UnboundID® Directory Server Administration Guide

Version 3.2.3

Version 3.2.3
April 25, 2012

Copyright

UnboundID Corp
13809 Research Blvd
Suite 500
Austin, TX 78750
www.UnboundID.com

Contents

1 Introduction

Overview

The UnboundID® Directory Server is a high performance, extensible, and scalable LDAP directory server, written completely in Java™. The Directory Server centralizes identity management information, subscriber data management, application configurations, and user credentials into a network, enterprise, and/or virtualized database environment. It provides seamless data management over a distributed system based on a standardized solution that meets the constant performance demands in today's markets. The Directory Server simplifies administration, reduces costs, and secures information in systems that scale for very large numbers of users.

This guide presents the basic procedures to quickly install and configure multiple directory instances and topologies. Future versions of this guide will continue to expand on the various concepts presented in this document as well as introduce new information. For specific questions, please contact your authorized support provider.

This chapter presents the following topics:

- Overview of the UnboundID Directory Server
- Administration Framework
- Location of the Tools
- Assumptions

Overview of the UnboundID Directory Server

The UnboundID Directory Server is a powerful, 100% Java, production-proven directory services solution for mission-critical and large-scale applications. The Directory Server provides an extensive feature-rich set of tools that can meet the production needs of your system.

- **Full LDAP Version 3 Implementation**. The Directory Server fully supports the Lightweight Directory Access Protocol version 3 (LDAP v3), which supports the Request For Comments (RFCs) specified in the protocol. The Directory Server provides a feature-rich solution that supports the core LDAPv3 protocol in addition to server-specific controls and extended operations.

- **High Availability**. The Directory Server supports N-way multi-master replication that eliminates single points of failure and ensures high availability for a networked topology. The Directory Server allows data to be stored across multiple machines and disk partitions for fast replication. The Directory Server also supports replication in entry-balancing proxy server deployments.

- **Administration Tools**. The Directory Server provides a full set of command-line tools, a Web-based administration management console, and a graphical user interface (GUI) Java-based setup tool to configure, monitor, and manage any part of the directory server. The Directory Server has a task-based subsystem that provides automated scheduling of basic functions, such as backups, restores, imports, exports, restarts, and shutdowns. The set of utilities also includes a troubleshooting support tool that aggregates system metrics into a zip file, which administrators can send to UnboundID for analysis.

- **Security Mechanisms**. The Directory Server provides extensive security mechanisms to protect data and prevent unauthorized access. Access control list (ACL) instructions are available down to the attribute value level and can be stored within each entry. The Directory Server allows connections over Secure Sockets Layer (SSL) through an encrypted communication tunnel. Clients can also use the StartTLS extended operation over standard, non-encrypted ports. Other security features include a privilege subsystem for fine-grained granting of rights, a password policy subsystem that allows configurable password validators and storage schemes, and SASL authentication mechanisms to secure data integrity, such as PLAIN, ANONYMOUS, EXTERNAL, CRAM-MD5, Digest-MD5, and GSSAPI. The Directory Server also supports various providers and mappers for certificate-based authentication in addition to the ability to encrypt specific entries or sensitive attributes.

- **Monitoring and Notifications**. The Directory Server supports monitoring entries using JConsole, over Simple Network Management Protocol (SNMP), or using the UnboundID® Directory Management Console. Administrators can track the response times for LDAP operations using a monitoring histogram as well as record performance statistics down to sub-second granularity. The Directory Server also provides configurable notifications, auditing, and logging subsystems with filtered logging capabilities.

- **Powerful LDAP SDK**. The Directory Server is based on a feature-rich LDAP SDK for Java, designed by UnboundID. The LDAP SDK for Java is a Java API standard for client application development that overcomes the many limitations of the Java Naming and Directory Interface (JNDI) model. (For example, JNDI does not address the use of LDAP controls and extended operations.) The LDAP SDK for Java provides support for controls and extended operations and allows clients to leverage the UnboundID Directory Server's extensible architecture for their applications. UnboundID provides Standard and Commercial versions of the LDAP SDK for Java. The Standard LDAP SDK for Java is available for free to public users and can be downloaded from the UnboundID web site. The Commercial version of the LDAP SDK for Java is part of the Directory Server distribution. For more information on the LDAP SDK for Java, go to the following site: **http://www.unboundid.com/products/ldapsdk**

- **Server SDK**. UnboundID also provides the UnboundID® Server SDK, which is a library of Java packages, classes, and build tools to help in-house or third-party developers create extensions for the UnboundID Directory Server, the UnboundID Directory Proxy Server, and the UnboundID® Synchronization Server. The servers were designed with a highly-extensible and scalable architecture with multiple plug-in points for your customization needs. The Server SDK provides APIs to alter the behavior of each server's components without affecting its code base.

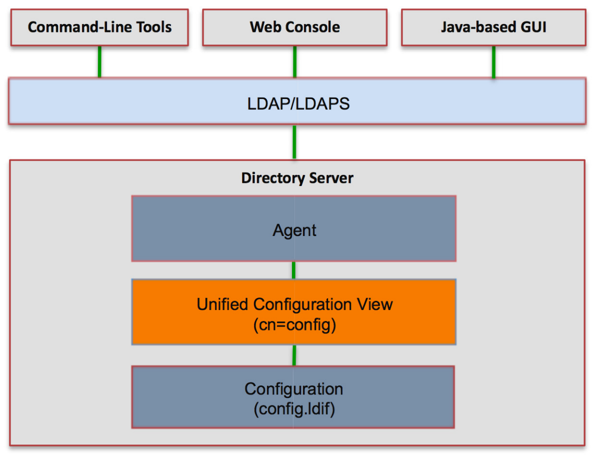
Note	Server SDK support is provided if you have purchased Premium Support for the product that you are developing extensions for. However, we do not provide support for the third party extensions developed using the Server SDK. You should contact your product level support group if you need assistance.

- **Multi-Platform Support**. The UnboundID® Directory Server is a pure Java application and is certified VMWare Ready™. It is intended to run within the Java Virtual Machine on any Java 6 Standard Edition (SE) or Enterprise Edition (EE) certified platform including, but not limited, to the following:
  - RedHat® Enterprise Linux® 5.5 (natively or running under VMware® ESX 4.0 Update 2)
  - Oracle® Solaris™ 10 update 9
  - RedHat®/CentOS 5.5 & 6
  - Windows Server® 2003 & 2008
  - Novell® SUSE® 11.3

Administration Framework

The UnboundID Directory Server provides an administration and configuration framework capable of managing stand-alone servers, server groups, and highly-available deployments that include multiple redundant directory server instances. Administrators can configure changes locally or remotely on a single directory server or on all servers in a server group. Each directory server configuration is stored as a flat file (LDIF) that can be accessed under the `cn=config` branch of the Directory Information Tree (DIT). Administrators can tune the configuration and perform maintenance functions over LDAP using a suite of command-line tools, a GUI-based Java console (for installs and uninstalls), or a web management console (for configuration and monitoring). The Directory Server also provides many plug-in points to extend the functionality of its components using custom code.

FIGURE 1-1. Directory Server Configuration Framework



Location of the Tools

UnboundID distributes the Directory Server, UnboundID Directory Server Management Console, and LDAP SDK for Java in zip format. After unzipping the file, you can access the `setup` utility in the instance root directory, located at the `UnboundID-DS` directory. You can quickly install a directory server instance in any location of your choice. The Directory Server stores a full set of command-line tools for maintaining your system in the `UnboundID-DS/bin` directory for UNIX® or Linux® machines and the `UnboundID-DS\bat` directory for Microsoft® Windows® machines.

Prior to installing the Directory Server, read Chapter 2 *Preparing Your Environment*, which presents important information on setting up your machines. Chapter 3 *Installing the Directory Server* presents procedures to install a server instance using the `setup` utility. This utility can be run in one of the three available installation modes: interactive command-line, non-interactive command-line, and GUI mode. Chapter 4 *Configuring the Directory Server* provides procedures to modify the configuration of a server instance or a group of servers using the command-line tools and the web management console.

Assumptions

This guide assumes that you have the following knowledge:

- Directory services and LDAP v3 concepts.
- System administration principles and practices.
- Understanding of Java and running Java applications.

2 Preparing Your Environment

Overview

The UnboundID Directory Server offers a highly portable and scalable Java-based architecture that can run on multiple platforms and operating systems. The Directory Server is specifically optimized for those operating systems used in production environments that process a very large number of entries.

This chapter presents topics and procedures to prepare your directory server environment. It provides information on filesystem recommendations, server settings, and other tips to get your production environment ready.

- Before You Begin
- Preparing the Operating System (Solaris)
- Preparing the Operating System (Linux)
- Running as a Non-Root User

Note	If you are conducting an initial evaluation of the UnboundID Directory Server, proceed to Chapter 3, Installing the Directory Server on page 27 to set up a stand-alone server using the GUI installer.

Before You Begin

The Directory Server requires certain software packages for the proper operation of the server. Make sure that you get the following software prior to your installation.

- **Java 6**. For optimized performance, the UnboundID Directory Server requires Java 6 (minimum required Sun/Oracle JDK 1.6.0_26, recommended Sun/Oracle JDK 1.6.0_31; minimum required JRE 1.6.0 IBM J9 2.4) for 32-bit and 64-bit architectures, respectively, depending on your system requirements.

Installing Java 6

The UnboundID Directory Server requires Java 6 (minimum required Sun/Oracle JDK 1.6.0_26, recommended Sun/Oracle JDK 1.6.0_31; minimum required JRE 1.6.0 IBM J9 2.4) for 32-bit and 64-bit architectures, respectively, depending on your system requirements. Even if your system already has Java installed, you may want to create a separate Java 6 installation for use by the UnboundID Directory Server to ensure that updates to the system-wide Java installation do not inadvertently impact the Directory Server. This setup requires that the JDK, rather than the JRE, for both the 32-bit and 64-bit versions or both, depending on your operating system, be downloaded.

To Install Java 6 (Oracle/Sun)

On Solaris™ systems, if you want to use the 64-bit version of Java, you need to install both the 32-bit and 64-bit versions. The 64-bit version of Java on Solaris is not a full stand-alone installation, but instead relies on a number of files provided by the 32-bit installation. Therefore, the 32-bit version should be installed first, and then the 64-bit version installed in the same location with the necessary additional files.

On other platforms (e.g., Linux, Microsoft Windows), the 32-bit and 64-bit versions of Java contain complete installations. If you only want to run the 64-bit version of Java, then it is not necessary to install the 32-bit JDK. If you do want to have both versions installed, then they should be installed in separate directories, because files cannot coexist in the same directory as they do on Solaris systems.

1. Open a browser and navigate to the following Oracle download site:

   `http://www.oracle.com/technetwork/java/javase/downloads/index.html`

2. Click the JDK link under the Java Platform (JDK), and select the Java version for your operating system.

To Install Java 6 (IBM)

1. Open a browser and navigate to the following IBM download site:

   `http://www.ibm.com/developerworks/java/jdk/`

2. Select the Java version for your operating system.

Preparing the Operating System (Solaris)

The UnboundID Directory Server has been extensively tested on multiple operating systems. We have found that several operating system optimizations lead to improved performance, which include using the ZFS filesystem on Solaris systems, restricting ZFS memory consumption, limiting transaction group writes, using compression and disabling access time updates.

Using ZFS

UnboundID strongly recommends the use of ZFS™ as the underlying filesystem on Solaris 10 and OpenSolaris™ systems. ZFS is a 128-bit filesystem that can store billions of times more data than traditional 64-bit systems. Based on a storage pool model, ZFS aggregates devices (mirrors, RAID-Z with single or double parity, concatenated or striped storage) into a virtual data source from which filesystems can be constructed. ZFS provides excellent performance, end-to-end data integrity, simple administration management, and unmatched scalability. It also provides many useful features, such as automatic checksum, dynamic striping, variable block sizes, compression, and unlimited constant-time snapshots. ZFS is part of the Solaris 10 and OpenSolaris operating systems.

All components of the Directory Server should be located on a single storage pool (zpool), rather than having separate pools configured for different server components (for example, one pool for the database and a second for log files). Single zpool configurations are the simplest and easiest to manage.

ZFS's copy-on-write transactional model does not require isolating I/O-intensive components. Therefore, all available disks should be placed in the same zpool, so that as many underlying spindles as possible can be used to provide the configuration with the greatest number of I/O operations per second.

Restricting ZFS Memory Consumption

Despite its excellent performance, ZFS does not release memory fast enough for some LDAP operations that might need it. This delay could cause some processes to fail to start while attempting to allocate a large amount of memory for a JVM heap.

To curb memory allocation problems, make sure that the system is configured to limit the amount of memory for caching (for example, up to two gigabytes). The Directory Server relies on database caching rather than filesystem caching for its performance. Thus, the underlying system should be configured so that the memory used by ZFS will not interfere with the memory used by the Directory Server. In most environments, UnboundID recommends that systems be configured to allow ZFS to use no more than 2 GB of memory for caching.

To Restrict ZFS Memory Consumption

1. Open the **`/etc/system`** file.

2. ZFS caches data from all active storage pools in the ARC cache. We can limit its memory consumption by setting the maximum size of the ARC cache using the **`zfs_arc_max`** property. For example, add the following line to the end of the **`/etc/system`** file. The property sets the maximum size of the ARC cache to 2 GB (0x80000000 or 2147483648 bytes) for ZFS. Note that your system may require a different value.

   ```
   set zfs:zfs_arc_max= 0x80000000
   ```

3. If your system processes large write operations, see the section on Limiting ZFS Transaction Group Writes. Otherwise, reboot the machine for the change to take effect. Also note

that this operation requires Solaris 10 update 4 (08/07) and Nevada (build 51) release or later.

Limiting ZFS Transaction Group Writes

UnboundID has found that the Directory Server can exhibit uneven throughput performance during continuous write loads for Oracle Berkeley DB Java Edition backends on ZFS systems. We have found that the ZFS Write Throttle feature stalls write operations when transaction groups are flushed to disk. During these periods, operation throughput can drop significantly with these large I/O bursts.

To smooth out write throughput and improve latency, we recommend setting the **`zfs_write_limit_override`** property in the **`etc/system`** file to the size of the available disk cache on the system.

To Limit ZFS Transaction Group Writes

1. Open the **`/etc/system`** file.
2. Add the following line to the end of the file. Set the value to the size of your onboard cache. For example, for a system that has a 32MB cache per disk, set the following parameter:

   ```
   set zfs:zfs_write_limit_override=0x2000000
   ```

3. For the change to take effect, reboot the machine. Also note that this operation requires Solaris 10 update 4 (08/07) or later.

ZFS Access to Underlying Disks

Storage requirements vary depending upon whether ZFS has access to the underlying disks. The following sections describe the implications of giving ZFS access to the disks versus not giving it access.

Giving ZFS Access to Underlying Disks

If possible, ZFS should be given direct access to the underlying disks that will be used to back the storage. Having direct access to the underlying disks makes it possible to configure the system with the greatest degree of reliability and flexibility.

In this configured setup, the zpool used for the Directory Server should have a RAID 1+0 configuration (a stripe across one or more 2-disk mirrors). Although this setup reduces the amount of available space when compared with other configurations, like RAID-Z (ZFS data-parity scheme with full dynamic stripe width) or RAID-Z2 (ZFS dual parity RAID-Z), RAID 1+0 provides dramatically better performance and much better reliability.

Denying ZFS Access to Underlying Disks

If ZFS cannot get direct access to the underlying disks, for example, the system only has access to a logical unit number (LUN) on a storage area network (SAN), then the provided storage should already include some level of redundancy (and again a RAID 1+0 configuration is recommended over other schemes like RAID 5 or RAID 6). If the storage includes redundancy, then the zpool should be created with only that LUN and should not add any additional redundancy. In such a configuration, ZFS is not able to take advantage of its advanced self-healing capabilities when it detects any corruption at the filesystem level. However, ZFS check-summing can still detect those types of problems.

Configuring ZFS Compression and Access Time Update

The ZFS filesystem used for the Directory Server should be configured with compression enabled. Enabling ZFS compression improves performance in the majority of cases, because it reduces the amount of data that needs to be written to or read from the underlying disks. In most cases, the reduced cost of the disk I/O outweighs the CPU cost of compressing and decompressing the data.

You should also disable the access time update (**`atime`**) tracking, so that the kernel will not update the access time of a file every time it is requested. ZFS stores the filesystem attributes within the filesystem itself, so that you can access the attributes using the **`zfs get all`** option.

To Configure ZFS Compression

If the ZFS filesystem is named **`ds`**, then use the commands listed below. The changes will take effect immediately with no need to reboot or perform any other action.

1. Turn on ZFS compression.

   ```
   # zfs set compression=on ds
   ```

2. Disable access time update for reads.

   ```
   # zfs set atime=off ds
   ```

3. To view all filesystem attributes, use the following command:

   ```
   # zfs get all ds
   ```

Caution	Knowing the actual size of files is useful when you need to: 1) back up files to a non-ZFS filesystem, 2) run a binary copy initialization over the network, or 3) estimate the amount of memory dedicated to caching. On traditional UNIX filesystems, the **`du`** command reports the sum of all the specified file sizes. However, on ZFS, **`du`** reports the amount of disk space consumed, which might not equal the sum of the file sizes if features like compression or multiple copies are enabled. Administrators should be aware of this difference when determining the database size using **`du`**. Instead of using **`du`**, UnboundID Directory Server provides a utility, **`bin/sum-file-sizes`**, that determines the size (in bytes, kilobytes, megabytes, or gigabytes) of the sum of a set of files even if ZFS compression or multiple copies are enabled.

Preparing the Operating System (Linux)

The UnboundID Directory Server has been extensively tested on multiple operating systems. We have found that several operating system optimizations lead to improved performance. These optimizations include increasing the file descriptor limit on Linux systems, setting filesystem flushes (Linux), editing OS-level environment variables, downloading some useful monitoring tools for Redhat Linux systems, and configuring for Large Page support.

Setting File Descriptor Requirements (for Linux Platforms)

The UnboundID Directory Server allows for an unlimited number of connections by default, but is restricted by the file descriptor limit on the operating system. Many Linux distributions have a default file descriptor limit of 1024 per process, which may be too low for the Directory Server if it needs to handle a large number of concurrent connections.

To fix this condition, we recommend setting the maximum file descriptor limit per process to 65,535 on Linux systems.

For example, for Linux kernel 2.6.27 or later, a default **`epoll`** resource limit on the amount of requested kernel memory is defined. In most cases, this **`epoll`** resource limit is too low for most directory applications (for example, 128 bytes on some systems). The **`epoll`** resource limit should be set to 65,535. See the instructions below to set this value.

Note	On Solaris systems (SPARC, x86, x64), run the Directory Server as a user, or role, that contains the **`sys_resource`** privilege. Any process with this privilege can automatically request a higher number of file descriptors.

To Increase the File Descriptor Limit on Linux Systems

1. Display the current kernel version of your system. Check if the kernel is 2.6.27 or later. If so, the `epoll` resource limit must be changed.

   ```
   $ uname -r
   ```

2. Display the current hard limit of your system. The hard limit is the maximum server limit that can be set without tuning the kernel parameters in the `proc` filesystem.

   ```
   $ ulimit -aH
   ```

3. Edit the `/etc/sysctl.conf` file. If there is a line that sets the value of the `fs.file-max` property, make sure its value is set to 65535. If there is no line that sets a value for this property, add the following line to the end of the file:

   ```
   fs.file-max = 65535
   ```

4. If the Linux kernel is 2.6.27, add the following line in the `/etc/sysctl.conf` to set the `epoll` resource limit:

   ```
   fs.epoll.max_user_instances = 65535
   ```

5. Edit the `/etc/security/limits.conf` file. If the file has lines that sets the soft and hard limits for the number of file descriptors, make sure the values are set to **65535**. If the lines are not present, add the following lines to the end of the file (before “#End of file”). Also note that you should insert a tab, rather than spaces, between the columns:

   ```
   * soft nofile 65535
   * hard nofile 65535
   ```

6. Reboot your system, and then use the `ulimit` command to verify that the file descriptor limit is set to 65535.

   ```
   $ ulimit -n
   ```

7. Check that the epoll resource limit was configured correctly.

   ```
   $ cat /proc/sys/fs/epoll/max_user_instances
   ```

Configuring the Number of File Descriptors Used by the Directory Server

By default, the Directory Server will attempt to use 65535 file descriptors. If this limit is not appropriate for your deployment, or you cannot configure your system to change the number of available file descriptors as presented in the previous section but feel that your system's existing maximum file descriptor limit is sufficient, then you can configure the number of file descriptors that the server will attempt to use. This can be accomplished by setting a `NUM_FILE_DESCRIPTORS` environment variable before attempting to start the server or run a tool which may require a large number of file descriptors. Alternately, you may create a file named "num-file-descriptors" in the server's `config` directory with a single line that sets the value of the `NUM_FILE_DESCRIPTORS` variable, like:

```
NUM_FILE_DESCRIPTORS=12345
```

Regardless of whether the server will attempt to use the default number of file descriptors or an explicitly-configured number, if an error occurs while trying to use that file descriptor limit then an error message will be written to the terminal. If this error occurs when trying to start the server, then that message will also be written to the server's error log.

Setting Filesystem Flushes (Linux)

With the out-of-the-box settings on Linux systems running the **`ext3`** filesystem, the data is only flushed to disk every five seconds. If the Directory Server is running on a Linux system using the **`ext3`** filesystem, consider editing the mount options for that filesystem to include **`commit=1`** in **`/etc/fstab`**, which changes the flush frequency from five seconds to one second.

Editing OS-Level Environment Variables

Certain environment variables can impact the Directory Server in unexpected ways. This is particularly true for environment variables that are used by the underlying operating system to control how it uses non-default libraries.

For this reason, the Directory Server explicitly overrides the values of key environment variables like **`PATH`**, **`LD_LIBRARY_PATH`**, and **`LD_PRELOAD`** to ensure that environment variables used to start the server do not inadvertently impact its behavior.

If there is a legitimate need to edit any of these environment variables, the values of those variables should be set by manually editing the **`set_environment_vars`** function of the **`lib/_script-util.sh`** script, which is set at the directory server's runtime. You will need to stop (**`bin/stop-ds`**) and restart (**`bin/start-ds`**) the server for the change to take effect.

Note	On Solaris systems, the UnboundID Directory Server automatically adds, if present, **`libumem.so`** to the **`LD_PRELOAD`** (and **`LD_PRELOAD_32`** and **`LD_PRELOAD_64`**) environment variables so that the **`umem`** memory manager will be automatically enabled. The Directory Server also uses **`priocntl`**, if present, to enable the fixed priority scheduler.

Install sysstat and pstack (Red Hat)

For Red Hat® Linux® systems, you should install a couple of packages, **`sysstat`** and **`pstack`**, that are disabled by default but are useful for troubleshooting purposes in the event that a problem occurs. The Directory Server's troubleshooting tool, **`collect-support-data`**, uses the **`iostat`**, **`mpstat`**, and **`pstack`** utilities plus others to collect monitoring, statistics and stack trace information on the server's processes.

Disable File System Swapping (Linux)

For all deployments, we recommend disabling disk swapping on the filesystem to protect the Directory Server JVM process from an overly aggressive filesystem cache.

In the **/etc/sysctl.conf** file, set:

```
vm.swappiness = 0
echo 0 > /proc/sys/vm/swappiness
```

or run:

```
# sysctl -w vm.swappiness=0
```

Set noatime on ext3 Systems

If you are using an ext3 filesystem, it is recommended that you set **noatime**, which turns off any **atime** updates during read accesses to improve performance.

```
# mount -t ext3 -o noatime /dev/fs1
```

Configure Large Page Support (Linux)

We recommend configuring Large Page support to provide a performance gain for your system of about a 5–10 percent performance improvement. Large Page support is especially useful for virtualized environments using VMWare. If you are configuring a VMWare-based system, you will need to calculate your memory configurations for your particular system. Follow the recommended VMWare Tuning Guidelines presented on their web site.

As a general guideline, you should ensure that the size of the large page is set slightly higher than the maximum size of your JVM settings. For example, for a total system memory of 96 GB, you could set the JVM memory to 80GB, then configure your system to provide ~85GB of large page support.

To Configure Large Page Support on Linux

1. Log in as root. For this example, we assume you are using at least Redhat Linux Enterprise 5.5.

2. Verify that your kernel supports large pages. On Linux systems, *Large Pages* are referred to as *Huge Pages*. If the file contains "HugePage_Total," "HugePages_Free" or "Hugepagesize," then your kernel supports large pages. Make note of the large page size of your system, which is dependent on your system architecture.

   ```
   $ cat /proc/meminfo | grep Huge
   HugePages_Total: 0
   HugePages_Free:  0
   HugePages_Rsvd:  0
   HugePagesize: 2048 kB
   ```

3. Set the maximum amount of memory in the server. For 64-bit JVMs that support large pages, you need to set the kernel for shared memory to be slightly higher than the maximum size of your JVM. This memory is "pinned" or reserved for the application once the JVM is started. For this example, set the maximum shared memory to 85 GB. In the **/etc/sysctl.conf** file, you need to add a line as follows:

```
kernel.shmmax = <number of bytes>
```

For example, for 85 GB, add the line:

```
kernel.shmmax = 91268055040
```

4. Next, set a virtual memory kernel parameter to tell the OS how many large pages you want to set aside. In the `/etc/sysctl.conf` file, you need to add the following line:

   ```
   vm.nr.hugepages = <number of pages>
   ```

   For example, we want to set the virtual memory to 85 GB (89128960 kB), which is 85 GB/ 2 MB (obtained in step 2 as the size of each large page). Thus, 89128960 kB/2048 kB = 43520, which is the number of large pages we want to reserve. This setting only takes effect at boot time.

   ```
   vm.nr.hugepages = 43520
   ```

   To set it without reboot, run one of the following commands:

   ```
   sysctl -w vm.nr_hugepages=43520
   ```

   If you want the setting to be present after reboot, then you have to modify the `/etc/sysctl.conf` file.

5. Next, create a new group called *hugetlb*, which appears in `/etc/group`. The group will set the permission to access the shared memory in the next step. Also you need to add the userid that the JVM is running as (for example, admin) to the group.

   ```
   # groupadd hugetlb
   # useradd -g hugetlb admin
   ```

6. In the `/etc/sysctl.conf` file, we want to give permission for the process to access the shared memory, so add the following line:

   ```
   vm.hugetlb_shm_group = gid
   ```

   where *gid* is the groupid. In this example, it is 501.

7. Next, we need to ensure that large page shared memory does not get swapped to disk. In the `/etc/security/limits.conf` file, add the following lines if not present:

   ```
   admin soft memlock n
   admin hard memlock n
   ```

   where *n* is equal to the total virtual memory (i.e., number of large pages times the page size. Thus, 43520 * 2048 = 89128960 kB).

8. Reboot the machine. Repeat step 2 to ensure that large page memory is configured.

   ```
   $ cat /proc/meminfo | grep Huge
   HugePages_Total: 44564
   HugePages_Free:  44564
   HugePages_Rsvd:  0
   HugePagesize: 2048 kB
   ```

9. Finally, go to the Directory Server root. Assuming you are using Sun JVM, set the **`-XX:+UseLargePages`** JVM option in the Directory Server's **`config/java.properties`** file for the **`start-ds`** tool and **`import-ldif`** tools.

```
# These JVM arguments can be used to run the Directory Server with an aggressive
memory tuning:

# start-ds.java-args=-d64 -server -Xmx80g -Xms80g -XX:+UseConcMarkSweepGC -
XX:+CMSConcurrentMTEnabled -XX:+CMSParallelRemarkEnabled -XX:+CMSParallelSurvivor-
RemarkEnabled -XX:+CMSScavengeBeforeRemark -XX:RefDiscoveryPolicy=1 -XX:ParallelC-
MSThreads=1 -XX:CMSMaxAbortablePrecleanTime=3600000
-XX:CMSInitiatingOccupancyFraction=80 -XX:+UseParNewGC -XX:+UseMembar -XX:+UseBi-
asedLocking -XX:+UseCompressedOops -XX:PermSize=64M -XX:+HeapDumpOnOutOfMemoryError
-XX:+UseLargePages

# These JVM arguments can be used to do an offline LDIF import with an aggressive
memory tuning:

# import-ldif.offline.java-args=-d64 -server -Xmx80g -Xms80g -XX:+UseParallelGC -
XX:+UseMembar -XX:NewRatio=8 -XX:+UseNUMA -XX:+UseCompressedOops -XX:+UseNUMA -
XX:+HeapDumpOnOutOfMemoryError -XX:+UseLargePages
```

10. On the Directory Server, run the **`dsjavaproperties`** tool to save the JVM settings.

```
$ bin/dsjavaproperties
```

You have successfully set up Large Page Support on your Linux systems.

Running as a Non-Root User

On UNIX systems, historically, only the root user has been allowed to bind to ports below 1024. However, for security reasons, network applications should not run as root, and in many environments, administrators for those applications cannot have root access to the underlying system.

Running as a Non-Root User on Solaris Systems

On systems running Solaris 10 and OpenSolaris, you can use the User and Process Rights Management subsystems with Role Based Access Control (RBAC) mechanisms to grant users (or roles) only the privileges necessary to accomplish a specific task. Using RBAC avoids the assignment of full super-user (root) privileges to the user. For example, you can grant the **`net_privaddr`** privilege to a non-root user (or role) that gives him or her the ability to listen on privileged ports (that is, ports 1024 and below). Similarly, granting the **`sys_resource`** privilege allows a user to bypass restrictions on resource limits, such as the number of file descriptors a process might use.

The Solaris User and Process Rights Management subsystems can also be used to remove capabilities from users. For example, removing the **`proc_info`** privilege from a user prevents that user from seeing processes owned by other users. Removing the **`file_link_any`** privilege can prevent users from creating hard links to files owned by other users. Hard links are not needed by the Directory Server and can represent a security risk under certain conditions. Table 2-1 summarizes the Solaris privileges that you may want to assign to non-root users.

TABLE 2-1. Summary of Notable Solaris Privileges

Privileges	Descriptions
net_privaddr	Provides the ability to listen on privileged network ports.
sys_resource	Provides the ability to bypass restrictions on resource limits (including the number of available file descriptors).
proc_info	Provides the ability for users to see processes owned by other users on the system. This privilege is available to all users by default, but it can pose a security risk in some cases. UnboundID recommends that it be removed from the role used by the Directory Server.
file_link_any	Provides the ability to create hard links to files owned by other users on the system. This privilege is available to all users by default, but it can pose a security risk in some cases. UnboundID recommends that it be removed from the role used by the Directory Server.

Running as a Non-Root User on Linux Systems

Linux systems do not provide a direct analog to the Solaris User and Process Rights Management subsystems. As a result, there is no easy way to allow a non-root user to listen on a privileged port. To run as a non-root user but still allow connections on a privileged port, two options are available:

- In many environments, the server can be run on a non-privileged port but can be hidden by a hardware load-balancer or LDAP proxy server.
- Alternately, the `netfilter` mechanism (exposed through the `iptables` command) can be used to automatically redirect any requests from a privileged port to the unprivileged port on which the server is listening.

Creating a Solaris Role for Multiple Administrators

To give multiple administrators access to the Directory Server, UnboundID recommends that a Solaris role be created to run the server and that all necessary administrators be added to that role. The Solaris role provides an audit trail that can be used to identify which administrator performed a given action, while still allowing administrators to run the server, to view and edit files, and to execute commands as that same user. As with normal user accounts, roles can be assigned privileges. The role used for the Directory Server should include the `net_privaddr` and `sys_resource` privileges and should exclude the `proc_info` and `file_link_any` privileges for improved security (that is, to eliminate the need for root access).

To Create a Solaris Role for the Directory Server

Create a Solaris role named `ds` with all of the appropriate privileges needed to run the Directory Server, and assign a password for that role.

1. Create the role.

```
$ roleadd -d /export/home/ds -m -s /usr/bin/bash \
  -K defaultpriv=basic,net_privaddr,sys_resource,-proc_info,-file_link_any ds
```

2. Assign a password.

```
$ passwd ds
```

3. For each administrator who is allowed to manage the Directory Server, assign the role with the **`usermod`** command. For example, to give someone with a user name of "john" the ability to assume the **`ds`** role, issue the following command:

```
$ usermod -R ds john
```

Note

If a user is already a member of one or more roles, then the entire list of existing roles, separated by commas, must also be provided or the user will be removed from those roles. For example, if the root account is also a role and the user "john" is also a member of that role, then the command would be:

```
$ usermod -R root,ds john
```

4. Log in using a normal user account and then use the **`bin/su`** command to assume the role created for the Directory Server. You cannot log directly into a system as a role. Only users that have been explicitly assigned to a role will be allowed to assume it.

3 Installing the Directory Server

Overview

After you have prepared your system based on the instructions in Chapter 2, you can begin the setup process using one of the UnboundID Directory Server's easy-to-use installation modes. This chapter presents the various installation options and procedures available to the administrator:

Getting the Installation Packages

To begin the installation process, obtain the latest ZIP release bundle from UnboundID and unpack it in a folder of your choice. The release bundle contains the Directory Server code, tools, package documentation, and the latest production release (JAR file) for the Oracle® Berkeley DB Java Edition (JE), which is used for the database backend.

To Unpack the Software for Installation

1. Download the latest ZIP distribution of the server software.
2. Unzip the file in a directory of your choice. In this example, the release bundle is unpacked in the **ds** directory.

   ```
   $ unzip UnboundID-DS-<version>.zip -d ds
   ```

   You can now set up the Directory Server.

About the Layout of the Directory Server Folders

Once you have unzipped the Directory Server ZIP distribution file, you will see the following folders and command-line utilities, shown in Table 3-1.

TABLE 3-1. Directory Server Directories and Tools Under Server Root

Under Server Root	Description
License.txt	Licensing agreement for the Directory Server.
README	README file that describes the steps to set up and start the Directory Server.
bak	Stores the physical backup files used with the **backup** command-line tool.
bat	Stores Windows-based command-line tools for the Directory Server.
bin	Stores UNIX/Linux-based command-line tools for the Directory Server.
classes	Stores any external classes for server extensions.
config	Stores the configuration files for the backends (admin, config) as well as the directories for messages, schema, tools, and updates.
db	Stores the Oracle Berkeley Java Edition database files for the Directory Server.
docs	Provides the release notes, Configuration Reference file and a basic Getting Started Guide (HTML).
import-tmp	Stores temporary imported items.
ldif	Stores any LDIF files that you may have created or imported.
legal-notices	Stores any legal notices for dependent software used with the Directory Server.
lib	Stores any scripts, jar, and library files needed for the server and its extensions.
locks	Stores any lock files in the backends.
logs	Stores log files for the Directory Server.
resource	Stores the MIB files for SNMP.
revert-update	The **revert-update** tool for UNIX/Linux systems.
revert-update.bat	The **revert-update** tool for Windows systems.
setup	The **setup** tool for UNIX/Linux systems.
setup.bat	The **setup** tool for Windows systems.

TABLE 3-1. **Directory Server Directories and Tools Under Server Root**

Under Server Root	Description
uninstall	The `uninstall` tool for UNIX/Linux systems.
uninstall.bat	The `uninstall` tool for Windows systems.
update	The `update` tool for UNIX/Linux systems.
update.bat	The `update` tool for Windows systems.

About the Directory Server Setup Modes

One of the strengths of the UnboundID Directory Server is the ease with which you can install a server instance using the `setup` tool. The `setup` tool allows you to install a stand-alone server instance, configure it with a minimum set of options, and populate it with generated sample data. The tool provides options to set up an SSL or StartTLS connection or set up a replication topology.

To install a stand-alone server instance, run the `setup` tool in one of three modes: graphical user interface (GUI), interactive command-line, or non-interactive command-line mode.

- **Graphical User Interface (GUI) Mode**. Used primarily for evaluation purposes, GUI mode launches a Java-based graphical `setup` tool that enables you to install the Directory Server, load it with data, and get it running quickly on desktop systems. If a graphical environment is not available, then `setup` will fall back to the interactive command-line mode.

- **Interactive Command-Line Mode**. Interactive command-line mode prompts for information during the installation process. To run the installation in this mode, use the `setup --cli` command. If the `--cli` option is not present, `setup` defaults to GUI mode if a graphical environment is available.

- **Non-interactive Command-Line Mode**. Non-interactive command-line mode is designed for setup scripts to automate installations or for command-line usage. To run the installation in this mode, `setup` must be run with the `--no-prompt` option as well as all other arguments that are required to define the appropriate initial configuration.

All installation and configuration steps should be performed while logged on to the system as the user or role under which the Directory Server will run.

Setting Up the Directory Server for Evaluation Purposes

To quickly evaluate the UnboundID Directory Server, use the Graphical user interface (GUI) installation mode to set up a server instance. GUI mode provides an easy-to-use Java-based graphical installer that allows fast deployment of your UnboundID Directory Server instance. If your system does not support a graphical environment, the installation falls back to interactive command-line mode. If you want to set up a multi-master replication topology using the GUI installer, skip to the next section “Setting Up a Replication Topology for Evaluation Purposes” on page 27 and follow its instructions.

To Set Up the Directory Server for Evaluation Purposes

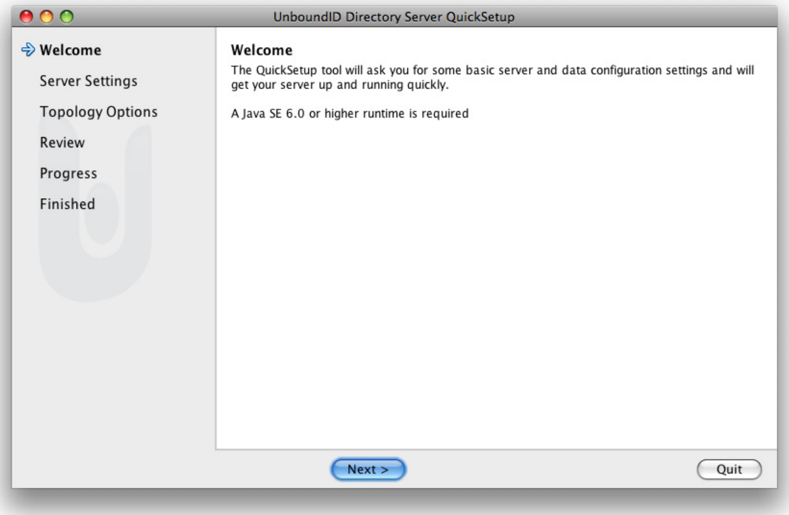
1. After you have unpacked the UnboundID software, go to the installation directory.

   ```
   $ cd /ds/UnboundID-DS
   ```

2. Install the Directory Server by launching **`setup`** with the appropriate **`JAVA_HOME`** directory. By default, the utility launches the graphical tool if your system supports a graphical environment.

   ```
   $ env JAVA_HOME=/ds/java ./setup
   ```

3. On the UnboundID Directory Server QuickSetup dialog, click **Next**.

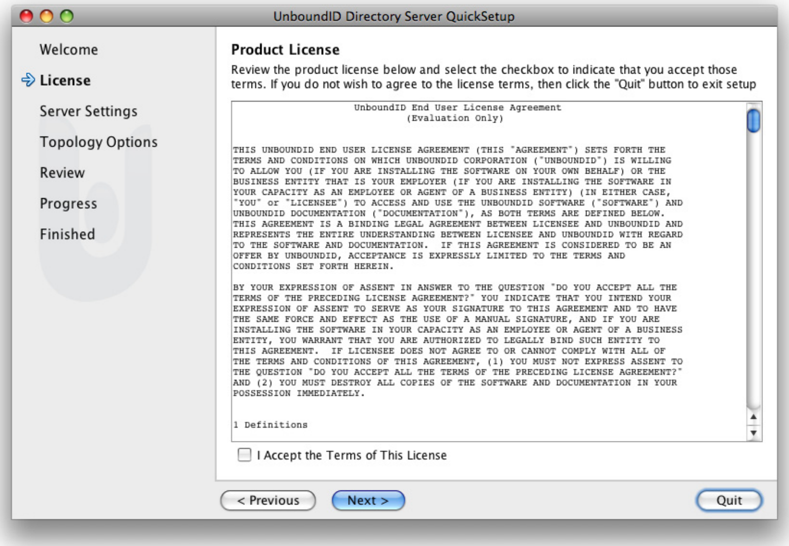


Welcome

The QuickSetup tool will ask you for some basic server and data configuration settings and will get your server up and running quickly.

A Java SE 6.0 or higher runtime is required

4. Read the licensing and terms of agreement. If you agree to its terms, select “**I accept the Terms of This License**,” and then click **Next**.



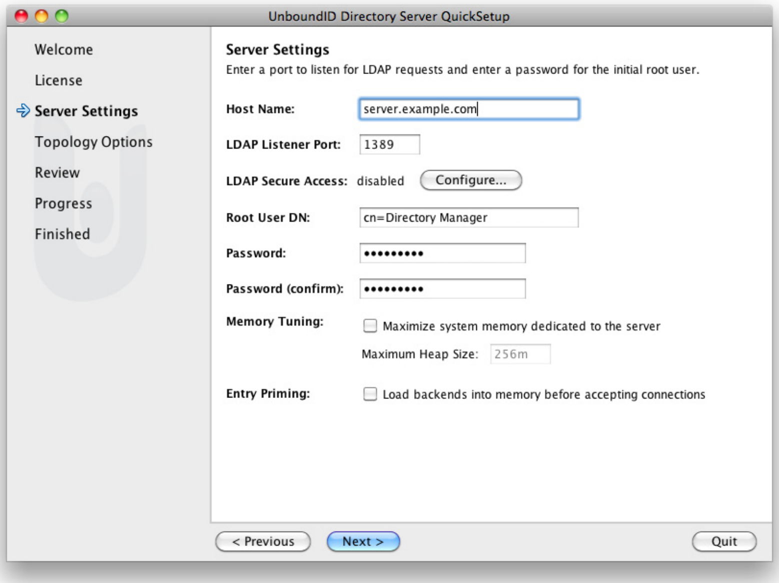
Product License

Review the product license below and select the checkbox to indicate that you accept those terms. If you do not wish to agree to the license terms, then click the "Quit" button to exit setup

```
                    UnboundID End User License Agreement
                             (Evaluation Only)


THIS UNBOUNDID END USER LICENSE AGREEMENT (THIS "AGREEMENT") SETS FORTH THE
TERMS AND CONDITIONS ON WHICH UNBOUNDID CORPORATION ("UNBOUNDID") IS WILLING
TO ALLOW YOU (IF YOU ARE INSTALLING THE SOFTWARE ON YOUR OWN BEHALF) OR THE
BUSINESS ENTITY THAT IS YOUR EMPLOYER (IF YOU ARE INSTALLING THE SOFTWARE IN
YOUR CAPACITY AS AN EMPLOYEE OR AGENT OF A BUSINESS ENTITY) (IN EITHER CASE,
"YOU" or "LICENSEE") TO ACCESS AND USE THE UNBOUNDID SOFTWARE ("SOFTWARE") AND
UNBOUNDID DOCUMENTATION ("DOCUMENTATION"), AS BOTH TERMS ARE DEFINED BELOW.
THIS AGREEMENT IS A BINDING LEGAL AGREEMENT BETWEEN LICENSEE AND UNBOUNDID AND
REPRESENTS THE ENTIRE UNDERSTANDING BETWEEN LICENSEE AND UNBOUNDID WITH REGARD
TO THE SOFTWARE AND DOCUMENTATION.  IF THIS AGREEMENT IS CONSIDERED TO BE AN
OFFER BY UNBOUNDID, ACCEPTANCE IS EXPRESSLY LIMITED TO THE TERMS AND
CONDITIONS SET FORTH HEREIN.

BY YOUR EXPRESSION OF ASSENT IN ANSWER TO THE QUESTION "DO YOU ACCEPT ALL THE
TERMS OF THE PRECEDING LICENSE AGREEMENT?" YOU INDICATE THAT YOU INTEND YOUR
EXPRESSION OF ASSENT TO SERVE AS YOUR SIGNATURE TO THIS AGREEMENT AND TO HAVE
THE SAME FORCE AND EFFECT AS THE USE OF A MANUAL SIGNATURE, AND IF YOU ARE
INSTALLING THE SOFTWARE IN YOUR CAPACITY AS AN EMPLOYEE OR AGENT OF A BUSINESS
ENTITY, YOU WARRANT THAT YOU ARE AUTHORIZED TO LEGALLY BIND SUCH ENTITY TO
THIS AGREEMENT.  IF LICENSEE DOES NOT AGREE TO OR CANNOT COMPLY WITH ALL OF
THE TERMS AND CONDITIONS OF THIS AGREEMENT, (1) YOU MUST NOT EXPRESS ASSENT TO
THE QUESTION "DO YOU ACCEPT ALL THE TERMS OF THE PRECEDING LICENSE AGREEMENT?"
AND (2) YOU MUST DESTROY ALL COPIES OF THE SOFTWARE AND DOCUMENTATION IN YOUR
POSSESSION IMMEDIATELY.


1 Definitions
```

☐ I Accept the Terms of This License

5. On the Server Settings dialog, type the server information described below, and click **Next**.



Server Settings

Enter a port to listen for LDAP requests and enter a password for the initial root user.

Host Name:	server.example.com
LDAP Listener Port:	1389
LDAP Secure Access:	disabled
Root User DN:	cn=Directory Manager
Password:	•••••••••
Password (confirm):	•••••••••
Memory Tuning:	☐ Maximize system memory dedicated to the server Maximum Heap Size: 256m
Entry Priming:	☐ Load backends into memory before accepting connections

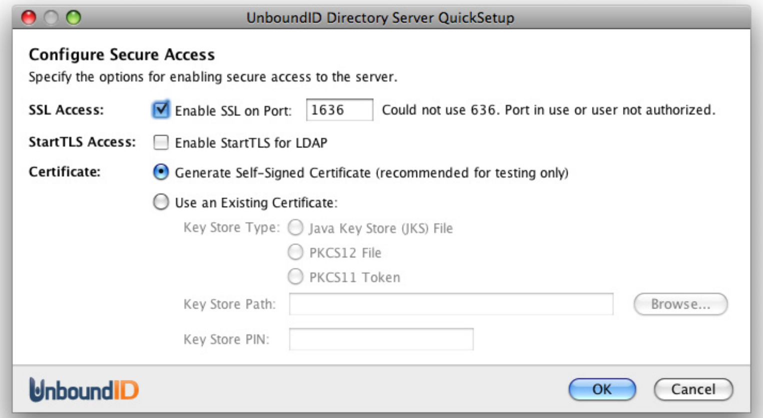
TABLE 3-2. Server Settings Dialog

Host Name	Directory Server’s host name or IP address. The default is your local host’s name.
LDAP Listener Port	LDAP port for the Directory Server. If you run the `setup` tool as the root user (or as a user or role with the net_privaddr privilege), the default port is 389. If you run the `setup` tool as a non-root user, the default port is 1389.
LDAP Secure Access	See step 6 below.
Root User DN	Default cn=Directory Manager.
Password	Root user bind DN password.

TABLE 3-2. Server Settings Dialog

Password Confirm	Confirm the password.
Memory Tuning	Maximize the memory for JVM optimization. This option should only be selected if the Directory Server is the primary application, and no other process consumes a significant amount of memory. You can specify the maximum heap size by entering the value with "m" for megabytes (for example, 256m) or "g" for gigabytes (for example, 1g). See "JVM Properties for Both Server and Command-Line Tools" on page 80 for more information on the Memory Tuning feature.
Entry Priming	Set the prime method property to preload, which loads the contents of the database from disk into cache memory at startup before accepting connections. Enabling priming ensures that the server provides optimum performance as soon as startup has completed. Note that entry priming can greatly increase the startup time of the Directory Server but will optimize overall performance. Select this option if you have strict throughput or response time performance requirements, or if you plan to have other replicas in a topology that can accept traffic while this server is starting. Setting the prime method to preload also allows the Directory Server to determine a recommended value for the CMSInitiatingOccupancyFraction when a Java garbage collection pause occurs. See "JVM Garbage Collection Using CMS" on page 81.

6. To set up SSL or StartTLS, click **Configure** (next to LDAP Secure Access). Fill in the information described below and click **OK** to continue.



Configure Secure Access

Specify the options for enabling secure access to the server.

SSL Access: [✓] Enable SSL on Port: 1636 Could not use 636. Port in use or user not authorized.

StartTLS Access: [ ] Enable StartTLS for LDAP

Certificate: [● Generate Self-Signed Certificate (recommended for testing only)] ○ Use an Existing Certificate:

Key Store Type: ○ Java Key Store (JKS) File ○ PKCS12 File ○ PKCS11 Token

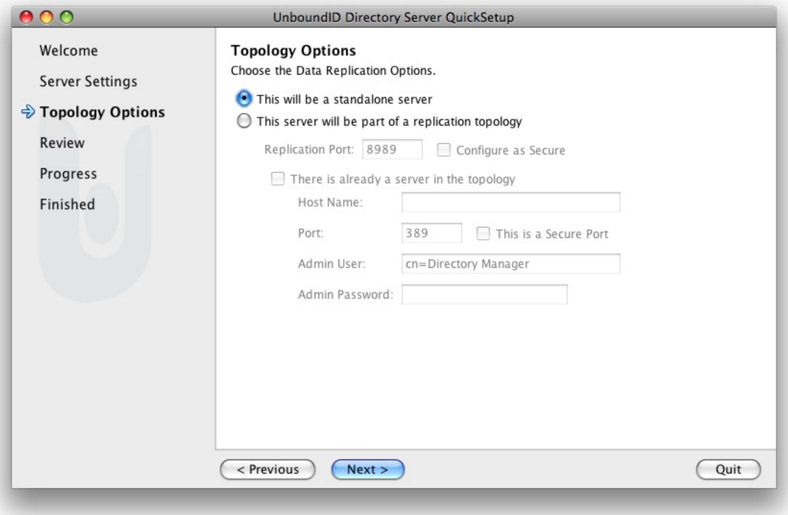
Key Store Path:

Key Store PIN:

TABLE 3-3. Configure Secure Access Dialog

SSL Access	Enable SSL on the specified port. If you run the `setup` tool as a root user or sufficiently privileged user, the default secure port is 636. If you run the `setup` tool as a non-root user, the default secure port is 1636.
StartTLS Access	Enable StartTLS for LDAP. StartTLS allows clients to promote a non-secure LDAP connection on the standard LDAP port to a secure connection.
Certificate	Generate a self-signed certificate if you are in a testing environment. Or use an existing certificate, and then select the keystore type, path and PIN if necessary.

7. On the Topology Options dialog, select whether the Directory Server is a standalone server or part of a replication topology, and then click **Next**. For instructions on setting up replication using the GUI tool, see "Setting Up a Replication Topology for Evaluation Purposes" on page 27 for more information.



Topology Options

Choose the Data Replication Options.

[✓ This will be a standalone server]

This server will be part of a replication topology

Replication Port: 8989 Configure as Secure

There is already a server in the topology

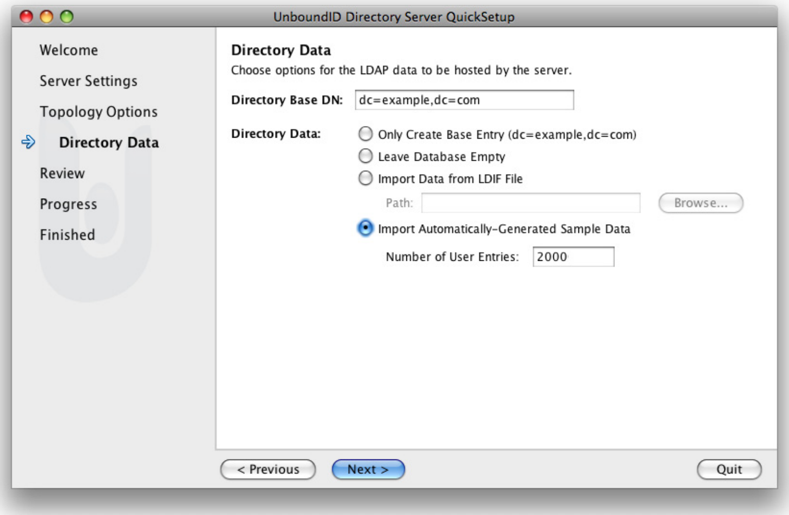
Host Name:

Port: 389 This is a Secure Port

Admin User: cn=Directory Manager

Admin Password:

8. On the Directory Data dialog, specify how to load data into your directory, and then click **Next**.



Directory Data

Choose options for the LDAP data to be hosted by the server.

Directory Base DN: dc=example,dc=com

Directory Data:

Only Create Base Entry (dc=example,dc=com)

Leave Database Empty

Import Data from LDIF File

Path:

[✓ Import Automatically-Generated Sample Data]

Number of User Entries: 2000

TABLE 3-4. Directory Data Dialog

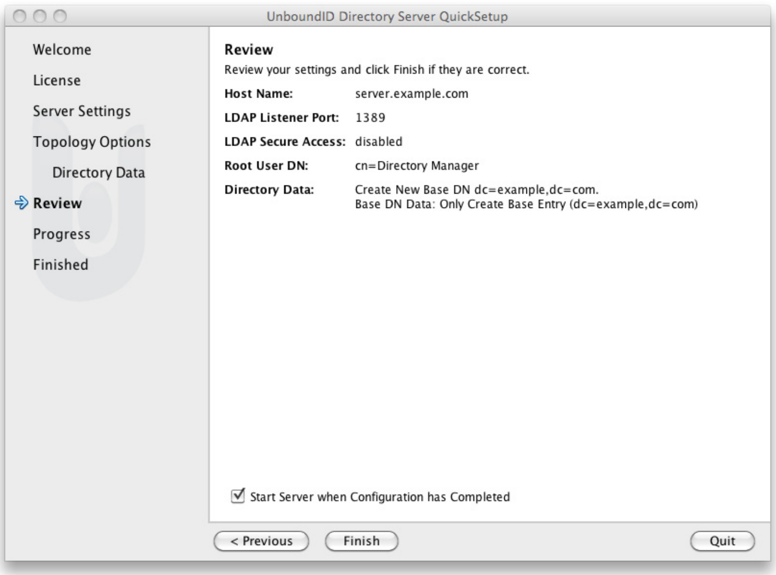
Directory Base DN	Base DN for your directory.
Directory Data:	One of the data options:
Only Create Base Entry	Create a base entry but no children entries below the base DN. By default, the Directory Server uses **`dc=example,dc=com`**. You can enter a base DN for your company.
Leave Database Empty	Create a stand-alone directory server that contains no data. See "Initializing the Directory Server" on page 42 for information on populating the database.

TABLE 3-4. **Directory Data Dialog**

Import Data from LDIF File	Populate the Directory Server with existing LDIF data. Type the path to the LDIF file or click Browse to select the path.

Import Automatically-Generated Sample Data	Populate the Directory Server with sample data for testing purposes, and type the number of sample user entries. The default number of entries is 2000. This option is convenient if you are quickly setting up and evaluating the Directory Server. The Directory Server generates sequential user entries for quick searches over specified DN ranges and loads the searchable entries that are sequentially ordered as follows:
	`uid=user.0,dc=People,dc=example,dc=com`
	`uid=user.1,dc=People,dc=example,dc=com`
	`uid=user.n,dc=People,dc=example,dc=com`
	where **`n`** is the number of entries specified during setup.

9. On the Review dialog, review your configuration. Select **Start Server when Configuration has Completed** to automatically start the server after it has been configured. Click **Finish** to complete the setup.



Review

Review your settings and click Finish if they are correct.

Host Name: server.example.com

LDAP Listener Port: 1389

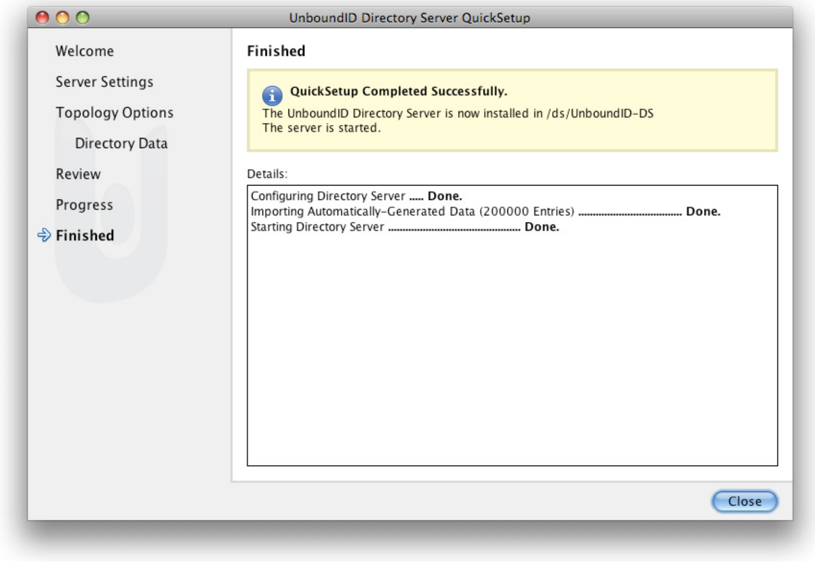
LDAP Secure Access: disabled

Root User DN: cn=Directory Manager

Directory Data: Create New Base DN dc=example,dc=com.
Base DN Data: Only Create Base Entry (dc=example,dc=com)

☑ Start Server when Configuration has Completed

10. On the Finished dialog, click **Close** to finish.



Finished

QuickSetup Completed Successfully.
The UnboundID Directory Server is now installed in /ds/UnboundID-DS
The server is started.

Details:

```
Configuring Directory Server ..... Done.
Importing Automatically-Generated Data (200000 Entries) .................................. Done.
Starting Directory Server ............................................ Done.
```

11. Read the section “Where To Go From Here” on page 39.

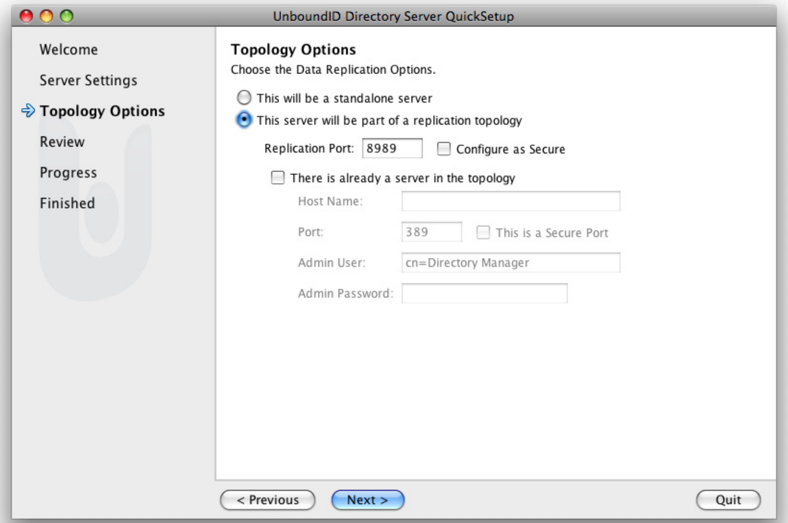
Setting Up a Replication Topology for Evaluation Purposes

The UnboundID Directory Server’s GUI setup tool allows you to quickly install a multi-master replication topology on desktop systems that support a graphical environment. The following procedures show how to configure three multi-master replication servers using the GUI setup tool for evaluation purposes. Production deployments require more fine-grained setup procedures that must be run from the command line or from scripts. For production environments, follow the instructions in “Replication Configuration” on page 383.

To Install a Replication Topology Using the GUI

1. For the first directory server instance, repeat steps 1–5 in “Setting Up the Directory Server for Evaluation Purposes” on page 21.

2. On the Topology screen, click "This server will be part of a replication topology" and then click **Next** to continue. Use the default replication port 8989 for this first replicated directory server.



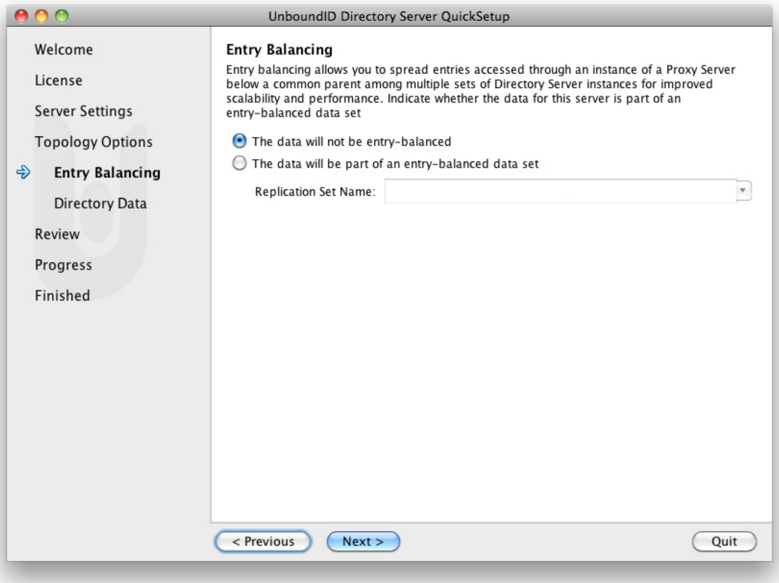
Topology Options
Choose the Data Replication Options.

( ) This will be a standalone server
(●) This server will be part of a replication topology

Replication Port: 8989 [ ] Configure as Secure

[ ] There is already a server in the topology
Host Name:
Port: 389 [ ] This is a Secure Port
Admin User: cn=Directory Manager
Admin Password:

3. Next, if you plan to use this directory server in an entry-balancing deployment with a directory proxy server, then you must specify the name of the replication set that will be replicated to other replicas. *Entry-balancing* allows you to spread entries among multiple sets of directory server instances, which are accessed through a Proxy Server instance below a common parent DN. On the Entry Balancing dialog, click if the data will be part of any entry-balancing dataset, and then select or enter a name for the replication set. For more information on entry-balancing deployments, see the *UnboundID Directory Proxy Server Administration Guid*e.



Entry Balancing
Entry balancing allows you to spread entries accessed through an instance of a Proxy Server below a common parent among multiple sets of Directory Server instances for improved scalability and performance. Indicate whether the data for this server is part of an entry-balanced data set

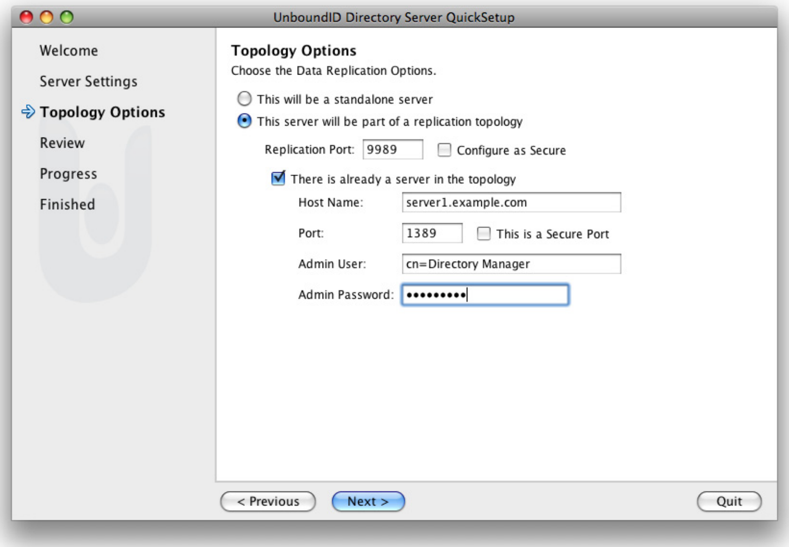
(●) The data will not be entry-balanced
( ) The data will be part of an entry-balanced data set

Replication Set Name:

4. Complete the installation. On the Directory Data dialog, generate and import 2000 sample entries. Then, on the Review dialog, click **Finish**. Check that the **Start Server when Configuration has Completed** option has been selected to automatically start the server after it has been configured. This first server must be online to set up the other servers for replication.

5. To install the second server, repeat steps 1–5 in "Setting Up the Directory Server for Evaluation Purposes" on page 21.

6. On the Topology dialog for the second server, type the Replication Port number, host name, LDAP listener port, bind DN (or Admin userID), and bind password (or Admin user password) for the first replica, `server1.example.com`. Click **Next**. If the first server was configured to use SSL to protect communication on the specified port, click **This is a Secure Port**.



Topology Options
Choose the Data Replication Options.

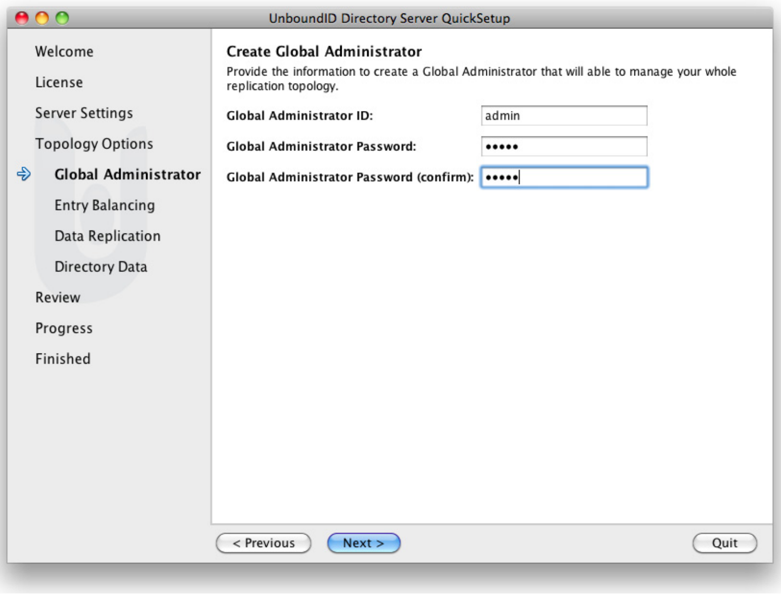
( ) This will be a standalone server
(●) This server will be part of a replication topology

Replication Port: 9989 [ ] Configure as Secure

[✓] There is already a server in the topology

Host Name: server1.example.com
Port: 1389 [ ] This is a Secure Port
Admin User: cn=Directory Manager
Admin Password: •••••••••

7. On the Global Administrator dialog, type a global administrator ID or accept the default, and then type the password. The Global Administrator is responsible for managing replication server groups.



Create Global Administrator

Provide the information to create a Global Administrator that will able to manage your whole replication topology.

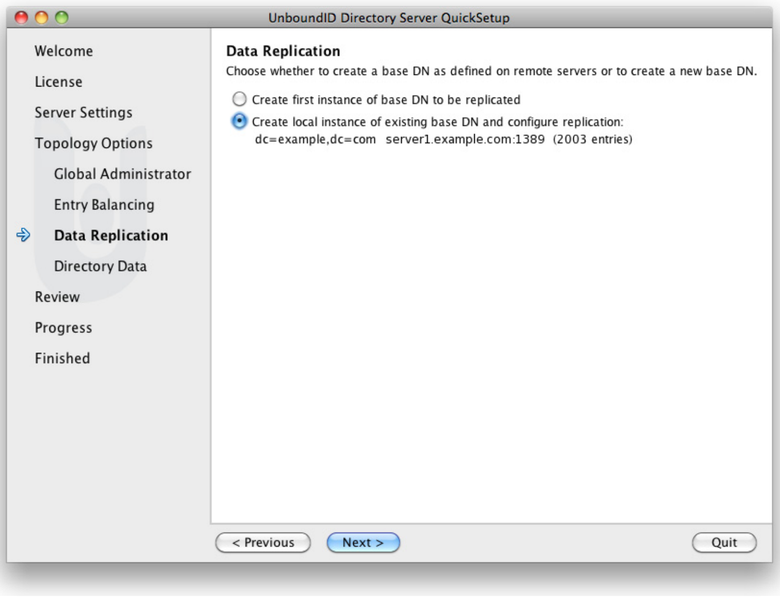
Global Administrator ID: admin

Global Administrator Password: •••••

Global Administrator Password (confirm): •••••

8. On the Entry Balancing dialog, click if the data will be part of any entry-balancing dataset, and then select or enter a name for the replication set.

9. On the Data Replication dialog, click the base DN that was configured on the first server.



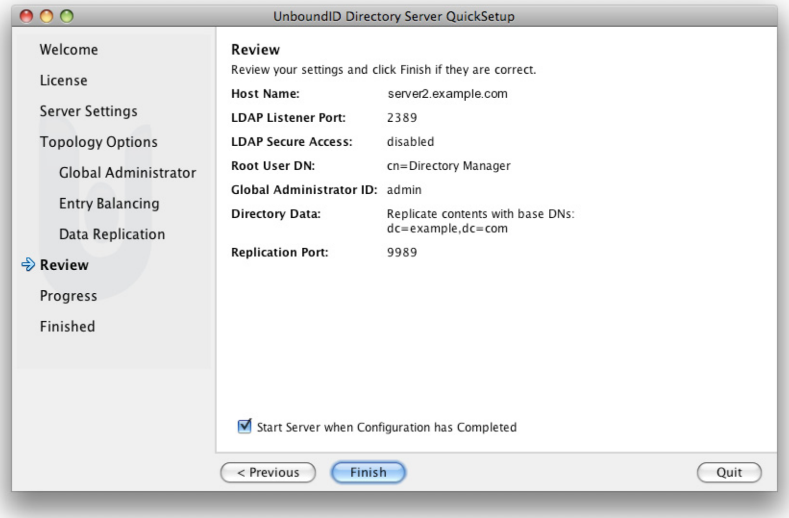
Data Replication

Choose whether to create a base DN as defined on remote servers or to create a new base DN.

○ Create first instance of base DN to be replicated

[✓ Create local instance of existing base DN and configure replication:]
dc=example,dc=com server1.example.com:1389 (2003 entries)

10. Review the installation on the second server and click **Finish**.



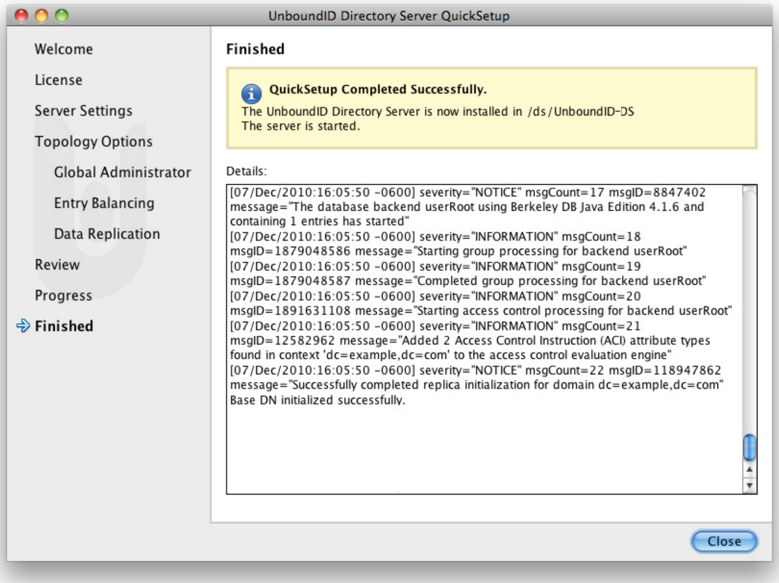
Review

Review your settings and click Finish if they are correct.

Host Name:	server2.example.com
LDAP Listener Port:	2389
LDAP Secure Access:	disabled
Root User DN:	cn=Directory Manager
Global Administrator ID:	admin
Directory Data:	Replicate contents with base DNs: dc=example,dc=com
Replication Port:	9989

☑ Start Server when Configuration has Completed

11. Click **Close** to exit the QuickSetup tool.



Finished

QuickSetup Completed Successfully.

The UnboundID Directory Server is now installed in /ds/UnboundID-DS
The server is started.

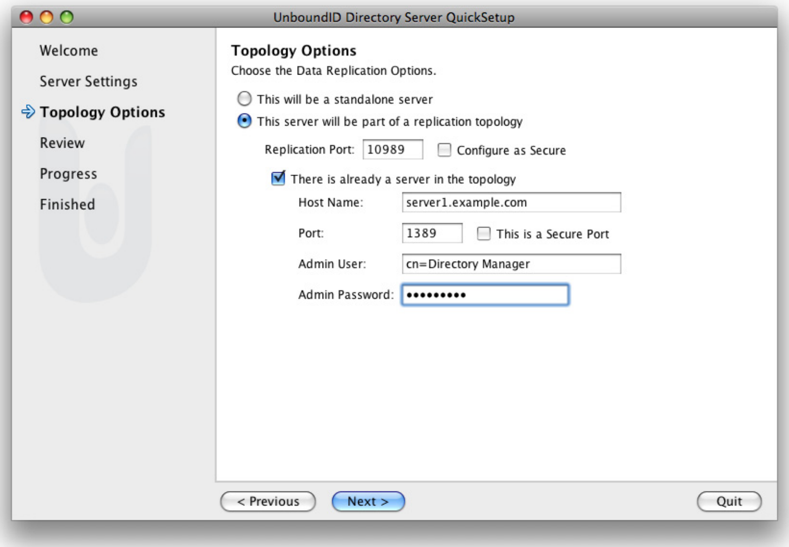
Details:

```
[07/Dec/2010:16:05:50 -0600] severity="NOTICE" msgCount=17 msgID=8847402
message="The database backend userRoot using Berkeley DB Java Edition 4.1.6 and
containing 1 entries has started"
[07/Dec/2010:16:05:50 -0600] severity="INFORMATION" msgCount=18
msgID=1879048586 message="Starting group processing for backend userRoot"
[07/Dec/2010:16:05:50 -0600] severity="INFORMATION" msgCount=19
msgID=1879048587 message="Completed group processing for backend userRoot"
[07/Dec/2010:16:05:50 -0600] severity="INFORMATION" msgCount=20
msgID=1891631108 message="Starting access control processing for backend userRoot"
[07/Dec/2010:16:05:50 -0600] severity="INFORMATION" msgCount=21
msgID=12582962 message="Added 2 Access Control Instruction (ACI) attribute types
found in context 'dc=example,dc=com' to the access control evaluation engine"
[07/Dec/2010:16:05:50 -0600] severity="NOTICE" msgCount=22 msgID=118947862
message="Successfully completed replica initialization for domain dc=example,dc=com"
Base DN initialized successfully.
```

12. For the third directory server instance, repeat steps 1–5 in “Setting Up the Directory Server for Evaluation Purposes” on page 21.

13. On the Topology dialog for the third server, type the replication port, host name, port number, bind DN (or administrative user ID), and bind password (or administrative user password) for the first replica. A *replica* is defined as the replication domain at one particular directory server instance. A *replication domain* is the replicated directory data at and below a specified base DN. Updates to entries at and below the base DN will be replicated.

14. Click **Next**. If the first server was configured for a secure port, click **This is a Secure Port**.



Topology Options
Choose the Data Replication Options.

- This will be a standalone server
- [✓ This server will be part of a replication topology]

Replication Port: 10989 Configure as Secure

[✓ There is already a server in the topology]

Host Name: server1.example.com

Port: 1389 This is a Secure Port

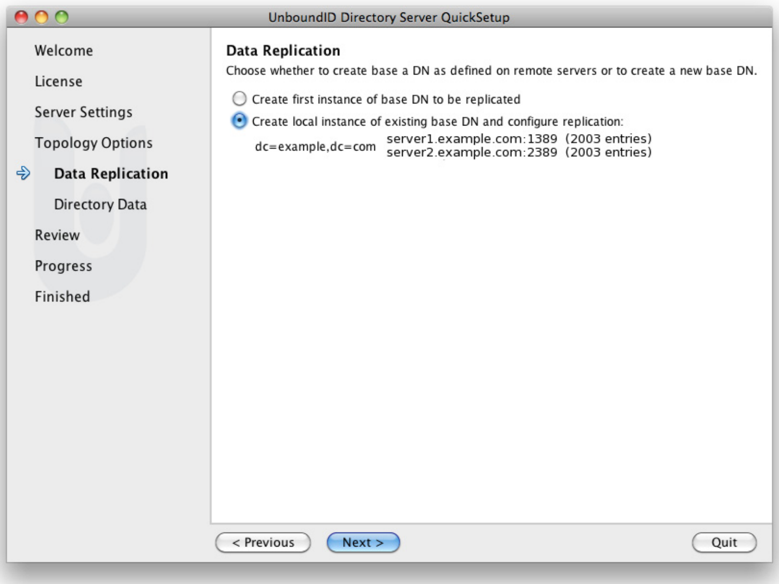
Admin User: cn=Directory Manager

Admin Password: •••••••••

15. On the Global Administrator dialog, type a global administrator ID or accept the default, and then type the password.

16. On the Entry Balancing dialog, click if the data will be part of any entry-balancing dataset, and then select or enter a name for the replication set.

17. On the Data Replication dialog, click the base DN that was configured on the first server. Notice that the other two servers are listed.

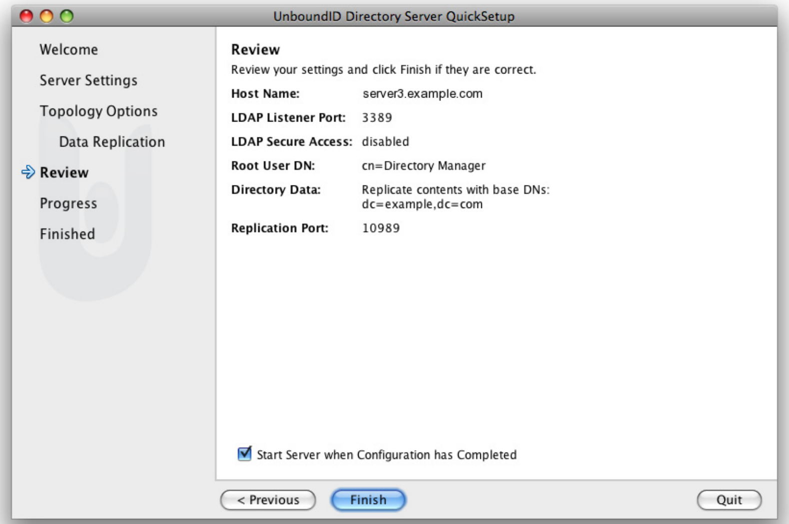


Data Replication
Choose whether to create base a DN as defined on remote servers or to create a new base DN.

- Create first instance of base DN to be replicated
- [✓ Create local instance of existing base DN and configure replication:]

dc=example,dc=com server1.example.com:1389 (2003 entries)
server2.example.com:2389 (2003 entries)

18. Review the installation for the third server and click **Finish**.



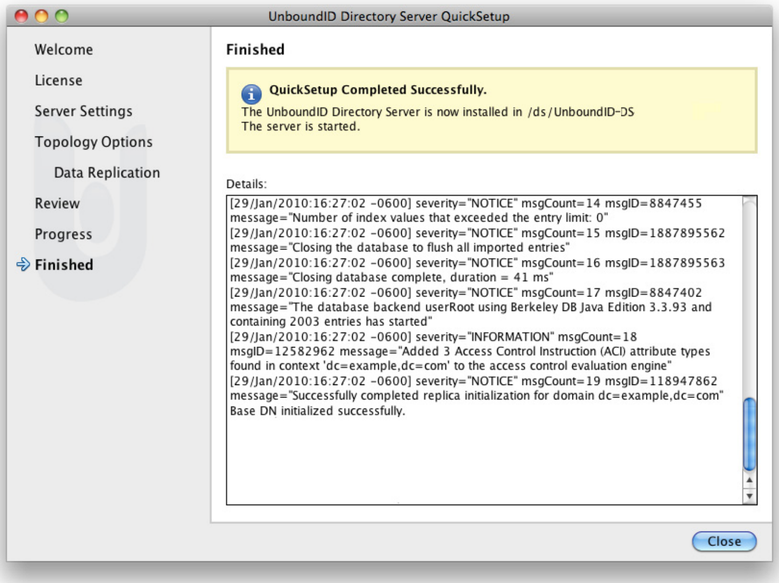
Review

Review your settings and click Finish if they are correct.

Host Name:	server3.example.com
LDAP Listener Port:	3389
LDAP Secure Access:	disabled
Root User DN:	cn=Directory Manager
Directory Data:	Replicate contents with base DNs: dc=example,dc=com
Replication Port:	10989

☑ Start Server when Configuration has Completed

19. Click **Close** to Finish.



Finished

QuickSetup Completed Successfully.

The UnboundID Directory Server is now installed in /ds/UnboundID-DS
The server is started.

Details:

```
[29/Jan/2010:16:27:02 -0600] severity="NOTICE" msgCount=14 msgID=8847455
message="Number of index values that exceeded the entry limit: 0"
[29/Jan/2010:16:27:02 -0600] severity="NOTICE" msgCount=15 msgID=1887895562
message="Closing the database to flush all imported entries"
[29/Jan/2010:16:27:02 -0600] severity="NOTICE" msgCount=16 msgID=1887895563
message="Closing database complete, duration = 41 ms"
[29/Jan/2010:16:27:02 -0600] severity="NOTICE" msgCount=17 msgID=8847402
message="The database backend userRoot using Berkeley DB Java Edition 3.3.93 and
containing 2003 entries has started"
[29/Jan/2010:16:27:02 -0600] severity="INFORMATION" msgCount=18
msgID=12582962 message="Added 3 Access Control Instruction (ACI) attribute types
found in context 'dc=example,dc=com' to the access control evaluation engine"
[29/Jan/2010:16:27:02 -0600] severity="NOTICE" msgCount=19 msgID=118947862
message="Successfully completed replica initialization for domain dc=example,dc=com"
Base DN initialized successfully.
```

You have successfully configured a multi-master replication topology with three directory servers. You can get a status of your replication topology by running the `dsreplication status` command-line tool. See “Command Line Interface” on page 384 for information on the `status` option.

Before You Begin

After you have unzipped the Directory Server ZIP file, you may want to carry out the following functions depending on your deployment requirements:

- **Custom Schema Elements**. If your deployment uses custom schema elements in a custom schema file (for example, `98-schema.ldif`), you may do one of the following:
  - Copy your custom schema file to the `config/schema` directory *before* running setup.
  - Copy your custom schema file to the `config/schema` directory *after* setup and restart the server. If replication is enabled, the restart will result in the schema replicating to other servers in the replication topology.
  - Use the Schema Editor *after* setup. If replication is enabled, schema definitions added through the Schema Editor will replicate to all servers in the replication topology without the need for a server restart.
- **Certificates**. If you are setting up a new machine instance, copy your keystore and truststore files to the `<server-root>/config` directory *prior* to running setup. The keystore and truststore passwords can be placed, in clear text, in corresponding `keystore.pin` and `truststore.pin` files in `<server-root>/config`.
- **Validate ACIs**. Many directory servers allow for less restrictive application of its access control instructions (ACIs), so that they accept invalid ACIs. For example, if an Oracle Directory Server Enterprise Edition (DSEE) (formerly Sun Directory Server Enterprise Edition) server encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the UnboundID Directory Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to an UnboundID Directory Server. If you are migrating from an Oracle DSEE deployment to an UnboundID Directory Server, the UnboundID Directory Server provides a `validate-acis` tool in the `bin` directory (UNIX or Linux systems) or `bat` directory (Windows systems) that identifies any ACI syntax problems before migrating data. For more information, see "Validating ACIs Before Migrating Data" on page 301.

Setting Up the Directory Server in Interactive Mode

The `setup` tool also provides an interactive text-based command-line interface to set up a directory server instance.

To Set Up the Directory Server in Interactive Mode

1. Unzip the distribution ZIP file, review “Before You Begin” on page 34, and then go to the installation directory if you are not already there. Use the **`setup`** command.

   ```
   $ ./setup --cli
   ```

   Note If your **`JAVA_HOME`** environment variable is set to an older version of Java, you must explicitly specify the path to the Java 6 JDK installation during setup. You can either set the **`JAVA_HOME`** environment variable with the Java 6 JDK path or execute the **`setup`** command in a modified Java environment using the **`env`** command:

   ```
   $ env  JAVA_HOME=/ds/java ./setup --cli
   ```

2. Read the UnboundID End-User License Agreement. If you agree to its terms, type **`yes`** to continue.

3. Type the root user DN, or press Enter or Return to accept the default (**`cn=Directory Manager`**), and then type and confirm the root user password.

4. Type the LDAP listener port number for your server, or press Enter or Return to accept the default port **`1389`**. If you logged in as a root user, the default port will be 389.

5. Type the base DN for the directory data, or press Enter or Return to accept the default base DN of **`dc=example,dc=com`**.

6. On the "Options for populating the database" menu, enter the option to generate and import sample data. Type the desired number of entries, or press Enter or Return to accept the default number (2000). This option is used for quick evaluation of the Directory Server. See “Initializing the Directory Server” on page 42 if you want to use other options to initialize the server.

7. The next two prompts asks if you want to use SSL, StartTLS or both. Type **`yes`** to enable one or both. Enabling SSL sets up a separate connection handler (i.e., LDAPS Connection Handler) to allow SSL over its client connections. Enabling StartTLS sets up the LDAP Connection Handler to allow StartTLS over its client connections. If you do not need a secure connection, press Enter or Return to accept the default (no) to use a standard LDAP connection.

8. If you typed **`yes`** for SSL or StartTLS, enter the keystore path or PIN options. If you use the Java or the PKCS#12 keystore, enter the keystore path, and the keystore PIN. If you use the PKCS#11 token, enter only the keystore PIN.

9. Next, type "yes" if you would like the Directory Server to listen for client connections on specific host names or IP addresses, and then enter the specific host name or IP address. You can enter as many host names or IP addresses as you would like. If you prefer the default (the server listens on all network interfaces), then press Enter or Return to accept the default (no).

10. Next, type "yes" if you would like to tune the amount of memory that will be consumed by the Directory Server and its tools. This option should only be selected if the Directory Server is the primary application on the machine, and no other processes consume a significant amount of memory.

11. If you selected optimized memory tuning, enter the maximum amount of memory (heap) to be allocated to the Directory Server. Use "m" for megabytes (for example, 256m), "g" for gigabytes (for example, 16g). The displayed value will be based on the available resources of your system. Otherwise, press Enter or Return to accept the default.

12. Next, type **`yes`** if you want to prime or preload the database cache at startup prior to accepting client connections. Note that priming the cache can increase the startup time for the Directory Server but provides optimum performance once startup has completed. This option is best used if you have strict throughput or response time performance requirements or if you plan to have other replicas in a replication topology that can accept traffic while this Directory Server instance is starting. Priming the cache also helps determine the recommended JVM **`CMSInitiatingOccupancyFraction`** option when a Java garbage collection pause occurs. See “JVM Garbage Collection Using CMS” on page 81.

13. Type **`yes`**, or press Enter or Return to accept the default (yes) to start the Directory Server after the configuration has completed. If you plan to configure additional settings or import data, you can type **`no`** to keep the server in shutdown mode.

14. In the "Setup Summary," confirm the configuration. Press Enter or Return to accept the default (set up with the parameters given), enter the option to repeat the installation process, or enter the option to cancel the setup completely.

Setting Up the Directory Server in Non-Interactive Mode

You can run the **`setup`** command in non-interactive mode to automate the installation process using a script or to run the command directly from the command line. Non-interactive mode is useful when setting up production or QA servers with specific configuration requirements.

The non-interactive command-line mode requires that all mandatory options be present for each command call. If there are missing or incorrect arguments, the **`setup`** tool fails and aborts the process. You must also use a **`--no-prompt`** option to suppress interactive output, except for errors, when running in non-interactive mode. Additionally, you must also use the **`--acceptLicense`** option and specify the port using the **`--ldapPort`** or **`--ldapsPort`** option. If neither option is specified, an error message is displayed. To view the license, run **`bin/review-license`** command.

To automatically tune the JVM to use maximum memory, use the **`--aggressiveJVMTuning`** and **`--maxHeapSize {memory}`** options. To preload the database at startup, use the **`--primeDB`** option.

To configure a deployment using a truststore, see the next section, “Installing the Directory Server in Non-Interactive Mode with a Truststore” on page 37.

To Install the Server in Non-Interactive Mode

The following procedure shows how to install a Directory Server in a production or QA environment with no security enabled.

- Unzip the distribution ZIP file, review “Before You Begin” on page 34, and then use **`setup`** with the **`--cli`** and **`--no-prompt`** options for non-interactive mode from the **`<server-root>`** directory. The following command uses the default root user DN (**`cn=Directory Manager`**) with the specified **`--rootUserPassword`** option. You must include the **`--acceptLicense`** option or the setup will generate an error message.

```
$ ./setup --cli --no-prompt --rootUserPassword "password" \
  --baseDN "dc=example,dc=com" --acceptLicense --ldapPort 389
```

Installing the Directory Server in Non-Interactive Mode with a Truststore

You can set up the Directory Server using an existing truststore for secure communication. This section assumes that you have an existing keystore and truststore with trusted certificates.

To Install the Server in Non-Interactive Mode with an Existing Truststore

- Unzip the distribution ZIP file, review “Before You Begin” on page 34, and then, from the installation directory, use **`setup`** with the **`--cli`** and **`--no-prompt`** options for non-interactive mode.

  The following example enables security using both SSL and StartTLS. It also specifies a JKS keystore and truststore that define the server certificate and trusted CA. The **`userRoot`** database contents will remain empty and the base DN entry will not be created. Make sure that **`truststore.jks`** file permission is writable by the owner. Also, the pin files (**`keystore.pin`** and **`truststore.pin`**) should be writable when invoking setup but can be changed to read-only afterwards.

```
$ ./setup --cli --no-prompt --rootUserPassword "password" \
  --baseDN "dc=example,dc=com" --ldapPort 389 --enableStartTLS \
  --ldapsPort 636 --useJavaKeystore config/keystore.jks \
  --keyStorePasswordFile config/keystore.pin \
  --certNickName server-cert --useJavaTrustStore config/truststore.jks \
  --trustStorePasswordFile config/truststore.pin --acceptLicense
```

Note	The password to the private key with the keystore is expected to be the same as the password to the keystore. If this is not the case, the private key password can be defined with the Directory Management Console or the **`dsconfig`** tool by editing the Trust Manager Provider standard configuration object.

Running the Status Tool

The Directory Server provides a **`status`** tool that outputs the current state of the server as well as other information, such as server version, JE Environment statistics, Operation Processing Times, Work Queue, and Administrative Alerts. The **`status`** tool is located in the bin directory (UNIX, Linux) or the **`bat`** directory (Windows).

To Run the Status Tool

Run the **`status`** command on the command line. The following command displays the current Directory Server status and limits the number of viewable alerts in the last 48 hours. It provides the current state of each connection handler, data sources, JE environment statistics, processing times by operation type and current state of the work queue.

```
$ bin/status --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret

          --- Server Status ---
Server Run Status:    Started 28/Mar/2012:10:47:17.000 -0500
Operational Status:   Available
Open Connections:     13
Max Connections:      13
Total Connections:    50

          --- Server Details ---
Host Name:            server1.example.com
Administrative Users: cn=Directory Manager
Installation Path:    ds-server-1
Server Version:       UnboundID Directory Server 3.2.3.0
Java Version:         1.6.0_31

          --- Connection Handlers ---
Address:Port : Protocol : State
-------------:----------:---------
0.0.0.0:1389 : LDAP     : Enabled
0.0.0.0:1689 : JMX      : Disabled
0.0.0.0:636  : LDAPS    : Disabled

          --- Data Sources ---
Base DN:     dc=example,dc=com
Backend ID:  userRoot
Entries:     1003
Replication: Enable
Replication Backlog: 0
Age of Oldest Backlog Change: <not available>

          --- JE Environment ---
ID                 : Cache Full :   Cache   : On-Disk  : Alert
-------------------:------------:----------:----------:------
replicationChanges : 6 %        : 328.8 kb : 30.4 kb  : None
userRoot           : 9 %        :  6.2 mb  : 146.6 mb : None

          --- Operation Processing Time ---
Op Type   : Total Ops : Avg Resp Time (ms)
----------:-----------:-------------------
Add       : 0         : 0.0
Bind      : 0         : 0.0
Compare   : 0         : 0.0
Delete    : 0         : 0.0
Modify    : 2788567   : 0.921
Modify DN : 0         : 0.0
Search    : 2267266   : 0.242
All       : 5055833   : 0.616

          --- Work Queue ---
           : Recent : Average : Maximum
-----------:--------:---------:--------
Queue Size : 4      : 0       : 10
% Busy     : 26     : 5       : 100
```

```
          --- Administrative Alerts ---
Severity : Time                       : Message
---------:----------------------------:--------------------------------------------
Info     : 28/Mar/2012 10:47:17 -0500 : The Directory Server has started successfully
Info     : 28/Mar/2012 10:47:14 -0500 : The Directory Server is starting
Info     : 28/Mar/2012 10:44:22 -0500 : The Directory Server has started successfully
Info     : 28/Mar/2012 10:44:18 -0500 : The Directory Server is starting

Shown are alerts of type [Info,Warning,Error,Fatal] from the past 48 hours
Use the --maxAlerts and/or --severity options to filter this list
```

Where To Go From Here

After you have set up your Directory Server instance, you can configure any specific server settings, import your user database, or run initial performance tests to optimize your server's throughput.

- **Apply Server Configurations**. Apply your server configuration changes individually or using a **`dsconfig`** batch file. The batch file defines the Directory Server configuration tool, **`dsconfig`**, commands necessary to configure your server instance. For more information on using batch files, see "Using dsconfig in Batch Mode" on page 94.

  If you are migrating from a Sun Java System 5.x, 6.x, 7.x directory server, you can use the **`bin/migrate-sun-ds-config`** command to migrate your configuration settings to this newly installed server instance.

- **Import Data**. Import your user data using the **`import-ldif`** tool. The import serves as an initial test of the schema settings.

```
$ bin/import-ldif --backendID userRoot --ldifFile ../user-data.ldif
```

- **Run Performance Tests**. The Directory Server provides two tools for functional performance testing using in-house LDAP clients that accesses the server directly: **`searchrate`** (tests search performance) and **`modrate`** (tests modification performance):

```
$ bin/searchrate --hostname server1.example.com --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --baseDN "dc=example,dc=com" --scope sub \
  --filter "(uid=user.[0-1999])" --attribute givenName --attribute sn \
  --attribute mail --numThreads 10

$ bin/modrate --hostname server1.example.com --port 389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --entryDN "uid=user.[0-1999],ou=People,dc=example,dc=com" \
  --attribute description --valueLength 12 --numThreads 10
```

Note	To simulate production loads, you can use the SLAMD tool (**`www.slamd.com`**), which was written by our engineers.

- **Add the Server to An Existing Replication Topology**. If you are adding the directory server to an existing replication topology, follow the instructions presented in the section, "Adding a Single Server to an Existing Topology" on page 392.

Configuring Base DNs

The Directory Server uses the Oracle® Berkeley DB Java Edition (JE) software as its primary database *backend* repository. The JE software is an open-source, transactional, high-performance key-value database that comes bundled with the ZIP distribution and is available from your provider. Each backend instance uses a JE environment, which supports multiple unique base DN suffixes. When a client sends a search request, the Directory Server compares the DN specified in the request with the base DN in the backend. If the requested DN matches the one in the backend, the request is processed.

Configuring a New Base DN to an Existing Backend

Assuming that you have established your DIT topology and set up a base DN for **`dc=example,dc=com`** on an existing backend with user entries, you can add another base DN, **`dc=test,dc=com`** using the **`dsconfig`** tool. The procedures to do so are presented below.

To Configure a New Base DN

1. From the root of the installation directory, use **`dsconfig`** with the **`--set-backend-prop`** option.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --add "base-dn:dc=test,dc=com" --hostname server1.test.com \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. Verify the base DN using the **`dbtest`** tool. The results show that the **`dc=example,dc=com`** base DN exists but does not have any data yet. See “Initializing the Directory Server” on page 42 for information on importing data.

```
$ bin/dbtest list-entry-containers --backendID userRoot

Base DN            JE Database Prefix  Entry Count
--------------------------------------------------
dc=example,dc=com  dc_example_dc_com   2003
dc=test,dc=com     dc_test_dc_com      0
```

To Configure a New Base DN in Interactive Mode

1. From the root of the installation directory, run **`bin/dsconfig`**.

2. Enter the connection parameters to authenticate to the server (**`hostname`**, **`port`**, **`bindDN`**, **`bindPassword`**).

3. On the UnboundID Directory Server configuration console menu, select Backend --> View and Edit an Existing Backend --> Local DB Backend.

4. On the Local DB Backend properties menu, type the number of corresponding to base DN.

Generating Sample Data

The UnboundID Directory Server provides LDIF templates that can be used to generate sample entries to initialize your directory. You can generate the sample data with the **`make-ldif`** utility together with template files that come bundled with the ZIP build, or you can use templates files that you create yourself. The templates create sequential entries that are convenient for testing the Directory Server with a range of dataset sizes. The Directory Server templates are located in the **`config/MakeLDIF`** folder and are labelled with the number of entries that it generates as shown in Table 3-5:

TABLE 3-5. **Available Sample Templates**

• example-10K.template	• example-5M.template
• example-100K.template	• example-10M.template
• example-500K.template	• example-50M.template
• example-1M.template	• example-100M.template

To randomize the data, the **`make-ldif`** command has a **`--randomSeed`** option that can be used to seed the random number generator. If this option is used with the same seed value, the template will always generate exactly the same LDIF file. Loading the data with the same seeding can be convenient for testing purposes when creating and loading data concurrently into multiple servers in a replicated topology or if it is desirable to be able to reproduce exactly the same LDIF file in the future.

If consistent datasets are not necessary for your application and if you want to generate the LDIF file more quickly, then omit the **`--randomSeed`** option and replace it with the **`--numThreads`** option, which generates entries in parallel based on the specified number of threads. The optimal number of threads depends on your underlying system and the template used to generate the data. Some trial and error might be necessary to find the number that provides the best performance. The **`make-ldif`** tool also provides an option to output the sample LDIF file in compressed gzip format. To use this option, use the **`--compress`** option with the previous example. You should specify the output file as **`--ldifFile /path/to/data.gz`**.

Note If possible, put very large LDIF files that will be imported (i.e., files that are larger than can be fully cached in the amount of available memory on the system) onto a separate disk from where the database resides. Doing so can help performance in cases where the imported LDIF file is large and the database cache fills, and when the database spends a lot of time reading pages from the disk.

To Generate a Sample LDIF File from the Template

Use the **`make-ldif`** command to generate sample data. The command generates 10,000 sample entries and writes them to an output file, **`data.ldif`**. The random seed generator is set to 0. If you generate another sample LDIF file using the same template and same random seed number, then **`make-ldif`** generates the same exact output file as the previous set:

```
$ bin/make-ldif --templateFile config/MakeLDIF/example-10K.template \
  --ldifFile /path/to/data.ldif --randomSeed 0
```

Note	The sample data templates generate a dataset with basic access control privileges that grants anonymous read access to anyone, grants users the ability to modify their own accounts, and grants the Admin account full privileges. The templates also include the **`uid=admin`** and **`ou=People`** entries necessary for a complete dataset. You can bypass the **`make-ldif`** command step entirely and use the **`--templateFile`** option with the **`import-ldif`** tool.

Initializing the Directory Server

In production environments, you typically set up the directory server and then import your data to the server using one of the following methods, depending on your dataset size:

- **Over LDAP**. You can add entries from an LDIF file over LDAP using the **`ldapmodify`** tool. This method is the slowest data initialization process and should only be used for very small datasets. It can typically take 10 or 100 times longer for large datasets than the other import methods. For a faster method than using the **`ldapmodify`** tool, you can use the **`parallel-update`** tool that allows you to run add, modify, delete, and modify DN operations concurrently using multiple threads.

- **Import**. You can import data using the **`import-ldif`** tool, which is faster than using the **`ldapmodify`** tool. The **`import-ldif`** tool has a large number of useful features, such as inclusion and exclusion filters that allow import of specific attributes or branches of a tree, task-based scheduling, schema validation, sample data import, and others. The utility has two modes of operation: offline and online. Offline import can be invoked using the **`import-ldif`** command without connection parameters. Online import can be run remotely, locally, or when it is inconvenient to stop the Directory Server for import. For online import, you must specify the connection parameters to the server to invoke the **`import-ldif`** tool.

- **Binary Copy**. You can run a binary copy by backing up the database from a source server and restoring it on the target server. This method is most convenient when dealing with a large number of entries and when installing the second to nth systems for a replicated topology. At least one system needs to be populated with complete data in order to obtain the database needed to initialize other systems using the binary copy method. For binary copy examples, see “Replication Configuration” on page 383.

Note	If an error occurs while initializing a backend, the Directory Server generates an administrative alert and records the error in the **`<server-root>/logs`** directory. The server will still start up even if the initialization fails but the administrative alert can be seen immediately.

To Initialize a Directory Server over LDAP Using Ldapmodify

1. Create an LDIF file that contains user entries, or locate an existing file. An LDIF file may typically contain a `changetype` attribute that indicates the type of modification to be made to the Directory Server. We assume for this example that *no* `changetype` attribute is present in the LDIF file, which means that the change records in the LDIF file will be added to the DIT by default. For information on adding entries to the directory server, see “Managing Entries” on page 147.

2. Use `ldapmodify` to initialize the Directory Server with the LDIF data. The following command does a bulk add operation using the `--defaultAdd` option and applies them to the Directory Server.

```
$ bin/ldapmodify --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --defaultAdd --stripTrailingSpaces \
  --ldifFile /path/to/data.ldif
```

Note	The `ldapmodify` and the `import-ldif` tools provide a `--stripTrailing-Spaces` option that can be used when adding entries from an LDIF file. Normally, if an attribute or DN has a trailing space, the `ldapmodify` tool will generate an error and abort the process. You can use the `--stripTrailingSpaces` option to properly parse and then add entries from an LDIF file.

To Initialize a Directory Server over LDAP Using Parallel-Update

1. Create an LDIF file that contains user entries, or locate an existing file. We assume for this example that no `changetype` attribute is present in the LDIF file, which means that the change records in the LDIF file will be added to the DIT by default. For information on adding entries to the directory server, see “Managing Entries” on page 147.

2. Use `parallel-update` to initialize the Directory Server with the LDIF data. The following commands uses 10 threads to update the DIT in parallel using a bulk add operation (`--defaultAdd`). Any entries that were rejected due to schema violations are written with a reason to the `rejects.ldif` file.

```
$ bin/parallel-update --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --defaultAdd --ldifFile /path/to/data.ldif \
  --rejectFile /path/to/rejects.ldif --numThreads 10
```

To Initialize a Directory Server Using Offline Import-ldif

1. Create the LDIF file that contains user entries, or locate an existing file. We assume for this example that no `changetype` attribute is present in the LDIF file, which means that the change records in the LDIF file will be added to the DIT by default. For information on adding entries to the directory server, see “Managing Entries” on page 147.

2. Stop the Directory Server.

```
$ bin/stop-ds
```

3. Use the offline **`import-ldif`** to import data from an LDIF file to the Directory Server. The data is imported from the **`data.ldif`** file to the **`userRoot`** backend. The **`--stripTrail-ingSpaces`** option strips trailing spaces on attributes that would otherwise result in a LDIF parsing error.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --skipTrailingSpaces
```

4. Restart the Directory Server.

```
$ bin/start-ds
```

To Initialize a Directory Server Using Online Import-ldif

The following example assumes that the data file is present on **`source.example.com`** and that the LDIF file is being loaded onto **`target.example.com`**. Any time that you specify any of the task backend options (**`host name`**, **`port`**, **`bindDN`**, **`bindPassword`**), then an online import is automatically run on that server. The Directory Server instances shown in Table 3-6 are used as an example in this section.

TABLE 3-6. Summary of Example Host Names and Ports

Directory Server	Port
source.example.com	1389
target.example.com	2389

1. Use the **`export-ldif`** tool to export data on **`source.example.com`** to an LDIF file.

```
$ bin/export-ldif --task --hostname source.example.com --port 1389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --backendID userRoot --ldifFile data.ldif
```

2. Use the online **`import-ldif`** tool remotely on **`target.example.com`** to import data from an LDIF file to the Directory Server. The data is imported from the **`data.ldif`** file to the **`userRoot`** backend. If any entry is rejected due to a schema violation, then the entry and the reason for the rejection is written to the **`rejects.ldif`** file. Skipped entries occur if an entry cannot be placed under a branch node in the DIT or if exclusion filtering is used (**`--excludeBranch`**, **`--excludeAttribute`**, or **`--excludeFilter`**). The **`--overwrite`** option indicates that any existing data in the backend should be overwritten. Finally, the **`--stripTrailingSpaces`** option strips trailing spaces on attributes that would otherwise result in a LDIF parsing error.

```
$ bin/import-ldif --task --hostname target.example.com --port 2389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --backendID userRoot --ldifFile /path/to/data.ldif \
  --rejectFile /path/to/rejects.ldif --skipFile /path/to/skipped.ldif \
  --overwrite --skipTrailingSpaces
```

To Schedule an Import Using Import-ldif

You can schedule an import to run as a Task by specifying the timestamp with the **`--start`** option, which is expressed in "YYYYMMDDhhmmss" format. If the option has a value of "0" then the task is scheduled for immediate execution.

1. Create the LDIF file that contains user entries, or locate an existing file. We assume for this example that no **`changetype`** attribute is present in the LDIF file, which means that the change records in the LDIF file will be added to the DIT by default. For information on adding entries to the Directory Server, see "Managing Entries" on page 147.

2. Use **`import-ldif`** with the **`--start`** option that will run the task at the specified date and time. If any entry is rejected due to a schema violation, then the entry and the reason for the rejection is written to **`rejects.ldif`**. Skipped entries occur if an entry cannot be placed under a branch node in the DIT or if exclusion filtering is used (**`--excludeBranch`**, **`--excludeAttribute`**, or **`--excludeFilter`**). When import is completed, the Directory Server issues a message to the specified email address. If an error occurs, the Directory Server sends an error message to the specified email address. Finally, the **`--stripTrailingSpaces`** option strips trailing spaces on attributes that would otherwise result in a LDIF parsing error.

```
$ bin/import-ldif --task --hostname server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --backendID userRoot --ldifFile /path/to/data.ldif \
  --start 20111201120100 --rejectFile /path/to/rejects.ldif \
  --skipFile /path/to/skipped.ldif --completionNotify admin@example.com \
  --errorNotify admin@example.com --stripTrailingSpaces
```

Note	The **`--task`** argument is optional but will be required in a future revision of the directory server. If a tool is invoked as a task without this **`--task`** argument, then a warning message will be displayed recommending that it be used. If the **`--task`** argument is provided but the tool was not given the appropriate set of authentication arguments to the server, then an error message will be displayed and the tool will exit with an error.

To Initialize a Directory Server with Sample Data Using Import-ldif

1. Create the LDIF file that contains user entries, or locate an existing file. We assume for this example that no **`changetype`** attribute is present in the LDIF file, which means that the change records in the LDIF file will be added to the DIT by default. For information on adding entries to the directory server, see "Managing Entries" on page 147.

2. Stop the Directory Server.

3. Use the offline **`import-ldif`** to import data from an LDIF template file to the Directory Server.

```
$ bin/import-ldif --backendID userRoot \
  --templateFile /config/MakeLDIF/example-10K.template
```

4. Restart the Directory Server.

Running the Directory Server

To start the Directory Server, run the **`bin/start-ds`** command on UNIX or Linux systems (**`bat\start-ds`** on Microsoft Windows systems). The **`start-ds`** command starts the Directory Server as a background process when no options are specified. To run the Directory Server as a foreground process, use the **`start-ds`** command with the **`--nodetach`** option.

To Start the Directory Server with start-ds

Use **`bin/start-ds`** to start the server.

```
$ bin/start-ds
```

To Start the Directory Server as a Foreground Process

1. Type **`bin/start-ds`** with the **`--nodetach`** option to launch the Directory Server as a foreground process.

   ```
   $ bin/start-ds --nodetach
   ```

2. You can stop the Directory Server by pressing **`Ctrl+C`** in the terminal window where the server is running or by running the **`stop-ds`** utility from another window.

Automatically Starting the Directory Server at Boot Time

By default, the UnboundID Directory Server does not start automatically when the system is booted. Instead, you must manually start it with the **`bin/start-ds`** command. To configure the Directory Server to start automatically when the system boots, use the **`create-rc-script`** utility to create a run control (RC) script, or create the script manually.

To Configure the Directory Server to Start at Boot

1. Create the startup script.

   ```
   $ bin/create-rc-script --outputFile UnboundID-DS.sh --userName ds
   ```

2. As a root user, move the generated **`UnboundID-DS.sh`** script into the **`/etc/init.d`** directory, and create symlinks to it from the **`/etc/rc3.d`** (starting with an “S” to ensure that the server is started) and **`/etc/rc0.d`** (starting with a “K” to ensure that the server is stopped).

   ```
   # mv UnboundID-DS.sh /etc/init.d/
   # ln -s /etc/init.d/UnboundID-DS.sh /etc/rc3.d/S50-UnboundID-DS.sh
   # ln -s /etc/init.d/UnboundID-DS.sh /etc/rc0.d/K50-UnboundID-DS.sh
   ```

Note	Some Linux implementations may not like the "-" in the scripts. If your scripts do not work, try renaming the scripts without the dashes. You can also try symlinking the S50* file into the **`/etc/rc3.d`** and/or **`/etc/rc0.d`** directory, based on whatever runlevel the server enters when it starts. Some Linux systems do not even use **`init.d`**-style startup scripts, so depending on whatever flavor of Linux you are using you might have to put the script somewhere else or use some other mechanism for having it launched at startup.

3. Log out as root, and re-assume the **`ds`** role if you are on a Solaris system.

Stopping the Directory Server

The Directory Server provides a simple shutdown script, **`bin/stop-ds`**, to stop the server. You can run it manually from the command line or within a script.

Note	If the Directory Server has been configured to use a large amount of memory, then it can take several seconds for the operating system to fully release the memory and make it available again. If you try to start the server too quickly after shutting it down, then the server can fail because the system does not yet have enough free memory. On UNIX systems, run the **`vmstat`** command and watch the values in the "free" column increase until all memory held by the Directory Server is released back to the system. You can also set a configuration option that specifies the maximum shutdown time a process may take.

To Stop the Directory Server

Use the **`stop-ds`** tool to shut down the server.

```
$ bin/stop-ds
```

To Schedule a Directory Server Shutdown

Use the **`bin/stop-ds`** tool with the **`--stopTime YYYYMMDDhhmmss`** option to schedule a server shutdown. The Directory Server schedules the shutdown and sends a notification to the **`server.out`** log file. The following example sets up a shutdown task that is scheduled to be processed on October 24, 2011 at 3:45 P.M. local time. The server uses the UTC time format if the provided timestamp includes a trailing “Z”, for example, 20111024154500Z. The command also uses the **`--stopReason`** argument that writes the reason for the shut down to the logs.

```
$ bin/stop-ds --stopTime 20111024154500Z --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --stopReason "Scheduled offline maintenance"
```

Restarting the Directory Server

You can restart the Directory Server using the **`stop-ds`** command with the **`--restart`** or **`-R`** option. Running the command is equivalent to shutting down the server, exiting the JVM session, and then starting up again, which requires a re-priming of the JVM cache. To avoid destroying and re-creating the JVM, use an *in-core restart*, which can be issued over LDAP. The in-core restart will keep the same Java process and avoid any changes to the JVM options.

To Restart the Directory Server

Go to the installation directory. Using an in-core restart (via the loopback interface), run the **`bin/stop-ds`** command with the **`-R`** or **`--restart`** options.

```
$ bin/stop-ds --restart --hostname 127.0.0.1 --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret
```

Uninstalling the Directory Server

The Directory Server provides an **`uninstall`** command-line utility for quick and easy removal of the code base. You can uninstall the Directory Server using one of the following modes:

- **Graphical User Interface (GUI) Mode**. GUI mode is the default uninstall option. The GUI provides an easy interface for removing a Directory Server installation.
- **Interactive Command-Line Mode**. Interactive command-line mode is a text-based interface that prompts the user for input. You can start the command using the **`bin/uninstall`** command with the **`--cli`** option. The utility prompts you for input if more data is required.
- **Non-Interactive Command-Line Mode**. Non-interactive mode suppresses progress information from being written to standard output during processing, except for fatal errors. This mode is convenient for scripting and is invoked using the **`bin/uninstall`** command with the **`--no-prompt`** option.

Note

For stand-alone installations with a single Directory Server instance, you can also manually remove the Directory Server by stopping the server and recursively deleting the directory and subdirectories using the following command:

```
$ rm -rf /ds/UnboundID-DS
```

Uninstalling the Directory Server in GUI Mode

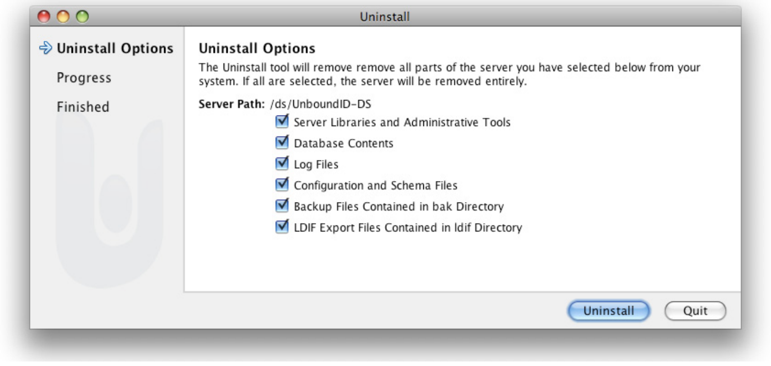
You can remove your Directory Server instance using the graphical uninstaller. If your system does not support a graphical environment, the command will run in interactive command-line mode.

To Uninstall the Directory Server in GUI Mode

1. From the UnboundID Directory Server installation directory, run the following command:

```
$ ./uninstall
```

2. Select the directories that you want to remove, and then click **Uninstall**.



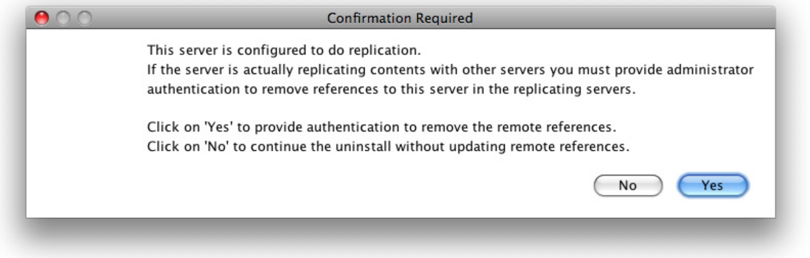
Uninstall Options

The Uninstall tool will remove remove all parts of the server you have selected below from your system. If all are selected, the server will be removed entirely.

Server Path: /ds/UnboundID-DS

- ✓ Server Libraries and Administrative Tools
- ✓ Database Contents
- ✓ Log Files
- ✓ Configuration and Schema Files
- ✓ Backup Files Contained in bak Directory
- ✓ LDIF Export Files Contained in ldif Directory

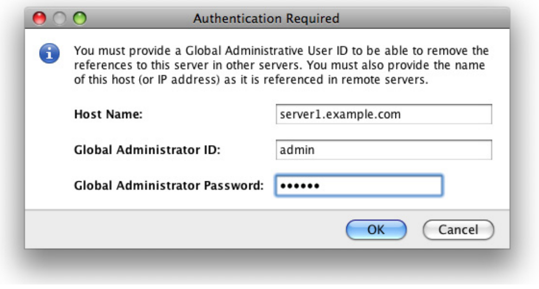
3. If the server is part of a replication topology, click `Yes` to disable replication. If the server is not part of a replication topology, continue to step 5.



This server is configured to do replication.
If the server is actually replicating contents with other servers you must provide administrator authentication to remove references to this server in the replicating servers.

Click on 'Yes' to provide authentication to remove the remote references.
Click on 'No' to continue the uninstall without updating remote references.

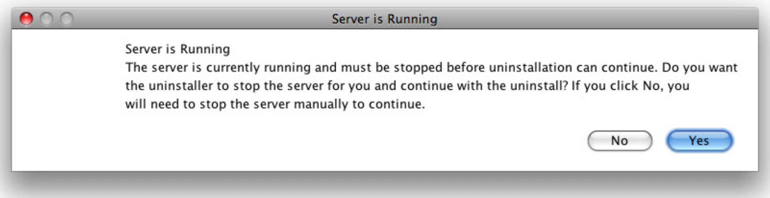
4. Type the Global Administrator password and click **OK**.The Global Administrator is responsible for managing replication server groups. For more information, see "Configuring a Global Administrator" on page 108.



You must provide a Global Administrative User ID to be able to remove the references to this server in other servers. You must also provide the name of this host (or IP address) as it is referenced in remote servers.

Host Name:	server1.example.com
Global Administrator ID:	admin
Global Administrator Password:	••••••

5. If the Directory Server is running, click **Yes** to stop the server.


Server is Running

Server is Running
The server is currently running and must be stopped before uninstallation can continue. Do you want the uninstaller to stop the server for you and continue with the uninstall? If you click No, you will need to stop the server manually to continue.

6. Click **Close** to exit the process. Manually remove any remaining files or directories if necessary.

Uninstalling the Directory Server in Interactive Command-Line Mode

Interactive mode uses a text-based, command-line interface to help you remove your instance. If `uninstall` cannot remove all of the Directory Server files, the `uninstall` tool generates a message with a list of the files and directories that must be manually deleted. The `uninstall` command must be run as either the root user or the same user (or role) that installed the Directory Server.

To Uninstall the Directory Server in Interactive Command-Line Mode

1. From the installation directory, run the `uninstall` command.

```
$ ./uninstall --cli
```

2. Select the components to be removed. If you want to remove all components, press Enter or Return to accept the default (remove all). Enter the option to specify the specific components that you want to remove.

```
Do you want to remove all components or select the components to remove?

    1)  Remove all components
    2)  Select the components to be removed

    q)  quit

Enter choice [1]:
```

3. For each type of server component, press Enter to remove them or type ’`no`’ to keep it.

```
Remove Server Libraries and Administrative Tools? (yes / no) [yes]:
Remove Database Contents? (yes / no) [yes]:
Remove Log Files? (yes / no) [yes]:
Remove Configuration and Schema Files? (yes / no) [yes]:
Remove Backup Files Contained in bak Directory? (yes / no) [yes]:
Remove LDIF Export Files Contained in ldif Directory? (yes / no) [yes]:
```

4. If the Directory Server is part of a replication topology, type yes to provide your authentication credentials (Global Administrator ID and password). If you are uninstalling a standalone server, continue to step 7.

5. Type the Global Administrator ID and password to remove the references to this server in other replicated servers. Then, type or verify the host name or IP address for the server that you are uninstalling.

6. Next, you will be asked how you want to trust the server certificate if you have set up SSL or StartTLS. For this example, press Enter to accept the default.

```
How do you want to trust the server certificate for the Directory Server on local-
host:389?

    1)  Automatically trust
    2)  Use a trust store
    3)  Manually validate

Enter choice [3]:
```

7. If your Directory Server is running, the server is shutdown before continuing the uninstall process. The uninstall processes the removal requests and completes. View the logs for any remaining files. Manually remove any remaining files or directories, if listed.

Uninstalling the Directory Server in Non-Interactive Mode

The **`uninstall`** utility provides a non-interactive method to enter the command with the **`--no-prompt`** option. Another useful argument is the **`--forceOnError`** option that continues the uninstall process when an error is encountered. If an option is incorrectly entered or if a required option is omitted and the **`--forceOnError`** option is not used, the command will fail and abort.

To Uninstall the Directory Server in Non-Interactive Mode

1. From the installation directory, run **`uninstall`** tool with the **`--remove-all`** option to remove all of the Directory Server's libraries. The **`--quiet`** option suppresses output information and is optional. The following command assumes that the directory server is stand-alone and not part of a replication topology.

```
$ ./uninstall --cli --remove-all --no-prompt --quiet --forceOnError
```

2. If any files or directories remain, manually remove them.

To Uninstall Selected Components in Non-Interactive Mode

- From the installation directory, run **`uninstall`** with the **`--backup-files`** option to remove the Directory Server's backup files. Use the **`--help`** or **`-H`** option to view the other options available to remove specific components.

```
$ ./uninstall --cli --backup-files --no-prompt --quiet --forceOnError
```

Installing the Directory Management Console

The UnboundID Directory Server provides a graphical web application tool, the UnboundID Directory Management Console. The Directory Management Console provides configuration and schema management functionality in addition to monitoring and server information.

Note	Like the **dsconfig** tool, all changes made using the Directory Management Console are recorded in **logs/config-audit.log**. In addition, anytime a configuration is made to the system, the configuration backend is automatically updated and saved as gzip-compressed files. You can access the changes in the **config/archived-configs** folder.

Installing the Console

UnboundID provides a ZIP file containing the war files necessary to install and configure the Directory Management Console. The following section assumes you are deploying the console on Apache Tomcat. The installation procedure for the console may vary depending on the version and servlet container on which you are deploying the console. For more details, consult the servlet container documentation that you are using.

To Install the Console Out of the Box

1. Download and install the latest Apache Tomcat version. For example, download **apache-tomcat-<version>.zip** from **http://tomcat.apache.org**/, and then unzip this file in a location of your choice.

2. Set the appropriate Apache Tomcat environment variables.

```
$ echo "BASEDIR=/path/to/tomcat" >> setclasspath.sh
$ echo "CATALINA_HOME=/path/to/tomcat" >> catalina.sh
```

3. Download the Directory Management Console ZIP file, **UnboundID-DS-web-console-<version>.zip** and unzip the file on your local host. You should see the following files:

```
3RD-PARTY-LICENSE.TXT
LICENSE.TXT
README
dsconsole.war
```

4. Create a **dsconsole** directory in **apache-tomcat-<version>/webapps/dsconsole**. Then, copy the **dsconsole.war** file to **apache-tomcat-<version>/webapps/dsconsole**.

```
$ mkdir apache-tomcat-<version>/webapps/dsconsole
$ cp dsconsole.war apache-tomcat-<version>/webapps/dsconsole
```

5. Go to the **`apache-tomcat-<version>/webapps/dsconsole`** directory to extract the contents of the console. The jar command is included with the JDK.

```
$ cd apache-tomcat-<version>/webapps/dsconsole
$ jar xvf dsconsole.war
```

6. With the out-of-the-box configuration, Tomcat will time out sessions after 30 minutes of inactivity forcing the user to log back in again. This can be changed on an servlet container wide basis by editing **`apache-tomcat-<version>/conf/web.xml`**, and updating the value of this configuration parameter:

```
<session-config>
   <session-timeout>120</session-timeout>
</session-config>
```

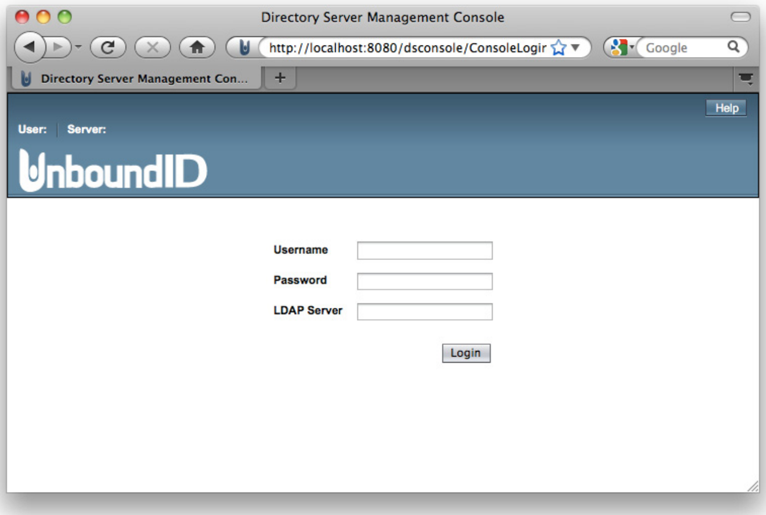
   Tomcat will expire sessions after the specified number of minutes. Changing the value to 120, for example, will extend the expiration to two hours. Changes to this setting might not take effect until the Tomcat instance is restarted, so consider changing the value before starting Tomcat for the first time.

7. Start the Directory Server if it is not already running, and then start the Directory Management Console using the **`apache-tomcat-<version>/bin/startup.sh`** script. Use **`shutdown.sh`** to stop the servlet container. (On Microsoft Windows, use **`startup.bat`** and **`shutdown.bat`**.) Note that the **`JAVA_HOME`** environment variable must be set to specify the location of the Java installation to run the server.

```
$ bin/startup.sh
```

8. Open a browser to **`http://hostname:8080/dsconsole`**. By default, Tomcat listens on port 8080 for HTTP requests. Login with your bind DN (e.g., **`uid=admin,dc=example,dc=com`**), bind DN password (see "Configuring Base DNs" on page 40 for more information). For the LDAP Server field, specify the server host name and port (e.g., **`localhost:1389`**).

If you restart the Directory Server, you must also log out of the current Directory Management Console session and then log back in to start a new console session.


Directory Server Management Console
http://localhost:8080/dsconsole/ConsoleLogin
Google
Directory Server Management Con...
Help
User:
Server:
UnboundID
Username
Password
LDAP Server
Login

Fine-tuning the Directory Management Console

The Directory Management Console uses a web.xml descriptor file for its configuration and deployment settings. Instead of specifying the host name and port on the Login page, you can configure one or more primary servers in the web.xml file as well as configure security and truststore settings for your directory server console. If you specify any servers using the web.xml file, the Login page will no longer display the LDAP Server field. It will automatically attempt to connect to the primary server(s) specified in the web.xml file in the order in which they are specified until one of the servers can authenticate the username and password. The console also uses this server to "discover" other servers in the topology, making them available for monitoring and management in the console.

To Configure One or More Primary Servers for the Console

1. Open the **`dsconsole/WEB-INF/web.xml`** file in a text editor to specify the server(s) that the console uses to authenticate. First, remove the comment tags (**`<!--`** and **`-->`**) in the **`ldap-servers`** section.

2. Next, specify the servers as **`host:port`** (e.g., **`server1.example.com:389`**) or using the LDAPS protocol to specify security information (e.g., **`ldaps://server1.example.com:389`**). If you specify more than one server, you must separate them using a space. For example, if you have two servers: one using standard LDAP communication, the other using SSL, you would see the following:

```
<context-param>
    <param-name>ldap-servers</param-name>
    <param-value>localhost:389 ldaps://svr1.example.com:389</param-value>
</context-param>
```

3. Save the file.

To Configure SSL for Primary Console Server(s)

You can configure the console so that it will communicate with all of its primary servers over SSL or StartTLS. See the previous section on how to specify one or more primary servers.

1. Open the **`dsconsole/WEB-INF/web.xml`** file in a text editor to specify the type of communication to authenticate. First, remove the comment tags (**`<!--`** and **`-->`**) in the **`security`** section.

2. Specify **`none`**, **`ssl`**, or **`starttls`** for the type of security that you are using to communicate with the directory servers.

```
<context-param>
<param-name>security</param-name>
<param-value>ssl</param-value>
</context-param>
```

3. Save the file.

To Configure a Truststore for the Web Console

For SSL and StartTLS communication, you can specify your truststore and its password (or password file) in the web.xml file. If no truststore is specified, all server certificates will be blindly trusted.

1. Open the **`dsconsole/WEB-INF/web.xml`** file in a text editor to specify the truststore. First, remove the comment tags (**`<!--`** and **`-->`**) in the **`truststore`** section.

2. Specify the path to your truststore.

```
<context-param>
<param-name>trustStore</param-name>
<param-value>/path/to/truststore</param-value>
</context-param>
```

3. Next, specify the password *or* the path to the password pin file.

```
<context-param>
<param-name>trustStorePassword</param-name>
<param-value>password</param-value>
</context-param>

<context-param>
<param-name>trustStorePasswordFile</param-name>
<param-value>/path/to/truststore/pin/file</param-value>
</context-param>
```

4. Save the file.

Upgrading the Directory Management Console

You can easily upgrade the Directory Management console by first moving the **`web.xml`** file to another location, unpacking the latest Directory Management Console distribution, and then replacing the newly deployed **`web.xml`** file with the previous build.

To Upgrade the Console

1. Shut down the console and servlet container.
2. In the current deployment of the Directory Management Console, move the **`<webapps>/dsconsole/WEB-INF/web.xml`** file to another location.
3. Download and deploy the latest version for the Directory Management Console. Follow steps 2–5 outlined in the section "To Install the Console Out of the Box" on page 52.
4. Assuming you had not renamed the **`.war`** file when you originally deployed the Directory Management Console, replace the newly deployed Directory Management Console's **`web.xml`** to **`<webapps/dsconsole/WEB-INF/web.xml`** file.
5. Start the servlet container.

Uninstalling the Directory Management Console

You can easily remove the existing Directory Management console by removing the **`webapps/dsconsole`** directory when no longer needed on your system.

To Uninstall the Console

1. Close the Directory Management Console, and shut down the servlet container. (On Microsoft Windows, use **`shutdown.bat`**).

```
$ apache-tomcat-<version>/bin/shutdown.sh
```

2. Remove the **`webapps/dsconsole`** directory.

```
$ rm -rf webapps/dsconsole
```

3. Restart the Tomcat instance if necessary. Alternatively, if no other applications are installed in the Tomcat instance, then the entire Tomcat installation can be removed by deleting the **`apache-tomcat`** directory tree.

4 Updating the Directory Server

Overview

UnboundID issues software release builds periodically with new features, enhancements, and fixes for improved server performance. Administrators can use the Directory Server's **`update`** utility to upgrade the current server code version. This chapter presents some update scenarios and their implications that you should consider when upgrading your server code.

About the Update Considerations

There are three main considerations when updating the directory server:

1. Is the server that is being upgraded part of a replication topology? If not, skip to the "Updating the Directory Server" on page 62.

2. If the server that is being upgraded is part of a replication topology, you might consider making the replication purge delay the same for all servers in a replication topology. Before starting your upgrade process, you might consider raising the purge delay for all servers in a topology, which may or may not require additional disk space for the changelog database. The *replication purge delay* represents the amount of time the administrator has to decide whether an updated server is working correctly and can be re-integrated into the topology without losing replication changes that would require the administrator to restore the data through either an LDIF import/export or restoring a backup (binary copy).

3. If the server that is being upgraded is part of a replication topology, then will the adminis-trator be updating the servers such that one or more servers in the topology will have

incompatible version of the Oracle Berkeley DB JE database at the same time? If yes, then read the section, “Backend Database Compatibility Considerations” on page 58.

Backend Database Compatibility Considerations

The UnboundID Directory Server centralizes identity management information, group data, application configurations, and user data into a network database using the Oracle® Berkeley DB Java Edition (BDB) software as its primary database backend. The BDB software provides an embedded, key-value repository for high performance transactional processing.

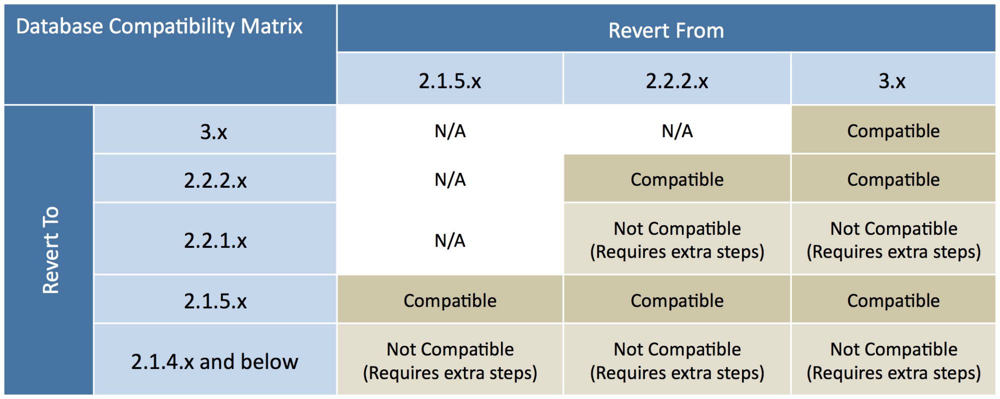
Prior releases of the UnboundID Directory Server (2.1.x, 2.1.4.x, 2.2.0, 2.2.1.x) required Oracle Berkeley DB Java Edition version 3.3.100 (BDB3). More recent Directory Server releases (2.1.5.x, 2.2.2.x, 3.x) require the latest available 4.x version of the Oracle Berkeley DB Java Edition software (BDB4). BDB4 provides improved caching policies and monitoring and other bug fixes for enhanced server performance.

While the Directory Server’s **`update`** tool can upgrade the server and the backend database to this new version of BDB4 without additional steps, you need to consider how the servers will be upgraded, so that you can maintain an easy reversion path should you decide to go back to the previous server version. We have created a Database Compatibility Matrix that shows which versions of the Directory Server are compatible with each other and which are not with regards to the backend database, specifically when doing a revert process.

For example, assume that you are running UnboundID Directory Server 2.1.2 (uses BDB3) and you plan to update your server instances to UnboundID Directory Server 2.1.5, which includes an update to BDB4. The matrix shows that reverting from UnboundID Directory Server 2.1.5 to 2.1.2 is not fully-compatible and additional steps are required during the reversion process to bring the database back into a compatible BDB3 format. If your system is already a BDB4-based Directory Server version (for example, 3.0) and you are upgrading to a newer version of the Directory Server (for example, 3.1), then the update and reversion process requires no additional processing steps.

To ensure that you can always revert your system, we recommend that you perform backups of your directory server environment prior to performing the update process. This backup may be in the form of a ZFS snapshot, binary backup, or an LDIF export. ZFS is a Solaris and Open-Solaris-based 128-bit filesystem that is based on a virtualized storage pool model. ZFS provides snapshot functionality to take read-only copies of the filesystem. Options for these methods are explained in more detail later in this chapter.

FIGURE 4-1. **Database Compatibility Matrix**



Database Compatibility Matrix		Revert From		
		2.1.5.x	2.2.2.x	3.x
Revert To	3.x	N/A	N/A	Compatible
	2.2.2.x	N/A	Compatible	Compatible
	2.2.1.x	N/A	Not Compatible (Requires extra steps)	Not Compatible (Requires extra steps)
	2.1.5.x	Compatible	Compatible	Compatible
	2.1.4.x and below	Not Compatible (Requires extra steps)	Not Compatible (Requires extra steps)	Not Compatible (Requires extra steps)

Database Compatibility Matrix Legend

- **Compatible.** The database versions are compatible between the Revert From and Revert To product versions. You can run the `revert-update` utility without any additional steps.

- **Not Compatible**. The database versions are not compatible between the Revert From and Revert To product versions. You will need to take additional steps when running the `revert-update` utility.

- **N/A**. A revert procedure between these versions is not applicable since the Revert To version is newer than the Revert From version.

About the Update Strategies

If you are updating between directory servers that have *incompatible* versions, then you should read this section. If not, you can skip to the next section.

There are two methods to update the UnboundID Directory Server depending on the level of backout or reversion protection that you need. The update strategies mitigate the update risk when a replicated server may contain data in incompatible BDB formats. The Update-All-But-One method updates directory server files on all but one machine to the newer version, so that a reversion can be run quickly if needed. The Update-All-Instances method updates the server code on all instances running on the BDB3-based directory server code. If you want to revert back to the prior version of the Directory Server, then you need to restore a previous backup or snapshot that you made earlier. In either case, we recommend only updating one server and verifying that it works as expected before updating all of your servers.

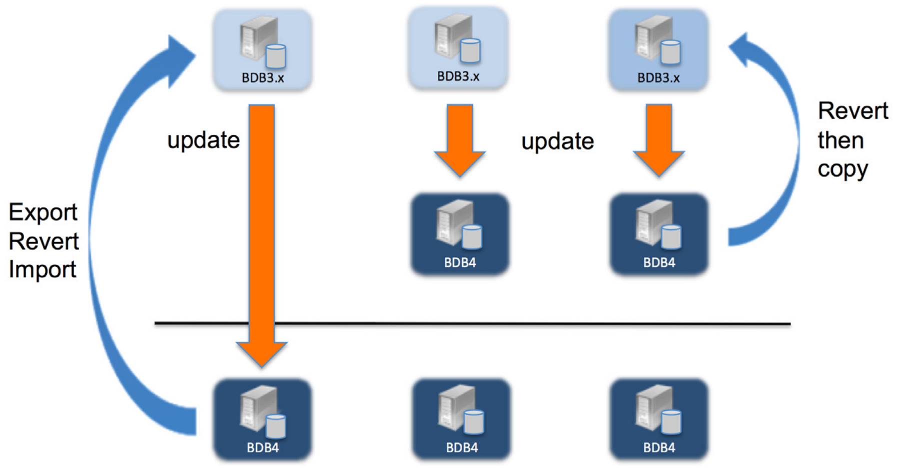
Update All But One Instance

The Update-All-But-One method is designed to maintain at least one copy of the BDB3 data in case you want to revert your BDB4 servers back to versions that require BDB3. This method updates all but one directory server, so that a reversion can be run as quickly as possible if necessary. The instance that is running on the older version can still receive replication changes from the other updated directory servers. Once the environment has been running on the newer version for a satisfactory period of time, then you can update the remaining directory server to the newer version.

This method provides the quickest way to revert back to the previous version of the directory server since there is always an up-to-date version of the data in the BDB3 (3.3.100) format. To revert back to BDB3, you just need to run the **`revert-update`** command, and then copy the database files to the machines running BDB4. Also, you need to ensure that the replication purge delay is large enough to span the whole reversion process.

Figure 4-2 shows a diagram of the process, where the top-left server remains on BDB3 while the other servers are updated to BDB4. If the updated servers require a reversion to BDB3, two servers can be reverted and then the database files can be copied from the BDB3 machine to the newly-reverted servers. After a sufficient amount of time (indicated by the line), the remaining server can be updated to BDB4. If a reversion is required, then the database files must be exported, the **`revert-update`** command run, and the database files must be re-imported.

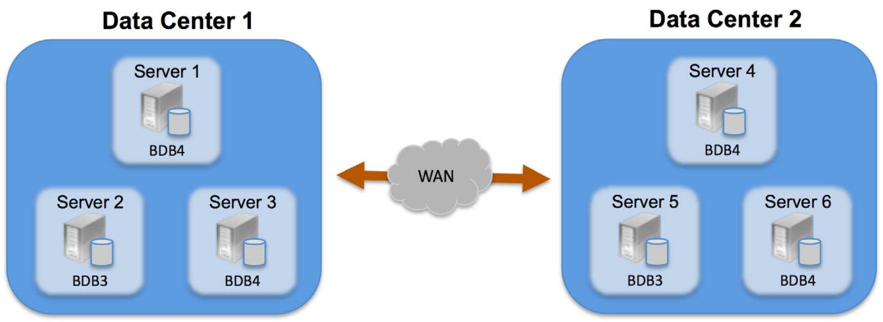
FIGURE 4-2. Oracle Berkeley Database Java Edition Update Process All But One



Important	Prior to updating the directory server, we strongly recommend that you do NOT disable replication before performing the upgrade. The **`dsreplication`** tool for Directory server version 2.1.x has known issues among mixed-version environments. As a result, disabling replication prior to updating from version 2.1.x (e.g., 2.1.4) to 2.2 (or 2.1.5 version), for example, could introduce problems.

Another advantage of the All-But-One strategy is that it avoids database copying or exporting over WAN when updating multiple data centers. When updating directory server instances across multiple data centers, we recommend that you update all but one server in each data center. Then, when you have determined that the update is working, update the remaining BDB3 machine. For example, in Figure 4-3, update servers 1 and 3 at Data Center 1 to the new version, and update servers 4 and 6 at Data Center 2. If you need to revert the directory server version back, then servers 2 and 5 are available for the process in their respective data centers. In this way, you avoid copying or exporting over WAN lines that could excessively extend the recovery time for the update/reversion process. Then, when you have determined that the updated servers are operational, you can update the remaining BDB3 machines.

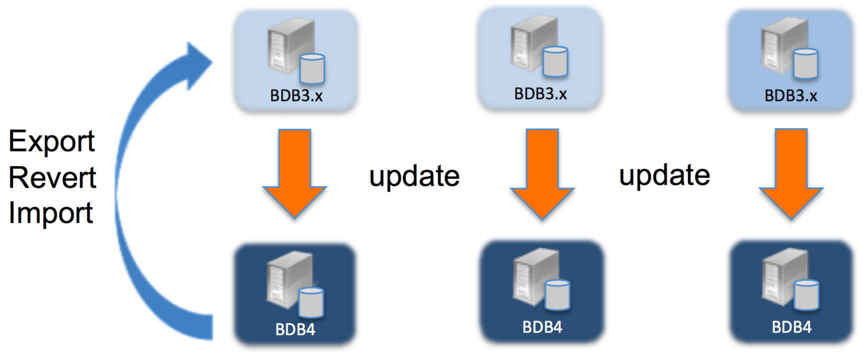
FIGURE 4-3. **Multi-Data Center Updates**



Update All Instances to the New Version

This method updates the server code on all instances running on the BDB3-based directory server code. If you want to revert back to the prior version of the directory server, then you need to restore a previous backup or snapshot that you made earlier. If a long period of time has passed (longer than the replication purge delay), then you will need to first export your data from the newly updated machine into an LDIF file, which will then be used to re-initialize the BDB3-based machine when you revert the update. This step is necessary as the BDB3 JAR files cannot read the BDB4 database files; thus, converting the data files to a standard LDIF format and re-importing them allows the BDB3 jar to read the data files from the newly-reverted instances.

FIGURE 4-4. **Oracle Berkeley Database Java Edition Update All Process**



Updating the Directory Server

The general update process involves downloading and unzipping a new version of the directory server ZIP build on the machine hosting the server that is to be updated. Then, the **`update`** utility on the *new* server is run with an option supplied to the tool that tells it where the server-to-be-updated exists on the local filesystem.

Prior to updating the directory server to a version that includes BDB4 from a version that includes BDB3, you must back up the existing databases using a ZFS snapshot or the directory server's **`backup`** utility. Then, run the **`<server-root>/update`** command to upgrade the Directory Server code, which uses the Oracle Berkeley DB Java Edition 4.x software. Note that BDB4 is not backwards compatible with BDB3, which means that a database updated by a server using BDB4 cannot be read by a server using BDB3.

Important	Prior to updating the directory server, we strongly recommend that you do NOT disable replication before performing the upgrade. The **`dsreplication`** tool for Directory server version 2.1.x has known issues among mixed-version environments. As a result, disabling replication prior to updating from version 2.1.x (e.g., 2.1.4) to 2.2 (or 2.1.5 version), for example, could introduce problems.

To Update the Directory Server

1. Before you begin the update, check the replication purge delay of the server. This step is only necessary if the Directory Server is part of a replication topology. The replication purge delay represents the window of time that the administrator can complete the update and make an evaluation that the Directory Server is performing as expected.

   ```
   $ bin/dsconfig get-replication-server-prop \
     --provider-name "Multimaster Synchronization" \
     --property "replication-purge-delay" \
     --port 389 --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
   ```

2. Shutdown all servers to be updated.

   ```
   $ bin/stop-ds
   ```

3. Back up your directory server code base using one of the following methods:

   - **Using ZFS (Solaris or OpenSolaris Only)**. For ZFS filesystems, perform a snapshot of all ZFS filesystems containing the directory server components (binaries, databases and log files). For example, assume that the ZFS filesystem is named **`rpool/ds`**. The **`-r`** option recursively takes a snapshot of the pool and its descendent filesystems.

     ```
     $ zfs snapshot -r rpool/ds@backup-bdb3
     ```

   - **Using Backup**. For non-ZFS systems, use the directory server's **`backup`** command to back up all of your local DB backends and configuration information. The following command creates a backup directory called *backups* and backs up all directories from the server root directory located at **`/ds/UnboundID-DS-current`**.

```
$ cd /ds/UnboundID-DS-current
$ bin/backup --backupDirectory backups --backUpAll
```

- **Using tar/zip**. For non-ZFS systems, create a ZIP or tar archive of the installation in case of any potential issues during the update or reversion process. This step is optional if you have already done a backup.

```
$ tar -cvf ds.tar ds/UnboundID-DS-current
$ zip -r ds.zip ds/UnboundID-DS-current
```

4. Download the new version of the Directory Server release in ZIP format, and unzip the file in a separate location in the directory server root directory. In the following example, unzip the software at `ds/UnboundID-DS-new.zip`.

```
$ unzip -f /ds/UnboundID-DS-new.zip
```

5. Verify the version number using the `status` tool's `--fullVersion` option. Compare this information to the target server's version information. Pay particular attention to the Oracle Berkeley DB JE Version information. If the major versions of the databases differ, make sure to read the "Backend Database Compatibility Considerations" on page 58.

```
$ /ds/UnboundID-DS-new/setup --version
```

6. Run the `update` tool provided with the newly unzipped build to update the existing directory server code. Make sure to specify the directory server instance that you are upgrading with the `--serverRoot` option. For example, the following command updates the code from the newly unzipped build to the current directory server instance at `/ds/UnboundID-DS-current`.

```
$ /ds/UnboundID-DS-new/update --serverRoot /ds/UnboundID-DS-current
```

7. During the update process if a file has been modified from its original version, the `update` tool prompts you as to whether or not you would like the tool to replace the existing version of the file with a new version.

```
File /UnboundID-DS/docs/admin-alerts-list.csv has been modified since it was
deployed.  How would you like to handle this file?

    1)  Replace it with the new version of the file
    2)  Keep the current version of the file
    3)  Replace the current file and any other files modified since deployment
    4)  Keep the current file and all other files modified since deployment

Enter choice: 3
```

8. View the log file to see which files were changed. The log file is located in the `<server-root>/history` directory. For example, the file will be labelled with the Directory Server version and revision numbers.

```
$ view <server-root>/history/1272307020420-3.1.1.<revision-num>/update.log
```

9. Repeat steps 1–7 on each directory server instance. When you are sure that the upgraded servers are operating without any problems, you can update the last instance. Alternatively, you can update all of the server instances at one time.

Reversion Procedures

After you have updated the Directory Server, you should make sure that the server meets your performance objectives. View the error logs, and then use the `status` command to see if the server has raised any unexpected alerts. You should also view the `dsreplication status` command to confirm that the Directory Server is accepting replication changes from other servers. If, for some reason, the Directory Server does not run as expected, you may want to consider reverting the last update performed.

Once the Directory Server has been updated, you can revert to the most recent version (one level back) using the `revert-update` tool. The `revert-update` tool accesses a file-actions log taken by the `update` tool in order to put the server's binaries and configuration back to its prior state. If you have run multiple updates, you can run the tool multiple times to revert to each prior update sequentially. Note, however, that you can only revert back one level at a time. For example, if you have run the `update` command twice since first installing the Directory Server, you can run the `revert-update` command to restore it to its previous state, and then run the `revert-update` command again to return it to its original state.

The following procedure assumes that you are restoring your Directory Server code to its previous version and that both server instances use compatible database backend revisions (for example, Oracle Berkeley DB JE version 4.x). If the Directory Server is part of a replication topology, then there may be additional administration action that must be performed following the reversion and *before* starting the server.

To Revert an Update

1. Run the `revert-update` command on the server to be restored. The `revert-update` command restores the LDIF backends (configuration, schema, and adminBackend):

```
$ revert-update
```

2. If the Directory Server is part of a replication topology, check the replication status to ensure that the server catches up with the other servers. Perform some updates on one of the other servers to ensure that replication is flowing properly.

```
$ bin/dsreplication status
```

   If the purge delay has passed, you may need to manually restore the databases if there have been replication changes to other servers in the replication topology

3. Check all logs (access, error, replication) for any issues.

Note	The `update` and `revert-update` tools never change the data in the local database backends. The tools do change the configuration data in the servers.

Reversion Procedures (Incompatible Databases)

The following reversion procedure depends on whether you backed up all but one BDB3-based directory server instance, or backed up all of the instances at one time. *The procedures are only necessary if reverting between incompatible BDB versions*.

Note	If the Directory Server's configuration changes in between the time that it is updated and the time it is reverted, the configuration changes will be lost.

Rollback a ZFS Snapshot (Solaris Only)

This reversion method is best used if you need to revert back to the original installation within a short period of time from the actual update process. For example, this method would be applicable within the scope of your Replication Purge delay.

Another consideration is how many updates have been made to the data. If the update rate is low, then restoring from ZFS or an original backup will be fast, because the original database will not have many updates to catch up on. If the rate of changes is high, it will take longer for the server to catch up once it has been reverted, because the database contains a large amount of out-of-date changes. In this case, using one of the other non-ZFS methods might be a better approach.

To Rollback a ZFS Snapshot

If you took a ZFS snapshot of the system prior to the update, you can simply rollback to that snapshot and restart the servers. You do not need to run the **`revert-update`** command with this option since the ZFS snapshot will have the original binaries and configuration in place.

1. Stop the server.

```
$ bin/stop-ds
```

2. Perform the ZFS rollback to restore the directory server code.

```
$ zfs rollback -r rpool/ds@backup-bdb3
```

3. Start the server, and then tail the error log to watch for a message indicating outstanding replication changes.

```
[07/Jul/2011:15:38:30 -0500] instanceName="x4600-01:5389" category=CORE sever-
ity=NOTICE msgID=458891 msg="The Directory Server has sent an alert notification
generated by class com.unboundid.directory.server.api.DirectoryThread (alert type
replication-backlogged, alert ID 118882379):  There are 39001 outstanding replica-
tion changes"
```

4. Check the replication status to ensure that the server catches up with the other servers. Perform some updates on one of the other servers to ensure that replication is flowing properly.

```
$ bin/dsreplication status
```

5. Check all logs (access, error, replication) for any issues.

Run the Restore Command

If ZFS is not an option or you did not use a ZFS snapshot and only ran the **`backup`** command, you can use the **`restore`** command to restore the configuration and databases to their original state after you revert the binary installation.

To Run the Restore Command

1. Run the **`revert-update`** command on the server to be restored. The **`revert-update`** command restores the LDIF backends (configuration, schema, and adminBackend). After running the revert command, DO NOT RESTART THE SERVER.

   ```
   $ revert-update
   ```

2. Run the **`restore`** command to restore specific databases (userRoot, changelog, and replicationChanges) to the previous version. For incompatible BDB databases, the actual database files must be restored since they are currently in BDB4 format.

   ```
   $ bin/restore --backupDirectory backups/userRoot
   $ bin/restore --backupDirectory backups/replicationChanges
   $ bin/restore --backupDirectory backups/changelog
   ```

3. Start the server.

   ```
   $ bin/start-ds
   ```

4. Monitor the error log on startup. You should see a message indicating the number of outstanding replication changes.

   ```
   [07/Jul/2011:16:27:16 -0500] category=CORE severity=NOTICE msgID=458891 msg="The
   Directory Server has sent an alert notification generated by class
   com.unboundid.directory.server.api.DirectoryThread (alert type replication-back-
   logged, alert ID 118882379):  There are 39001 outstanding replication changes"
   ```

5. Check the replication status to ensure that the server catches up with the other servers. Perform some updates on one of the other servers by making sure that the server has no backlog or that the backlog is shrinking.

   ```
   $ bin/dsreplication status
   ```

6. Check all logs (access, error, replication) for any issues.

Restore from a Backup of an Existing BDB3 Instance

This restore method is best used when the time since you updated the code base is longer than the Replication Purge Delay. By restoring a current backup from the BDB3 machine, you bring the reverted server into a near-synchronized state with replication, so that the time for it to catch up will be very short.

To Restore from a Backup of an Existing BDB3 Instance Using Backup/Restore

1. Run the `revert-update` command on the server to be restored. The `revert-update` command restores the LDIF backends (configuration, schema, and adminBackend).

   ```
   $ revert-update
   ```

2. On the BDB3 instance, you need to do **one** of the following backup methods:

   - **Online Remote Backup**. Run an online remote backup of all BDB databases (userRoot, changelog, replicationChanges). Store this backup in a new location outside of other backups so it is easy to identify.

     ```
     $ bin/backup --backupDirectory /ds/revertBackup --backupUpAll \
       --port 389 --bindDN "cn=Directory Manager" --bindPassword password
     ```

   - **Offline Local Backup**. Run an offline local backup for all of the BDB databases (userRoot, changelog, replicationChanges).

     ```
     $ bin/stop-ds
     $ bin/backup --backupDirectory backups --backUpAll
     $ bin/start-ds
     ```

3. Remove the current DB files on the local destination directory server (the reverted instance).

   ```
   $ rm db/userRoot/*
   $ rm changelogDB/*
   $ rm db/changelog/*
   ```

4. Transfer the backup files from the BDB3 instance to the server that you are reverting. If you used the `backup` utility (online or offline), restore the BDB database using one of the following methods:

   - **Online Remote Restore**. On the remote server that you are reverting, run a remote restore of all of the BDB databases, such as userRoot, changelog, and replicationChanges.

     ```
     $ bin/restore --backupDirectory /ds/revertBackup/userRoot \
       --hostname server1 --port 389 --bindDN "cn=Directory Manager" \
       --bindPassword password

     $ bin/restore --backupDirectory /ds/revertBackup/changelog \
       --hostname server1 --port 389 --bindDN "cn=Directory Manager" \
       --bindPassword password

     $ bin/restore --backupDirectory /ds/revertBackup/replicationChanges \
       --hostname server1 --port 389 --bindDN "cn=Directory Manager" \
       --bindPassword password
     ```

- **Offline Local Restore**. On the local server that you are reverting, run a local restore of all of the BDB databases, such as userRoot, changelog, and replicationChanges.

```
$ bin/restore --backupDirectory /ds/revertBackup/userRoot
$ bin/restore --backupDirectory /ds/revertBackup/changelog
$ bin/restore --backupDirectory /ds/revertBackup/replicationChanges
```

5. Start the server.

```
$ bin/start-ds
```

6. Check that the server has started properly, and that all replication changes have been replayed to this server. Since the database on this server instance was restored from a recent backup, the backlog of changes and the time to catch up with replication may be short. If the rate of change in your environment is high, then it may take longer (minutes or hours) to catch up.

```
$ bin/dsreplication status
```

To Restore from a Backup of an Existing BDB3 Instance Using ZFS

1. Run the **`revert-update`** command on the server to be restored. The **`revert-update`** command will restore the LDIF backends (configuration, schema, and adminBackend).

```
$ revert-update
```

2. On the BDB3 instance, shutdown the server, and create a ZFS snapshot of the databases. Once the ZFS snapshot is created, you can start the BDB3 server backup. If there is no replication or updates happening to the BDB3 server, you can perform the ZFS snapshot while the server is running.

```
$ zfs snapshot -r rpool/ds@backup-bdb3
```

3. Remove the current DB files on the local destination directory server (the reverted instance).

```
$ rm db/userRoot/*
$ rm changelogDB/*
$ rm db/changelog/*
```

4. Transfer the backup files from the BDB3 instance to the server that you are reverting. Do one of the following, depending if you are reverting a local machine or a remote one:

   - **Local Machine**. If you are using a ZFS snapshot, copy the database files for the above databases into their appropriate locations on the server being reverted.

   ```
   $ zfs restore rpool/ds@backup-bdb3
   ```

   - **Remote Machine**. On the source directory server, copy the files over from the BDB3 machine:

   ```
   $ scp /ds/.zfs/snapshot/ds@backup-bdb3/UnboundID-DS/db/userRoot/*  user@remote-
   system:/ds/UnboundID-DS/db/userRoot
   ```

```
$ scp /ds/.zfs/snapshot/ds@backup-bdb3/UnboundID-DS/changelogDB/*  user@remote-
system:/ds/UnboundID-DS/changelogDB

$ scp /ds/.zfs/snapshot/ds@backup-bdb3/UnboundID-DS/db/changelog/*  user@remote-
system:/ds/UnboundID-DS/db/changelog
```

5. Start the server.

```
$ bin/start-ds
```

6. Check that the server has started properly, and that all replication changes have been replayed to this server. Since the database on this server instance was restored from a recent backup, the replication backlog may be small and the time to catch up may be short. If the rate of change in your environment is high, then it may take longer (minutes to hours) to catch up.

```
$ bin/dsreplication status
```

Restore via Export/Import

In this method, you have updated all of your directory servers and no longer have any instances running BDB3 in your environment. Because the database files are not backwards compatible from BDB4 to BDB3, you need to perform an LDIF export/import when restoring these servers

For example, if your deployment has three servers, then the best approach to restoring all three servers to a BDB3 version would be to restore one of the servers using the export/import method. Then, from that server, use a database backup or ZFS snapshot to restore the other two servers. If possible, export the user databases (e.g., userRoot) on one server to LDIF prior to reverting to the previous version of the server. This export must be done while still using the BDB4 version of the BDB.JAR file, so that the database can be read.

When using the **`export-ldif`** command, you cannot export and re-import the changelog and **`replicationChanges`** databases from BDB4 to BDB3. Remove the DB files in the instance **`db/changelog`** directory and in the **`root/changelogDB`** directory if present. Any changes that occur on the other servers while this server is being restored will show up in the changelog, so that the most recent changes will be present.

To Restore via Export/Import

1. Stop the server that you want to revert and run the following commands if appropriate.

```
$ bin/export-ldif --backendID userRoot --ldifFile ldif/userRoot.ldif
```

2. Run the **`revert-update`** command on the server.

```
$ revert-update
```

3. Import the userRoot files as appropriate for your installation.

```
$ bin/import-ldif --backendID userRoot --ldifFile ldif/userRoot.ldif
```

4. Start the server.

```
$ bin/start-ds
```

5. Monitor the error log on startup. You should see a message indicating the number of outstanding replication changes.

```
[07/Jul/2011:16:27:16 -0500] category=CORE severity=NOTICE msgID=458891 msg="The
Directory Server has sent an alert notification generated by class
com.unboundid.directory.server.api.DirectoryThread (alert type replication-back-
logged, alert ID 118882379):  There are 39001 outstanding replication changes"
```

6. Check the replication status to ensure that the server catches up with the other servers. Perform some updates on one of the other servers by making sure that the server has no backlog or that the backlog is shrinking.

```
$ bin/dsreplication status
```

7. Check all logs (access, error, replication) for any issues.

8. To restore the remaining BDB4 instances, follow the procedures in “Restore from a Backup of an Existing BDB3 Instance” on page 66.

5 Tuning the Directory Server

Overview

The UnboundID Directory Server's installation process automatically determines the optimal Java Virtual Machine (JVM) settings based on calculations of the machine running setup. While the majority of the default configuration and JVM settings are suitable for most deployments, it is not uncommon in high performance environments to make slight adjustments to the directory server's JVM settings as well as performance and resource-related configuration changes with the **`dsconfig`** tool.

This chapter provides guidance for tuning the directory server and its tools for both optimum performance with regard to throughput and disk space usage. This chapter presents the following topics:

About Minimizing Disk Access

Most critical to directory server performance is minimizing disk access. Defining a JVM heap size that can contain the entire contents of the database cache in memory is essential to minimizing read operations from disk and achieving optimal performance. While directory write operations will always require an associated write to disk, an environment that sustains a high load of write operations may consider tuning the background database cleaner to minimize the size of the database.

Minimizing the size of the database not only reduces hard disk requirements but also reduces the memory requirements for the database cache. The Directory Server's ability to compact common parent DNs is an example of optimizing the database size based on known characteristics of the data. It is also important to understand that the database on-disk is comprised of transaction log files, which are only appended to. After an initial database import, the size on-disk will grow by a factor of at least 25% as inactive records accumulate within the transaction logs. Therefore, during normal operation, the on-disk size of the database transaction logs do not represent the memory needed to cache the database.

Memory Allocation and Database Cache

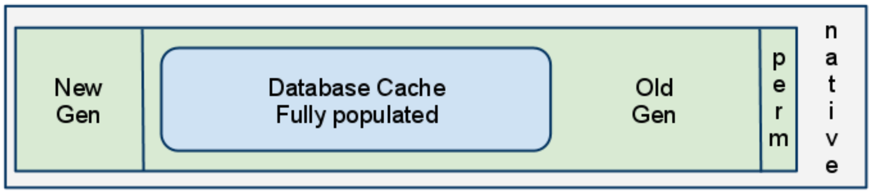
The Directory Server's optimal performance is dependent on the proper allocation of memory to the JVM heap, the number of processor cores in the system, and the correct combination of JVM options for optimized garbage collection. The Directory Server's setup process automatically assigns the JVM options and determines the memory allocation based on the total amount of memory on the system but additional tuning may be required to meet the performance objectives for your system.

Most often, Directory Server performance tuning can be accomplished by adjusting a few settings. Tuning these settings, which include both JVM and configuration options, require an understanding of the JVM heap structure as well as the expected database usage. This section describes the basic components of the Directory Server footprint and logic behind the automated tuning of the setup process.

Directory Server Process Memory

Being a Java process, the Directory Server is comprised mostly of a JVM heap and a marginal amount of memory allocated by the JVM's execution of native code. While we frequently refer to the JVM Heap as the maximum memory consumed by the directory server, the actual process size will be slightly larger than the Xmx value due to accumulation of small chunks of native code that Java requires for things, such as SSL sockets.

FIGURE 5-1. **JDK 1.6 Heap Structure**



Within the JDK 1.6 JVM Heap, the principal memory components are the New, Old and Permanent Generations. Each area serves a specific purpose. Old Generation is responsible for longer-lived objects. It is in the Old Generation that the database cache will eventually need to reside with room enough to allow expected object churn. The Old Generation size is computed from the leftover heap after defining New and Permanent Generation sizes; therefore, it is not explicitly stated in the JVM options. A typical set of Generation definitions for the JVM is as follows:

-Xmx16g -Xms16g -XX:MaxNewSize=2g -XX-NewSize=2g -XX:PermSize=64M

The **`-Xms`** value instructs the JVM to allocate an initial heap of 16G while the **`-Xmx`** represents the maximum the JVM is allowed to allocate. We recommend setting these values identically to prevent the JVM from spending time re-sizing the heap between the initial and maximum values.

The NewSize value instructs the JVM to set the default size of the New Generation to 2G while the MaxNewSize represents the maximum size of the New Generation. Again, we recommend setting these values identically. Note also that the definitions of MaxNewSize and NewSize are not required; the JVM will define a small New Generation in the event the parameters are not provided.

The PermSize value instructs the JVM to allocate an additional heap of 64M in addition to the JVM heap specified by **`-Xmx`**. Permanent Generation never undergoes garbage collection and stores class or method objects.

Note	For MaxNewSize and PermSize, no matter how large the heap (specified by the -Xmx/Xms values), the MaxNewSize/NewSize should be defined at or below 2 GB. Likewise, the PermSize is less dependent on database size and should not need to be raised above 64M.

Performance testing of the Directory Server has shown that New and Permanent Generation areas should be defined within a relatively tight threshold with our setup process, doing the best job at predicting the needed New and Permanent sizes. Considering the New and Permanent Generations as fixed points helps simplify the tuning process of Old Generation with just the modification of **`-Xmx/Xms`**.

A Method for Determining Heap and Database Cache Size

The most straightforward approach to defining the proper memory allocation of Directory Server components is to use the Directory Server setup process on hardware, which suitably represents the target production platform, especially with regard to process and memory, and assign a large heap, perhaps the most the setup process will allow. After running setup, any schema and production database settings should be defined in preparation for the database import using the **`import-ldif`** tool.

Note	At the moment after an import-ldif, the database is at its most optimized state on disk, with no inactive records. Over time, the on-disk representation of the database will grow as much as 25-50% as inactive records accumulate before being removed by the server's cleaner thread.

After the database is imported, the server should be started and a configuration change made to the backend. In this scenario, set the prime-method to "preload" on the **`userRoot`** backend configuration. Once the change is made, restart the Directory Server and watch for a successful preload message at the end of startup. If preloading did not complete, the setup process should be run again with a larger heap size. If preloading completed, the database cache utilization percentage will be of interest. The **`status`** command will display something like the following:

```
     --- JE Environment ---
ID       : Cache Full :  Cache : On-Disk : Alert
---------:------------:--------:---------:------
userRoot : 30%        : 1.1 gb : 868.6mb : None
```

Looking at the above output and knowing that the database is fully loaded into cache, the 30% utilization is comfortable cushion for future database growth. In general, it is best to leave at least 10-20% cache headroom available.

During this scenario, it was clear from the **`start-ds`** output that the database primed completely and our interpretation of the status output was sound. To see the state of the database cache with more detail, perform an **`ldapsearch`** on the backend monitor.

Automatic DB Cache Percentages

The setup process automatically tunes the percentage of the **`db-cache-percent`** property for the **`userRoot`** backend based on the maximum configured JVM heap size.

TABLE 5-1. **Percentage assigned to db-cache-percent property**

Maximum Heap Size	DB Cache Percent
Greater than or equal to 16 GB	75 * (MaxHeapSize-2GB) / MaxHeapSize
Greater than or equal to 8 GB	75 * (MaxHeapSize-1GB) / MaxHeapSize
Greater than or equal to 3 GB	Set to 50 %
Else	Set to 25 %

Automatic Memory Allocation if Memory Tuning is Enabled

If the Memory Tuning feature is enabled during setup, the setup algorithm determines the maximum JVM heap size based on the total amount of available system memory. The Directory Server also allows you to specify the maximum heap size during the setup process. You can enable Memory Tuning during the setup process by clicking the check box in the graphical installer (seen on page 21), selecting the feature during the interactive command-line mode (seen on page 34), adding the **`--aggressiveJVMTuning`** option using the **`setup`** tool in non-interactive command-line mode (seen on page 36), or regenerating the java properties file with the **`--jvmTuningParameter`** options (seen on page 80).

If Memory Tuning is selected, the server allocates the maximum JVM heap dependent on the total system memory. The following table displays the automatically allocated maximum JVM heap memory based on available system memory.

TABLE 5-2. **Allocated Max JVM Memory if Memory Tuning is Enabled**

Available System Memory	Allocated Max JVM Memory if Memory Tuning is Enabled
16 GB or more using a 64-bit JVM	The maximum JVM heap size will be set to 70% of total system memory. If the maximum JVM heap size is less than or equal to 128GB of memory (which should be the case for systems with up to 160 GB of memory), then the initial heap size will be set to equal the maximum heap size. If the maximum heap size is greater than 128GB, then the initial heap size will be set to 128GB in order to work around an apparent bug in the JVM.
6 GB–16 GB using a 64-bit JVM	total system memory - 4 GB
4 GB–6 GB using a 64-bit JVM, or over 4 GB using a 32-bit JVM	2 GB
2 GB–4 GB	512 MB
1 GB–2 GB	256 MB
Less than 1 GB	128 MB

Automatic Memory Allocation if Memory Tuning is Not Enabled

If Memory Tuning is not selected, the server allocates the maximum JVM heap to at most 256 MB for systems with one gigabyte of available system memory or more, or 64 MB for systems with less than one gigabyte of total memory. The table below displays the default maximum JVM heap size.

TABLE 5-3. **Allocated Max JVM Memory if Memory Tuning is Not Enabled**

Available System Memory	Allocated Max JVM Memory if Memory Tuning is Not Enabled
At least 1 GB	256 MB
Less than 1 GB	128 MB

Automatic Memory Allocation for the Command-Line Tools

At setup, the Directory Server automatically allocates memory to each command-line utility based upon the maximum JVM heap size. The server sets each command-line utility in the **`config/java.properties`** with **`-Xmx/Xms`** values depending on the expected memory needs of the tools. Because some tools can be invoked as a server task while the server is online, there are two definitions of the tool in the **`config/java.properties`** file: one with **`.online`** and one with **`.offline`** added to the name. The online invocations of the tools typically require minimal memory as the task is performed within the Directory Server's JVM. The offline invocations of the tools, for example, **`import-ldif.online`** and **`rebuild-index.offline`**, can require the same amount of memory that is needed by the Directory Server.

Beyond the offline tool invocations, some tools, such as **`ldap-diff`** and **`verify-index`**, may need more than the minimal memory if large databases are involved. The table below lists the tools that are expected to have more than the minimal memory needs along with the rules for defining the default heap size.

TABLE 5-4. **Default Memory Allocation to Command-Line Tools**

Command-Line Tools	Allocated Memory
start-ds import-ldif (offline) rebuild-index (offline)	MaxHeapSize
backup (offline) dbtest export-ldif (offline) ldap-diff restore (offline) scramble-ldif summarize-access-log verify-index	If Max System Memory is: Greater than or equal to 16 GB: set Heap to 3 GB Greater than or equal to 8 GB: set Heap to 1 GB Greater than or equal to 4 GB: set Heap to 512 MB Under 4 GB: set Heap to 128 MB

Database Preloading

Most influential to Directory Server performance is the ability to maintain database contents either in Directory Server JVM memory or in the Operating System's filesystem cache. The former is preferred since the latter is subject to uncontrollable movement of data in and out of the filesystem cache by other processes.

For a Directory Server deployment where enough memory is available to cache the database, it is recommended to set the priming method of the database to "preload" as opposed to the default value "none". With a properly sized database cache (See section on Memory Allocation and Database Cache), a priming method of "preload" directs the server to load the database contents into memory at server startup before accepting the first client connection. The time needed to preload the database is proportional to the database size and can be optionally avoided with the **`--skipPrime`** option to **`start-ds`**. If the priming method is "none" or the **`--skipPrime`** option is specified at startup, the database cache will slowly build as entries are accessed due to client activity.

The "preload" priming method is suitable for nearly all Directory Server deployments. In the case where the size of the database precludes storing the whole database in memory, there are a number of priming alternatives that provide methods for optimizing server performance. This type of deployment is referred to as *disk-bound* since disk access is expected more often. See the section "Tuning Disk-Bound Deployments" on page 84 for more information. The remaining priming options are applicable to these environments.

The Directory Server provides advanced options to configure how the caches get primed, what gets primed (data, internal nodes, system indexes) and where it gets primed (database cache, filesystem cache, or both) using the **`dsconfig prime-method`** property. The **`prime-method`** property is a multi-valued option. For example, it allows you to prime the filesystem cache, and then preload the internal nodes into the database cache. For more details, see the *UnboundID Directory Server Configuration Reference*.

The following is a summary of some of the priming methods:

- **Preload All Data**. Prime the contents of the backend into the database cache.
- **Preload Internal Nodes only**. Prime only internal database structure information into the database cache but do not prime any actual data.
- **Prime to Filesystem Cache**. Use sequential reads across the database files to prime the filesystem cache.
- **Prime to Filesystem Cache Non-Sequential**. Uses non-sequential reads across the database files to prime the filesystem cache. This is beneficial for ZFS filesystems (Solaris) that detect sequential reads and prevents them from being used to populate the cache.
- **System Index to Prime**. Prime specific system indexes when the backend is initialized. Note that any system index which is not present in this set or in the `System Index to Prime Internal Nodes Only` set will not be primed at all. Any index in both sets will be completely primed.
- **System Index to Prime Internal Nodes Only**. Prime only the internal nodes for specific system indexes when the backend is initialized. Note that any system index which is not present in this set or in the `System Index to Prime` set will not be primed at all. Any index in both sets will be completely primed.

Configuring Database Preloading

You can use the `dsconfig` tool to set the database priming method. If multiple prime methods are used, then the order in which they are specified in the configuration is the order in which they will be performed. The preloading option requires that you restart the Directory Server. The following procedure shows how to configure database preloading.

To Configure Database Preloading

1. Set the prime method to "preload" by removing the value of "none".

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --set prime-method:preload
```

2. Restart the Directory Server to apply the changes using `bin/stop-ds` and then, `bin/start-ds`.

To Configure Multiple Preloading Methods

1. Assuming the database is too large to hold entirely in database cache but the sum of the internal nodes does fit, set the method to Prime to Filesystem Cache Non-Sequential followed by Preload Internal Nodes.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --set prime-method:prime-to-filesystem-cache-non-sequential \
  --set prime-method:preload-internal-nodes-only
```

2. Restart the Directory Server to apply the changes using **`bin/stop-ds`**, **`bin/start-ds`**.

To Configure System Index Preloading

1. Assuming the database is too large to hold entirely in database cache but some system indexes can fit by themselves, use the **`system-index-to-prime`** property of the backend and set the **`preload-method`** to "preload". To preload specific indexes, the **`prime-all-indexes`** property must be set to false. Also, you may not use system index preloading for filesystem priming (that is, using the **`prime-to-filesystem-cache`** property). The following example turns off priming for all indexes, and sets the system index priming for all but the referral database, **`dn2uri`**:

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --set prime-method:preload \
  --set prime-all-indexes:false \
  --set system-index-to-prime:dn2id \
  --set system-index-to-prime:id2children \
  --set system-index-to-prime:id2entry \
  --set system-index-to-prime:id2subtree
```

2. Restart the Directory Server to apply the changes using **`bin/stop-ds`** and then, **`bin/start-ds`**.

The Database Cleaner

Production environments that have a high volume of write operations may require cleaner thread tuning to control on-disk database size as log files with inactive nodes wait to be cleaned and deleted. JE uses one or more cleaner threads that run in the background to compact the number of JE database (db) files. The Directory Server stores its Oracle® Berkeley DB Java Edition (JE) database files on-disk in the **`db`** directory. Each JE database log file is labelled **`nnnnnnnn.jdb`**, where **`nnnnnnnn`** is an 8-digit hexadecimal number that starts at 00000000 and is increased by 1 for each file written to disk. JE only appends data to the end of each file and does not overwrite any existing data.

The cleaner threads begin by scanning the records in each db file, starting with the file that contains the smallest number of active records. Next, the cleaner threads append any active records to the most recent database file. If a record is no longer active due to modifications or deletions, the cleaner threads leave it untouched. After the db file no longer has active records, the cleaner threads can either delete the file or rename the discarded file. Note that because of this approach to cleaning, the database size on-disk can temporarily increase when cleaning is being performed and files are waiting to be removed.

The Local DB Backend configuration object has two properties that control database cleaning: `db-cleaner-min-utilization` and `db-num-cleaner-threads`. The `db-cleaner-min-utilization` property determines, by percentage, when to begin cleaning out inactive records from the database files. By default, the property is set to 75, which indicates that database cleaning ensures that at least 75% of the total log file space is devoted to live data. Note that this property only affects the on-disk representation of the database and not the in-memory database cache—only live data is ever cached in memory.

The `db-num-cleaner-threads` property determines how many threads are configured for db cleaning. The default single cleaner thread is normally sufficient. However, environments with a high volume of write traffic may need to increase this value to ensure that database cleaning can keep up.

If the number of database files grow beyond your expected guidelines or if the Directory Server is experiencing an increased number of update requests, you can increase the number of cleaner threads using the `dsconfig` tool (select Backend -> select advanced properties -> `db-num-cleaner-threads`).

Compacting Common Parent DNs

The UnboundID Directory Server compacts entry DNs by tokenizing common parent DNs. Tokenizing the common parent DNs allows you to increase space usage efficiency when encoding entries for storage. The Directory Server automatically defines tokens for base DNs for the backend (for example, `dc=example,dc=com`). You can also define additional common base DNs that you want to tokenize. For example, use the following configuration to tokenize two branches, `ou=people,dc=example,dc=com` and `ou=customers,dc=example,dc=com`:

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --add "compact-common-parent-dn:ou=people,dc=example,dc=com" \
  --add "compact-common-parent-dn:ou=customers,dc=example,dc=com"
```

Import Thread Count

For most systems, the default setting of 16 threads is sufficient and provides good import performance. On some systems, increasing the import thread count may lead to improved import performance, while selecting a value that is too large can actually cause import performance to degrade. If minimizing LDIF import time is crucial to your deployment, you must determine the optimal number of import threads for your system, which is dependent on both the underlying system and the dataset being imported.

You can use the `dsconfig` command to set the number of import threads as follows:

```
$ bin/dsconfig set-backend-prop --backend-name userRoot --set import-thread:24
```

JVM Properties for Both Server and Command-Line Tools

The UnboundID Directory Server and tools refer to the **config/java.properties** file for JVM options that include important memory settings. The **java.properties** file sets the default Java arguments for the Directory Server and each command-line utility including the default **JAVA_HOME** path.

The **java.properties** is generated at server setup time and defines memory-related JVM settings based on the user-provided value for max heap size if aggressive memory tuning option was selected at setup. Most of the JVM options specified for both server and tools are in no need of customization after setup. The exception is the **-Xmx/Xms** options, which specify max/ initial JVM heap size. See the section on Memory Footprint and Database Cache for advice on tailoring the -Xmx/Xms values.

Other than altering the heap size of the server process (start-ds) or command-line tools, the most common change required to **java.properties** is when it is desired to update the JVM version. A single edit will apply the new JVM to all server and tool use.

Applying the Changes Using dsjavaproperties

To apply the changes to the **config/java.properties** file, edit the file manually, and then run the **bin/dsjavaproperties** utility. The **dsjavaproperties** tool uses the information contained in the **config/java.properties** file to generate a **lib/set-java-home** script (or **lib\set-java-home.bat** on Microsoft Windows systems), which is used by the Directory Server and all of its supporting tools to identity the Java environment and its JVM settings. During the process, **dsjavaproperties** calculates an MD5 digest of the contents of the **config/java.properties** file and stores the digest in the generated **set-java-home** script.

The **dsjavaproperties** utility also performs some minimal validation whenever the property references a valid Java installation by verifying that **$(java-home)/bin/java** exists and is executable.

If you make any changes to the **config/java.properties** file but forget to run **bin/dsjavaproperties**, the Directory Server compares the MD5 digest with the version stored in **set-java-home** and sends a message to standard error if the digests differ:

```
WARNING -- File /ds/UnboundID-DS/config/java.properties has been edited without running
dsjavaproperties to apply the changes
```

To Regenerate the Java Properties File

The **dsjavaproperties** command provides a **--initialize** option that allows you to regenerate the Java Properties file specifically if you set up the Directory Server using standard memory usage but opt for aggressive memory tuning after setup. Rather than reconfigure the Java Properties file by re-running setup or manually editing the **java.properties** file, you can regenerate the properties file for aggressive memory tuning. Any existing file will be renamed with a ".old" suffix.

- Run the **`dsjavaproperties`** command to regenerate the java properties file for aggressive memory tuning:

```
$ bin/dsjavaproperties --initialize --jvmTuningParameter AGGRESSIVE
```

To Update the Java Properties File

The **`dsjavaproperties`** command provides an **`--update`** option that allows you to verify that all of the server's tools have a VM configuration setting in the properties file. If there are tools missing from the file due to accidental removal, the missing tools are added to the bottom of the **`java.properties`** file.

- Run the **`dsjavaproperties`** command to update the java properties file for the VM configurations for any missing tool:

```
$ bin/dsjavaproperties --update
```

JVM Garbage Collection Using CMS

To ensure reliable server performance with Java JDK version 6, the Directory Server depends on Java's Concurrent Mark and Sweep process (CMS) for background garbage collection. There are several garbage collection options in JDK 6, with CMS being the ideal choice for consistent system availability. The CMS collector runs, for the most part, as one or more background threads within the JVM, freeing up space in JVM Heap from an area called the Old Generation. One of the criteria used by CMS in order to determine when to start background garbage collection is a parameter called **`CMSInitiatingOccupancyFraction`**. This percentage value, which applies to the Old Generation, is a recommendation for the JVM to initiate CMS when data occupancy in Old Generation reaches the threshold.

In order to understand this CMS property, it is important to know how large the Old Generation is and how much data in the Old Generation is expected to be occupied by the database cache. Ideally, the database cache takes less than 70% of the space available in the Old Generation, and the **`CMSInitiatingOccupancyFraction`** value of 80 leaves plenty of headroom to prevent the JVM from running out of space in Old Generation due to an inability for CMS to keep up. Because CMS takes processing resources away from the Directory Server, it is not recommended to set the **`CMSInitiatingOccupancyFraction`** at or below the expected database cache size, which would result in the constant running of CMS in the background. See the section on Memory Footprint and Database Cache for a description of determining Old Generation size.

When the CMS collection process cannot keep pace with memory demands in the Old Generation, the JVM will resort to pausing all application processing to allow a full garbage collection. This event, referred to as a *stop-the-world* pause, does not break existing TCP connections or alter the execution of the directory server requests. The goal in tuning CMS is to prevent the occurrence of these pauses. When one does occur, the Directory Server will generate an alert, after the pause, and record the pause time in the error log.

Because determining an ideal **`CMSInitiatingOccupancyFraction`** can be difficult, the approach we have taken is to warn if the Directory Server detects a garbage collection pause by generating a recommended value for the occupancy threshold based on the current amount of memory being consumed by the backend caches. Unfortunately, it is not possible for an administrator to determine the ideal occupancy threshold value in advance. Therefore, to warn of any impending garbage collection pauses, the Directory Server calculates a recommended value for the **`CMSInitiatingOccupancyFraction`** property and exposes it in the JVM Memory Usage monitor entry in the following attribute:

```
recommended-cms-initiating-occupancy-fraction-for-current-data-set
```

Also, when you start the server, you will see an administrative alert indicating the current state of the **`CMSInitiatingOccupancyFraction`** and its recommended value.

```
$ bin/start-ds
[20/April/2012:10:35:25 -0500] category=CORE severity=NOTICE msgID=458886 msg="UnboundID Directory
Server 3.2.3.0 (build 20120420135933Z, R6226) starting up"
... (more output) ...
[20/April/2012:10:35:53 -0500] category=UBID_EXTENSIONS severity=NOTICE msgID=1880555580 msg="Mem-
ory-intensive Directory Server components are configured to consume 71750382 bytes of memory:
['userRoot local DB backend' currently consumes 26991632 bytes and can grow to a maximum of 64323584
bytes, 'changelog cn=changelog backend' currently consumes 232204 bytes and can grow to a maximum of
2426798 bytes, 'Replication Changelog Database' currently consumes 376661 bytes and can grow to a
maximum of 5000000 bytes].  The configured value of CMSInitiatingOccupancyFraction is 36 which is
less than the minimum recommended value (43) for the server's current configuration. Having this
value too low can cause the Concurrent Mark and Sweep garbage collector to run too often, which can
cause a degradation of throughput and response time.  Consider increasing the CMSInitiatingOccupan-
cyFraction value to at least the minimum value, preferably setting it to the recommended value of 55
by editing the config/java.properties file, running dsjavaproperties, and restarting the Directory
Server.  If the server later detects that this setting actually leads to a performance degradation,
a separate warning message will be logged.  If this server has not yet been fully loaded with data,
then you can disregard this message"
[20/April/2012:10:35:53 -0500] category=CORE severity=NOTICE msgID=458887 msg="The Directory Server
has started successfully"
```

The Directory Server only makes a recommendation if all of the backends are preloaded and the **`CMSInitiatingOccupancyFraction`** JVM property is explicitly set, which is done automatically. For example, if you installed the Directory Server and specified that the database be preloaded (or "primed") at startup, then the Directory Server can make a good recommendation for the **`CMSInitiatingOccupancyFraction`** when a pause occurs. If the backend database cache is not full and has not been preloaded, then the recommended value may be an inaccurately low value.

Note

The generated value for the **`CMSInitiatingOccupancyFraction`** property could change over time with each Directory Server build, Java release, or changes in data set. If the current value is fairly close to the recommended value, then there is no need to change the property unless the server experiences a JVM pause.

If the Directory Server experiences a JVM garbage collection pause, you can retrieve the recommended value from the server, reset the **`CMSInitiatingOccupancyFraction property`**, run **`dsjavaproperties`**, and restart the server.

To Determine the CMSInitiatingOccupancyFraction Value

1. If you set the Preload Database at startup option during the installation, then skip to step 3. If you are not sure, retrieve the **`prime-method`** property for the backend as follows:

```
$ bin/dsconfig get-backend-prop --backend-name userRoot \
  --property prime-method --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

2. If the **`prime-method`** property was not configured, use **`bin/dsconfig`** to set the property to **`PRELOAD`**, and then, restart the Directory Server to preload the database cache.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
  --set prime-method:preload --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt

$ bin/stop-ds
$ bin/start-ds
```

3. At startup, you will see an administrative message if the current **`CMSInitiatingOccupancyFraction`** property is below the recommended value. You can get the recommended value from this message and change it in the **`config/java.properties`** file in step 5.

4. If you were unable to see the recommended **`CMSInitiatingOccupancyFraction`** property at startup presented in the previous step, first you must pre-tune the value of the **`CMSInitiatingOccupancyFraction`** property to ensure that all of the data is imported into the server and preloading is enabled in the backend. Next, retrieve the recommended **`CMSInitiatingOccupancyFraction`** value by issuing the following search. If the **`recommended-cms-initiating-occupancy-fraction-for-current-data-set`** is not present, then make sure that the server has been restarted since enabling preload for the backend(s).

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN "cn=monitor" \
  "(objectclass=ds-memory-usage-monitor-entry)" \
  cms-initiating-occupancy-fraction \
  recommended-cms-initiating-occupancy-fraction-for-current-data-set

dn: cn=JVM Memory Usage,cn=monitor
cms-initiating-occupancy-fraction:80
recommended-cms-initiating-occupancy-fraction-for-current-data-set:55
```

5. Open the **`config/java.properties`** file using a text editor, manually edit the **`CMSInitiatingOccupancyFraction`** or any other property to its recommended value in the **`start-ds.java-args`** property, and then, save the file when finished. (The following arguments are recommended for a Sun 5440 server. Contact your authorized support provider for specific assistance.):

```
start-ds.java-args=-d64 -server -Xmx20g -Xms20g -XX:MaxNewSize=1g -XX:NewSize=1g
-XXParallelGCThreads=16 -XX:+UseConcMarkSweepGC -XX:+CMSConcurrentMTEnabled
-XX:+CMSParallelRemarkEnabled -XX:+CMSParallelSurvivorRemarkEnabled
-XX:ParallelCMSThreads=8 -XX:CMSMaxAbortablePrecleanTime=3600000
-XX:+CMSScavengeBeforeRemark -XX:RefDiscoveryPolicy=1
-XX:CMSInitiatingOccupancyFraction=55 -XX:+UseParNewGC
-XX:+UseBiasedLocking -XX:+UseLargePages -XX:+HeapDumpOnOutOfMemoryError
```

Note	The **-XX:ParallelGCThreads** should be limited to 16 (default) or to 8 for smaller systems. Also, the **-XX:ParallelCMSThreads** should be limited to 8.

6. Run the **bin/dsjavaproperties** command to apply the changes.

```
$ bin/dsjavaproperties
```

7. Restart the Directory Server.

Tuning Disk-Bound Deployments

For the best server performance, we recommend that you configure your Directory Server so that it fully caches your DIT in the backend database cache. Most of the instructions in this manual assume this scenario. However, for databases that are too large or cost-prohibitive to fit in memory, it is best to do this as part of running setup in order to have the preferred value for the New Generation space. During setup, elect *not* to prime, or preload, the database at startup.

We recommend the following procedures to configure your Directory Server for optimum performance. You might also consider using a proxy server in an entry-balancing deployment, which would allow all of the data to be cached in a partitioned environment. See the *UnboundID Directory Proxy Server Administration Guide* for more information.

To Configure the Directory Server in Disk-Bound Deployments

1. Configure the JVM size to use 10-20% of the memory on the system.

2. Configure the backend **db-cache-percent** property to be 50%. (The extra memory is needed for garbage collection head room.)

3. Set the **default-cache-mode** of the backend to **evict-bin-immediately**.

4. Add **je.nodeMaxEntries=32** to the **je-property** property on the backend. The Local DB Backends need to be re-imported if data already exists.

5. Use **dsconfig** to set the **db-cache-percent**, **default-cache-mode**, and **je-property** properties.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot \
--set db-cache-percent:50 \
--set default-cache-mode:evict-bin-immediately \
--add je-property:je.nodeMaxEntries=32
```

6. Set the **`CMSInitiatingOccupancyFraction`** property in the **`java.properties`** file to be 60. For more information, see the section "JVM Garbage Collection Using CMS" on page 81.

7. Either completely uncap or set a very high limit for the filesystem cache. On Solaris with ZFS, this can be done by editing **`zfs:zfs_arc_max`** in **`/etc/system`** and restarting the system. For example, to set the limit to 80GB, you would specify the following:

```
set zfs:zfs_arc_max = 85899345920

or

set zfs:zfs_arc_max = 0x1400000000
```

8. On Linux, we recommend setting **`vm.swappiness`** to zero to protect the Directory Server JVM process from an overly aggressive filesystem cache.

```
echo 0 > /proc/sys/vm/swappiness
echo "" >> /etc/sysctl.conf
echo "# Disable filesystem cache swappiness" >> /etc/sysctl.conf
echo "vm.swappiness = 0" >> /etc/sysctl.conf
```

Once configured and restarted, the Directory Server may exhibit uneven initial performance as the more frequent access database files are read into filesystem cache. Once the filesystem cache is warmed, Directory Server performance should be at optimal rates.

Oracle DSEE Compatibility

For companies that are migrating from a Oracle Directory Server Enterprise Edition (DSEE) server to the UnboundID Directory Server, the UnboundID Directory Server provides a **`dsconfig`** batch file, **`sun-ds-compatibility.dsconfig`**, which describes the various components that can be configured to make the server exhibit behavior closer to an DSEE configuration.

The default configuration for the UnboundID Directory Server varies in several ways from the default configuration for DSEE, primarily by providing a better out-of-the-box configuration and better standards compliance. DSEE administrators can make a number of changes to their installations and may wish to apply similar changes when migrating to the UnboundID Directory Server. Administrators can use the **`sun-ds-compatibility.dsconfig`** batch file to apply the Directory Server's configuration with the necessary **`dsconfig`** commands. Simply uncomment the example commands listed in the file, and then run the **`dsconfig`** command specifying the batch file. Note that this batch file is not comprehensive and must be used together with the **`migrate-sun-ds-config`** tool, located in the **`bin`** folder (or **`bat`** folder for Windows systems) during the migration process. Both the tool and the batch file overlap in some areas but provide good initial migration support from the DSEE server to an UnboundID server.

Another useful tool is the **`migrate-ldap-schema`** tool in the **`bin`** folder (or **`bat`** folder for Windows systems), which migrates schema information from an existing LDAP server onto this UnboundID Directory Server instance. All attribute type and objectclass definitions that are contained in the source LDAP server will be added to the targeted instance or written to a schema file.

To understand the overall migration process, you can also review the *UnboundID Directory Server Migration Guide*.

To Configure the Directory Server for Oracle DSEE Compatibility

1. From the Directory Server installation directory, open the **`sun-ds-compatibility.dsconfig`** file in the **`docs`** folder. You can use a text editor to view the file.

2. Read the file completely.

3. Apply any changes to the file by removing the comment symbol at any **`dsconfig`** command example, and then applying the **`dsconfig`** command specifying the batch file.

   ```
   $ bin/dsconfig --no-prompt --bindDN "cn=Directory Manager" \
     --bindPassword "password" --batch-file /path/to/dsconfig/file
   ```

4. Run the **`migrate-ldap-schema`** to move the schema definitions on the source LDAP server to this server.

   ```
   $ bin/migrate-ldap-schema
   ```

5. Next, run the **`migrate-sun-ds-config`** tool to see what differences exist in the UnboundID configuration versus the DSEE configuration. On the UnboundID Directory Server, run the following command:

   ```
   $ bin/migrate-sun-ds-config
   ```

6. Test the server instance for further settings that may not have been set with the batch file, the **`migrate-sun-ds-config`** tool or the **`migrate-ldap-schema`** tool.

7. If you notice continued variances in your configuration, contact your authorized support provider.

6 Configuring the Directory Server

Overview

The out-of-the-box, initial configuration settings for the UnboundID Directory Server provide an excellent starting point for most general-purpose directory applications. However, additional tuning might be necessary to meet the performance, hardware, operating system, and memory requirements for your production environments.

The Directory Server stores its configuration settings in an LDIF file, **`config/config.ldif`**. Rather than editing the file directly, the Directory Server provides command-line and GUI tools that administrators can use to configure the server. The Directory Server also includes tools to create server groups, so that configuration changes can be applied to multiple servers at one time.

This chapter presents the following topics:

Accessing the Directory Server Configuration

The UnboundID Directory Server configuration can be accessed and modified in the following ways:

- **Using the dsconfig Command-Line Tool**. The `dsconfig` tool is a text-based menu-driven interface to the underlying configuration. The tool interacts with the configuration using three operational modes: interactive command-line mode, non-interactive command-line mode, and batch mode. All configuration changes made using the tool are recorded in `logs/config-audit.log`.

- **Using the UnboundID Directory Management Console**. The UnboundID Directory Server provides a web console for graphical server management and monitoring. For viewing and editing configurations, the console provides the equivalent functionality as the `dsconfig` command. All configuration changes made using this tool are recorded in `logs/config-audit.log`.

Configuring the Directory Server Using dsconfig

The `dsconfig` tool is the text-based management tool used to configure the underlying directory configuration. The tool has three operational modes: interactive mode, non-interactive mode, and batch mode.

The `dsconfig` tool also offers an offline mode using the `--offline` option, in which the server does not have to be running to interact with the configuration. In most cases, the configuration should be accessed with the server running in order for the server to give the user feedback about the validity of the configuration.

Using dsconfig in Interactive Command-Line Mode

In interactive mode, the `dsconfig` tool offers a filtering mechanism that only displays the most common configuration elements. The user can specify that more expert level objects and configuration properties be shown using the menu system.

Running `dsconfig` in interactive command-line mode provides a user-friendly, menu-driven interface for accessing and configuring the UnboundID Directory Server. To start `dsconfig` in interactive command-line mode, simply invoke the `dsconfig` script without any arguments. You will be prompted for connection and authentication information to the Directory Server, and then a menu will be displayed of the available operation types.

In some cases, a default value will be provided in square brackets. For example, `[389]` indicates that the default value for that field is port `389`. You can press Enter or Return to accept the default. To skip the connection and authentication prompts, provide this information using the command-line options of `dsconfig`.

To Configure the Directory Server using dsconfig Interactive Command-Line Mode

1. Launch the **`dsconfig`** tool in interactive command-line mode.

   ```
   $ bin/dsconfig
   ```

2. Next, enter the LDAP connection parameters. Enter the Directory Server host name or IP address, or press Enter or Return to accept the default.

3. Enter the number corresponding to the type of LDAP connection (1 for LDAP, 2 for SSL, 3 for StartTLS) that you are using on the Directory Server, or press **`Enter`** or **`Return`** to accept the default (1).

4. Next, type the LDAP listener port number, or accept the default port. The default port is the port number of the server local to the tool.

5. Enter the user bind DN (default, **`cn=Directory Manager`**) and the bind DN password.

6. On the Directory Server Configuration Console main menu, type a number corresponding to the configuration that you want to change. Note that the number can change between releases or within the same release, depending on the options selected (for example, in cases where more expert level objects and properties are displayed).

   In this example, select 1 for **`Backend`**. Then, set the **`db-cache-percent`** to **`40%`**. The optimal cache percentage depends on your system performance objectives and must be tuned as determined through analysis. In many cases, the default value chosen by the **`setup`** utility is sufficient.

7. On the Backend management menu, enter the number corresponding to view and edit an existing backend.

8. Select the backend to work with. In this example, using the basic object menu, only one backend that can be viewed in the directory, **`userRoot`**. Type **`Enter`** or **`Return`** to accept the default.

9. From the Local DB Backend properties menu, type the number corresponding to the **`db-cache-percent`** property.

10. Enter the option to change the value, and then type the value for the **`db-cache-percent`** property. In this example, type 40 for "40 %“.

11. Review the changes, and then type **`f`** to apply them.

    Before you apply the change, the **`dsconfig`** interactive command-line mode provides an option to view the equivalent non-interactive command based on your menu selections. This is useful in building **`dsconfig`** script files for configuring servers in non-interactive or batch mode. If you want to view the equivalent **`dsconfig`** non-interactive command, type **`d`**. See “Getting the Equivalent dsconfig Non-Interactive Mode Command” on page 93 for more information.

12. In the Backend management menu, type **`q`** to quit the **`dsconfig`** tool.

Using dsconfig Interactive Mode: Viewing Advanced Properties

For most configuration settings, some properties are more likely to be modified than others. The `dsconfig` interactive mode provides an option that hides or shows additional advanced properties that administrators might want to configure.

To View Advanced Properties

1. Repeat steps 1–9 in the previous section using `dsconfig` in Interactive Command-Line Mode.

2. From the Local DB Backend properties menu, type `a` to display the advanced properties, which toggles any hidden properties.

Using dsconfig Interactive Mode: Viewing Object Menus

Because some configuration objects are more likely to be modified than others, the UnboundID Directory Server provides four different object menus that hide or expose configuration objects to the user.

The following object menus are available:

- **Basic**. Only includes the components that are expected to be configured most frequently.

- **Standard**. Includes all components in the Basic menu plus other components that might occasionally need to be altered in many environments.

- **Advanced**. Includes all components in the Basic and Standard menus plus other components that might require configuration under special circumstances or that might be potentially harmful if configured incorrectly.

- **Expert**. Includes all components in the Basic, Standard, and Advanced menus plus other components that should almost never require configuration or that could seriously impact the functionality of the server if not properly configured.

The purpose of object menus is to simply present only those properties that an administrator will likely use and is a convenience feature designed to unclutter menu readability.

To Change the dsconfig Object Level

1. Repeat steps 1–6 in the section using `dsconfig` in Interactive Command-Line Mode.

2. On the UnboundID Directory Server configuration main menu, type o (letter “o”) to change the object level. By default, Basic objects are displayed.

3. Enter a number corresponding to a object level of your choice: `1` for Basic, `2` for Standard, `3` for Advanced, `4` for Expert.

4. View the menu at the new object level. You should see additional configuration options for the Directory Server components.

```
>>>> UnboundID Directory Server configuration console main menu

What do you want to configure?

    1)   Account Status Notification Handler  15)  Log Retention Policy
    2)   Alert Handler                         16)  Log Rotation Policy
    3)   Backend                               17)  Password Generator
    4)   Certificate Mapper                    18)  Password Policy
    5)   Client Connection Policy              19)  Password Validator
    6)   Connection Criteria                   20)  Plugin
    7)   Connection Handler                    21)  Request Criteria
    8)   Global Configuration                  22)  Result Criteria
    9)   Identity Mapper                       23)  Search Entry Criteria
    10)  Key Manager Provider                  24)  Search Reference Criteria
    11)  Local DB Index                        25)  Trust Manager Provider
    12)  Location                              26)  Virtual Attribute
    13)  Log Field Mapping                     27)  Work Queue
    14)  Log Publisher

    o)   'Standard' objects are shown - change this
    q)   quit

Enter choice:
```

Using dsconfig Interactive Mode: Viewing Administrative Action Alerts

The **dsconfig** tool and the web console provide a useful feature that displays notifications for certain operations that require further administrator action to complete the process. If you change a certain backend configuration property, the admin action will appear in two places during a **dsconfig** interactive session: when configuring the property and before you apply the change. For example, if you change the **db-directory** property on the **userRoot** backend (that is, specify the path to the filesystem path that holds the Oracle Berkeley DB Java Edition backend files), you will see an admin action reminder during one of the steps (shown below).

The admin action alert will also appear as a final confirmation step. The alert allows you to continue and apply the change or back out of the configuration if the resulting action cannot be conducted at the present time. For example, after you type "f" to apply the **db-directory** property change, the admin alert message appears:

```
Enter choice [b]: f

One or more configuration property changes require administrative action or confirma-
tion/notification. Those properties include:
 * db-directory:  Modification requires that the Directory Server be stopped, the data-
base directory manually relocated, and then the Directory Server restarted. While the
Directory Server is stopped, the directory and files pertaining to this backend in the
old database directory must be manually moved or copied to the new location.

Continue?  Choose 'no' to return to the previous step (yes / no) [yes]:
```

Currently, only a small set of properties display an admin action alert appear in **dsconfig** interactive mode and the web console. For more information on the properties, see the *UnboundID Directory Server Configuration Reference.*

Using dsconfig in Non-Interactive Command-Line Mode

The **dsconfig** non-interactive command-line mode provides a simple way to make arbitrary changes to the Directory Server by invoking it from the command line. To use administrative scripts to automate configuration changes, run the **dsconfig** command in non-interactive mode, which is more convenient than using interactive mode. Note, however, that if you plan to make changes to multiple configuration objects at the same time, then the batch mode might be more appropriate.

You can use the **dsconfig** tool to update a single configuration object using command-line arguments to provide all of the necessary information. The general format for the non-interactive command line is:

```
$ bin/dsconfig --no-prompt {globalArgs} {subcommand} {subcommandArgs}
```

The **--no-prompt** argument indicates that you want to use non-interactive mode. The **{subcommand}** is used to indicate which general action to perform. The **{globalArgs}** argument provides a set of arguments that specify how to connect and authenticate to the Directory Server. Global arguments can be standard LDAP connection parameters or SASL connection parameters depending on your setup. For example, using standard LDAP connections, you can invoke the **dsconfig** tool as follows:

```
$ bin/dsconfig --no-prompt list-backends \
  --hostname server.example.com \
  --port 389 \
  --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password
```

If your system uses Kerberos V, you can invoke **dsconfig** as follows:

```
$ ./dsconfig --no-prompt list-backends \
  --saslOption mech=GSSAPI \
  --saslOption authid=admin@example.com \
  --saslOption ticketcache=/tmp/krb5cc_1313 \
  --saslOption useticketcache=true
```

The**{subcommandArgs}** argument contains a set of arguments specific to the particular subcommand that you wish to invoke. To always display the advanced properties, use the **--advanced** command-line option.

Note	Global arguments can appear anywhere on the command line (including before the subcommand, and after or intermingled with subcommand-specific arguments). The subcommand-specific arguments can appear anywhere after the subcommand.

To View the List of dsconfig Properties

1. Use the **dsconfig** command with the **list-properties** option to view the list of all **dsconfig** properties. Remember to add the LDAP connection parameters.

   ```
   $ bin/dsconfig list-properties
   ```

2. Use the **dsconfig** command with the **list-properties** option and the **--complexity <menu level>** to view objects at and below the menu object level. You can also add the **--includeDescription** argument that includes a synopsis and description of each property in the output. Remember to add the LDAP connection parameters.

```
$  bin/dsconfig list-properties --complexity advanced --includeDescription
```

3. If the server is offline, you can run the command with the **--offline** option. You do not need to enter the LDAP connection parameters.

```
$ bin/dsconfig list-properties --offline --complexity advanced --includeDescription
```

Note	You can also view the **<server-root>/docs/config-properties.txt** that contains the property information provided with the server.

To Configure Additional Tuning Using dsconfig Non-Interactive Mode

1. Use the **dsconfig** command in non-interactive mode to change the amount of memory used for caching database contents and to specify common parent DNs that should be compacted in the underlying database.

```
$ bin/dsconfig set-backend-prop \
  --host server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --backend-name userRoot \
  --set db-cache-percent:40 \
  --add "compact-common-parent-dn:ou=accts,dc=example,dc=com" \
  --add "compact-common-parent-dn:ou=subs,dc=example,dc=com" \
  --no-prompt
```

Getting the Equivalent dsconfig Non-Interactive Mode Command

While the **dsconfig** non-interactive command-line mode is convenient for scripting and automating processes, obtaining the correct arguments and properties for each configuration change can be quite time consuming.

To facilitate easy and quick configuration, you can use an option to display the equivalent non-interactive command using **dsconfig** interactive mode. The command displays the equivalent **dsconfig** command to recreate the configuration in a scripted configuration or to more quickly enter any pending changes on the command line for another server instance.

Note	There are two other ways to get the equivalent **dsconfig** command. One way is by looking at the **logs/config-audit.log**. It might be more convenient to set the Directory Server up the way you want and then get the **dsconfig** arguments from the log. Another way is by configuring an option using the Directory Management Console. The console shows the equivalent **dsconfig** command prior to applying the change.

To Get the Equivalent dsconfig Non-Interactive Mode Command from Interactive Mode

1. Using **dsconfig** in interactive mode, make changes to a configuration but do not apply the changes (that is, do not enter "f").

2. Enter **d** to view the equivalent non-interactive command.

3. View the equivalent command (seen below), and then press **Return** to continue. For example, based on an example in the previous section, changes made to the **db-cache-percent** returns:

   ```
   Command line to apply pending changes to this Local DB Backend:

   dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:40
   ```

   The command does not contain the LDAP connection parameters required for the tool to connect to the host since it is presumed that the command would be used to connect to a different remote host.

Using dsconfig in Batch Mode

The UnboundID Directory Server provides a **dsconfig** batching mechanism that reads multiple **dsconfig** invocations from a file and executes them sequentially. The batch file provides advantages over standard scripting by minimizing LDAP connections and JVM invocations that normally occur with each **dsconfig** call. If a **dsconfig** command has a missing or incorrect argument, the command will fail and abort the batch process. Any command that was successfully executed prior to the abort will be applied to the Directory Server. The **--no-prompt** option is required with **dsconfig** in batch mode.

You can view the **logs/config-audit.log** file to review the configuration changes made to the Directory Server and use them in the batch file. The batch file can have blank lines for spacing and lines starting with a pound sign (#) for comments. The batch file also supports a "\" line continuation character for long commands that require multiple lines.

The Directory Server also provides a **docs/sun-ds-compatibility.dsconfig** file for migrations from Oracle Directory Server Enterprise Edition (DSEE) to UnboundID Directory Server machines.

To Configure the Directory Server in dsconfig Batch Mode

1. Create a text file that lists each **dsconfig** command with the complete set of properties that you want to apply to the Directory Server. The items in this file should be in the same format as those accepted by the **dsconfig** command.

   ```
   # Set the DB cache percent to 40
   set-backend-prop --backend-name userRoot --set db-cache-percent:40

   # Disable prime-method to none
   set-backend-prop --backend-name userRoot --reset prime-method
   ```

```
# Create a local db backend
create-backend --backend-name accts --type local-db \
--set description:Accounts \
--set db-cache-percent:75 \
--set prime-thread-count:4 \
--set enabled:true \
--set base-dn:ou=accts,dc=example,dc=com \
--set db-num-cleaner-threads:2 \
--set "compact-common-parent-dn:ou=accts,dc=example,dc=com" \
--set prime-all-indexes:false \
--set system-index-to-prime:dn2id \
--set system-index-to-prime:dn2uri \
--set system-index-to-prime:id2children \
--set system-index-to-prime:id2subtree
```

2. Use **`dsconfig`** with the **`--batch-file`** option to read and execute the commands.

```
$ bin/dsconfig --hostname host1 --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt \
  --batch-file /path/to/config-batch.txt
```

Configuring the Directory Server Using the Directory Management Console

The UnboundID Directory Server provides a graphical web console for server management and monitoring that has the same functionality as that of the **`dsconfig`** command. When logging on to the Directory Management Console, the console does not persistently store any credentials for authenticating to the Directory Server but uses the credentials provided by the user when logging in. When managing multiple directory instances, the provided credentials must be valid for each instance.

For information on installing the Directory Management Console, see “Installing the Directory Management Console” on page 52.

Logging into the Directory Management Console

To log into the console, you can either use a DN (for example, **`cn=admin,dc=example,dc=com`**) or provide the name of an administrator, which is stored under **`cn=admin data`**, such as **`admin`** or **`cn=admin2,cn=Administrators,cn=Admin Data`**. See “Configuring a Global Administrator” on page 108 for instructions.

To Log into the Console

1. Start the directory server.

```
$ bin/start-ds
```

2. Start the Apache Tomcat servlet container.

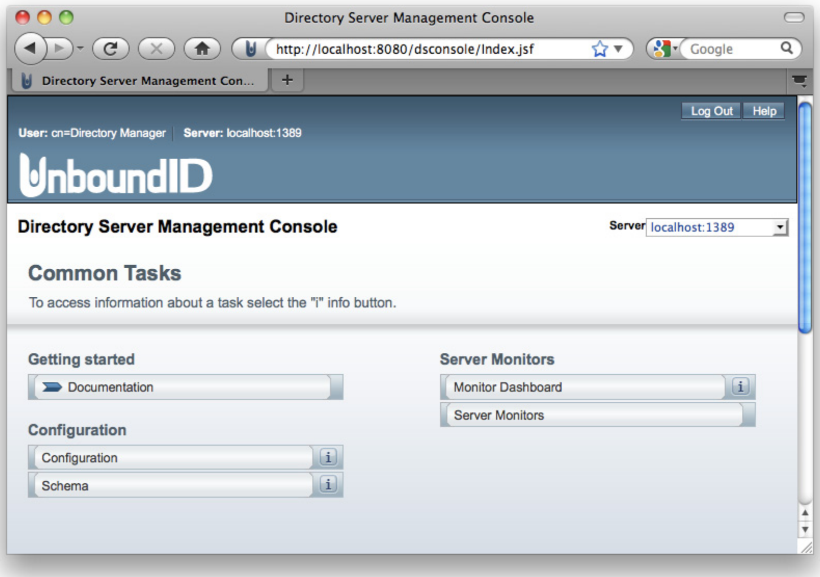
```
$ /apache-tomcat-<version>/bin/startup.sh
```

3. Open a browser to `http://hostname:8080/dsconsole/`.

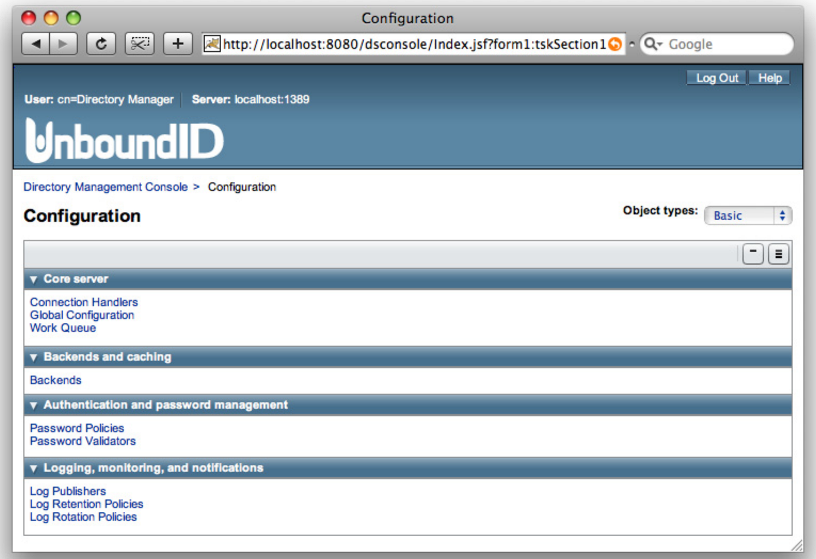
4. Enter the root user DN (or any authorized administrator user name) and password, and then click **Login**.

   The console does not persistently store any credentials for accessing a directory server. Instead, it uses the credentials provided by the user when logging into the console. When managing multiple directory instances, the provided credentials must be valid at each instance.

5. On the Directory Management Console, click **Configuration**.

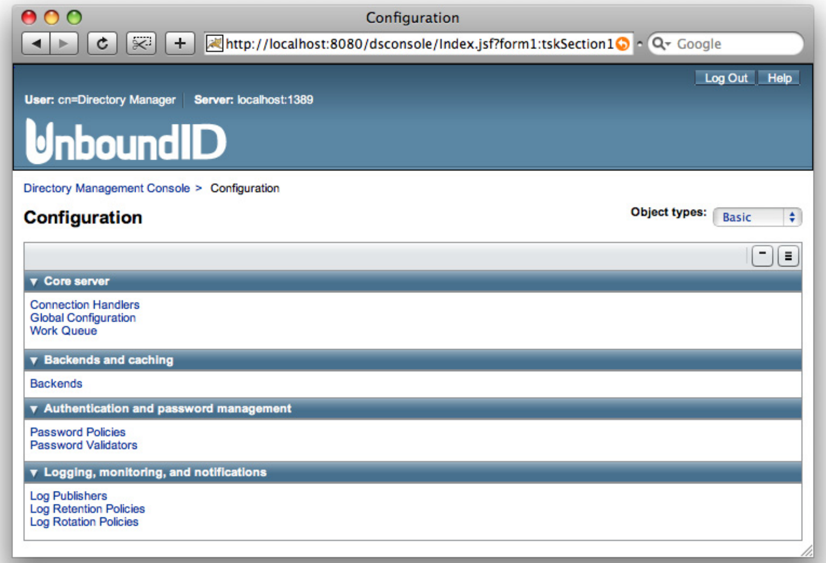

Directory Server Management Console
http://localhost:8080/dsconsole/Index.jsf
Google
Directory Server Management Con...
Log Out
Help
User: cn=Directory Manager
Server: localhost:1389
UnboundID
Directory Server Management Console
Server localhost:1389
Common Tasks
To access information about a task select the "i" info button.
Getting started
Documentation
Configuration
Configuration
Schema
Server Monitors
Monitor Dashboard
Server Monitors

6. View the Configuration menu. By default, the console displays the Basic object type properties. You can change the object level of the object types using the **Object Types** drop-down list.

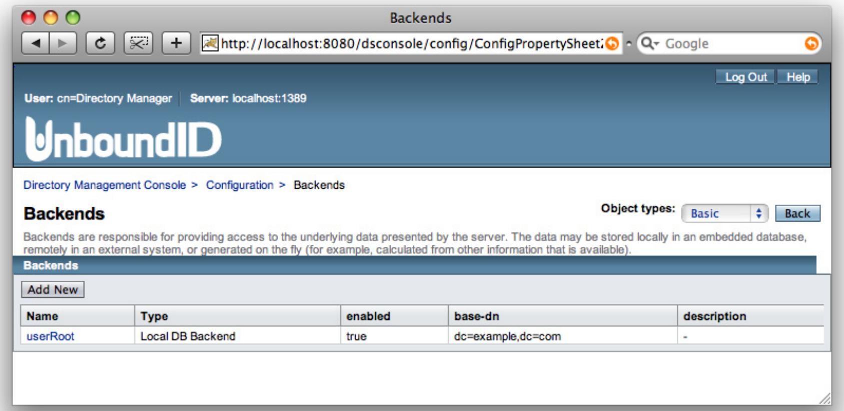


To Configure the Directory Server Using the Directory Management Console

1. Log into the Directory Management Console. Click **Configuration** to open the Configuration menu, and then click **Backends**.



2. In the Backends menu, click the backend to work with. For this example, only one backend is displayed. Click **userRoot**.



Backends

http://localhost:8080/dsconsole/config/ConfigPropertySheet

User: cn=Directory Manager | Server: localhost:1389

Directory Management Console > Configuration > Backends

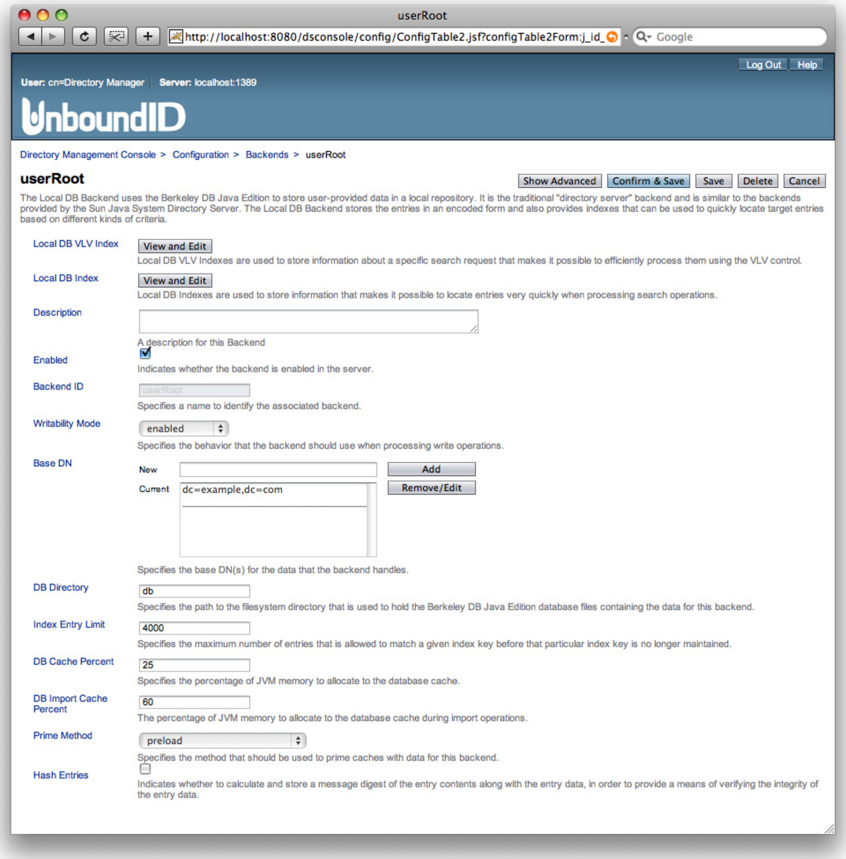
Backends

Object types: Basic

Backends are responsible for providing access to the underlying data presented by the server. The data may be stored locally in an embedded database, remotely in an external system, or generated on the fly (for example, calculated from other information that is available).

Name	Type	enabled	base-dn	description
userRoot	Local DB Backend	true	dc=example,dc=com	-

3. Change or enter values in the userRoot configuration. For this example, change the DB Cache Percent value to `40`, and then click **Confirm & Save** to apply the change without further confirmation.



userRoot

http://localhost:8080/dsconsole/config/ConfigTable2.jsf?configTable2Form:j_id_

User: cn=Directory Manager | Server: localhost:1389

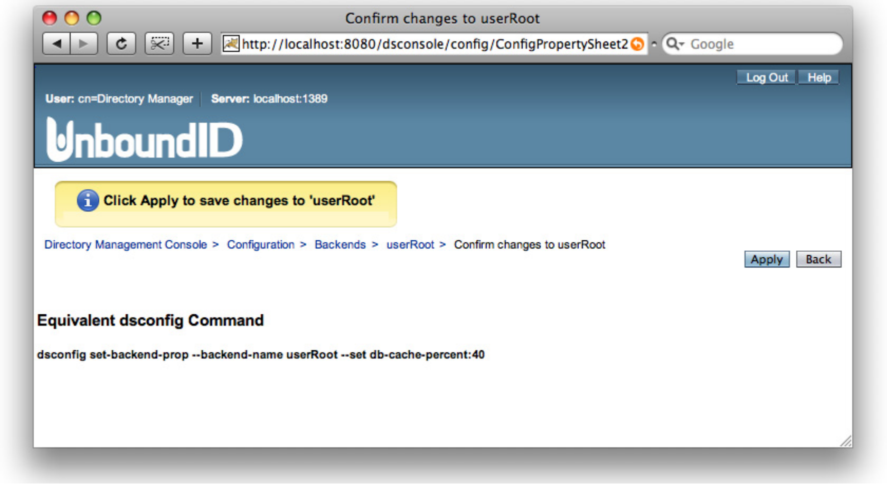
Directory Management Console > Configuration > Backends > userRoot

userRoot

The Local DB Backend uses the Berkeley DB Java Edition to store user-provided data in a local repository. It is the traditional "directory server" backend and is similar to the backends provided by the Sun Java System Directory Server. The Local DB Backend stores the entries in an encoded form and also provides indexes that can be used to quickly locate target entries based on different kinds of criteria.

Local DB VLV Index	Local DB VLV Indexes are used to store information about a specific search request that makes it possible to efficiently process them using the VLV control.
Local DB Index	Local DB Indexes are used to store information that makes it possible to locate entries very quickly when processing search operations.
Description	A description for this Backend
Enabled	✓ Indicates whether the backend is enabled in the server.
Backend ID	userRoot — Specifies a name to identify the associated backend.
Writability Mode	enabled — Specifies the behavior that the backend should use when processing write operations.
Base DN	Current: dc=example,dc=com — Specifies the base DN(s) for the data that the backend handles.
DB Directory	db — Specifies the path to the filesystem directory that is used to hold the Berkeley DB Java Edition database files containing the data for this backend.
Index Entry Limit	4000 — Specifies the maximum number of entries that is allowed to match a given index key before that particular index key is no longer maintained.
DB Cache Percent	25 — Specifies the percentage of JVM memory to allocate to the database cache.
DB Import Cache Percent	60 — The percentage of JVM memory to allocate to the database cache during import operations.
Prime Method	preload — Specifies the method that should be used to prime caches with data for this backend.
Hash Entries	☐ Indicates whether to calculate and store a message digest of the entry contents along with the entry data, in order to provide a means of verifying the integrity of the entry data.

4. Click **Apply** to save the changes. Note that you can see the equivalent non-interactive command-line command on this page. If the property has any associated admin action requirements, such as a system or component restart, it will appear on this page:


Confirm changes to userRoot
http://localhost:8080/dsconsole/config/ConfigPropertySheet2
Google
Log Out
Help
User: cn=Directory Manager
Server: localhost:1389
UnboundID
Click Apply to save changes to 'userRoot'
Directory Management Console > Configuration > Backends > userRoot > Confirm changes to userRoot
Apply
Back

Equivalent dsconfig Command

```
dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:40
```

Generating a Summary of Configuration Components

The Directory Server provides a **`summarize-config`** tool that generates a summary of the configuration in a local or remote directory server instance. The tool is useful when comparing configuration settings on a directory server instance when troubleshooting issues or when verifying configuration settings on newly-added servers to your network. The tool can interact with the local configuration regardless of whether the server is running or not.

By default, the tool generates a list of basic components. To include a list of advanced components, use the **`--advanced`** option. To run the tool on an offline server, use the **`--offline`** option. Run the **`summarize-config --help`** option to view other available tool options.

To Generate a Summary of Configuration Components

Run the **`summarize-config`** tool to generate a summary of the configuration components on the directory server instance. The following command runs a summary on a local online server.

```
$ bin/summarize-config

   Backends:
     Local DB Backend: userRoot
       backend-id: userRoot
       enabled: true
       writability-mode: enabled
       base-dn: "dc=example,dc=com"
       set-degraded-alert-when-disabled: true
       return-unavailable-when-disabled: true
       db-directory: db
       index-entry-limit: 4000
       db-cache-percent: 25
```

```
  db-import-cache-percent: 60
  prime-method: none

  hash-entries: false
  set-degraded-alert-for-untrusted-index: true
  return-unavailable-for-untrusted-index: true
  Local DB Indexes:
    Local DB Index: mail
      attribute: mail
      index-entry-limit: 4000
      index-type: equality
    Local DB Index: uniqueMember
      attribute: uniquemember
      index-entry-limit: 4000
      index-type: equality
    Local DB Index: sn
      attribute: sn
      index-entry-limit: 4000
      index-type: equality, substring
    Local DB Index: givenName
      attribute: givenName
      index-entry-limit: 4000
      index-type: equality, substring
    Local DB Index: uid
      attribute: uid
      index-entry-limit: 4000
      index-type: equality
    Local DB Index: objectClass
      attribute: objectClass
      index-entry-limit: 4000
      index-type: equality
    Local DB Index: member
      attribute: member
      index-entry-limit: 4000
      index-type: equality
    Local DB Index: cn
      attribute: cn
      index-entry-limit: 4000
      index-type: equality, substring
    Local DB Index: aci
      attribute: aci
      index-entry-limit: 4000
      index-type: presence
    Local DB Index: telephoneNumber
      attribute: telephonenumber
      index-entry-limit: 4000
      index-type: equality
    Local DB Index: ds-entry-unique-id
      attribute: ds-entry-unique-id
      index-entry-limit: 4000
      index-type: equality

Backend: replicationChanges
  java-class: com.unboundid.directory.server.replication.server.ReplicationBackend
  backend-id: replicationChanges
  enabled: true
  writability-mode: disabled
  base-dn: dc=replicationChanges
  set-degraded-alert-when-disabled: true
  return-unavailable-when-disabled: true

Changelog Backend: changelog
  backend-id: changelog
  enabled: false
  writability-mode: disabled
```

```
base-dn: cn=changelog
set-degraded-alert-when-disabled: false
return-unavailable-when-disabled: false
db-directory: db
changelog-maximum-age: 2 d
db-cache-percent: 1
changelog-include-attribute: -
changelog-exclude-attribute: -
changelog-deleted-entry-include-attribute: -
changelog-deleted-entry-exclude-attribute: -
write-lastmod-attributes: true
use-reversible-form: false
apply-access-controls-to-changelog-entry-contents: false
report-excluded-changelog-attributes: none
```

About Root Users, Administrators, and Global Administrators

The Directory Server provides three different classes of administrator accounts: root user, administrator, and global administrator. The *root user* is the LDAP-equivalent of a UNIX super-user account and inherits its privileges from the default root user privilege set (see “Default Root Privileges” on page 102). The root user "account" is an entry that is stored in the server’s configuration under the **`cn=Root DNs,cn=config`** and bypasses access control evaluation. This account has full access to the entire set of data in the directory information tree (DIT) as well as full access to the server configuration and its operations. One important difference between other vendors’ servers and the Directory Server’s implementation is that the root user’s rights are granted through a set of privileges. This allows the Directory Server to have multiple root users on its system if desired; however, the normal practice is to set up administrator user entries. Also, the Root User has no resource limits by default.

The *administrator user* can have a full set of root user privileges but often has a subset of these privileges to limit the accessible functions that can be performed. The administrators entries typically have limited access to the entire set of data in the directory information tree (DIT), which is controlled by access control instructions. These entries reside in the backend configuration (for example, **`uid=admin,dc=example,dc=com`**) and are replicated between servers in a replication topology. In some cases, administrator user accounts may be unavailable when the server enters lockdown mode unless the administrator is given the **`lock-down`** mode privilege.

A *global administrator* is primarily responsible for managing configuration server groups and can be used to log in to the Directory Management Console. A configuration server group is an administration domain that allows you to synchronize configuration changes to one or all of the servers in the group. For example, you can set up a group when configuring a replication topology, where configuration changes to one server can be applied to all of the servers at one time. Global Administrator(s) are stored in the **`cn=admin data`** backend. These entries along with other admin backend data are always replicated between servers in a replication topology. These users can be assigned privileges like other admin users but are typically used to manage the data under **`cn=admin data`** and **`cn=config`**.

Managing Root User Accounts

The UnboundID Directory Server provides a default root user, `cn=Directory Manager`, that is stored in the server's configuration file (for example, under `cn=Root DNs,cn=config`). The root user is the LDAP-equivalent of a UNIX super-user account and inherits its read-write privileges from the default root privilege set. Root user entries are stored in the server's configuration and not in backend data. Root users have access to all of the data in the directory.

To limit full access to all of the Directory Server, we recommend that you create separate administrator user accounts with limited privileges so that you can identify the administrator responsible for a particular change. Having separate user accounts for each administrator also makes it possible to enable password policy functionality (such as password expiration, password history, and requiring secure authentication) for each administrator.

Default Root Privileges

The UnboundID Directory Server contains a privilege subsystem that allows for a more fine-grained control of privilege assignments. The following set of root privileges are available to each root user DN:

TABLE 6-1. Default Root Privileges

Privilege	Description
audit-data-security	Allows the associated user to execute data security auditing tasks.
backend-backup	Allows the user to perform backend backup operations.
backend-restore	Allows the user to perform backend restore operations.
bypass-acl	Allows the user to bypass access control evaluation.
config-read	Allows the user to read the server configuration.
config-write	Allows the user to update the server configuration.
disconnect-client	Allows the user to terminate arbitrary client connections.
ldif-export	Allows the user to perform LDIF export operations.
ldif-import	Allows the user to perform LDIF import operations.
lockdown-mode	Allows the user to request a server lockdown.
modify-acl	Allows the user to modify access control rules.
password-reset	Allows the user to reset user passwords but not their own. The user must also have privileges granted by access control to write the user password to the target entry.
privilege-change	Allows the user to change the set of privileges for a specific user, or to change the set of privileges automatically assigned to a root user.
server-restart	Allows the user to request a server restart.
server-shutdown	Allows the user to request a server shutdown.
stream-values	Allows the user to perform a stream values extended operation that obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
third-party-task	Allows the associated user to invoke tasks created by third-party developers.
unindexed-search	Allows the user to perform an unindexed search in the Oracle Berkeley DB Java Edition backend.

TABLE 6-1. **Default Root Privileges**

Privilege	Description
update-schema	Allows the user to update the server schema.
use-admin-session	Allows the associated user to use an administrative session to request that operations be processed using a dedicated pool of worker threads.

The Directory Server provides other privileges that are not assigned to the root user DN by default but can be added using the **`ldapmodify`** tool (see "Modifying Individual Root User Privileges" on page 104 for more information).

TABLE 6-2. **Other Available Privileges**

Privilege	Description
bypass-read-aci	Allows the associated user to bypass access control checks performed by the server for bind, compare, and search operations. Access control evaluation may still be enforced for other types of operations.
jmx-notify	Allows the associated user to subscribe to receive JMX notifications.
jmx-read	Allows the associated user to perform JMX read operations.
jmx-write	Allows the associated user to perform JMX write operations.

Viewing the Default Root User Privileges Using dsconfig

The root DN accounts are the only user accounts that are stored within the Directory Server's configuration under **`cn=Root DNs,cn=config`**. You can view the default privileges automatically granted to root users using the **`dsconfig`** tool.

To View the Default Privileges Assigned to Root Users

Use **`dsconfig`** to view the root DN.

```
$ bin/dsconfig get-root-dn-prop --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

Modifying a Root User Password

Root users are governed by the Root Password Policy and by default, their passwords never expire. However, in the event that you want to change a root user's password, you can use the **`ldappasswordmodify`** tool.

To Modify a Root User Password

1. Open a text editor and create a text file containing the new password. In this example, name the file **`rootuser.txt`**.

```
$ echo password > rootuser.txt
```

2. Use **ldappasswordmodify** to change the root user's password.

```
$ bin/ldappasswordmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --newPasswordFile rootuser.txt
```

3. Remove the text file.

```
$ rm rootuser.txt
```

Creating a Root User

You can create another root user by adding an entry under **cn=Root DNs,cn=config**. By default, the new root user will receive the default set of root privileges. If you want the new root user to have a limited set of privileges, you can remove the privileges using the **dsconfig** tool.

To Create a Root User

1. Open a text editor, and create a file containing the root user entry.

```
dn: cn=Directory Manager2,cn=Root DNs,cn=config
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: ds-cfg-root-dn-user
userPassword: password
cn: Directory Manager2
sn: Manager2
ds-cfg-alternate-bind-dn: cn=Directory Manager2
givenName: Directory
```

2. Use **ldapmodify** to add the entry.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --defaultAdd --filename "rootuser.ldif"
```

Modifying Individual Root User Privileges

All root users automatically inherit the default root privileges defined in the **default-root-privilege-name** configuration property. However, you can give individual root users additional privileges that are not included in the set of default root privileges. You can also remove default root privileges from individual root users.

Modifying the privileges of the root user can be accomplished by adding the **ds-privilege-name** operational attribute to the entry for the root user. Any values containing a privilege name will grant that privilege to the user in addition to the set of default root privileges. Any values containing a minus sign followed by a privilege name will remove that privilege from that root user, even if it is included in the set of default root privileges.

To Modify the Privileges for an Individual Root User

1. Open a text editor, and create a file containing the changes to the root user entry. The following example grants the **`proxied-auth`** privilege and removes the **`server-shutdown`** and **`server-restart`** privileges.

```
dn: cn=Directory Manager2,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
ds-privilege-name: -server-shutdown
ds-privilege-name: -server-restart
```

2. Use **`ldapmodify`** to apply the change.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --filename "modifyRootUserPrivileges.ldif"
```

Configuring Administrator Accounts

The administrator user has a subset of root user privileges to ensure that only the functions that are needed to carry out their jobs are assigned to them. Typically, administrator user entries are controlled by access control evaluation to limit access to the entire set of data in the directory information tree (DIT). Administrator entries reside in the backend configuration (for example, **`uid=admin,dc=example,dc=com`**) and are replicated between servers in a replication topology.

The following examples show how to configure administrator accounts. The first procedure shows how to set up a single, generic **`uid=admin,dc=example,dc=com`** account with limited privileges. Note that if you generated sample data at install, you can view an example **`uid=admin`** entry using **`ldapsearch`**. The second example shows a more realistic example, where the user is part of a Directory Administrators group. Note that both examples are based on a simple Directory Information Tree (DIT). Actual deployment cases depends on your schema.

To Set Up a Single Administrator Account

1. Create an LDIF file with an example Administrator entry.

```
dn: uid=admin,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
givenName: Admin
uid: admin
cn: Admin User
sn: User
userPassword: password
```

Then add the entry using the **ldapmodify** tool.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --defaultAdd --filename admin.ldif
```

2. Create another LDIF file to add the access control instruction (ACI) to the root suffix, or base DN to provide full access to the directory to the new administrator.

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr != "aci")
  (version 3.0; acl "Grant full access for the admin user";
     allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

Then add the entry using the **ldapmodify** tool.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --filename admin.ldif
```

3. Verify the additions using **ldapsearch**. The first command searches for the entry that contains **uid=Admin** and returns it if the search is successful. The second command searches for the base DN and returns only those operational attributes, including access control instructions, associated with the entry.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN dc=example,dc=com "(uid=admin)"

$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN dc=example,dc=com --searchScope base \
  "(objectclass=*)" "+"
```

4. Add specific privileges to the Admin account. In this example, add the **password-reset** privilege to the admin account from the command line.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: password-reset
(Press CTRL-D)
Processing MODIFY request for uid=admin,dc=example,dc=com
MODIFY operation successful for DN uid=admin,dc=example,dc=com
```

5. Assign a password policy for the Admin account. For example, create an "Admin Password Policy", then add the password policy to the account.

```
$ bin/dsconfig create-password-policy \
  --policy-name "Admin Password Policy" \
  --set "description:Password policy for directory administrators" \
  --set password-attribute:userpassword \
  --set "default-password-storage-scheme:Salted SHA-1" \
  --set password-change-requires-current-password:true \
  --set force-change-on-reset:true \
  --set "max-password-age:25 weeks 5 days" \
  --set grace-login-count:3 \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret \
  --no-prompt
```

6. Apply the password policy to the account. In this example, the password policy is being added from the command line.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changtype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Admin Password Policy,cn=Password Policies,cn=config
```

To Set Up an Administrator Group

The following example shows how to set up a group of administrators that have access rights to the whole directory. The example uses a static group using the **`GroupOfUniqueNames`** object class.

Note	You can also create a virtual static group, where each attribute is represented by a virtual attribute.

1. Create an LDIF file with an example Administrator group.

```
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalunit
objectClass: top
ou: Groups

dn: cn=Dir Admins,ou=Groups,dc=example,dc=com
objectClass: groupofuniquenames
objectClass: top
uniqueMember: uid=user.0, ou=People, dc=example,dc=com
uniqueMember: uid=user.1, ou=People, dc=example,dc=com
cn: Dir Admins
ou: Groups
```

Then add the entry using the **`ldapmodify`** tool:

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --defaultAdd --filename admin-group.ldif
```

2. Create another LDIF file to add the access control instruction (ACI) to the root suffix, or base DN to provide full access to the directory to the new administrator.

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=example,dc=com")
  (targetattr != "aci")
  (version 3.0; acl "allow all Admin group";
    allow(all) groupdn = "ldap:///cn=Dir Admins,ou=Groups,dc=example,dc=com";)
```

Then add the entry using the **`ldapmodify`** tool:

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --filename admin.ldif
```

3. Verify the additions using `ldapsearch`. The first command searches for the entry that contains `cn=Dir Admins` and returns it if the search is successful. The second command searches for the base DN and returns only those operational attributes, including access control instructions, associated with the entry.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN dc=example,dc=com \
  "(cn=Dir Admins)"

$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN dc=example,dc=com --searchScope base \
  "(objectclass=*)" "+"
```

4. Add specific privileges to each Admin account. In this example, add the `password-reset` privilege to the `user.0` admin account from the command line. Repeat the process for the other administrators configured in the Admin group.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: password-reset
(Press CTRL-D)
Processing MODIFY request for uid=user.0,dc=example,dc=com
MODIFY operation successful for DN uid=user.0,dc=example,dc=com
```

5. Assign a password policy for the Admin account. For example, create an "Admin Password Policy", then add the password policy to the account. Apply the password policy to the account.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=user.0,dc=example,dc=com
changtype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Admin Password Policy,cn=Password Policies,cn=config
```

Configuring a Global Administrator

A global administrator is responsible for managing configuration server groups or can be used to log in to the Directory Management Console. A configuration server group is an administration domain that allows you to synchronize configuration changes to one or all of the servers in the group. For example, you can set up a group when configuring a replication topology, where configuration changes to one server can be applied to all of the servers at a time.

Global Administrator(s) are stored in the `cn=admin data` backend. These entries along with other admin backend data is always replicated between servers in a replication topology. Global Administrators can be assigned privileges like other admin users but are typically used to manage the data under `cn=admin data` and `cn=config`. You can create new global administrators and remove existing global administrators using the `dsframework` tool. The global administrator entries are located in the `cn=Administrators,cn=admin data` branch. Once you have set up a global admin, you can use it to log on to the Directory Management Console. For example, the Directory Management Console accepts the root user DN (for example,

cn=Directory Manager), the global admin ID (for example, **admin2**), or the full global admin DN (for example, **cn=admin2,cn=Administrators,cn=Admin Data**).

By default, the **dsframework** tool connects to any registered server interface when querying for configuration and status-related information. To connect over SSL or StartTLS, the **dsframework set-server-properties** command can be used to set the **preferredSecurity** property (possible values: **none**, **ssl**, or **starttls**). This feature allows the administrator to have control over which security protocol that the tool should use for such queries. To view the server properties, use the **dsframework list-server-properties** command.

To Create a New Global Administrator Using dsframework

1. Use **dsframework** to create a new global administrator.

```
$ bin/dsframework create-admin-user \
  --userID admin2 --set password:secret \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
```

2. To verify the creation of the new administrator, use the **list-admin-user** subcommand with **dsframework**.

```
$ bin/dsframework list-admin-user --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
id: admin2
id: admin
```

To Remove a Global Administrator

1. Use **dsframework** to delete an existing global administrator.

```
$ bin/dsframework delete-admin-user --userID admin2 \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
```

2. To verify the deletion of the global administrator, use the **list-admin-user** option with **dsframework**.

```
$ bin/dsframework list-admin-user --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
id: admin
```

Configuring Configuration Server Groups

The UnboundID Directory Server provides a mechanism for setting up administrative domains that synchronize configuration changes among servers in a server group. After you have set up a server group, you can make an update on one server using **dsconfig**, then you can apply the change to the other servers in the group using the **--applyChangeTo server-group** option of the **dsconfig** non-interactive command. If you want to apply the change to one server in the group, use the **`--applyChangeTo single-server`** option. When using **`dsconfig`** in interactive command-line mode, you will be asked if you want to apply the change to a single server or to all servers in the server group.

Configuring a Server Group

You can create an administrative server group using the **`dsframework`** tool. The general process is to create a group, register each server, add each server to the group, and then set a global configuration property to use the server group. If you are configuring a replication topology, then you must configure the replicas to be in a server group as outlined in “Replication Configuration” on page 383.

The following example procedure adds three directory server instances into the server group labelled "group-one". The commands are run on server1.example.com.

Server	Host Name	LDAP Port
instance 1	server1.example.com	1389
instance 2	server2.example.com	2389
instance 3	server3.example.com	3389

To Create a Server Group

1. Create a group called “group-one” using **`dsframework`**.

```
$ bin/dsframework create-group --groupName group-one \
  --description "Server Group One" --hostname server1.example.com \
  --port 1389 --bindDN "cn=admin,cn=administrators,cn=admin data" \
  --bindPassword secret
```

2. Register each directory server that you want to add to the server group. If you have set up replication between a set of servers, these server entries will have already been created by the **`dsreplication enable`** command.

```
$ bin/dsframework register-server \
  --serverID server1.example.com:1389 --set hostname:server1.example.com \
  --set ldapport:1389 --set ldapEnabled:true \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret

$ bin/dsframework register-server \
  --serverID server2.example.com:2389 --set hostname:server2.example.com \
  --set ldapport:2389 --set ldapEnabled:true \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret

$ bin/dsframework register-server \
  --serverID server3.example.com:3389 --set hostname:server3.example.com \
  --set ldapport:3389 --set ldapEnabled:true \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret
```

3. Add each directory server to the group.

```
$ bin/dsframework add-to-group \
  --groupName group-one --memberName server1.example.com:1389 \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret

$ bin/dsframework add-to-group \
  --groupName group-one --memberName server2.example.com:2389 \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret

$ bin/dsframework add-to-group \
  --groupName group-one --memberName server3.example.com:3389 \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret
```

4. Set a global configuration property using the `dsconfig` tool.

```
$ bin/dsconfig set-global-configuration-prop \
  --set configuration-server-group:group-one \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" \
  --bindPassword secret --no-prompt
```

5. Test the server group. In this example, enable the log publisher for each directory server in the group, `server-group` by using the `--applyChangeTo server-group` option.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:true --applyChangeTo server-group \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" \
  --bindPassword secret --no-prompt
```

6. View the property on the first directory server instance.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" --property enabled \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" \
  --bindPassword secret --no-prompt

Property : Value(s)
---------:---------
enabled  : true
```

View the property on the second directory server instance.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" --property enabled \
  --hostname server2.example.com --port 2389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret --no-
prompt

Property : Value(s)
---------:---------
enabled  : true
```

View the property on the third directory server instance.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled \
  --hostname server3.example.com --port 3389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret \
  --no-prompt

Property : Value(s)
---------:---------
enabled  : true
```

7. Test the server group by disabling the log publisher on the first directory server instance by using the `--applyChangeTo single-server`.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:disabled \
  --applyChangeTo single-server \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret \
  --no-prompt
```

   View the property on the first directory server instance. The first directory server instance should be disabled.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled \
  --hostname server1.example.com --port 1389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret \
  --no-prompt

Property : Value(s)
---------:---------
enabled  : false
```

8. View the property on the second directory server instance.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled \
  --hostname server2.example.com --port 2389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" --bindPassword secret \
  --no-prompt

Property : Value(s)
---------:---------
enabled  : true
```

9. View the property on the third directory server instance.

```
$ bin/dsconfig get-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --property enabled \
  --hostname server3.example.com --port 3389 \
  --bindDN "cn=admin,cn=administrators,cn=admin data" \
  --bindPassword secret --no-prompt

Property : Value(s)
---------:---------
enabled  : true
```

Configuring Client Connection Policies

Client connection policies help distinguish what portions of the DIT the client can access. They also enforce restrictions on what clients can do in the server. A client connection policy specifies criteria for membership based on information about the client connection, including client address, protocol, communication security, and authentication state and identity. The client connection policy does not control membership based on the type of request being made.

Every client connection is associated with exactly one client connection policy at any given time. A client connection policy is assigned to the client when the connection is established. The choice of which client connection policy to use will be reevaluated when the client attempts a bind to change its authentication state or uses the StartTLS extended operation to convert an insecure connection to a secure one. Any changes you make to the client connection policy do not apply to existing connections. The changes only apply to new connections.

Client connections are always unauthenticated when they are first established. If you plan to configure a policy based on authentication, you must define at least one client connection policy with criteria that match unauthenticated connections.

Once a client has been assigned to a policy, you can determine what operations they can perform. For example, your policy might allow only SASL bind operations. Client connection policies are also associated with one or more subtree views, which determine the portions of the DIT a particular client can access. For example, you might configure a policy that prevents users connecting over the extranet from accessing configuration information. The client connection policy is evaluated in addition to access control, so even a root user connecting over the extranet would not have access to the configuration information.

Understanding the Client Connection Policy

Client connection policies are based on two things:

- **Connection criteria**. The connection criteria are used in many areas within the server. They are used by the client connection policies, but they can also be used in other instances when the server needs to perform matching based on connection-level properties, such as filtered logging. A single connection can match multiple connection criteria definitions.

- **Evaluation order index**. If multiple client connection policies are defined in the server, then each of them must have a unique value for the `evaluation-order-index` property. The client connection policies are evaluated in order of ascending evaluation order index. If a client connection does not match the criteria for any defined client connection policy, then that connection will be terminated.

If the connection policy matches a connection, then the connection is assigned to that policy and no further evaluation occurs. If, after evaluating all of the defined client connection policies, no match is found, the connection is terminated.

When a Client Connection Policy is Assigned

A client connection policy can be associated with a client connection at the following times:

- When the connection is initially established. This association occurs exactly once for each client connection.
- After completing processing for a StartTLS operation. This association occurs at most once for a client connection, because StartTLS cannot be used more than once on a particular connection. You also may not stop using TLS while keeping the connection active.
- After completing processing for a bind operation. This association occurs zero or more times for a client connection, because the bind request can be processed many times on a given connection.

StartTLS and bind requests will be subject to whatever constraints are defined for the client connection policy that is associated with the client connection at the time that the request is received. Once they have completed, then subsequent operations will be subject to the constraints of the new client connection policy assigned to that client connection. This policy may or may not be the same client connection policy that was associated with the connection before the operation was processed. That is, any policy changes do not apply to existing connections and will be applicable when the client reconnects.

All other types of operations will be subject to whatever constraints are defined for the client connection policy used by the client connection at the time that the request is received. The client connection policy assigned to a connection never changes as a result of processing any operation other than a bind or StartTLS. So, the server will not re-evaluate the client connection policy for the connection in the course of processing an operation. For example, the client connection policy will never be re-evaluated for a search operation.

Setting Resource Limits

Your client connection policy can specify resource limits, helping to ensure that no single client monopolizes server resources. You can limit the total number of connections to a server from a particular client or from clients that match specified criteria. You can also limit the duration of the connection.

A client connection policy may only be used to enforce additional restrictions on a client connection. You can never use it to grant a client capabilities that it would not otherwise have.

Any change to any of these new configuration properties will only impact client connections that are assigned to the client connection policy after the change is made. Any connection associated with the client connection policy before the configuration change was made will continue to be subject to the configuration that was in place at the time it was associated with that policy.

TABLE 6-3. Resource Limiting Properties

Property	Description
maximum-concurrent-connections	Specifies the maximum number of client connections that can be associated with that client connection policy at any given time. The default value of zero indicates that no limit will be enforced. If the server already has the maximum number of connections associated with a client connection policy, then any attempt to associate another connection with that policy (e.g., newly-established connections or an existing connection that has done something to change its client connection policy, such as perform a bind or StartTLS operation) will cause that connection to be terminated.
terminate-connection	Specifies that any client connection for which the client connection policy is selected (whether it is a new connection or an existing connection that is assigned to the client connection policy after performing a bind or StartTLS operation) will be immediately terminated. This property can be used to define criteria for connections that you do not want to be allowed to communicate with the Directory Server.
maximum-connection-duration	Specifies the maximum length of time that a connection associated with the client connection policy can remain established to the Directory Server, regardless of the amount of activity on that connection. A value of "0 seconds" (default) indicates that no limit will be enforced. If a connection associated with the client connection policy has been established for longer than this time, then it will be terminated.
maximum-idle-connection-duration	Specifies the maximum length of time that a connection associated with the client connection policy can remain established with the Directory Server without any requests in progress. A value of "0 seconds" (default) indicates that no additional limit will be enforced on top of whatever idle time limit might already be in effect for an associated connection. If a nonzero value is provided, then the effective idle time limit for any client connection will be the smaller of the **`maximum-idle-connection-duration`** from the client connection policy and the idle time limit that would otherwise be in effect for that client. This property can be used to apply a further restriction on top of any value that may be enforced by the **`idle-time-limit`** global configuration property (which defines a default idle time limit for client connections) or the **`ds-rlim-idle-time-limit`** operational attribute (which may be included in a user entry to override the default idle time limit for that user).
maximum-operation-count-per-connection	Specifies the maximum number of operations that a client associated with the client connection policy will be allowed to request. A value of zero (default) indicates that no limit will be enforced. If a client attempts to request more than this number of operations on the same connection, then that connection will be terminated.
maximum-concurrent-operations-per-connection	Specifies the maximum number of operations that may be active at any time from the same client. This limit is only applicable to clients that use asynchronous operations with multiple outstanding requests at any given time. A value of zero (default) indicates that no limit will be enforced. If a client already has the maximum number of outstanding requests in progress and issues a new request, then that request will be delayed and/or rejected based on the value of the **`maximum-concurrent-operation-wait-time-before-rejecting`** property.

TABLE 6-3. Resource Limiting Properties

Property	Description
maximum-concurrent-operation-wait-time-before-rejecting	Specifies the maximum length of time that a client connection should allow an outstanding operation to complete if the maximum number of concurrent operations for a connection are already in progress when a new request is received on that connection. A value of “0 seconds” (default) indicates that any new requests received while the maximum number of outstanding requests are already in progress for that connection will be immediately rejected. If an outstanding operation completes before this time expires, then the server may be allowed to process that operation. If the time expires, the new request will be rejected.
allowed-request-control	Specifies the OIDs of the request controls that clients associated with the client connection policy will be allowed to use. If any allowed-request-control OIDs are specified, then any request which includes a control not in that set will be rejected. If no **`allowed-request-control`** values are specified (default), then any control whose OID is not included in the set of **`denied-request-control`** values will be allowed.
denied-request-control	Specifies the OIDs of the request controls that clients associated with the client connection policy will not be allowed to use. If there are any denied-request-control values, then any request containing a control whose OID is included in that set will be rejected. If there are no **`denied-request-control`** values (default), then any request control will be allowed if the **`allowed-request-control`** property is also empty, or only those controls whose OIDs are included in the set of **`allowed-request-control`** values will be allowed if at least one **`allowed-request-control`** value is provided.
allowed-filter-type	Specifies the types of components which may be used in filters included in search operations with a non-base scope that are requested by clients associated with the client connection policy. Any non-base scoped search request whose filter contains a component not included in this set will be rejected. The set of possible filter types include: • and • or • not • equality • sub-initial • sub-any • sub-final • greater-or-equal • less-or-equal • present • approximate-match • extensible-match By default, all filter types will be allowed. Also note that no restriction will be placed on the types of filters which may be used in searches with a base scope.

TABLE 6-3. Resource Limiting Properties

Property	Description
allow-unindexed-searches	Specifies whether clients associated with the client connection policy will be allowed to request searches which cannot be efficiently processed using the configured set of indexes. Note that clients will still be required to have the **`unindexed-search`** privilege, so this option will not grant the ability to perform unindexed searches to clients that would not have otherwise had that ability, but it may be used to prevent clients associated with the client connection policy from requesting unindexed searches when they might have otherwise been allowed to do so. By default, this has a value of "true", indicating that any client associated with the client connection policy that has the **`unindexed-search`** privilege will be allowed to request unindexed searches.
minimum-substring-length	Specifies the minimum number of bytes, which may be present in any **`subInitial`**, **`subAny`**, or **`subFinal`** element of a substring search filter component in a search with a non-baseObject scope. A value of one (which is the default) indicates that no limit will be enforced. This property may be used to prevent clients from issuing overly-vague substring searches that may require the Directory Server to examine too many entries over the course of processing the request.
maximum-search-size-limit	Specifies the maximum number of entries that may be returned from any single search operation requested by a client associated with this client connection policy. Note that this property only specifies a maximum limit and will never increase any limit that may already be in effect for the client via the **`size-limit`** global configuration property or the **`ds-rlim-size-limit`** operational attribute. A value of zero (default) indicates that no additional limit will be enforced on top of whatever size limit might already be in effect for an associated connection. If a nonzero value is provided, then the effective maximum size limit for any search operation requested by the client will be the smaller of the size limit from that search request, the **`maximum-search-size-limit`** from the client connection policy, and the size limit that would otherwise be in effect for that client.

TABLE 6-3. **Resource Limiting Properties**

Property	Description
maximum-search-time-limit	Specifies the maximum length of time that the server may spend processing any single search operation requested by a client associated with this client connection policy. Note that this property only specifies a maximum limit, and will never increase any limit which may already be in effect for the client via the **`time-limit`** global configuration property or the **`ds-rlim-time-limit`** operational attribute. A value of "0 seconds" (default) indicates that no additional limit will be enforced on top of whatever time limit might already be in effect for an associated connection. If a nonzero value is provided, then the effective time limit for any search operation requested by the client will be the smaller of the time limit from that search request, the **`maximum-search-time-limit`** from the client connection policy, and the time limit that would otherwise be in effect for that client.
maximum-search-lookthrough-limit	Specifies the maximum number of candidate entries that the server may examine while processing any single search operation requested by a client associated with this client connection policy. Note that this property only specifies a maximum limit and will never increase any limit which may already be in effect for the client via the **`lookthrough-limit`** global configuration property or the **`ds-rlim-lookthrough-limit`** operational attribute. A value of zero (default) indicates that no additional lookthrough limit will be enforced on top of whatever lookthrough limit might already be in effect for an associated connection. If a nonzero value is provided, then the effective lookthrough limit for any search operation requested by a client will be the smaller of the **`maximum-search-lookthrough-limit`** from the client connection policy and the lookthrough limit that would otherwise be in effect for that client.

Defining the Operation Rate

You can configure the maximum operation rate for individual client connections as well as collectively for all connections associated with a client connection policy. If the operation rate limit is exceeded, the Directory Server may either reject the operation or terminate the connection. You can define multiple rate limit values, making it possible to fine tune limits for both a long term average operation rate and short term operation bursts. For example, you can define a limit of one thousand operations per second and one million operations per day, which works out to an average of less than twelve operations per second, but with bursts of up to one thousand operations per second.

Rate limit strings should be specified as a maximum count followed by a slash and a duration. The count portion must contain an integer, and may be followed by a multiplier of **`k`** (to indicate that the integer should be interpreted as thousands), **`m`** (to indicate that the integer should be interpreted as millions), or **`g`** (to indicate that the integer should be interpreted as billions). The duration portion must contain a time unit of milliseconds (**`ms`**), seconds (**`s`**), minutes (**`m`**), hours (**`h`**), days (**`d`**), or weeks (**`w`**), and may be preceded by an integer to specify a quantity for that unit.

For example, the following are valid rate limit strings:

- 1/s (no more than one operation over a one-second interval)
- 10K/5h (no more than ten thousand operations over a five-hour interval)
- 5 m / 2 d (no more than five million operations over a two-day interval)

You can provide time units in many different formats. For example, a unit of seconds can be signified using s, sec, sect, second, and seconds.

Client Connection Policy Deployment Example

In this example scenario, we assume the following:

- Two external LDAP clients are allowed to bind to the directory server.
- Client 1 should be allowed to open only 1 connection to the server.
- Client 2 should be allowed to open up to 5 connections to the server.

Defining the Connection Policies

We need to set a per-client connection policy limit on the number of connections that may be associated with a particular client connection policy. We have to define at least two client connection policies, one for each of the two clients. Each policy must have different connection criteria for selecting the policy with which a given client connection should be associated.

Because the criteria is based on authentication, we must create a third client connection policy that applies to unauthenticated clients, because client connections are always unauthenticated as soon as they are established and before they have sent a bind request. Plus, clients are not required to send a bind request as their first operation.

Therefore, we define the following three client connection policies:

- Client 1 Connection Policy, which only allows client 1, with an evaluation order index of 1.
- Client 2 Connection Policy, which only allows client 2, with an evaluation order index of 2.
- Unauthenticated Connection Policy, which allows unauthenticated clients, with an evaluation order index of 3.

We define simple connection criteria for the Client 1 Connection Policy and the Client 2 Connection Policy with the following properties:

- The **`user-auth-type`** must not include **`none`**, so that it will only apply to authenticated client connections.

- The **`included-user-base-dn`** should match the bind DN for the target user. This DN may be full DN for the target user, or it may be the base DN for a branch that contains a number of users that you want treated in the same way.

To create more generic criteria that match more than one user, you could list the DNs of each of the users explicitly in the `included-user-base-dn` property. If there is a group that contains all of the pertinent users, then you could instead use the `[all|any|not-all|not-any]-included-user-group-dn` property to apply to all members of that group. If the entries for all of the users match a particular filter, then you could use the `[all|any|not-all|not-any]-included-user-filter` property to match them.

How the Policy is Evaluated

Whenever a connection is established, the server associates the connection with exactly one client connection policy. The server does this by iterating over all of the defined client connection policies in ascending order of the evaluation-order-index. Policies with a lower `evaluation-order-index` value will be examined before those with a higher `evaluation-order-index` value. The first policy that the server finds whose criteria match the client connection will be associated with that connection. If no client connection policy is found with criteria matching the connection, then the connection will be terminated.

So, in our example, when a new connection is established, the server first checks the connection criteria associated with the Client 1 Connection Policy because it has the lowest `evaluation-order-index` value. If it finds that the criteria do not match the new connection, the server then checks the connection criteria associated with the Client 2 Connection Policy because it has the second lowest evaluation order index. If these criteria do not match, the server finally checks the connection criteria associated with the Unauthenticated Connection Policy because it has the third lowest evaluation order index. It finds a match, so the client connection is associated with the Unauthenticated Connection Policy.

After the client performs a bind operation to authenticate to the server, then the client connection policies will be reevaluated. If client 2 performs the bind, then the Client 1 Connection Policy will not match but the Client 2 Connection Policy will, so the connection will be re-associated with that client connection policy. Whenever a connection is associated with a client connection policy, the server will check to see if the maximum number of client connections have already been associated with that policy. If so, then the newly-associated connection will be terminated.

For example, Client 1 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthorized Connection Policy. Client 1 then sends a bind request. The determination of whether the bind operation is allowed is made based on the constraints defined in the Unauthorized Connection Policy, because it is the client connection policy already assigned to the client connection. Once the bind has completed, then the server will reevaluate the client connection policy against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to the Client 1 Connection Policy.

Next, Client 2 opens a new connection. Because it is a new connection not yet associated with connection criteria, it is assigned to the Unauthorized Connection Policy. When Client 2 sends a bind request, the operation is allowed based on the constraints defined in the Unauthorized Connection Policy. Once the bind is complete, the client connection is evaluated against the connection criteria associated with Client 1 Connection Policy, because it has the lowest evaluation order index. The associated connection criteria do not match, so the client 2 connection is evaluated against the connection criteria associated with Client 2 Connection Policy, because it has the next lowest evaluation order index. The associated connection criteria match, so processing stops and the client connection is assigned to Client 2 Connection Policy.

Client 1 sends a search request. The Client 1 Connection Policy is used to determine whether the search operation should be allowed, because this is the client connection policy assigned to the client connection for client 1. The connection is not reevaluated, before or after processing the search operation.

To Configure a Client Connection Policy Using the Console

1. Open the Directory Server Management Console. Provide a username and password, and then click Login.

2. In the Core Server section, click Client Connection Policies. If you do not see Client Connection Policies on the menu, change the Object Types filter to Standard.

3. Click Add New to add a new policy.

4. Enter a Policy ID. If you want to base your new client connection policy on an existing policy, select it from the Template menu.

5. Configure the properties of the client connection policy. To enable the policy, select Enabled.

6. Enter the order in which you want the new policy to be evaluated in the Evaluation Order Index box, and then click Continue. A policy with a lower index is evaluated before a policy with a higher index. The Directory Server uses the first evaluated policy that applies to a client connection.

7. Select the connection criteria that match the client connection for this policy. Click View and edit to change the criteria. Click Select New to add new criteria. Select the operations allowed for clients that are members of this connection group. Use the Add and Remove buttons to make operations available to clients. Specify the extended operations that clients are allowed and denied to use.

8. Enter the type of authorization allowed and the SASL mechanisms that are allowed and denied in response to client requests.

9. Check the Include Backend Subtree Views check box if you want to automatically include the subtree views of backends configured in the Directory Server. You can also choose to include and exclude specific base DNs using the appropriate fields.

10. Once you have finished configuring the properties of your client connection policy, click "Confirm then Save" to review the `dsconfig` command equivalent and save your changes. Click Save Now to save your changes without first reviewing the `dsconfig` output.

Configuring a Client Connection Policy Using dsconfig

You can configure a client connection policy using the **`dsconfig`** tool in interactive mode from the command line. You can access the Client Connection Policy menu on the Standard objects menu.

To Configure a Client Connection Policy Using dsconfig

1. Use the **`dsconfig`** tool to create and configure a client connection policy. Specify the host name, connection method, port number, and bind DN as described in previous procedures.

   ```
   $ bin/dsconfig
   ```

2. On the UnboundID Directory Server configuration console main menu, change to the Standard objects menu by entering "o" and then entering "2" for the Standard menu.

3. On the UnboundID Directory Server configuration console main menu, enter the number associated with Client Connection Policy.

4. On the Client Connection Policy management menu, type the number corresponding to Create a new connection policy.

5. Enter **`n`** to create a new client connection policy from scratch.

6. Next, enter a name for the new client connection policy.

7. On the Enabled Property menu, select **`true`** to enable the connection policy.

8. On the "Evaluation-Order" Property menu, type a value between 0 and 2147483647 to set the evaluation order for the policy. A client connection policy with a lower evaluation-order will be evaluated before one with a higher number. For this example, type 9999.

9. On the Client Connection Policy menu, review the configuration. If you want to make any further modifications, enter the number corresponding to the property. Enter `f` to finish the creation of the client connection policy.

Note	Any changes that you make to the client connection policy do not apply to existing connections. They will only apply to new connections.

Restricting Directory Server Access Based on Client IP Address

Another common use case is to limit client access to the Directory Server. Two methods are available:

- **Connection Handlers**. You can limit the IP addresses using the LDAP or LDAPS connection handlers. The connection handlers provide an **`allowed-client`** property and a **`denied-client`** property. The **`allowed-client`** property specifies the set of allowable address masks that can establish connections to the handler. The `denied-client` property specifies the set of address masks that are not allowed to establish connections to the handler.

- **Client Connection Policies**. You can take a more fine-grained approach by restricting access by configuring a new Client Connection Policy, then create a new connection criteria and associate it with the connection policy. Connection criteria define sets of criteria for grouping and describing client connections based on a number of properties, including the protocol, client address, connection security, and authentication state for the connection. Each client connection policy may be associated with zero or more Connection Criteria, and server components may use Connection Criteria to indicate which connections should be processed and what kind of processing should be performed (e.g., to select connections and/or operations for filtered logging, or to classify connections for network groups).

To Restrict Directory Server Access Using the Connection Handlers

Use `dsconfig` to set the `allowed-client` property for the LDAP connection handler. You should specify the address mask for the range of allowable IP addresses that can establish connections to the Directory Server. You should also specify the loopback address, `127.0.0.1`, so that you will still be able to configure the server using the `dsconfig` tool on the local host.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAP Connection Handler" \
  --set "allowed-client:10.6.1.*" --set allowed-client:127.0.0.1 \
  --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

To Restrict Directory Server Access using Client Connection Policies

1. Create a simple connection criteria. The following example uses the `dsconfig` tool in non-interactive mode. It allows only the Directory Server’s IP address and loopback to have access.

```
$ bin/dsconfig set-connection-criteria-prop \
  --criteria-name allowed-ip-addrs \
  --add included-client-address:10.6.1.80 \
  --add included-client-address:127.0.0.1 \
  --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

2. Assign the criteria to the client connection policy. After you have run the following command, access is denied to remote IP addresses. The Directory Server does not require a restart.

```
$ bin/dsconfig set-client-connection-policy-prop \
  --policy-name new-policy \
  --set connection-criteria:allowed-ip-addrs \
  --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

3. Add a remote IP range to the criteria. For this example, add 10.6.1.*. Access from any remote servers is allowed. The Directory Server does not require a restart.

```
$ bin/dsconfig set-connection-criteria-prop \
  --criteria-name allowed-ip-addrs \
  --add "included-client-address:10.6.1.* \
  --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

4. To restore default behavior, you can just remove the criteria from the connection policy. The Directory Server does not require a restart. Remember to include the LDAP or LDAPS connection parameters (**hostname**, **port**, **bindDN**, **bindPassword**) with the **dsconfig** command.

```
$ bin/dsconfig set-client-connection-policy-prop \
  --policy-name new-policy --remove connection-criteria:allowed-ip-addrs \
  --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

To Automatically Authenticate Clients That Have A Secure Communication Channel

The Directory Server provides an option to automatically authenticate clients that have a secure communication channel (either SSL or StartTLS) and presented their own certificate. This option is disabled by default, but when enabled, the net effect will be as if the client issued a SASL EXTERNAL bind request on that connection.

This option will be ignored if the client connection is already authenticated (e.g., because it is using StartTLS but the client had already performed a bind before the StartTLS request). If the bind attempt fails, then the connection will remain unauthenticated but usable. If the client subsequently sends a bind request on the connection, then it will be processed as normal and any automatic authentication will be destroyed.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAPS Connection Handler" \
  --set "auto-authenticate-using-client-certificate:true"
```

Securing the Server with Lockdown Mode

The Directory Server provides tools to enter and leave lockdown mode if the server requires a security lockdown. In lockdown mode, only users with the **lockdown-mode** privilege can perform operations, while those without the privilege are rejected. Root users have this privilege by default; other administrators can be given this privilege.

The Directory Server can be manually placed into lockdown mode to perform some administrative operation while ensuring that other client requests are not allowed to access any data in the server. In addition, some configuration problems (particularly problems that could lead to inadvertent exposure of sensitive information, like an access control rule that cannot be properly parsed) cause the server to place itself in lockdown mode, so that an administrator can manually correct the problem. Lockdown mode does not persist across restarts, so a directory server can be taken out of lockdown mode by using either the **leave-lockdown-mode** tool or by restarting the server. If administrators want to start a server in lockdown mode, they can use the **`start-ds --lockdownMode`** option.

Any client request to the Directory Server in lockdown mode receives an "Unavailable" response.

To Manually Enter Lockdown Mode

Use **`enter-lockdown-mode`** to begin a lockdown mode.

```
$ bin/enter-lockdown-mode --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword password
```

To Start the Server in Lockdown Mode

Use the **`--lockdownMode`** option with the **`start-ds`** tool to start a server in lockdown mode.

```
$ bin/start-ds --lockdownMode
```

To Leave Lockdown Mode

Use **`leave-lockdown-mode`** to end lockdown mode.

```
$ bin/leave-lockdown-mode --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword spassword
```

Configuring the Maximum Shutdown Time

During shutdown, some database checkpointing and cleaning threads may remain active even after the default time period on systems with very large or very busy database backends. If checkpointing or cleaning is aborted prematurely, it could possible lead to significantly longer startup times for the Directory Server. The Directory Server provides an option for administrators to set the maximum time a shutdown process should take. When a shutdown process is initiated, the server begins stopping all of its internal components and waits up to 5 minutes for all threads to complete before exiting.

Administrators can use the **`dsconfig`** tool to increase the maximum shutdown time to allow database operations to complete.

To Configure the Maximum Shutdown Time

Use the **`dsconfig`** tool to increase the maximum shutdown time for your system. The following command increases the maximum shutdown time from 5 minutes to 6 minutes. The command allows time values of **`w`** (weeks), **`d`** (days), **`h`** (hours), **`m`** (minutes), **`s`** (seconds), **`ms`** (milliseconds).

Remember to include the LDAP or LDAPS connection parameters (host name, port, bindDN and bindDN password) with the **`dsconfig`** command.

```
$ bin/dsconfig set-global-configuration-prop --set "maximum-shutdown-time:6 m"
```

Note	The **`maximum-shutdown-time`** property can also be changed using the **`dsconfig`** tool in interactive mode. From the main menu, select Global Configuration, and then select the option to display advanced properties.

Working with Referrals

A referral is a redirection mechanism that tells client applications that a requested entry or set of entries is not present on the Directory Server but can be accessed on another server. Referrals can be used when entries are located on another server. The Directory Server implements two types of referrals depending on the requirement.

- **Referral on Update Plug-in**. The Directory Server provides a Referral on Update Plug-in to create any referrals for update requests (add, delete, modify, or modDN operations) on read-only servers. For example, given two replicated directory servers where one server is a master (read-write) and the other, a read-only server, you can configure a referral for any client update requests on the second directory server to point to the master server. If a client application sends an add request, for example, on the second directory server, the directory server responds with a referral that indicates any updates should be made on the master server. All read requests on the read-only server will be processed as normal. See "Working with Referrals" on page 126.

- **Smart Referrals**. The Directory Server supports smart referrals that map an entry or a specific branch of a DIT to an LDAP URL. Any client requests (reads or writes) targeting at or below the branch of the DIT will return a referral to the server designated in the LDAP URL.

Specifying LDAP URLs

Referrals use LDAP URLs to redirect a client application's request to another server. LDAP URLs have a specific format, described in RFC 4516 and require that all special characters be properly escaped and any spaces indicated as "%20". LDAP URLs have the following syntax:

```
ldap[s]://hostname:port/base-dn?attributes?scope?filter
```

where

- *ldap[s]* indicates the type of LDAP connection to the directory server. If the Directory Server connects over a standard, non-encrypted connection, then **`ldap`** is used; if it connects over SSL, then **`ldaps`** is used. Note that any search request initiated by means of an LDAP URL is anonymous by default, unless an LDAP client provides authentication.

- *hostname* specifies the host name or IP address of the directory server.

- *port* specifies the port number of the directory server. If no port number is provided, the default LDAP port (389) or LDAPS port (636) is used.

- *base-dn* specifies the distinguished name (DN) of an entry in the DIT. The directory server uses the base DN as the starting point entry for its searches. If no base DN is provided, the search begins at the root of the DIT.

- *attributes* specifies those attributes for which the directory server should search and return. You can indicate more than one attribute by providing a comma-separated list of attributes. If no attributes are provided, the search returns all attributes.

- *scope* specifies the scope of the search, which could be one of the following: *base* (only search the specified base DN entry), *one* (only search one level below the specified base DN), *sub* (search the base entry and all entries below the specified base DN). If no scope is provided, the server performs a base search.

- *filter* specifies the search filter to apply to entries within the scope of the search. If no filter is provided, the server uses "**`(objectclass=*)`**".

Creating Smart Referrals

You can create a smart referral by adding an entry with the **`referral`** and **`extensibleObject`** object classes or adding the object classes to a specific entry. The **`referral`** object class designates the entry as a referral object. The **`extensibleObject`** object class allows you to match the target entry by matching any schema attribute. The following example shows how to set up a smart referral if a portion of a DIT is located on another server.

To Create a Smart Referral

1. Create an LDIF file with an entry that contains the **`referral`** and **`extensibleObject`** object classes.

```
dn: ou=EngineeringTeam1,ou=People,dc=example,dc=com
objectClass: top
objectClass: referral
objectClass: extensibleObject
ou: Engineering Team1
ref: ldap://server2.example.com:6389/ou=EngineeringTeam1,ou=People,dc=exam-
ple,dc=com
```

2. On the first server, add the referral entry using the **ldapmodify** command.

```
$ bin/ldapmodify --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --defaultAdd --fileName referral-entry.ldif
```

3. Verify the addition by searching for a user.

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --baseDN ou=People,dc=example,dc=com "(uid=user.4)"
SearchReference(referralURLs={ldap://server2.example.com:6389/
ou=EngineeringTeam1,ou=People,dc=example,dc=com??sub?})
```

To Modify the Referral

Use **ldapmodify** with the **manageDSAIT** control to modify the **ref** attribute on the referral entry.

```
$ bin/ldapmodify --port 1389 --bindDN "uid=admin,dc=example,dc=com"
  --control manageDSAIT
dn: ou=EngineeringTeam1,ou=People,dc=example,dc=com
changetype: modify
replace: ref
ref: ldap://server3.example.com/ou=EngineeringTeam1,ou=People,dc=example,dc=com
```

To Delete a Referral

Use **ldapdelete** with the **manageDSAIT** control to delete the referral entry.

```
$ bin/ldapdelete --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --control manageDSAIT "ou=EngineeringTeam1,ou=People,dc=example,dc=com"
```

Configuring a Read-Only Directory Server

The UnboundID Directory Server provides a means to configure a hub-like, read-only directory server for legacy systems that require it. The read-only directory server participates in replication but cannot respond to any update requests from an external client. You can configure the Directory Server by setting the **writability** mode to **internal-only**, which makes the server operate in read-only mode. Read-only mode directory servers can process update operations from internal operations but reject any write requests from external clients. Because the Directory Server cannot accept write requests, you can configure the server to send a referral, which redirects a client's request to another master server. The client must perform the operation again on the server named in the referral.

Note	Many Server SDK extensions use the InternalConnection interface to process operations in the server, rather than issuing LDAP requests over the network. If an extension does so in response to an external update request, then any Directory Server using that extension will effectively respond to external update requests, even though the Directory Server is configured to operate in read-only mode, as described above. One possible workaround is to split the extension into two extensions, one for reads and one for writes, then disabling (or not deploying) the write-only extension when configuring a directory server in read-only mode.

To Configure a Read-Only Directory Server

1. Configure two replication servers. See “Replication Configuration” on page 383 for various ways to set up your servers.

2. On the second server, use the **`dsconfig`** command to set the writability mode of the server to internal-only.

```
$ bin/dsconfig set-global-configuration-prop \
  --set writability-mode:internal-only \
  --hostname server2.example.com \
  --port 2389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

3. Use the **`dsconfig`** tool to create a referral using an update plug-in. This command sets up the server to process read operations but redirects all write operations to another server. The following example sets a referral for a specific base DN.

```
$ bin/dsconfig create-plugin --plugin-name "Refer Updates" \
  --type referral-on-update --set enabled:true \
  --set referral-base-url:ldap://server3.example.com:3389/ \
  --set "base-dn:dc=example,dc=com" \
  --hostname server2.example.com \
  --port 2389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

4. Modify an attribute on the second server to view the referral. The administrator will need to manually update the entry on the third server.

```
$ bin/ldapmodify -p 2389 -D "cn=Directory Manager" -w secret
dn: uid=user.12,ou=People,dc=example,dc=com
changetype:modify
```

```
replace:telephoneNumber
telephoneNumber: +1 408 555 1155
```

Note	To remove or modify the referral itself, you must use the Manage DSAIT control as defined in RFC 3296. The control tells the Directory Server to treat referral entries (i.e., smart referrals) as regular entries rather than generating a referral when targeting them. You can add the `--control managedsait` option with `ldapdelete` or `ldapmodify` to work with the referral itself.

Configuring HTTP Access for the Directory Server

Although most clients communicate with the UnboundID Directory Server using LDAP, the server also provides support for an HTTP connection handler that uses Java servlets to serve content to clients over HTTP. UnboundID offers an extension that uses this HTTP connection handler to add support for the Simple Cloud Identity Management (SCIM) protocol. Third-party developers can also use the UnboundID Server SDK to write extensions that leverage this HTTP support.

The following sections describe how to configure HTTP servlet extensions and how to configure an HTTP connection handler.

Configuring HTTP Servlet Extensions

To use the HTTP connection handler, you must first configure one or more servlet extensions. Servlet extensions are responsible for obtaining Java servlets (using the Java Servlet 3.0 specification as described in JSR 315) and registering them to be invoked using one or more context paths. If you plan to deploy the SCIM extension, then you should follow the instructions included in the documentation for the extension. For custom servlet extensions created using the Server SDK, the process varies based on whether you are using a Java-based or Groovy-scripted extension.

Java-Based Servlet Extensions

For Java-based extensions, first use the Server SDK to create and build the extension bundle as described in the Server SDK documentation. Then, install it using the `manage-extension` tool as follows:

```
$ bin/manage-extension --install/path/to/extension.zip
```

The Java-based extension may then be configured for use in the server using `dsconfig` or the web-based administration console. Create a new Third Party HTTP Servlet Extension, specifying the fully-qualified name for the `HTTPServletExtension` subclass in the `extension-`

`class` property, and providing any appropriate arguments in the `extension-argument` property.

Groovy-Scripted Extensions

For Groovy-scripted extensions, place the necessary Groovy scripts in the appropriate directory (based on the package for those scripts) below the `lib/groovy-scripted-extensions` directory. Then, create a new Groovy Scripted HTTP Servlet Extension, specifying the fully-qualified Groovy class name for the `script-class` property, and providing any appropriate arguments in the `script-argument` property.

Configuring HTTP Operation Loggers

Servlet extensions may write error log messages in the same way as any other kind of server component, but interaction with HTTP clients will not be recorded in the server access log. However, if a servlet extension performs internal operations to interact with data held in the directory server, then those operations may be captured in the access log. To capture information about communication with HTTP clients, you must configure one or more HTTP operations loggers.

By default, the server comes with a single HTTP operation logger implementation, which uses the standard W3C common log format. It records messages in a format like the following:

```
127.0.0.1 - - [01/Jan/2012:00:00:00 -0600]"GET/hello HTTP/1.1" 200 113]
```

The log message contains the following elements:

- The IP address of the client that issued the request.
- The RFC 1413 (ident) identity of the client. Because the ident protocol information is not typically provided by HTTP clients, the HTTP connection handler never requests this information. This identity will always be represented as a dash to indicate that the information is not available.
- The authenticated identity determined for the request by HTTP authentication, or a dash to indicate that the request was not authenticated.
- The time that the request was received.
- The request issued by the client, including the HTTP method, path and optional query string, and the HTTP protocol version used.
- The integer representation of the HTTP status code for the response to the client.
- The number of bytes included in the body of the response to the client.

To configure an HTTP operation logger to use this common log format, create a new instance of a Common Log File HTTP Operation Log Publisher object, specifying the path and name for the active log file to be written and the rotation and retention policies that should be used to manage the log files. In general, properties for Common Log File HTTP Operation Log Publisher objects have the same meaning and use as they do for other kinds of loggers.

You can use the Server SDK to create custom Java-based or Groovy-scripted HTTP operation loggers using the Third Party HTTP Operation Log Publisher and Groovy Scripted HTTP Operation Log Publisher object types.

Example HTTP Log Publishers

When troubleshooting HTTP Connection Handler issues, administrators should first look at the logs to determine any potential problems. The following section shows some **`dsconfig`** commands and their corresponding log files.

Default Configuration Example

You can configure a default detailed HTTP Log Publisher with default log rotation and retention policies as follows:

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Detailed Access Logger" \
  --type detailed-http-operation \
  --set enabled:true \
  --set log-file:logs/http-detailed-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy" \
  --set "retention-policy:Size Limit Retention Policy"
```

The corresponding log file provides access information with the following elements:

- The time that the request was received.
- The request ID issued by the client, including the IP address, port, HTTP method, and URL.
- The authorization type, request contect type, and status code.
- The respone content length.
- The redirect URI
- The response content type.

The HTTP log file is shown as follows:

```
[23/Feb/2012:01:19:45 -0600] RESULT requestID=4300604 from="10.5.1.10:53269"
method="GET" url="https://10.2.1.113:443/Gimel/Users/uid=user.402914,ou=Peo-
ple,dc=gimel" authorizationType="Basic" requestContentType="application/json"
statusCode=200 etime=4.145 responseContentLength=1530 redirectURI="https://x2270-
11.example.lab:443/Gimel/Users/uid=user.402914,ou=people,dc=gimel"
responseContentType="application/json"
[23/Feb/2012:01:19:45 -0600] RESULT requestID=4300605 from="10.5.1.10:53269"
method="PUT" url="https://10.2.1.113:443/Gimel/Users/uid=user.207585,ou=peo-
ple,dc=gimel" authorizationType="Basic" requestContentType="application/json"
statusCode=200 etime=4.872 responseContentLength=1532 redirectURI="https://x2270-
11.example.lab:443/Gimel/Users/uid=user.207585,ou=people,dc=gimel"
responseContentType="application/json"
[23/Feb/2012:11:31:18 -0600] RESULT requestID=4309872 from="10.5.1.10:3
```

Configuration with Request/Response Header Names and Values

The following example adds request/response header names and values, including the "Content-Type" request header, which is normally suppressed by default.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "HTTP Detailed Access Logger" \
  --set log-request-headers:header-names-and-values \
  --remove suppressed-request-header-name:Content-Type \
  --set log-response-headers:header-names-and-values
```

Using the previous dsconfig example command, the server generates a log showing a query request by a SCIM Server using the property, **`scim-query-rate`**.

```
[23/Feb/2012:11:39:41 -0600] RESULT requestID=4665307 from="10.5.0.20:56044"
method="GET" url="https://10.2.1.113:443/Beth/Users?attributes=userName,title,
emails,urn:scim:schemas:extension:custom:1.0:descriptions,urn:scim:schemas: exten-
sion:enterprise:1.0:manager,groups,urn:scim:schemas:extension:custom:1.0: blob&fil-
ter=userName+eq+%22user.18935%22" requestHeader="Host: x2270-11.example.lab:443"
requestHeader="Accept: application/json" requestHeader="Content-Type: application/
json"  requestHeader="Connection: keep-alive" requestHeader="User-Agent: Wink Client
v1.1.2"  authorizationType="Basic" requestContentType="application/json" status-
Code=200 etime=140.384 responseContentLength=11778 responseHeader="Access-Control-
Allow-Credentials: true" responseContentType="application/json"
```

Another log example shows an example user creation event. The client is **`curl`**.

```
[23/Feb/2012:11:50:11 -0600] RESULT requestID=4802791 from="10.8.1.229:52357"
method="POST" url="https://10.2.1.113:443/Aleph/Users/" requestHeader="Host: x2270-
11.example.lab" requestHeader="Expect: 100-continue" requestHeader="Accept: applica-
tion/xml" requestHeader="Content-Type: application/xml" requestHeader="User-Agent:
curl/7.21.4 (universal-apple-darwin11.0) libcurl/7.21.4 OpenSSL/0.9.8r zlib/1.2.5"
authorizationType="Basic" requestContentType="application/xml" requestContent-
Length=1773 statusCode=201 etime=11.598 responseContentLength=1472 redirec-
tURI="https://x2270-11.example.lab:443/Aleph/Users/b2cef63c-5e46-11e1-974b-
60334b1a0d7a" responseContentType="application/xml"
```

The final example shows a user deletion request. The client is the UnboundID Synchronization Server.

```
[23/Feb/2012:11:38:06 -0600] RESULT requestID=4610261 from="10.5.1.114:34558"
method="DELETE" url="https://10.2.1.113:443/Aleph/Users/b8547525-24e0-41ae-b66b-
0b441800de70" requestHeader="Host: x2270-11.example.lab:443" requestHeader="Accept:
application/json" requestHeader="Content-Type: application/json" requestHeader="Con-
nection: keep-alive" requestHeader="User-Agent: UnboundID-Sync-3.2.2.0 (Build
20120222173845Z, Revision 11281)" authorizationType="Basic" requestContentType="appli-
cation/json" statusCode=200 etime=10.615 responseContentLength=0
```

Configuring HTTP Connection Handlers

HTTP connection handlers are responsible for managing the communication with HTTP clients and invoking servlets to process requests from those clients. Each HTTP connection handler must be configured with one or more HTTP servlet extensions and zero or more HTTP operation log publishers.

The configuration properties available for use with an HTTP connection handler include:

- **`listen-address.`** Specifies the address on which the connection handler will listen for requests from clients. If not specified, then requests will be accepted on all addresses bound to the system.
- **`listen-port.`** Specifies the port on which the connection handler will listen for requests from clients. Required.
- **`use-ssl`**. Indicates whether the connection handler will use SSL/TLS to secure communications with clients (whether it uses HTTPS rather than HTTP). If SSL is enabled, then key-manager-provider and trust-manager-provider values must also be specified.
- **`http-servlet-extension`**. Specifies the set of servlet extensions that will be enabled for use with the connection handler. You can have multiple HTTP connection handlers (listening on different address/port combinations) with identical or different sets of servlet extensions. At least one servlet extension must be configured.
- **`http-operation-log-publisher`**. Specifies the set of HTTP operation log publishers that should be used with the connection handler. By default, no HTTP operation log publishers will be used.
- **`key-manager-provider`**. Specifies the key manager provider that will be used to obtain the certificate presented to clients if SSL is enabled.
- **`trust-manager-provider`**. Specifies the trust manager provider that will be used to determine whether to accept any client certificates presented to the server.
- **`num-request-handlers`**. Specifies the number of threads that should be used to process requests from HTTP clients. These threads are separate from the worker threads used to process other kinds of requests. The default value of zero means the number of threads will be automatically selected based on the number of CPUs available to the JVM.

To Configure the HTTP Connection Handler and Logger

An HTTP connection handler has two dependent configuration objects: one or more HTTP servlet and optionally, an HTTP log publisher. The HTTP servlet extension and log publisher must be configured prior to configuring the HTTP connection handler. The log publisher is optional but in most cases, you want to configure one or more logs to troubleshoot any issues with your HTTP connection.

1. The first step is to configure your HTTP servlet extension. The following example uses the **`ExampleHTTPSerletExtension`** in the Server SDK.

```
$ bin/dsconfig create-http-servlet-extension \
  --extension-name "Hello World Servlet" \
  --type third-party \
  --set "extension-class:com.unboundid.directory.sdk.examples.ExampleHTTPServletExtension" \
  --set "extension-argument:path=/" \
  --set "extension-argument:name=example-servlet"
```

2. Next, configure one or more HTTP log publishers. The following example configures two log publishers: one for common access; the other, detailed access. Both log publishers use the default configuration settings for log rotation and retention.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Common Access Logger" \
  --type common-log-file-http-operation \
  --set enabled:true \
  --set log-file:logs/http-common-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"

$ bin/dsconfig create-log-publisher \
  --publisher-name "HTTP Detailed Access Logger" \
  --type detailed-http-operation --set enabled:true \
  --set log-file:logs/http-detailed-access \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

3. Configure the HTTP connection handler by specifying the HTTP servlet extension and log publishers. Note that some configuration properties can be later updated on the fly while others, like **`listen-port`**, require that the HTTP connection handler be disabled, then re-enabled for the change to take effect.

```
$ bin/dsconfig create-connection-handler \
  --handler-name "Hello World HTTP Connection Handler" \
  --type http \
  --set enabled:true \
  --set listen-port:8443 \
  --set use-ssl:true \
  --set "http-servlet-extension:Hello World Servlet" \
  --set "http-operation-log-publisher:HTTP Common Access Logger" \
  --set "http-operation-log-publisher:HTTP Detailed Access Logger" \
  --set "key-manager-provider:JKS" \
  --set "trust-manager-provider:JKS"
```

4. By default, the HTTP connection handler has an advanced monitor entry property, **`keep-stats`**, that is set to TRUE by default. You can monitor the connection handler using the **`ldapsearch`** tool.

```
$ bin/ldapsearch --baseDN "cn=monitor" \
  "(objectClass=ds-http-connection-handler-statistics-monitor-entry)"
```

Working with the Referential Integrity Plug-in

Referential integrity is a plug-in mechanism that maintains the DN references between an entry and a group member attribute. For example, if you have a group entry consisting of member attributes specifying the DNs of printers, you can enable the referential integrity plug-in to ensure that the group entry is automatically removed if a printer entry is removed from the directory server.

The Referential Integrity Plug-in is disabled by default. When enabled, the plug-in performs integrity updates on the specified attributes (for example, member or uniquemember) after a delete, modify DN, or a rename (i.e., subordinate modifyDN) operation is logged to the **`logs/referint`** file. If an entry is deleted, the plug-in checks the log file and makes the corresponding change to the associated group entry.

Three important points about the Referential Integrity Plug-in:

- All specified attributes that are configured for Referential Integrity must be indexed.
- On replicated servers, the Referential Integrity Plug-in configuration is not propagated to other replicas; therefore, you must manually enable the plug-in on each replica.
- The plug-in settings must also be identical on all machines.

To Enable the Referential Integrity Plug-in

1. Determine the attributes needed for your system. By default, the **`member`** and the **`uniquemember`** attributes are set for the plug-in.
2. Run the **`dsconfig`** tool to enable the Referential Integrity Plug-in.

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" --set
enabled:true --hostname example.server.com --port 389 --bindDN "uid=admin,dc=exam-
ple,dc=com" --bindDNPassword secret
```

Working with the UID Unique Attribute Plug-in

Attribute uniqueness is a plug-in mechanism that ensures the uniqueness of an attribute value in a DIT or subtree. The plug-in checks for uniqueness prior to any add, modify, or modify DN operation and determines if the newly added or updated entry will result in an LDAP violation due to an identical **`uid`** attribute value.

The plug-in is disabled by default as it can affect performance in heavy write load environments. Once the plug-in is enabled, it does not check for attribute uniqueness on existing entries, but only on new ADD, MODIFY, or MODDN operations.

For replicated environments, you can use the plug-in on a naming attribute. You should check that it is enabled for the same attribute on all of the masters.

To Enable the UID Unique Attribute Plug-in

1. Determine which attributes must be unique in your data.
2. Run the **`dsconfig`** tool to enable the plug-in. By default, the plug-in type property is set to **`postsynchronizationadd`**, **`postsynchronizationmodify`**, **`postsynchronizationmodi-`**

fydn, **preoperationadd**, **preoperationmodify**, and **preopertionmodifydn**. If you want to set one plug-in type, use the **--set plugin-type:<operation-type>** option. For example, use **--set plugin-type:preoperationadd** with the following command if you only want to check for attribute uniqueness prior to ADD operation.

```
$ bin/dsconfig set-plugin-prop --plugin-name "UID Unique Attribute" --set
enabled:true --port 389 --bindDN "uid=admin,dc=example,dc=com" --bindDNPassword
secret
```

Managing Directory Server Extensions

You can create extensions that use the Server SDK to extend the functionality of your Directory Server. Extension bundles are installed from a zip archive or a file system directory. You can use the **manage-extension** tool to install any extension that is packaged using the extension bundle format. This tool can be used with any HTTP servlet extension. It first opens and loads the extension bundle, confirms the correct extension to install, stops the server if necessary, copies the bundle to the server install root, and restarts the server.

Run the `manage-extension` tool to install and copy the files. For example, you can install the SCIM extension, **scim-extension-1.1.0**, as follows:

```
$ bin/manage-extension --install scim-extension-1.1.0
```

7 Managing Data

Overview

The UnboundID Directory Server supports the LDAP version 3 protocol and provides a full set of LDAP utilities for server administration as well as additional tools to efficiently manage the Directory Server and its data.

This chapter presents the following topics and procedures:

Running Task-Based Utilities

The Directory Server has a Tasks subsystem that allows you to schedule basic operations, **`such as import-ldif`**, **`export-ldif`**, **`backup`**, **`restore`**, **`start-ds`**, and **`stop-ds`**. All task-based utilities require the **`--task`** option that explicitly indicates the utility is intended to run as a task rather than in offline mode.

The **import-ldif**, **export-ldif**, **backup**, **restore**, **start-ds**, **stop-ds** utilities have the following task-based utilities (an example can be seen in “To Schedule an Import Using Import-ldif” on page 45):

TABLE 7-1. Task-based Tool Options

Option	Description
--task	Indicates that the tool is invoked as a task. The **--task** argument is optional but it will be required in a future revision of the Directory Server. If a tool is invoked as a task without this **--task** argument, then a warning message will be displayed recommending that it be used. If the **--task** argument is provided but the tool was not given the appropriate set of authentication arguments to the server, then an error message will be displayed and the tool will exit with an error.
--start	Indicates the date and time, expressed in the format '**YYYYMMDDhhmmss**', when the operation starts when scheduled as a server task. A value of '0' causes the task to be scheduled for immediate execution. When this option is used, the operation is scheduled to start at the specified time, after which this utility will exit immediately.
--dependency	Specifies the ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution. This option can be used multiple times in a single command.
--failedDependencyAction	Specifies the action this task will take should one of its dependent tasks fail. The value must be one of the following: **PROCESS**, **CANCEL**, **DISABLE**. If not specified, the default value is **CANCEL**. This option can be used multiple times in a single command.
--completionNotify	Specifies the email address of a recipient to be notified when the task completes. This option can be used multiple times in a single command.
--errorNotify	Specifies the email address of a recipient to be notified if an error occurs when this task executes. This option can be used multiple times in a single command.

Importing Data

The UnboundID Directory Server provides initialization mechanisms to import or export database files. The **import-ldif** command-line tool imports data from an LDAP Data Interchange Format (LDIF) file. The data imported by the **import-ldif** command can include all or a portion of the entries (a subset of the entries or a subset of the attributes within entries or both) contained in the LDIF file. The command also supports importing data that has been compressed, encrypted or digitally signed or both.

The **import-ldif** utility can be run with the server offline or online. If the server is online, administrators can initiate the import from a local or remote client. The LDIF file that contains the import data must exist on the server system. During an online import, the target database repository, or backend, will be removed from service and data held in that backend will not be available to clients.

The **import-ldif** tool has been modified to help guard against accidental overwriting of existing backend data with the addition of the **-r/--overwriteExistingEntries** option. This option must be present when performing an import into a backend with a branch that already contains entries (although the option is not needed if a branch contains just a single base entry). Without this option the tool will exit with an error message indicating the backend and branch containing entries. For Directory Server versions after 2.2.0, existing scripts that perform an **`import-ldif`** may need to be modified to include this option in order to function as before.

Note	The **`export-ldif`** tool works in an analogous manner as the **`import-ldif`** tool, except it is used to export data from the server to an outputted LDIF file. Features that filter attributes or branches are available with the **`export-ldif`** tool. Run **`export-ldif --help`** to view the available options.

Validating an LDIF File

Prior to importing data, you can validate an import file using the directory server's **`validate-ldif`** tool. When run, the tool binds to the directory server, locally or remotely, and validates the LDIF file to determine whether it violates the server's schema. Those elements that do not conform to the schema will be rejected and written to standard output. You can specify the path to the output file to which the rejected entries are written and the reasons for their rejection. The **`validate-ldif`** tool works with regular non-compressed LDIF files or gzip-compressed LDIF files.

To process large files faster, you can also set the number of threads for validation. The tool also provides options to skip specified schema elements if you are only validating certain items, such as attributes only. Use the **`--help`** option to view the arguments.

To Validate an LDIF File

Use the **`validate-ldif`** tool to validate an LDIF file. Make sure the server is online before running this command.

```
$ bin/validate-ldif --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --ldifFile /path/to/data.ldif \
  --rejectFile rejectedEntries

1 of 200 entries (0 percent) were found to be invalid.
1 undefined attributes were encountered.
Undefined attribute departmentname was encountered 1 times.
```

Tracking Skipped and Rejected Entries

During import, entries can be skipped if they do not belong in the specified backend, or if they are part of an excluded base DN or filter. The **`--skipFile {path}`** argument can be used on the command line to indicate that any entries that are skipped should be written to a specified file. You can add a comment indicating why the entries were skipped.

Similarly, the **`--rejectFile {path}`** argument can be added to obtain information about which entries were rejected and why. An entry can be rejected if it violates the server's schema constraints, if its parent entry does not exist, if another entry already exists with the same DN, or if it was rejected by a plug-in.

Running an Offline LDIF Import

You can run the **`import-ldif`** tool offline to import LDIF data encoded with the UTF-8 character set. This data can come from LDIF files, compressed LDIF files (GZIP format), or from data generated using a MakeLDIF template. You do not need to authenticate as a directory administrator when performing offline LDIF imports.

Note	The **`export-ldif`** tool works in an analogous manner as the **`import-ldif`** tool, except it is used to export data from the server to an outputted LDIF file. Features that filter attributes or branches are available with the **`export-ldif`** tool. Run **`export-ldif --help`** to view the available options.

To Perform an Offline LDIF Import

Use the **`import-ldif`** command to import data from an LDIF file. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

```
$ bin/import-ldif --backendID userRoot --ldifFile /path/to/data.ldif \
  --rejectFile /path/to/reject.ldif --skipFile /path/to/skip.ldif
```

To Perform an Offline LDIF Import Using a Compressed File

Use the **`import-ldif`** command to import data from a compressed gzip formatted file. You must also use the **`--isCompressed`** option to indicate that the input file is compressed. Make sure the Directory Server is offline before running this command. Do not specify any connection arguments when running the command.

```
$ bin/import-ldif --backendID userRoot --isCompressed \
  --ldifFile /path/to/data.gz --rejectFile /path/to/reject.ldif \
  --skipFile /path/to/skip.ldif
```

To Perform an Offline LDIF Import Using a MakeLDIF Template

Use the **`import-ldif`** command to import data from a MakeLDIF template, which is located in the **`<directory-instance>/config/MakeLDIF`**. Make sure the directory server is offline before running this command. Do not specify any connection arguments when running the command. The following command uses the standard data template and generates 10,000 sample entries, and then imports the file to the server.

```
$ bin/import-ldif --backendID userRoot --templateFile config/MakeLDIF/example.template
```

Running an Online LDIF Import

Administrators can run LDIF imports while the server is online from another remote server. The online import resembles the offline import, except that you must provide information about how to connect and authenticate to the target server. You can schedule the import of an LDIF file to occur at a particular time using the **`--task`** and **`--start YYYYMMDDhhmmss`** options of the **`import-ldif`** tool.

You can also specify email addresses for users that should be notified whenever the import process completes (regardless of success or failure, or only if the import fails). See "Working with the SMTP Account Status Notification Handler" on page 495 to set up SMTP notifications.

To Perform an Online LDIF Import

Use the **`import-ldif`** tool to import data from an LDIF. Make sure the Directory Server is online before running this command.

```
$ bin/import-ldif --task --hostname server.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --backendID userRoot --ldifFile /path/to/data.ldif
```

To Schedule an Online LDIF Import

1. Use the **`import-ldif`** tool to import data from an LDIF file at a scheduled time. To specify a time in the UTC time zone, include a trailing "Z". Otherwise, the time will be treated as a local time in the time zone configured on the server. Make sure the Directory Server is online before running this command.

```
$ bin/import-ldif --task --hostname server.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --backendID userRoot --ldifFile /path/to/data.ldif \
  --start 20111025010000 \
  --completionNotify import-complete@example.com \
  --errorNotify import-failed@example.com

Import task 2011102417321510 scheduled to start Oct 26, 2011 1:00:00 AM CDT
```

2. Confirm that you successfully scheduled your import task using the **`manage-tasks`** tool to view a summary of all tasks on the system.

```
$ bin/manage-tasks --summary --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt

  ID                Type    Status
  ------------------------------------------
  2011102417321510  Import  Waiting on start time
```

3. Use the **`manage-tasks`** tool to monitor the progress of this task. Use the task ID of the import task. If you cannot find the task ID, use the **`--summary`** option to view a list of all tasks scheduled on the directory server.

```
$ bin/manage-tasks --info 2011102417321510 \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt

  Task Details
  -----------------------------------------------------
  ID                         2011102417321510
  Type                       Import
  Status                     Waiting on start time
  Scheduled Start Time       Oct 26, 2011 1:00:00 AM CDT
  Actual Start Time
  Completion Time
  Dependencies               None
  Failed Dependency Action   None
  Email Upon Completion      admin@example.com
  Email Upon Error           admin@example.com

  Import Options
  ----------------------------
  LDIF File       /path/to/data.ldif
  Backend ID      userRoot
```

To Cancel a Scheduled Import

Use the **`manage-tasks`** tool to cancel the scheduled task.

```
$ bin/manage-tasks --no-prompt --cancel 2011102417321510 --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
```

Appending Entries to an Existing DIT

The **`import-ldif`** command no longer supports an **`--append`** option. The one method to import and append a number of entries to the directory server while preserving operational attributes, **`such`** as **`ds-create-time`** or **`modifiersName`**, is to use the **`Ignore No User Modification`** control. The **`Ignore No User Modification`** control allows modification of certain attributes that have the No User Modification constraint. Special care should be used with this control.

To Append Entries to an Existing DIT

- Use **`ldapmodify`** with the **`Ignore No User Modification`** control (i.e., the OID is 1.3.6.1.4.1.30221.2.5.5).

```
$ bin/ldapmodify --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --control 1.3.6.1.4.1.30221.2.5.5 --filename change-record.ldif
```

Filtering Data Imports

The **`import-ldif`** command provides a way to either include or exclude specific attributes or entries during an import. The arguments are summarized in Table 7-1.

TABLE 7-2. **Inclusion and Exclusion Arguments for import-ldif**

Argument	Description
--includeBranch	Base DN of a branch to include in the LDIF import (can be specified multiple times)
--excludeBranch	Base DN of a branch to exclude from the LDIF import (can be specified multiple times)
--includeAttribute	Attribute to include in the LDIF import (can be specified multiple times)
--excludeAttribute	Attribute to exclude from the LDIF import (can be specified multiple times)
--includeFilter	Filter to identify entries to include in the LDIF import (can be specified multiple times)
--excludeFilter	Filter to identify entries to exclude from the LDIF import (can be specified multiple times)

Encrypting LDIF Exports and Importing Encrypted LDIF Files

The Directory Server provides features to encrypt data during an LDIF export using the **`export-ldif --encryptLDIF`** option and to allow the encrypted LDIF file to be imported onto the same instance or another server in the same replication topology using the **`import-ldif --isEncrypted`** option. Both encrypted export and import features require online access to the **`ads-truststore`** backend, which stores the server instance key and is replicated to all other servers in the topology. These features are only available when running the tools online as a task using the **`--task`** argument along with the arguments to connect and authenticate to the Directory Server (e.g., **`hostname`**, **`port`**, **`bindDN`**, and **`bindPassword`**).

The Directory Server also provides an additional argument that digitally signs the contents of the LDIF file, which ensures that the content has not been altered since the export. To digitally sign the contents of the exported LDIF file, use the **`--sign`** option. This option also requires that it be run as an online task. Note that there is not much added benefit to both signing and encrypting the same data, since encrypted data cannot be altered without destroying the ability to decrypt it.

Important	Because the tools require online access to the **`ads-truststore`** backend, you must ensure that the **`ads-truststore`** backend along with the **`adminRoot`** backend be properly backed up, so that the server instance key can be accessed to decrypt or encrypt the LDIF data as well as check the digital-signature of its contents. If signature validation fails, the server will enter lock-down mode.

To Encrypt an LDIF Export

1. Run **`export`**-ldif tool as a task to encrypt the data during an export to an output LDIF file. The following command runs an online export of the **`userRoot`** backend, and encrypts the

file when written to an output file called **data.ldif.** The **--task** option plus the LDAP connection arguments (e.g., **hostname**, **port**, **bindDN**, and **bindPassword**) are required to run the operation as an online task.

```
$ bin/export-ldif --task --hostname server1 --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --backendID userRoot --ldifFile /path/to/data.ldif --encryptLDIF
```

To Import an Encrypted LDIF File

An encrypted LDIF file can be imported into the same instance from which it was exported, or into any other server in the same replication topology with that instance. You cannot import an encrypted LDIF file into a server that is not in some way connected to the instance from which it was exported.

- Run the **import-ldif** tool to import the encrypted LDIF file from the previous example. The following command requires that it be run as an online task; therefore, you must include the **--task** option and the LDAP connection parameters (e.g., **hostname**, **port**, **bindDN**, and **bindPassword**). The command imports the **data.ldif** file, decrypts the contents while overwriting the existing contents to the **userRoot** backend. The command requires the **--isEncrypted** option, which instructs the tool that the LDIF file is encrypted.

```
$ bin/import-ldif --task --hostname server1 --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --backendID userRoot --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries --isEncrypted
```

To Export a Signed LDIF File

1. Run **export**-ldif tool as a task to digitally sign the data during an export to an output LDIF file. The following command runs an online export of the **userRoot** backend, and signs the content when written to an output file called **data.ldif.** The **--task** option plus the LDAP connection arguments (e.g., **hostname**, **port**, **bindDN**, and **bindPassword**) are required to run the operation as an online task.

```
$ bin/export-ldif --task --hostname server1 --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --backendID userRoot --ldifFile /path/to/data.ldif --sign
```

To Import a Signed LDIF File

- Run the **import-ldif** tool to import the signed LDIF file from the previous example. The following command requires that it be run as an online task; therefore, you must include the **--task** option and the LDAP connection parameters (e.g., **hostname**, **port**, **bindDN**, and **bindPassword**). The command imports the **data.ldif** file, checks the signature of the contents while overwriting the existing contents to the **userRoot** backend. The command requires the **--isSigned** option, which instructs the tool that the contents of the LDIF file is signed.

```
$ bin/import-ldif --task --hostname server1 --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword password \
  --backendID userRoot --ldifFile /path/to/data.ldif \
  --overwriteExistingEntries --isSigned
```

Getting the Number of Entries in a Subtree

To obtain the number of entries in a subtree, you can use one of three methods depending on your directory size:

- Run **`ldapsearch`** using the base DN of the **`cn=monitor`** subtree to determine the number of entries.

  ```
  $ bin/ldapsearch --port 389 --bindDN "uid=admin,dc=example,dc=com" \
    --bindPassword password --baseDN "cn=userRoot Backend,cn=monitor" \
    "(objectclass=*)" ds-backend-entry-count
  ```

- Set up VLV indexes, and then run a search using the indexes to see the resulting numbers. See “Working with Indexes” on page 165 for more information.

- Run the **`dump-dns`** tool to an output file. This solution is the quickest of the three methods and is best for very large databases.

  ```
  $ bin/dump-dns --hostname example.server.com --port 389 --bindDN "cn=Directory Man-
  ager" --bindPassword secret --baseDN dc=example,dc=com --outputFile dns.txt
  5300000 DNs written
  Processing completed. 5300000 DNs written.
  ```

Managing Entries

The Directory Server is a fully LDAPv3-compliant server that comes with a comprehensive set of LDAP command-line tools to search, add, modify, and delete entries.

Searching Entries

The Directory Server provides an **`ldapsearch`** tool to search for entries or attributes within your directory. The tool requires the LDAP connection parameters needed to bind to the server, the **`baseDN`** option to specify the starting point of the search within the directory, and the search scope. The **`searchScope`** option determines the depth of the search: *base* (search only the entry specified), *one* (search only the children of the entry and not the entry itself), *sub* (search the entry and its descendents). The **`ldapsearch`** tool provides basic functionality as specified by the RFC 2254 but provides additional features that takes advantage of the Directory Server’s control mechanisms. For more information, run the **`ldapsearch --help`** function.

To Search the Root DSE

The Root DSE is a special directory entry that resides at the root of the directory information tree (DIT). The entry holds operational information about the server and its supported controls. Specifically, the root DSE entry provides information about the supported LDAPv3 controls, SASL mechanisms, password authentication schemes, supported LDAP protocols, additional features, naming contexts, extended operations, and server information.

- Use **ldapsearch** to view the root DSE entry on the Directory Server. Be sure you include the "+" to display the operational attributes in the entry.

```
$ bin/ldapsearch --port 389 --bindDN uid=admin,dc=example,dc=com \
  --bindPassword secret --baseDN "" --searchScope base "(objectclass=*)" "+"
```

Note	The Directory Server provides an option to retrieve the Root DSE's operational attributes and add them to the user attribute map of the generated entry. This feature allows client applications that have difficulty handling operational attributes to access the root DSE using the **show-all-attributes** configuration property. Once this property is set, the associated attribute types are re-created and re-registered as user attributes in the schema (in memory, not on disk). Once you set the property, you can use **ldapsearch** without "+" to view the root DSE. Use the **dsconfig** tool to set the **show-all-attributes** property to **TRUE**, as follows: `$ bin/dsconfig set-root-dse-backend-prop --set show-all-attributes:true`

To Search All Entries in the Directory

- Use **ldapsearch** to search all entries in the directory. The filter **"(objectclass=*)"** matches all entries. If the **--searchScope** option is not specified, the command defaults to a search scope of **sub**:

```
$ bin/ldapsearch --port 389 --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN dc=example,dc=com \
  --searchScope sub "(objectclass=*)"
```

To Search for an Access Control Instruction

- Use **ldapsearch** to search the **dc=example,dc=com** base DN entry. The filter **"(objectclass=*)"** matches all entries, and the **aci** attribute is specified so that only it is returned. The **cn=Directory Manager** bind DN has the privileges to view an **aci**.

```
$ bin/ldapsearch --port 389 --bindDN "cn=Directory Manager" \
  --bindPassword password --baseDN dc=example,dc=com --searchScope base \
  "(objectclass=*)" aci

dn: dc=example,dc=com
aci: (targetattr!="userPassword")
  (version 3.0; acl "Allow anonymous read access for anyone";
    allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

To Search for the Schema

- Use `ldapsearch` to search the `cn=schema` entry. The base DN is specified as `cn=schema`, and the filter `"(objectclass=*)"` matches all entries. The command uses a special attribute "+" to return all operational attributes:

```
$ bin/ldapsearch --port 389 --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN cn=schema --searchScope base "(objectclass=*)" \
  "+"
```

To Search for a Single Entry using Base Scope and Base DN

- Use `ldapsearch` to search for a single entry by specifying the 'base' scope and DN:

```
$ bin/ldapsearch --port 389 --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN uid=user.14,ou=People,dc=example,dc=com \
  --searchScope base "(objectclass=*)"
```

To Search for a Single Entry using the Search Filter

- Search for a single entry by specifying the 'sub' scope and a search filter that describes a single entry:

```
$ bin/ldapsearch --port 389 --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN ou=People,dc=example,dc=com \
  --searchScope sub "(uid=user.14)"
```

To Search for All Immediate Children for Restricted Return Values

- Search for all immediate children of `ou=People,dc=example,dc=com`. The attributes returned are restricted to `sn` and `givenName`. The special attribute "+" returns all operational attributes:

```
$ bin/ldapsearch --port 389 --bindDN uid=admin,dc=example,dc=com \
   --bindPassword password --baseDN ou=People,dc=example,dc=com \
   --searchScope one '(objectclass=*)' sn givenName "+"
```

To Search for All Children of an Entry in Sorted Order

- Search for all children of the `ou=People,dc=example,dc=com` subtree. The resulting entries are sorted by the server in ascending order by `sn` and then in descending order by `givenName`:

```
$ bin/ldapsearch --port 389 --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN ou=People,dc=example,dc=com \
  --searchScope sub --sortOrder sn,-givenName '(objectclass=*)'
```

To Limit the Number of Returned Search Entries and the Search Time

- Search for a subset of the entries in the **`ou=People,dc=example,dc=com`** subtree by specifying a compound filter. No more than 200 entries will be returned and the server will spend no more than 5 seconds processing the request. Returned attributes are restricted to a few operational attributes:

```
$ bin/ldapsearch --port 389 --bindDN uid=admin,dc=example,dc=com \
  --bindPassword password --baseDN ou=People,dc=example,dc=com \
  --searchScope sub --sizeLimit 200 --timeLimit 5 \
  "(&(sn<=Doe)(employeeNumber<=1000))" ds-entry-unique-id entryUUID
```

Adding Entries

Depending on the number of entries that you want to add to your Directory Server, you can use the **`ldapmodify`** tool for small additions. The **`ldapmodify`** tool provides two methods for adding a single entry: using a LDIF file or from the command line.

The attributes must conform to your schema and contain the required object classes.

To Add an Entry Using an LDIF File

1. Open a text editor and create an entry that conforms with your directory schema. For example, add the following entry in the file and save the file as "add-user.ldif." For the **`userPassword`** attribute, enter the cleartext password. The Directory Server encrypts the password and stores its encrypted value in the server. Make sure that the LDIF file has limited read permissions for only authorized administrators.

```
dn: uid=user.2000,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Toby Hall$73600 Mash Street$Cincinnati, OH  50563
postalCode: 50563
description: This is the description for Toby Hall.
uid: user.2000
userPassword: wordsmith
employeeNumber: 2000
initials: TBH
givenName: Toby
pager: +1 596 232 3321
mobile: +1 039 311 9878
cn: Toby Hall
sn: Hall
telephoneNumber: +1 097 678 9688
street: 73600 Mash Street
homePhone: +1 214 233 8484
l: Cincinnati
mail: user.2000@maildomain.net
st: OH
```

2. Use the **`ldapmodify`** tool to add the entry specified in the LDIF file. You will see a confirmation message of the addition. If the command is successful, you will see generated success messages with the "#" symbol.

```
$ bin/ldapmodify --hostname server.example.com --port 389 \
  --bindDN "cn=admin,dc=example,dc=com" --bindPassword password \
  --defaultAdd --filename add-user.ldif
# Processing ADD request for uid=user.2000,ou=People,dc=example,dc=com
# ADD operation successful for DN uid=user.2000,ou=People,dc=example,dc=com
```

To Add an Entry Using the Changetype LDIF Directive

RFC 2849 specifies LDIF directives that can be used within your LDIF files. The most commonly used directive is **changetype**, which follows the **dn:** directive and defines the operation on the entry. The main advantage of using this method in an LDIF file is that you can combine adds and modifies in one file.

1. Open a text editor and create an entry that conforms with your directory schema. For example, add the following entry in the file and save the file as "**add-user2.ldif**." Note the use of the **changetype** directive in the second line.

```
dn: uid=user.2001,ou=People,dc=example,dc=com
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
postalAddress: Seely Dorm$100 Apple Street$Cincinnati, OH  50563
postalCode: 50563
description: This is the description for Seely Dorm.
uid: user.2001
userPassword: pleasantry
employeeNumber: 2001
initials: SPD
givenName: Seely
pager: +1 596 665 3344
mobile: +1 039 686 4949
cn: Seely Dorm
sn: Dorm
telephoneNumber: +1 097 257 7542
street: 100 Apple Street
homePhone: +1 214 521 4883
l: Cincinnati
mail: user.2001@maildomain.net
st: OH
```

2. Use the **ldapmodify** tool to add the entry specified in the LDIF file. You will see a confirmation message of the addition. In this example, you do not need to use the **--defaultAdd** or its shortform **-a** option with the command.

```
$ bin/ldapmodify --hostname server.example.com --port 389 \
  --bindDN "cn=admin,dc=example,dc=com" --bindPassword password \
  --filename add-user2.ldif
```

To Add Multiple Entries in a Single File

You can have multiple entries in your LDIF file by simply separating each DN and its entry with a blank line from the next entry.

1. Open a text editor and create a couple of entries that conform to your directory schema. For example, add the following entries in the file and save the file as "**`add-user3.ldif`**." Separate each entry with a blank line.

   ```
   dn: uid=user.2003,ou=People,dc=example,dc=com
   objectClass: top
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   ...(similar attributes to previous examples)...

   dn: uid=user.2004,ou=People,dc=example,dc=com
   objectClass: top
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   ...(similar attributes to previous examples)...
   ```

2. Use the **`ldapmodify`** tool to add the entries specified in the LDIF file. In this example, we use the short form arguments for the **`ldapmodify`** tool. The **`-h`** option specifies the host name, the **`-p`** option specifies the LDAP listener port, **`-D`** specifies the bind DN, **`-w`** specifies the bind DN password, **`-a`** specifies that entries that omit a changetype will be treated as add operations, and **`-f`** specifies the path to the input file. If the operation is successful, you will see commented messages (those begining with "#") for each addition.

   ```
   $ bin/ldapmodify -h server.example.com -p 389 \
   -D "cn=admin,dc=example,dc=com" -w password -a -f add-user3.ldif
   # Processing ADD request for uid=user.2003,ou=People,dc=example,dc=com
   # ADD operation successful for DN uid=user.2003,ou=People,dc=example,dc=com
   # Processing ADD request for uid=user.2004,ou=People,dc=example,dc=com
   # ADD operation successful for DN uid=user.2004,ou=People,dc=example,dc=com
   ```

Deleting Entries Using ldapdelete

You can delete an entry using the **`ldapdelete`** tool. You should ensure that there are no child entries below the entry as that could create an orphaned entry. Also, make sure that you have properly backed up your system prior to removing any entries.

To Delete an Entry Using ldapdelete

- Use **`ldapdelete`** to delete an entry. The following example deletes the **`uid=user.14`** entry.

  ```
  $ bin/ldapdelete --hostname server1.example.com --port 389 --bindDN
  "uid=admin,dc=example,dc=com" --bindPassword password uid=user.14,ou=Peo-
  ple,dc=example,dc=com
  ```

To Delete Multiple Entries Using a File

1. You can generate a file of DNs that you would like to delete from the Directory Server. The following command searches for all entries in the **ou=Accounting branch** and uses the special LDAP attribute "**1.1**," which returns the DNs of the matched entries. The command uses the UNIX redirection symbol ">" to redirect the output stream to a file. The file will contain only the DNs returned by the search filter.

```
$ bin/ldapsearch --hostname server1.example.com --port 389 --bindDN
"uid=admin,dc=example,dc=com" --bindPassword password --baseDN "ou=Account-
ing,ou=People,dc=example,dc=com" --searchScope sub "(objectclass=*)" "1.1" > /usr/
local/entry_dns.txt
```

2. Run the **ldapdelete** command with the file to delete the entries. The command uses the **--continueError** option, which will continue deleting through the whole list even if an error is encountered for a DN entry.

```
$ bin/ldapdelete --hostname server1.example.com --port 389 --bindDN
"uid=admin,dc=example,dc=com" --bindPassword password \
--filename /usr/local/entry_dns.txt --continueError
```

Deleting Entries Using ldapmodify

You can use the LDIF **changetype** directive to delete an entry from the directory using the **ldapmodify** tool. You can only delete leaf entries.

To Delete an Entry Using ldapmodify

- From the command line, use the **ldapmodify** tool with the **changetype:delete** directive. Enter the DN, press Enter to go to the next line, then enter the changetype directive. Press Control-D twice to enter the EOF sequence (UNIX) or Control-Z (Windows).

```
$ bin/ldapmodify --hostname server1.example.com -port 389 --bindDN
"uid=admin,dc=example,dc=com" --bindPassword password
dn:uid=user.14,ou=People,dc=example,dc=com
changetype: delete
```

Modifying Entries Using ldapmodify

You can use the **ldapmodify** tool to modify entries from the command line or by using an LDIF file that has the **changetype:modify** directive and value. If you have more than one change, you can separate them using the - (dash) symbol.

To Modify an Attribute from the Command Line

1. Use the **ldapsearch** tool to locate a specific entry.

```
$ bin/ldapsearch -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
-w password -b dc=example,dc=com "(uid=user.2004)"
```

2. Use the **ldapmodify** command to change attributes from the command line. Specify the modification using the **changetype:modify** directive, and then specify which attributes are to be changed using the **replace** directive. In this example, we change the telephone number of a specific user entry. When you are done typing, you can press CTRL-D (Unix EOF escape sequence) twice or CTRL-Z (Windows) to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" -w
password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 097 453 8232
```

To Modify Multiple Attributes in an Entry from the Command Line

1. Use the **ldapsearch** tool to locate a specific entry.

```
$ bin/ldapsearch -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
-w password -b dc=example,dc=com "(uid=user.2004)"
```

2. Use the **ldapmodify** command to change attributes from the command line. Specify the modification using the **changetype:modify** directive, and then specify which attributes are to be changes using the **add** and **replace** directive. In this example, we add the **postOfficeBox** attribute, change the mobile and telephone numbers of a specific user entry. The **postOfficeBox** attribute must be present in your schema to allow the addition. The three changes are separated by a dash ("-"). When you are done typing, you can press CTRL-D (Unix EOF escape sequence) twice or CTRL-Z (Windows) to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: postOfficeBox
postOfficeBox: 111
-
replace: mobile
mobile: +1 039 831 3737
-
replace: telephoneNumber
telephoneNumber: +1 097 453 8232
```

To Add an Attribute from the Command Line

- Use the **ldapmodify** command from the command line. Specify the modification using the **changetype:modify** directive, and then specify which attributes are to be added using the **add** directive. In this example, we add another value for the **cn** attribute, which is multi-valued. When you are done typing, you can press CTRL-D (UNIX EOF escape sequence) twice to process the request.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
```

```
add: cn
cn: Sally Tea Tree
```

Note	An error could occur if the attribute is single-valued, if the value already exists, if the value does not meet the proper syntax, or if the value does not meet the entry's objectclass requirements. Also, make sure there are no trailing spaces after the attribute value.

To Add an Attribute Using the Language Subtype

The Directory Server provides support for attributes using language subtypes. The operation must specifically match the subtype for successful operation. Any non-ASCII characters must be in UTF-8 format.

- Use the **ldapmodify** command to add an attribute with a language subtype for Korean.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" -w
password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: postalAddress; lang-ko
postalAddress; lang-ko:Byung-soon Kim$2020-14 Seocho 2-dong, Yongsan-gu$135-283
Seoul
```

To Add an Attribute Using the Binary Subtype

The Directory Server provides support for attributes using binary subtypes, which are typically used for certificates or JPEG images that could be stored in an entry. The operation must specifically match the subtype for successful operation. The version directive with a value of "1" must be used for binary subtypes. Typical binary attribute types are **userCertificate** and **jpegPhoto**.

- Use the **ldapmodify** command to add an attribute with a binary subtype. The attribute points to the file path of the certificate.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" -w
password
version: 1
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
add: userCertificate;binary
userCertificate;binary:<file:///path/to/cert
```

To Delete an Attribute

- Use ldapmodify with the LDIF **delete** directive to delete an attribute.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
delete: employeeNumber
```

To Delete an Attribute with Multiple Values

You can use the LDIF **`delete`** directive to delete a specific attribute. For this example, assuming you have multiple values of **`cn`** in an entry (e.g., cn: Sally Tree, cn: Sally Tea Tree).

- Use **`ldapmodify`** to delete a specific attribute of a multi-valued pair, then specify the attribute pair that you want to delete. In this example, we keep **`cn:Sally Tree`** and delete the **`cn: Sally Tea Tree`**.

```
$ bin/ldapmodify -h server.example.com -p 389 -D "cn=admin,dc=example,dc=com" \
  -w password
dn: uid=user.2004,ou=People,dc=example,dc=com
changetype: modify
delete: cn
cn: Sally Tea Tree
```

To Rename an Entry

Renaming an entry involves changing the relative distinguished name (RDN) of an entry. You cannot rename a RDN if it has children entries as this violates the LDAP protocol.

- Use the **`ldapmodify`** tool to rename an entry. The following command changes **`uid=user.14`** to **`uid=user.2014`** and uses the **`changetype`**, **`newrdn`**, and **`deleteoldrdn`** directives.

```
$ bin/ldapmodify --hostname server1.example.com --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword password
dn: uid=user.14,ou=People,dc=example,dc=com
changetype:moddn
newrdn: uid=user.2014
deleteoldrdn: 1
```

To Move an Entry Within a Directory

You can use **`ldapmodify`** to move an entry from one base DN to another base DN. Before running the **`ldapmodify`** command, you must assign access control instructions (ACIs) on the parent entries. The source parent entry must have an ACI that allows export operations **`allow(export)`**. The target parent entry must have an ACI that allows import operations **`allow(import)`**. For more information on access control instructions, see “Working with Access Control” on page 291.

- Use the **`ldapmodify`** command to move an entry from the Contractor branch to the **`ou=People`** branch.

```
$ bin/ldapmodify --hostname server1.example.com --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword password
dn: uid=user.14,ou=contractors,dc=example,dc=com
changetype:moddn
newrdn: uid=user.2014
deleteoldrdn: 0
newsuperior: ou=People,dc=example,dc=com
```

To Move an Entry from One Machine to Another

The Directory Server provides a tool, **`move-entry`**, to move one entry on one machine to another. The entry must exist on the source server and must not be present on the target server. The source server must also support the use of interactive transactions and the 'real attributes only' request control. The target server must support the use of interactive transactions and the ignore NO-USER-MODIFICATION request control.

- Use the **`move-entry`** tool to move an entry (e.g., **`uid=test.user,ou=People,dc=example,dc=com`**) from the source host to the target host.

```
$ bin/move-entry --sourceHost source.example.com --sourcePort 389 \
--sourceBindDN "uid=admin,dc=example,dc=com" --sourceBindPassword password \
--targetHost target.example.com --targetPort 389 \
--targetBindDN "uid=admin,dc=example,dc=com" --targetBindPassword password \
--entryDN uid=test.user,ou=People,dc=example,dc=com
```

Note	The **`move-entry`** tool moves entries from one machine to another. It does not copy the entries, so that once the entries are moved, they are no longer present on the source server.

To Move Multiple Entries from One Machine to Another

The **`move-entry`** tool provides the ability to move multiple entries listed in a DN file from one machine to another. Empty lines and lines beginning with the octothorpe character (#) will be ignored. Entry DNs may optionally be prefixed with "**`dn: `**", but long DNs cannot be wrapped across multiple lines.

1. Open a text file, enter a list of DNs, one DN per line, and then save the file. You can also use the **`ldapsearch`** command with the special character "**`1.1`**" to create a file containing a list of DNs that you want to move. The following example searches for all entries that match (**`department=Engineering`**) and returns only the DNs that match the criteria. The results are re-directed to an output file, **`test-dns.ldif`**:

```
$ bin/ldapsearch --hostname source.example.com --port 389 --bindDN
"uid=admin,dc=example,dc=com" --bindDNPassword password --baseDN dc=example,dc=com
--searchScope sub "(department=Engineering)" "1.1" > test-dns.ldif
```

2. Run the **`move-entry`** tool with the **`--entryDNFile`** option to specify the file of DNs that will be moved from one machine to another.

```
$ bin/move-entry --sourceHost source.example.com --sourcePort 389 \
--sourceBindDN "uid=admin,dc=example,dc=com" --sourceBindPassword password \
--targetHost target.example.com --targetPort 389 \
--targetBindDN "uid=admin,dc=example,dc=com" --targetBindPassword password \
--entryDNFile /path/to/file/test-dns.ldif
```

3. If an error occurs with one of the DNs in the file, the output message shows the error. The **`move-entry`** tool will continuing processing the remaining DNs in the file.

```
An error occurred while communicating with the target server:  The entry
uid=user.2,ou=People,dc=example,dc=com cannot be added because an entry with that
name already exists
```

```
Entry uid=user.3,ou=People,dc=example,dc=com was successfully moved from
source.example.com:389 to target.example.com:389

Entry uid=user.4,ou=People,dc=example,dc=com was successfully moved from
source.example.com:389 to target.example.com:389
```

Scrambling Data Files

The Directory Server provides an LDIF scrambling tool (`scramble-ldif`) that obscures the contents of a specified set of attributes. Scrambling data can be useful if you want to use real production data to test the UnboundID Directory Server, but do not want to expose the data internally to employees. The tool is also useful to scramble data when providing directory data to a third party to investigate a problem.

The `scramble-ldif` tool was designed to ensure that the information is completely obscured in a non-reversible manner. The tool maintains the entry's indexing characteristics that are consistent with the real data while adhering to the associated attribute syntax. Alphabetic characters are replaced with random alphabetic characters (preserving the case of those letters), and numeric digits are replaced with random numeric digits. The algorithm used to select the letters and digits ensures that the same clear-text value will always result in the same scrambled value in the LDIF file (although the values can be scrambled differently each time the tool is run). Non-alphanumeric characters will be left unaltered.

The `scramble-ldif` tool has options to sequentially generate numbers for specified attribute values. This feature is useful when running tests on your scrambled data. The tool also has an option to gzip compress the scrambled LDIF output file.

To Scramble the LDIF File

This procedure scrambles the data in the following example LDIF entry:

```
dn: uid=smith,ou=People,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
postalAddress: Aaron Smith$01251 Chestnut Street$Panama City, DE  50369
postalCode: 50369
description: This is the description for Aaron Smith.
uid: asmith
userPassword: password
employeeNumber: 0
initials: AGS
givenName: Aaron
pager: +1 779 041 6341
mobile: +1 010 154 3228
cn: Aaron Smith
sn: Smith
telephoneNumber: +1 685 622 6202
street: 01251 Chestnut Street
homePhone: +1 225 216 5900
```

```
l: Panama City
mail: asmith@example.com
st: DE
```

1. Use the **`scramble-ldif`** tool to scramble an import LDIF file.

```
$ bin/scramble-ldif --sourceLDIF real-data.ldif --targetLDIF scramble.ldif \
  --attributeName postalAddress --attributeName description \
  --attributeName uid --attributeName userPassword  --attributeName employeenumber \
  --attributeName initials --attributeName givenName --attributeName pager \
  --attributeName mobile --attributeName cn --attributeName sn \
  --attributeName telephoneNumber --attributeName street --attributeName homephone \
  --attributeName mail --attributeName uid --processDNs
Processing complete.  100004 entries written
```

2. Check the LDIF file using the **`ldifsearch`** command with a general search filter, such as **`(objectclass=*)`**. The directory server provides utilities to search (**`ldifsearch`**), modify (**`ldifmodify`**), or compare (**`ldif-diff`**) LDIF files without binding to the server.

```
$ bin/ldifsearch --ldifFile scramble.ldif "(objectclass=*)"

dn: uid=xjgtld,ou=People,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
postalAddress: Mucvm Sdzyv$44727 Yjsyybhr Bbxmun$Rzgafz Byul, BW  46369
postalCode: 50369
description: Isho pk uyp mexwfvfpuig srn Vkpta Dutrx.
uid: xjgtld
userPassword: wtxooikk
employeenumber: 3
initials: JHU
... (more output) ...
```

To Sequentially Scramble Specific Attributes

Use the **`scramble-ldif`** command to obfuscate specific attributes from an import LDIF file. The following command scrambles the **`givenName`**, **`cn`**, **`sn`**, and **`uid`** attributes. The **`uid`** attribute is replaced with the value 10000. Each subsequent **`uid`** attribute will be sequentially ordered; thus, the next entry will have a **`uid`** attribute with the value of 10001. The **`--processDNs`** option scrambles the DNs with the same scrambled value for the **`uid`** attribute.

```
$ bin/scramble-ldif --sourceLDIF real-data.ldif --targetLDIF scramble.ldif \
  --attributeName uid --attributeName givenName --attributeName cn \
  --attributeName sn --attributeName uid --sequentialAttributeName uid \
  --initialSequentialValue 10000 --processDNs
```

Backing Up and Restoring Data

Directory administrators should have a comprehensive backup strategy and schedule that comprise of daily, weekly, and monthly backups including incremental and full backups of the directory data, configuration, and backends. Administrators should also have a backup plan for the underlying filesystem; for example, taking regular snapshots on ZFS systems on Solaris-based machines. This dual purpose approach provides excellent coverage in the event that a server database must be restored for any reason.

The UnboundID Directory Server provides efficient backup and restore command-line tools that support full and incremental backups. The backups can also be scheduled using the UNIX-based **`cron`** scheduler or using the Directory Server's Task-based scheduler. Backups can also be performed with the server online while processing other requests, so that there is no need to shut down the server or place it in read-only mode prior to starting a backup.

If you back up more than one backend, the backup tool creates a subdirectory below a specified backup directory for each backend. If you back up only a single backend, then the backup files will be placed in the specified directory. A single directory can only contain files from one backend, so that you cannot have backup files from multiple different backends in the same backup directory.

The Directory Server's **`backup`** command and ZFS snapshots provide good coverage in the event a database needs to be restored. For example, the server database would require restoration if the following occurs: 1) the underlying disks have gone bad and the database is inaccessible, 2) the filesystem is operational but the database has become out-of-sync, 3) some application error has occurred data. The Directory Server's **`backup`** command can help with the first case as it provides a way to store backups on a separate disk from the database itself. ZFS snapshots can help with the last two cases.

Note If you have an implementation using the Large Attributes backend, the backup procedure is a bit more complicated due to the additional time it takes to back up and restore the backend. This additional time may be problematic in some production environments. See "Running Backups for Large Attributes" on page 193 for an alternative backup procedure.

To Back Up All Backends

- Use **`backup`** to save the all of the server's backends. The optional **`--compress`** option can reduce the amount of space that the backup consumes, but can also significantly increase the time required to perform the backup.

```
$ bin/backup --backUpAll --compress --backupDirectory /path/to/backup
```

To Back Up a Single Backend

Go to the installation directory, and use the **`backup`** tool to save the single backend, **`userRoot`**.

```
$ bin/backup --backendID userRoot --compress --backupDirectory /path/to/backup
```

To Assign an ID to a Backup

Go to the installation directory, and use the **backup** tool to save the single backend, **userRoot**. The following command assigns the backup ID "weekly" to the **userRoot** backup. The backup file appears under **backups/userRoot** directory as **userRoot-backup-weekly**.

```
$ bin/backup --backupDirectory /path/to/backups/userRoot --backendID userRoot \
  --backupID weekly
```

To Run an Incremental Backup on All Backends

The Directory Server provides support for incremental backups, which backs up only those items that have changed since the last backup (whether full or incremental) on the system, or since a specified earlier backup. Incremental backups must be placed in the same backup directory as the full backup on which they are based.

The following command runs an incremental backup on all backends based on the most recent backup:

```
$ bin/backup --backUpAll --incremental --backupDirectory /path/to/backup
```

To Run an Incremental Backup on a Single Backend

Go to the installation directory, and use **backup** to save the single backend, **userRoot**.

```
$ bin/backup --backendID userRoot --incremental --backupDirectory /path/to/backup
```

To Run an Incremental Backup based on a Specific Prior Backup

You can run an incremental backup based on a specific prior backup that is not the most current version on the system. To get the backup ID, use the **restore --listBackups** command (see below).

```
$ bin/backup --backUpAll --incremental --backupDirectory /path/to/backup \
  --incrementalBaseID backup-ID
```

To Schedule an Online Backup

You can schedule an backup to run as a Task by specifying the timestamp with the **--task** and **--start** options. The option is expressed in "YYYYMMDDhhmmss’" format. If the option has a value of "0" then the task is scheduled for immediate execution. You cannot run recurring tasks, so daily operations must be run using cron or through some system that can submit the task.

For online (remote) backups, the backup operation can be conducted while the Directory Server is online if you provide information about how to connect and to authenticate to the target directory. You can schedule the backup to occur at a specific time using the Task-based \ **--start YYYYMMDDhhmmss** option. To specify a time in the UTC time zone format, add a trailing "Z" to the time. Otherwise, the time will be treated as a local time in the time zone configured on the server.

```
$ bin/backup --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --backUpAll --task --start 20111025010000 \
  --backupDirectory /path/to/backup --completionNotify admin@example.com \
  --errorNotify admin@example.com

Backup task 2011102500084110 scheduled to start Oct 28, 2011 1:00:00 AM CDT
```

To List the Available Backups on the System

Use the **`restore`** tool to list the backups in a backup directory.

```
$ bin/restore --listBackups --backupDirectory /mybackups

[13:26:21]  The console logging output is also available in '/ds/UnboundID-DS/logs/
tools/restore.log'

Backup ID:          20120212191715Z
Backup Date:        12/Feb/2012:13:17:19 -0600
Is Incremental:     false
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash:  false
Has Signed Hash:    false
Dependent Upon:     none
Backup ID:          20120212192411Z
Backup Date:        12/Feb/2012:13:24:16 -0600
Is Incremental:     true
Is Compressed:      false
Is Encrypted:       false
Has Unsigned Hash:  false
Has Signed Hash:    false
Dependent Upon:     20120212191715Z
```

To Perform an Offline Restore

Use the **`restore`** command to restore the **`userRoot`** backend. Only a single backend can be restored at a time. The Directory Server must be shut down before performing an offline restore.

```
$ bin/restore --backupDirectory /path/to/backup/userRoot
```

Note	If you ran a backup using the **`--backupAll`** option to back up all of the directories, you can still only restore one backend at a time. If it is necessary to rerun a test or back out of a failed change, you can create an archive (e.g., tarball) of the server root or take a ZFS snapshot.

To Schedule an Online Restore

By providing connection and authentication information (and an optional start time), the restore can be performed via the tasks subsystem while the server is online. The Tasks subsystem allows you to schedule certain operations, **`such as import-ldif`**, **`export-ldif`**, **`backup`**, **`restore`**, **`start-ds`**, and **`stop-ds`**. You can schedule a restore to run as a Task by specifying the timestamp with the **`--task`** and **`--start`** options. The option is expressed in "YYYYMMDDhhmmss'" format. If the option has a value of "0" then the task is scheduled for immediate execution. You cannot run recurring tasks, so daily operations must be run using cron or through some system that can submit the task.

The backend that is being restored will be unavailable while the restore is in progress. To specify a time in the UTC time zone, add a trailing "Z" to the time. Otherwise, the time will be treated as a local time in the configured time zone on the server.

```
$ bin/restore --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --task --start 20111025010000 \
  --backupDirectory /path/to/backup/userRoot \
  --completionNotify admin@example.com --errorNotify admin@example.com
```

To Restore an Incremental Backup

The process for restoring an incremental backup is exactly the same as the process for restoring a full backup for both the online and offline restore types. The **`restore`** tool will automatically ensure that the full backup and any intermediate incremental backups are restored first before restoring the final incremental backup. The tool will not restore any files in older backups that are no longer present in the final data set.

Moving or Restoring a User Database

Part of any disaster recovery involves the restoration of the user database from one server to another. You should have a well-defined backup plan that takes into account whether or not your data is replicated among a set of servers. The plan is the best insurance against significant downtime or data loss in the event of unrecoverable database issue.

Keep in mind the following general points about database recovery:

- **Regular ZFS Snapshots (Solaris or OpenSolaris Only)**. If you have been taking regular ZFS snapshots (e.g., every 15-60 minutes), then you can roll back to the most recent snapshot before the problem. This should be the fastest recovery, and replication should bring you back up to date fairly quickly. However, in the case of a "disk full" problem, you may also need to do other things like destroy older snapshots to free up more space to avoid running into the same problem again in the course of replaying other changes.

- **Regular Backup from Local Replicated Directory Server**. Take a backup from a local replicated directory server and restore to the failed server. This will be more recent than any other backup you have.

- **Restore the Most Recent Backup**. Restore the most recent backup from a local server. In some cases, this may be preferred over taking a new backup if that would adversely impact performance of the server being backed up although it will take longer for replication to play back changes.

- **Contact Support**. If all else fails, contact your authorized support provider and we can work with you (and possibly in cooperation with the Oracle Berkeley DB JE engineers) to try a low-level recovery, including tools that attempt to salvage whatever data we can obtain from the database.

To Move or Restore a User Database (Solaris or OpenSolaris Only)

The following procedure presents how to move or restore the user database from a source server and then restore the database on the target server.

1. Shutdown the target directory server.

2. Remove the existing `db/userRoot` and `changelogDb` directories from the target directory server if present. The `db/userRoot` is the user database and the `changelogDb` is the replication server changelog, respectively.

```
# rm -rf changelogDb/
# rm -rf db/userRoot/
```

3. On the source server, run ’`zfs list`’ to see the list of ZFS pools.

```
# zfs list
```

   If you want to see a list of snapshots, use the `-t` option:

```
# zfs list -t snapshot
```

4. On the source server, create a ZFS snapshot of the mirror at the source. Assume that the name of the ZFS filesystem is `rpool/ds`. The source server does not need to be shutdown. Also, because of the atomic capability of zfs, you can use the `-r` option to recursively take a snapshot for the pool and all of its descendent filesystems.

```
# zfs snapshot -r rpool/ds@backup-020810
```

5. Run zfs list to verify the snapshot.

```
# zfs list -t snapshot
```

6. Perform a secure copy of the `changelogDb` files from the snapshot to the target. You can treat the snapshot as a read-only filesystem.

```
# cd /ds/.zfs/snapshot/backup-020210/path/to/ds/UnboundID-DS/
# scp -r changelogDb/ root@host:/ds/path/to/ds/UnboundID-DS/
```

7. Perform a secure copy of the `userRoot` files from the snapshot to the target. You can treat the snapshot as a read-only filesystem.

```
# cd /ds/.zfs/snapshot/backup-020210/path/to/ds/UnboundID-DS/
# scp -r userRoot/ root@host:/ds/path/to/ds/UnboundID-DS/
```

8. Start the target directory server using **`bin/start-ds`** (or **`bat\start-ds`** on Windows systems).

9. On the source server, remove the snapshot.

   ```
   # zfs destroy -r rpool/ds@backup-020810
   ```

   You have successfully moved or restored a database.

Working with Indexes

The UnboundID Directory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the Directory Information Tree (DIT). Indexes are associated with attributes and stored in database index files, which are managed separately for each base DN in the Directory Server. The Directory Server automatically creates index files when you first initialize a base DN or when you use the **`dsconfig`** tool to create a local DB backend. During modify operations, the Directory Server updates the database index files.

The UnboundID Directory Server comes with the following types of indexes:

- Default system indexes to ensure that the server operates efficiently. These indexes consists of database files that contain index keys mapping to the list of entry IDs.
- Default Local DB indexes that are created for each database suffix. You can modify the index to meet your system's requirements using the **`dsconfig`** tool.
- Local DB VLV indexes that allow a client to request the server to send search results using the Virtual List View control.
- Filtered Indexes that provide the ability to index an attribute but only for entries that match a specified filter based on an equality index. The filtered index can only be used for searches containing that filter. The filtered index can be maintained independently of the equality index for that attribute and even if a normal equality index is not maintained for that attribute.

General Tips on Indexes

Directory administrators should keep the following tips in mind when working with indexes:

- **Important Critical Indexes.** The Directory Server has several built-in indexes on the Local DB Backend that are critical to internal server processing and should *never* be removed.

  aci, ds-entry-unique-id, objectClass

- **Built-in Indexes for Efficient Queries.** The Directory Server has built-in indexes on the Local DB Backend that support efficient queries over standard LDAP. These indexes are not required for all deployments.

  cn, givenName, mail, member, sn, telephoneNumber, uid, uniqueMember

  If the data in the directory does not include these attributes, then the indexes can be removed. Note however that there is also *no cost* to having them present either. If the data includes these attributes, but they are not used in queries, then they can be removed, which will reduce the size of the database both on disk and in memory.

- **Online Rebuilds**. Whenever an online index rebuild is in progress, the data in that backend will be available and writable although the index being rebuilt will not be used; therefore, searches which attempt to use that attribute might be unindexed.

- **Index Rebuild Administrative Alert**. The Directory Server generates an administrative alert when the rebuild process begins and ends. It will have a degraded-alert-type of "index-rebuild-in-progress" so that a proxy server, such as the UnboundID Directory Proxy Server can avoid using that server while the rebuild is in progress.

- **Avoid Online System Index Rebuilds**. Avoid rebuilding system indexes (primarily dn2id, dn2uri, id2children, and id2subtree) with the backend online, because it can have a significant impact on the availability of that data.

- **Indexing Certain Attributes.** You should ensure that the following recommendations are used when setting up the indexes.

  - Equality and substring indexes should not be used for attributes that contain binary data.

  - Approximate indexes should be avoided for attributes containing numbers, such as telephone numbers.

- **Unindexed Searches**. Attributes that are not indexed are *unindexed* and result in longer search times as the database itself has to be searched instead of the database index file. Only users with the `unindexed-search` privilege are allowed to carry out unindexed searches. In general, applications should be prevented from performing unindexed searches, so that searches that are not indexed would be rejected rather than tying up a worker thread. The ways to achieve this include:

  - Make sure that only the absolute minimum set of users have the `unindexed-search` privilege.

  - Make sure that `allow-unindexed-searches` property is set to false in all client connection policies in which unindexed searches should never be necessary.

  - Set a nonzero value for the `maximum-concurrent-unindexed-searches` global configuration property to ensure that if unindexed searches are allowed, only a limited number of them will be active at any given time. Administrators can configure the maximum number of concurrent unindexed searches by setting a property under Global Configuration.

To change the maximum number of concurrent unindexed searches, use the **`dsconfig`** tool to set a value for the number. A value of "0" (default) represents no limit on the number of concurrent unindexed searches.

```
$ bin/dsconfig set-global-configuration-prop --set maximum-concurrent-unindexed-
searches:2
```

- **Index Entry Limit**. The Directory Server specifies an *index entry limit* property. This property defines the maximum number of entries that are allowed to match a given index key before it is no longer maintained by the server. If the index keys have reached this limit (default value is 4000), then you must rebuild the indexes using the **`rebuild-index`** tool. If an index entry limit value is set for the local DB backend, it overrides the value set for the overall JE backend index entry limit configuration (i.e., 4000).

  To change the default index entry limit, use the **`dsconfig`** tool as seen in the following example:

  ```
  $ bin/dsconfig set-local-db-index-prop --backend-name userRoot --index-name cn \
    --set index-entry-limit:5000
  ```

- **Rebuild Index vs Full Import**. You can expect a limited amount of database growth due to the existence of old data when running rebuild-index versus doing a full import of your database.

Index Types

The UnboundID Directory Server supports several types of indexes to quickly find entries that match search criteria in LDAP operations. The Directory Server uses an attribute's matching rules to normalize its values and uses those values as index keys to a list of matching entry IDs. Entry IDs are integer values that are used to uniquely identify an entry in the backend by means of a set of database index files (id2entry, dn2id, dn2uri, id2children, id2subtree).

Matching rules are elements defined in the schema that tell the server how to interact with the particular attribute. For example, the **`uid`** attribute has an equality matching rule defined in the schema, and thus, has an equality index maintained by the directory server. The following table describes the index types.

TABLE 7-3. UnboundID Directory Server Index Types

Index Type	Description
Approximate	Used to efficiently locate entries that match the approximate search filter. It is used to identify which entries are approximately equal to a given assertion. Approximate indexes can only be applied to attributes that have a corresponding approximate matching rule.
Equality	Used to efficiently locate entries that match the equality search filter. It is used to identify which entries are exactly equal to a given assertion. Equality indexes can only be applied to attributes that have a corresponding equality matching rule. An offshoot of the equality index is the filtered index, which uses a defined search filter for a specific attribute. The filtered index can be maintained independently of the equality index for a specific attribute.
Ordering	Used to efficiently locate entries that match the ordering search filter. It is used to identify which entries have a relative order of values for an attribute. Ordering indexes can only be applied to attributes that have a corresponding ordering matching rule.
Presence	Used to efficiently locate entries that match the presence search filter. It is used to identify which entries have at least one value for a specified attribute. There is only one presence index key per attribute.
Substring	Used to efficiently locate entries that match the substring search filter. It is used to identify which entries contain specific substrings to a given assertion. Substring indexes can only be applied to attributes that have a corresponding substring matching rule.

System Indexes

The UnboundID Directory Server contains a set of default system database index files that ensure that the server operates efficiently. These default indexes cannot be modified or deleted.

TABLE 7-4. System Indexes

Index	Description
dn2id	Allows quick retrieval of DNs. The DN database, or dn2id, has one record for each entry. The key is the normalized entry DN and the value is the entry ID.
id2entry	Allows quick retrieval of entries. The id2entry database contains the LDAP entries. The database key is the entry ID and the value is the entry contents.
referral	Allows quick retrieval of referrals. The referral database called dn2uri contains URIs from referral entries. The key is the DN of the referral entry and the value is that of a labeled URI in the ref attribute for that entry.
id2children	Allows quick retrieval of an entry and its children. The id2children database provides a mapping between an entry's unique identifier and the entry unique identifiers of the corresponding entry's children.
id2subtree	Allows quick retrieval of an entry's subtree. The id2subtree database provides a mapping between an entry's unique identifier and its unique identifiers in its subtree.
state	Stores the configuration state of the JE backend for a particular base DN.
recentChanges	Stores the recent changes to the Directory Server.
objectclass	Allows quick retrieval of subtree searches. Stored as an equality index.

TABLE 7-4. System Indexes

Index	Description
givenName	Allows quick retrieval of givenName searches. Stored as an equality and a substring indexes.
ds-entry-unique-id	Allows quick retrieval of specific entries. Stored as an equality index.
member	Allows quick retrieval of member searches in group entries. Stored as an equality index.
uid	Allows quick retrieval of uid searches. Stored as an equality index.
cn	Allows quick retrieval of cn searches. Stored as equality and substring indexes.
uniquemember	Allows quick retrieval of uniqueMember searches in group entries. Stored as an equality index.
telephoneNumber	Allows quick retrieval of telephoneNumber searches. Stored as an equality index.
sn	Allows quick retrieval of sn searches. Stored as equality and substring indexes.
mail	Allows quick retrieval of mail searches. Stored as an equality index.
aci	Allows quick retrieval of aci searches. Stored as a presence index.
vlvIndex	Allows quick retrieval of VLV indexes.

To View the System Indexes

Use the **dbtest** command to view the system and user indexes. The index status indicates if it is a trusted index. A status of "false" indicates that the index is no longer trusted and needs to be rebuilt. In this case, the Directory Server will issue an alert if any index is operating in a degraded state (index status = false) and must be rebuilt.

```
$ bin/dbtest list-index-status --baseDN dc=example,dc=com --backendID userRoot

[19/Mar/2010:11:10:52 -0500] The Directory Server has sent an alert notification gener-
ated by class com.unboundid.directory.server.backends.jeb.EntryContainer (alert type
index-degraded, alert ID 8847510):  Due to changes in the configuration, index
dc_example_dc_com_employeeNumber.equality is currently operating in a degraded state
and must be rebuilt before it can used

Index Name                    Index Type  JE Database Name                                  Index
                                                                                             Status
---------------------------------------------------------------------------------------------------
id2children                   Index       dc_example_dc_com_id2children                      true
id2subtree                    Index       dc_example_dc_com_id2subtree                       true
objectClass.equality          Index       dc_example_dc_com_objectClass.equality             true
givenName.equality            Index       dc_example_dc_com_givenName.equality               true
givenName.substring           Index       dc_example_dc_com_givenName.substring              true
ds-entry-unique-id.equality   Index       dc_example_dc_com_ds-entry-unique-id.equality      true
member.equality               Index       dc_example_dc_com_member.equality                  true
uid.equality                  Index       dc_example_dc_com_uid.equality                     true
cn.equality                   Index       dc_example_dc_com_cn.equality                      true
cn.substring                  Index       dc_example_dc_com_cn.substring                     true
uniqueMember.equality         Index       dc_example_dc_com_uniqueMember.equality            true
telephoneNumber.equality      Index       dc_example_dc_com_telephoneNumber.equality         true
sn.equality                   Index       dc_example_dc_com_sn.equality                      true
sn.substring                  Index       dc_example_dc_com_sn.substring                     true
mail.equality                 Index       dc_example_dc_com_mail.equality                    true
aci.presence                  Index       dc_example_dc_com_aci.presence                     true
employeeNumber.equality       Index       dc_example_dc_com_employeeNumber.equality         false

Total: 17
```

Preloading a System Index

You can preload a system index when a backend is initialized for your system. You can use the **dsconfig** tool to specify the system indexes that you want to prime using the **system-index-to-prime**. To set the property, you must also set the **prime-all-indexes** property to **FALSE**. For more information, see “JVM Properties for Both Server and Command-Line Tools” on page 80.

To Preload a System Index

Use the **dsconfig** tool to set the **prime-all-indexes** property to **FALSE** for the **userRoot** backend. Also, set the **system-index-to-prime** propery for each system index for the back-end.

```
$ bin/dsconfig set-backend-prop --no-prompt --backend-name userRoot \
  --set prime-all-indexes:false --set system-index-to-prime:dn2id \
  --set system-index-to-prime:id2entry \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret
```

Managing Local DB Indexes

You can modify the local DB indexes to meet your system's requirements using the **dsconfig** tool. If you are using the **dsconfig** tool in interactive command-line mode, you can access the Local DB Index menu from the Basic object menu.

To View the List of Local DB Indexes

Use **dsconfig** with the **list-local-db-indexes** option to view the default list of indexes.

```
$ bin/dsconfig list-local-db-indexes --backend-name userRoot \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt

Local DB Index     : Type    : index-type
-------------------:---------:--------------------
aci                : generic : presence
cn                 : generic : equality, substring
ds-entry-unique-id : generic : equality
givenName          : generic : equality, substring
mail               : generic : equality
member             : generic : equality
objectClass        : generic : equality
sn                 : generic : equality, substring
telephoneNumber    : generic : equality
uid                : generic : equality
uniqueMember       : generic : equality
```

To View a Property for All Local DB Indexes

Use **`dsconfig`** with the **`--property`** option to view a property assigned set for all local DB indexes. Repeat the option for each property that you want to list. In this example, the **`prime-index`** property specifies if the backend is configured to prime the index at startup.

```
$ bin/dsconfig list-local-db-indexes --property index-entry-limit \
  --property prime-index --backend-name userRoot --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

To View the Configuration Parameters for Local DB Index

To view the configuration setting of a local DB index, use **`dsconfig`** with the **`get-local-db-index-prop`** option and the **`--index-name`** and **`--backend-name`** properties. If you want to view the advanced properties, add the **`--advanced`** option to your command.

```
$ bin/dsconfig get-local-db-index-prop --index-name aci --backend-name userRoot \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret \
  --no-prompt
```

To Modify the Configuration of a Local DB Index

You can easily modify an index using the **`dsconfig`** tool. Any modification or addition of an index requires the indexes to be rebuilt. In general, an index only needs to be built once after it has been added to the configuration.

If you add an index, then import the data using the **`import-ldif`** tool, then the index will be automatically rebuilt. If you add an index, then add the data using some other method than **`import-ldif`**, you must rebuild the index using the **`rebuild-index`** tool.

1. Use **`dsconfig`** with the **`set-local-db-index-prop`** option and the **`--index-name`** and **`--backend-name`** properties. In this example, update the **`prime-index`** property, which loads the index at startup. This command requires the **`--advanced`** option to access this property.

```
$ bin/dsconfig set-local-db-index-prop --index-name uid \
  --backend-name userRoot --set prime-index:true \
  --advanced --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. View the index to verify the change.

```
$ bin/dsconfig get-local-db-index-prop --index-name uid \
  --backend-name userRoot --advanced --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in “General Tips on Indexes” on page 165.

```
$ bin/stop-ds
```

4. Run the **`rebuild-index`** tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index uid
```

5. Restart the Directory Server if shutdown.

```
$ bin/start-ds
```

To Create a New Local DB Index

1. To create a new local DB index, use **`dsconfig`** with the **`--create-local-db-index`** option and the **`--index-name`**, **`--backend-name`**, and **`--set index-type: (propertyValue)`** options. If you do not set any property values, the default values are assigned.

```
$ bin/dsconfig create-local-db-index \
  --index-name roomNumber --backend-name userRoot --set index-type:equality \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. View the index.

```
 $ bin/dsconfig get-local-db-index-prop \
  --index-name roomNumber --backend-name userRoot --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in “General Tips on Indexes” on page 165.

```
$ bin/stop-ds
```

4. Rebuild the index using the **`rebuild-index`** tool.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index roomNumber
```

5. Restart the Directory Server.

```
$ bin/start-ds
```

To Delete a Local DB Index

You can delete an index using the **`dsconfig`** tool. Check that no plug-in applications are using the index before deleting it. When the index is deleted, the corresponding index database will also be deleted. The disk space is reclaimed once the cleaner threads begin.

Use **`dsconfig`** with the **`delete-local-db-index`** option to remove it from the database.

```
$ bin/dsconfig delete-local-db-index \
  --index-name roomNumber --backend-name userRoot --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

Working with Local DB VLV Indexes

Local DB VLV indexes allow a client to request that the Directory Server send only a subset of the search results, which is made possible by the virtual list view (VLV) control. These indexes allow the client to run **`ldapsearch`** with the **`--virtualListView`** option. If you are using the **`dsconfig`** tool in interactive command-line mode, you can access the Local DB Index menu from the Standard object menu.

To View the List of Local DB VLV Indexes

Use **`dsconfig`** with the **`list-local-db-indexes`** option to view the default list of indexes. In the example, no VLV indexes are defined.

```
$ bin/dsconfig list-local-db-vlv-indexes --backend-name userRoot \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword password --no-prompt
```

To Create a New Local DB VLV Index

1. Use **`dsconfig`** with the **`create-local-db-vlv-index`** option and the **`--index-name`**, **`--backend-name`**, and **`--set index-type:(propertyValue)`** options. If you do not set any property values, the default values are assigned.

```
$ bin/dsconfig create-local-db-vlv-index \
  --index-name givenName --backend-name userRoot --set base-dn:dc=example,dc=com \
  --set scope:whole-subtree --set filter:"(objectclass=*)" \
  --set sort-order:givenName \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. View the index.

```
$ bin/dsconfig get-local-db-vlv-index-prop \
  --index-name givenName --backend-name userRoot --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret --no-prompt
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in “General Tips on Indexes” on page 165.

```
$ bin/stop-ds
```

4. Rebuild the index using the **`rebuild-index`** tool. You must add the “**`vlv.`**” prefix to the index name to rebuild the VLV index.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.givenName
```

5. Restart the Directory Server.

```
$ bin/start-ds
```

To Modify a VLV Index's Configuration

1. Use **`dsconfig`** with the **`set-local-db-vlv-index-prop`** option and the **`--index-name`** and **`--backend-name`** properties. In this example, update the **`base-dn`** property.

```
$ bin/dsconfig set-local-db-vlv-index-prop --index-name givenName \
  --backend-name userRoot --set base-dn:ou=People,dc=example,dc=com \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. View the index to verify the change.

```
$ bin/dsconfig get-local-db-vlv-index-prop \
  --index-name givenName --backend-name userRoot --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --no-prompt
```

3. Stop the Directory Server. You can do an index rebuild with the server online; however, keep in mind the tips presented in "General Tips on Indexes" on page 165.

```
$ bin/stop-ds
```

4. Rebuild the index using the **`rebuild-index`** tool. You must add the prefix **`vlv`** to the index name.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index vlv.givenName
```

5. Restart the directory server.

To Delete a VLV Index

You can delete a VLV index using the **`dsconfig`** tool. Check that the index is not being used in any plug-in applications before deleting it.

1. Use **`dsconfig`** with the **`delete-local-db-vlv-index`** option to remove it from the data-base.

```
$ bin/dsconfig delete-local-db-vlv-index --index-name givenName \
  --backend-name userRoot --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. Verify the deletion by trying to view the vlv index.

```
$ bin/dsconfig get-local-db-vlv-index-prop --index-name givenName \
  --backend-name userRoot --port 1389 --bindDB "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

Working with Filtered Indexes

The UnboundID Directory Server provides an advanced feature that uses an equality search filter for indexing a specified attribute. If the Directory Server has a large number of entries and multiple categories for an attribute, such as **`employeeType`** (full-time, part-time, contract), the server can improve search performance by using a filtered index for a specific attribute, such as **`employeeNumber`**, together with a search filter of "**`(employeeType=full-time)`**". The filtered index can then be used for equality searches for the attribute and the filter, expressed as the following:

```
(&(employeeNumber=12345)(employeeType=full-time))
(&(employeeType=full-time)(employeeNumber=12345))
(&(employeeNumber=12345)(&(employeeType=full-time)))
(&(employeeNumber=12345)(employeeType=full-time(foo=bar))
(&(employeeNumber=12345)(&(employeeType=full-time)(foo=bar)))
```

The filtered index can be maintained independently from the equality filter for that attribute. Further, the index will be used only for searches containing the equality component with the associated attribute type ANDed with this filter. When configuring a filtered index, you should be aware of the following properties:

TABLE 7-5. Filtered Index Properties

Filtered Index Properties	Description
equality-index-filter	Specifies a search filter that may be used in conjunction with an equality component for the associated attribute type. If an equality index filter is defined, then an additional equality index will be maintained for the associated attribute, but only for entries that match the provided filter. Further, the index will be used only for searches containing an equality component with the associated attribute type ANDed with this filter.
maintain-equality-index-without-filter	Specifies whether to maintain a separate equality index for the associated attribute without any filter, in addition to maintaining an index for each equality index filter that is defined. If this is false, then the attribute will not be indexed for equality by itself but only in conjunction with the defined equality index filters.

To Create a Filtered Index

1. Use the **`dsconfig`** tool to create a filtered index. The following command creates an equality index on the **`employeeNumber`** attribute and maintains an index for the equality filter defined "**`(employeeType=full-time)`**". After you have created the index, you must rebuild the indexes.

```
$ bin/dsconfig create-local-db-index --backend-name "userRoot" \
  --index-name employeeNumber --set maintain-equality-index-without-filter:true \
  --set index-type:equality --set equality-index-filter:"(employeeType=full-time)"\
  --no-prompt
```

2. Stop the Directory Server using **`bin/stop-ds`**.

3. Run the **`rebuild-index`** tool to rebuild your index.

```
$ bin/rebuild-index --baseDN dc=example,dc=com --index employeeNumber
```

4. Start the Directory Server using **`bin/start-ds`**.

Working with LDAP Transactions

The UnboundID Directory Server provides support for two different types of transactions: *batched transactions*, which are processed all at once at commit time, and *interactive transactions*, which are processed as requested in isolation from other server processing.

Batched transactions are write operations (add, delete, modify, modify DN, and password modify) that are processed as a single atomic unit when the commit request is received. If an abort request is received or an error occurs during the commit request, the changes are rolled back. The batched transaction mechanism is based on RFC 5805.

Interactive transactions are processed as they are requested, but these changes are isolated from other operations in progress on the directory server at the same time. Interactive transactions consist of any combination of add, compare, delete, modify, modify DN, search, and password modify operations. Changes that are part of an interactive transaction are not visible to other clients until the commit request is received, after which the entire set of changes are applied as a single atomic unit.

Applications developed to perform either batched or interactive transactions should include as few operations in the transaction as possible. For batched transactions, the changes are not actually processed until the commit request is received. Therefore, the client cannot know whether the changes will be successful until commit time. If any of the operations fail, then the entire set of operations fails. Keeping the number of operations in a batched transaction as small as possible helps reduce the number of changes that will not be applied in the event of a problem with any of those requests.

For interactive transactions, the Directory Server might need to lock access to entries that are involved in the transaction until the transaction is committed. This lock increases the chance that other requests can be significantly delayed. The lock could potentially create deadlocks that cause one or more of those contending requests to be aborted, requiring a rollback of all the changes. Therefore, minimizing the number of transactions helps ensure successful operation processing.

Note

Batched and interactive transactions assume the existence of the `userRoot` backend. If the `userRoot` backend does not exist, then the transaction operation handlers will be disabled at startup and report errors.You must disable the transaction handlers until you can re-install the userRoot backend:

```
dsconfig set-extended-operation-handler-prop --handler-name
"Batched Transactions" --set enabled:false
```

```
dsconfig set-extended-operation-handler-prop --handler-name
"Interactive Transactions" --set enabled:false
```

Or reset the handlers to point to some other `userRoot` backend:

```
dsconfig set-extended-operation-handler-prop --handler-name
"Batched Transactions" --set backend-id:someOtherUserRoot
```

```
dsconfig set-extended-operation-handler-prop --handler-name
"Interactive Transactions" --set backend-id:someOtherUserRoot
```

To Request a Batched Transaction Using ldapmodify

1. Use the `ldapmodify` tool’s `--useTransaction` option. It provides a mechanism for processing multiple operations as part of a single batched transaction. Create a batch text file with the changes that you want to apply as a single atomic unit:

```
dn:uid=user.3,ou=People,dc=example,dc=com
changetype: delete

dn:uid=user.1,ou=People,dc=example,dc=com
changetype: modify
replace: pager
pager: +1 383 288 1090
```

2. Use `ldapmodify` with the `--useTransaction` and `--filename` options to run the batched transaction.

```
$ bin/ldapmodify --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --useTransaction --filename test.ldif

#Successfully created a transaction with transaction ID 400
#Processing DELETE request for uid=user.3,ou=People,dc=example,dc=com
#DELETE operation successful for DN uid=user.3,ou=People,dc=example,dc=com
#This operation will be processed as part of transaction 400
#Processing MODIFY request for uid=user.1,ou=People,dc=example,dc=com
#MODIFY operation successful for DN uid=user.1,ou=People,dc=example,dc=com
#This operation will be processed as part of transaction 400
#Successfully committed transaction 400
```

Working with the Parallel-Update Tool

The UnboundID Directory Server provides a `parallel-update` tool, which reads change information (add, delete, modify, and modify DN) from an LDIF file and applies the changes in parallel. This tool is a multi-threaded version of the `ldapmodify` tool that is designed to process a large number of changes as quickly as possible.

The `parallel-update` tool provides logic to prevent conflicts resulting from concurrent operations targeting the same entry or concurrent operations involving hierarchically-dependent entries (for example, modifying an entry after it has been added, or adding a child after its parent). The tool also has a retry capability that can help ensure that operations are ultimately successful even when interdependent operations are not present in the correct order in the LDIF file (for example, the change to add a parent entry is provided later in the LDIF file than a change to add a child entry).

After the tool has applied the changes and reaches the end of the LDIF file, it automatically displays the update statistics described in the following table.

TABLE 7-6. Parallel-Update Tool Result Statistics

Processing Statistic	Description
Attempts	Number of update attempts
Successes	Number of successful update attempts
Rejects	Number of rejected updates
ToRetry	Number of updates that will be retired
AvgOps/S	Average operations per second
RctOps/S	Recent operations per second. Total number of operations from the last interval of change updates.
AvgDurMS	Average duration in milliseconds
RctDurMS	Recent duration in milliseconds. Total duration from the last interval of change updates.

To Run the Parallel-Update Tool

1. Create an LDIF file with your changes. The third change will generate a rejected entry because its `userPassword` attribute contains an encoded value, which is not allowed.

```
dn:uid=user.2,ou=People,dc=example,dc=com
changetype: delete

dn:uid=user.99,ou=People,dc=example,dc=com
changetype: moddn
newrdn: uid=user.100
deleteoldrdn: 1

dn:uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
```

```
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI  49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==

dn:uid=user.100,ou=People,dc=example,dc=com
changetype: modify
replace: st
st: TX
-
replace: employeeNumber
employeeNumber: 100
```

2. Use **`parallel-update`** to apply the changes in the LDIF file to a target server. In this example, we use ten concurrent threads. The optimal number of threads depends on your underlying system. The **`--ldifFile`** and **`--rejectFile`** options are also required.

```
$ bin/parallel-update --hostname 127.0.0.1 --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --ldifFile changes.ldif --rejectFile reject.ldif --numThreads 10

Reached the end of the LDIF file
 Attempts Successes   Rejects   ToRetry  AvgOps/S  RctOps/S  AvgDurMS  RctDurMS
--------- --------- --------- --------- --------- --------- --------- ---------
        4         3         1         0         3         3        26        26

All processing complete
Attempted 4 operations in 1 seconds
```

3. View the rejects file for any failed updates.

```
# ResultCode=53, Diagnostic Message=Pre-encoded passwords are not allowed for
# the password attribute userPassword
dn: uid=user.101,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
postalAddress: Ziggy Zad$15172 Monroe Street$Salt Lake City, MI  49843
postalCode: 49843
description: This is the description for Ziggy Zad.
uid: user.101
userPassword: {SSHA}IK57iPozIQybmIJMMdRQOpIRudIDn2RcF6bDMg==
```

Comparing the Data in Two Directory Servers

The UnboundID Directory Server provides an **`ldap-diff`** tool to compare the data on two LDAP servers to determine any differences that they may contain. The differences are identified by first issuing a subtree search on both servers under the base DN using the default search filter **`(objectclass=*)`** to retrieve the DNs of all entries in each server. When the tool finds an entry is on both servers, it retrieves the entry from each server and compares all of its attributes. The tool writes any differences it finds to an LDIF file in a format that could be used to modify the content of the source server, so that it matches the content of the target server. Any non-synchronized entries can be compared again for a configurable number of times with an optional pause between each attempt to account for replication delays.

You can control the specific entries to be compared with the **`--searchFilter`** option. In addition, only a subset of attributes can be compared by listing those attributes as trailing arguments of the command. You can also exclude specific attributes by prepending a ^ character to the attribute. (On Windows operating systems, excluded attributes must be quoted, for example, "^attrToExclude".) The **`@objectClassName`** notation can be used to compare only attributes that are defined for a given objectclass.

The **`ldap-diff`** tool can be used on servers actively being modified by checking differing entries multiple times without reporting false positives due to replication delays. By default, it will re-check each entry twice, pausing two seconds between checks. These settings can be configured with the **`--numPasses`** and **`--secondsBetweenPass`** options. If the utility cannot make a clean comparison on an entry, it will list any exceptions in comments in the output file.

The directory user specified for performing the searches must be privileged enough to see all of the entries being compared and to issue a long-running, unindexed search. For the Directory Server, the out-of-the-box **`cn=Directory Manager`** user has these privileges, but you can assign the necessary privileges by setting the following attributes in the user entry:

```
ds-cfg-default-root-privilege-name: unindexed-search
ds-cfg-default-root-privilege-name: bypass-acl
ds-rlim-size-limit: 0
ds-rlim-time-limit: 0
ds-rlim-idle-time-limit: 0
ds-rlim-lookthrough-limit: 0
```

The **`ldap-diff`** tool tries to make efficient use of memory, but it must store the DNs of all entries in memory. For directories that contain hundreds of millions of entries, the tool might require a few gigabytes of memory. If the progress of the tool slows dramatically, it might be running low on memory. The memory used by the **`ldap-diff`** tool can be customized by editing the **`ldap-diff.java-args`** setting in the **`config/java.properties`** file and running the **`dsjavaproperties`** command.

If you do not want to use a subtree search filter, you can use an input file of DNs for the source, target, or both. The format of the file can accept various syntaxes for each DN:

```
dn: cn=this is the first dn
dn: cn=this is the second
 dn and it is wrapped
cn=this is the third dn
# The following DN is base-64 encoded
dn::
Y249ZG9uJ3QgeW91IGhhdmUgYmV0dGVyIHRoaW5ncyB0byBkbyB0aGFuIHNlZSB3aGF0IHRoaXMgc2F5cw==

# There was a blank line above
dn: cn=this is the final entry.
```

Caution	Do not manually update the servers when the tool identifies differences between two servers involved in replication. First contact your authorized support provider for explicit confirmation, because manual updates to the servers risk introducing additional replication conflicts.

To Compare Two Directory Servers Using ldap-diff

1. Use `ldap-diff` to compare the entries in two directory server instances. Ignore the `user-password` attribute due to the one-way password hash used for the password storage scheme.

   ```
   $ bin/ldap-diff --outputLDIF difference.ldif \
     --sourceHost server1.example.com --sourcePort 1389 \
     --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
     --targetHost server2.example.com --targetPort 2389 \
     --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
     --baseDN dc=example,dc=com --searchFilter "(objectclass=*)" "^userpassword"
   ```

2. Open the output file in a text editor to view any differences. The file is set up so that you can re-apply the changes without any modification to the file contents. The file shows any deletes, modifies, and then adds from the perspective of the source server as the authoritative source.

   ```
   # This file contains the differences between two LDAP servers.
   #
   # The format of this file is the LDIF changes needed to bring server
   # ldap://server1.example.com:1389 in sync with server ldap://server2.exam-
   ple.com:2389.
   #
   # These differences were computed by first issuing an LDAP search at both
   # servers under base DN dc=example,dc=com using search filter (objectclass=*)
   # and search scope SUB to first retrieve the DNs of all entries. And then each
   # entry was retrieved from each server and attributes: [^userpassword] were
   # compared.
   #
   # Any entries that were out-of-sync were compared a total of 3 times waiting a
   # minimum of 2 seconds between each attempt to account for replication delays.
   #
   # Comparison started at [24/Feb/2010:10:34:20 -0600]

   # The following entries were present only on ldap://server2.example.com:2389 and
   need to
   # be deleted.

   # This entry existed only on ldap://server1.example.com:1389
   # Note: this entry might be incomplete.  It only includes attributes:
   # [^userpassword]dn: uid=user.200,ou=People,dc=example,dc=com
   # objectClass: person
   # objectClass: inetOrgPerson
   ... (more attributes not shown) ...
   # st: DC
   dn: uid=user.200,ou=people,dc=example,dc=com
   changetype: delete

   # The following entries were present on both servers but were out of sync.

   dn: uid=user.199,ou=people,dc=example,dc=com
   changetype: modify
   add: mobile
   mobile: +1 300 848 9999
   -
   delete: mobile
   mobile: +1 009 471 1808

   # The following entries were missing on ldap://server2.example.com:2389 and need to
   be
   # added.
   ```

```
# This entry existed only on ldap://server2.example.com:2389
# Note: this entry might be incomplete.  It only includes attributes:
# [^userpassword]
dn: uid=user.13,ou=People,dc=example,dc=com
changetype: add
objectClass: person
objectClass: inetOrgPerson
... (more attributes not shown) ...
# Comparison completed at [24/Feb/2010:10:34:25 -0600]
```

To Compare Configuration Entries Using ldap-diff

Use **ldap-diff** to compare the configuration entries in two directory server instances. The filter searches all configuration entries. Ignore the **userpassword** attribute due to the password storage scheme that uses a one-way hashing algorithm.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN cn=config --searchFilter "(objectclass=*)" \
  "^userpassword"
```

To Compare Entries Using Source and Target DN Files

Use **ldap-diff** to compare the entries in two directory server instances. In the following example, the utility uses a single DN input file for the source and target servers, so that no search filter is used. Ignore the **userpassword** attribute due to the password storage scheme that uses a one-way hashing algorithm.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN "dc=example,dc=com" --sourceDNsFile input-file.ldif \
  --targetDNsFile input-file.ldif "^userpassword"
```

To Compare Directory Servers for Missing Entries Only Using ldap-diff

Use **ldap-diff** to compare two directory servers and return only those entries that are missing on one of the servers using the **--missingOnly** option, **which** can significantly reduce the run-time for this utility.

```
$ bin/ldap-diff --outputLDIF difference.ldif \
  --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN dc=example,dc=com --searchFilter "(objectclass=*)" "^userpassword"
  --missingOnly
```

Working with Virtual Attributes

The UnboundID Directory Server provides mechanisms to support virtual attributes in an entry. Virtual attributes are abstract, dynamically generated attributes that are invoked through an LDAP operation, such as **`ldapsearch`**, but are not stored in the directory backend. While most virtual attributes are operational attributes, providing processing-related information that the server requires, the virtual attribute subsystem allows you to create user-defined virtual attributes to suit your directory requirements.

Viewing the List of Default Virtual Attributes

The Directory Server has a default set of virtual attributes that can be viewed using the **`dsconfig`** tool. Some virtual attributes are enabled by default and are useful for most applications. You can easily enable or disable each virtual attribute using the **`dsconfig`** tool.The default set of virtual attributes are described in the table below. You can enable or disable these attributes using the **`dsconfig`** tool.

TABLE 7-7. Description of Virtual Attributes

Virtual Attributes	Description
ds-entry-checksum	Generates a simple checksum of an entry’s contents, which can be used with an LDAP assertion control to ensure that the entry has not been modified since it was last retrieved.
ds-instance-name	Generates the name of the Directory Server instance from which the associated entry was read. This virtual attribute can be useful in load-balancing environments to determine the instance from which an entry was retrieved.
entryDN	Generates an **`entryDN`** operational attribute in an entry that holds a normalized copy of the entry's current distinguished name (DN). Clients can use this attribute in search filters.
hasSubordinates	Creates an operational attribute that has a value of TRUE if the entry has subordinate entries.
isMemberOf	Generates an **`isMemberOf`** operational attribute that contains the DNs of the groups in which the user is a member.
numSubordinates	Generates an operational attribute that returns the number of child entries. While there is no cost if this operational attribute is enabled, there could be a performance cost if it is requested. Note that this operational attribute only returns the number of immediate children of the node.
subschemaSubentry	A special entry that provides information in the form of operational attributes about the schema elements defined in the server. It identifies the location of the schema for that part of the tree. • ldapSyntaxes - set of attribute syntaxes • matchingRules - set of matching rules • matchingRuleUse - set of matching rule uses • attributeTypes - set of attribute types • objectClasses - set of object classes • nameForms - set of name forms • dITContentRules - set of DIT content rules • dITStructureRules - set of DIT structure rules

TABLE 7-7. **Description of Virtual Attributes**

Virtual Attributes	Description
User Defined Virtual Attribute	Generates virtual attributes with user-defined values in entries that match the criteria defined in the plug-in's configuration. User-defined virtual attributes are intended to specify a hard-coded value for entries matching a given set of criteria.
Virtual Static Member	Generates a member attribute whose values are the DNs of the members of a specified virtual static group. Virtual static groups are best used in client applications with a large number of entries that can only support static groups and obtains all of its membership from a dynamic group. Do not modify the filter in the Virtual Static Member attribute as it is an advanced property and modifying it can lead to undesirable side effects.
Virtual Static Uniquemember	Generates a **`uniqueMember`** attribute whose values are the DNs of the members of a specified virtual static group. Virtual static groups are best used in client applications with a large number of entries that can only support static groups and obtains all of its membership from a dynamic group. Do not modify the filter in the Virtual Static Uniquemember attribute as it is an advanced property and modifying it can lead to undesirable side effects.

To View the List of Default Virtual Attributes Using dsconfig Non-Interactive Mode

Use **`dsconfig`** to view the virtual attributes.

```
$ bin/dsconfig list-virtual-attributes --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret --no-prompt
```

Viewing Virtual Attribute Properties

Each virtual attribute has basic properties that you can view using the **`dsconfig`** tool. The complete list of properties is described in the *UnboundID Directory Server Configuration Reference*. Some basic properties are as follows:

- **Description**. A description of the virtual attribute.
- **Enabled**. Specifies whether the virtual attribute is enabled for use.
- **Base-DN**. Specifies the base DNs for the branches containing entries that are eligible to use this virtual attribute. If no values are given, the server generates virtual attributes anywhere in the server.
- **Group-DN**. Specifies the DNs of the groups whose members can use this virtual attribute. If no values are given, the group membership is not taken into account when generating the virtual attribute. If one or more group DNs are specified, then only members of those groups are allowed to have the virtual attribute.
- **Filter**. Specifies the filters that the server applies to entries to determine if they require virtual attributes. If no values are given, then any entry is eligible to have a virtual attribute value generated.

To View Virtual Attribute Properties

Use **dsconfig** to view the properties of a virtual attribute.

```
$ bin/dsconfig get-virtual-attribute-prop --name isMemberOf --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret --no-prompt
```

Enabling a Virtual Attribute

You can enable a virtual attribute using the **dsconfig** tool. If you are using **dsconfig** in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Enable a Virtual Attribute using dsconfig Interactive Mode

1. Use **dsconfig** to enable a virtual attribute. Specify the connection port, bind DN, password, and host information. Then type the LDAP connection parameter for your directory server: 1 for LDAP, 2 for SSL, 3 for StartTLS.

   ```
   $ bin/dsconfig --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
     --bindPassword secret --hostname localhost
   ```

2. On the UnboundID Directory Server configuration console main menu, type **o** to change the object menu, and then type the number corresponding to Standard.

3. On the UnboundID Directory Server configuration console main menu, type the number corresponding to virtual attributes.

4. On the Virtual Attribute management menu, type **3** to view and edit an existing virtual attribute.

5. From the list of existing virtual attributes on the system, select the virtual attribute to work with. For this example, type the number corresponding to the **numSubordinates** virtual attribute.

6. On the **numSubordinates** Virtual Attribute Properties menu, type **2** to enable the virtual attribute. On the Enabled Property menu for the **numSubordinates** virtual attribute, type **2** to change the value to **TRUE**.

7. On the **numSubordinates** Virtual Attribute Properties menu, type **f** to apply the changes.

8. Verify that the virtual attribute is enabled. Note that this example assumes you have configured the group entries.

   ```
   $ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
     --bindPassword secret --baseDN dc=example,dc=com \
     "(ou=People)" numSubordinates

   dn: ou=People,dc=example,dc=com
   numSubordinates: 1000
   ```

To Enable a Virtual Attribute Using dsconfig Non-Interactive Mode

Use **`dsconfig`** to enable the **`numSubordinates`** virtual attribute.

```
$ bin/dsconfig set-virtual-attribute-prop --name numSubordinates \
  --set enabled:true --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

Creating User-Defined Virtual Attributes

If you require your own user-defined virtual attributes, you can create them using the **`dsconfig`** tool. User-defined virtual attributes are intended to specify a hard-coded value for entries matching a given set of criteria. You must ensure that the new virtual attribute conforms to your plug-in schema, otherwise you will see an error message when you configure it. If you are using **`dsconfig`** in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Create a User-Defined Virtual Attribute in Interactive Mode

1. Use **`dsconfig`** to enable a virtual attribute. Specify the connection port, bind DN, password, and host information. Then type the LDAP connection parameter for your directory server: 1 for LDAP, 2 for SSL, 3 for StartTLS.

2. On the UnboundID Directory Server configuration console main menu, type **`o`** to change the object menu, and then type **`2`** to select Standard.

3. On the UnboundID Directory Server configuration console main menu, type the number corresponding to virtual attributes.

4. On the Virtual Attribute management menu, type **`2`** to create a new virtual attribute.

5. Next, you can use an existing virtual attribute as a template for your new attribute, or your can create a new attribute from scratch. In this example, type **`n`** to create a new Virtual Attribute from scratch.

6. On the Virtual Attribute Type menu, enter a number corresponding to the type of virtual attribute that you want to create. In this example, type the number corresponding to User Defined Virtual Attribute.

7. Next, enter a name for the new virtual attribute. In this example, enter "dept-number".

8. On the Enabled Property menu, enter **`1`** to set the property to true (enable).

9. On the Attribute-Type Property menu, type the **`attribute-type`** property for the new virtual attribute. You can enter the OID number or attribute name. The **`attribute-type`** property must conform to your schema. For this example, type "departmentNumber".

10. Enter the value for the virtual attribute, and then press Enter or Return to continue. In this example, enter some number corresponding to a department, such as "110", and then type Enter or Return to continue.

11. On the User Defined Virtual Attributes menu, enter a description for the virtual attribute. Though optional, this step is useful if you plan to create a lot of virtual attributes.

12. Enter the option to change the value, and then type a description of the virtual attribute. In this example, enter "dept number for engineering staff".

13. On the User Defined Virtual Attribute menu, type the number corresponding to the filter.

14. On the Filter Property menu, enter the option to add one or more filter properties, type the filter, and then type Enter or Return to continue. In this example, type **`"(&(object-class=Person)(ou=engineering))"`**. Press **`1`** to use the filter value entered.

15. On the User Defined Virtual Attribute menu, type **`f`** to finish creating the virtual attribute.

16. Verify that the attribute was created successfully. This example assumes that you have set up a user (**`uid=user.1`**) with the **`ou=engineering`** attribute and another user (**`uid=user.6`**) with the **`ou=accounting`** attribute. Use **`ldapsearch`** to search for a user in the engineering staff (**`ou=engineering`**). You will notice the **`departmentNumber`** attribute. If you run the command again, using another user who is not in engineering (for example, **`ou=accounting`**), the attribute will not be present.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN dc=example,dc=com "(uid=user.1)"
dn: uid=user.1,ou=People,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
ou: Engineering
departmentNumber: 110

$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --bindDN dc=example,dc=com (uid=user.6)
dn: uid=user.6,ou=People,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
...(more attributes)...
ou: Accounting
```

To Create a Virtual Attribute Using dsconfig in Non-Interactive Mode

You can also use **`dsconfig`** in non-interactive command-line mode to create a virtual attribute. The following command sets up the **`dept-number`** virtual attribute introduced in the previous section:

```
$ bin/dsconfig create-virtual-attribute --name dept-number --type user-defined \
  --set "description:Dept number for engineering staff" \
  --set enabled:true --set attribute-type:departmentNumber \
  --set "filter:(&(objectclass=person)(ou=engineering))" --set "value:110" \
  --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

Creating Mirror Virtual Attributes

The UnboundID Directory Server provides a feature to mirror the value of another attribute in the same entry or mirror the value of the same or a different attribute in an entry referenced by the large entry. For example, given a DIT where users have a manager attributed with a value of the DN of the employee as follows:

```
dn: uid=apeters,ou=people,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
manager:uid=jdoe,ou=people,dc=example,dc=com
uid: apeters
... (more attributes) ...
```

You can set up a mirror virtual attribute, so that the returned value for the **managerName** virtual attribute can be the **cn** value of the entry referenced by the **manager** attribute as follows:

```
$ bin/ldapsearch --port 1389 --baseDN dc=example,dc=com "(uid=apeters)" \
  dn: uid=apeters,ou=people,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
manager:uid=jdoe,ou=people,dc=example,dc=com
managerName: John Doe
uid: apeters... (more attributes not shown) ...
```

To Create a Mirror Virtual Attribute in Non-Interactive Mode

You can also use **dsconfig** in non-interactive command-line mode to create a mirror virtual attribute. The following example sets up the **managerName** virtual attribute introduced in the previous section:

1. Update the schema to define the **managerName** attribute. In a text editor, create a file with the following schema definition for the attribute and save it as **98-myschema.ldif**, for example, in the **<server-root>/config/schema** folder. You can optionally add the attribute to an object class. See "Extending the Schema Using a Custom Schema File" on page 332 for more information.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.4 NAME 'managerName'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{256}
  X-ORIGIN 'UnboundID Directory Server Example' )
```

2. Restart the Directory Server.

```
$ bin/stop-ds --restart
```

3. Use **dsconfig** to create the virtual attribute.

```
$ bin/dsconfig create-virtual-attribute \
  --name "managerName" --type mirror \
  --set "description:managerName from manager cn" \
  --set enabled:true --set attribute-type:managerName \
  --set source-attribute:cn --set source-entry-dn-attribute:manager \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

4. Verify the mirror virtual attribute by searching for an entry.

```
$ bin/ldapsearch --port 1389 --baseDN dc=example,dc=com "(uid=apeters)"

dn: uid=apeters,ou=People,dc=example,dc=com
... (attributes) ...
manager: uid=jdoe,ou=People,dc=example,dc=com
managerName: John Doe
```

Editing a Virtual Attribute

You can edit virtual attributes using the **dsconfig** tool. You must ensure that the virtual attribute conforms to your plug-in schema, otherwise you will see an error message when you edit the virtual attribute. If you are using **dsconfig** in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Edit a Virtual Attribute Using dsconfig in Non-Interactive Mode

Use **dsconfig** to change a property's value.

```
$ bin/dsconfig set-virtual-attribute-prop --name dept-number --set "value:111" \
  --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

Deleting a Virtual Attribute

You can delete virtual attributes using the **dsconfig** tool. If you are using **dsconfig** in interactive command-line mode, you can access the virtual attribute menu on the Standard object menu.

To Delete a Virtual Attribute

Use **dsconfig** to delete an existing virtual attribute.

```
$ bin/dsconfig delete-virtual-attribute --name dept-number \
  --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

Working with Large Attributes

The Directory Server supports storing large attributes (for example, JPEG images, voice files, XML blobs, etc.) in an alternate backend that is separate from the rest of the entry contents. The virtual attribute can be used to make the large attributes appear as part of the entry contents if it is requested by a client or is needed during the course of processing the operation. The use of an alternate backend makes it possible to configure caching differently for those large attributes and makes it efficient for dealing with entries when large attribute values are not needed.

When the Large Attribute virtual attribute is configured, the Large Attribute Backend must be created to support it. The backend uses a specialized version of the existing Oracle Berkeley DB JE backend but does not support indexing as clients are not intended to access it directly. Only one large attribute backend can be used in the Directory Server. Large attributes can be created, updated, or removed using LDAP operations, and all changes for those values will be transparently applied to the Large Attribute backend.

If replication is enabled in the Directory Server, there is no change to the way that replication works. Changes involving large attribute values will be processed in exactly the same way as changes to other attributes. If an operation involves a change to both normal and large attributes, the entire set of changes will be replicated as a single atomic unit.

Note Large attributes should only be used for limited operations due to the possible performance penalty for accessing and decoding the large value every time a client accesses the entry. Also, large attributes require a change in backup strategy due to the extra time it may take to back up the Large Attributes backend. For more information, see "Running Backups for Large Attributes" on page 193.

Large attribute support currently has the following limitations:

- **No Batched or Interactive Transaction Support**. Large attributes cannot be altered in any operation that is part of a batched or interactive transaction. Any attempt to run a batched or interactive transaction will fail.
- **Limited Subtree Delete Request Control Support**. The subtree delete request control cannot be used to remove a subtree if the base entry or any of its subordinates have large attribute values.
- **Importing Large Attributes**. When importing data into a backend with one or more large attribute types defined, the values for those large attributes will be excluded from the entry being imported, and they will not automatically be populated in the Large Attribute backend. Contact your authorized support provider for assistance to import data into the Large Attribute backend.
- **No Historical Data for Replication**. Historical data is not maintained for large attributes. As a result, replication is unable to resolve conflicting LDAP modify operations involving large attributes.

To Create a Large Attribute Virtual Attribute

The following example shows how to create a Large Attribute Virtual Attribute, labelled **`myLargeAttr`**.

1. Create the Large Attributes backend using **`dsconfig`**.

   ```
   $ bin/dsconfig create-backend --backend-name Large-Attributes \
     --type large-attribute --set enabled:true --port 1389 \
     --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret --no-prompt
   ```

2. Set a global configuration property to access the Large Attributes backend.

   ```
   $ bin/dsconfig set-global-configuration-prop \
     --set large-attribute-backend:Large-Attributes --port 1389 \
     --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret --no-prompt
   ```

3. Define an operational attribute in the server schema that will be used to hold large attribute values. Making it an operational attribute ensures that it is only returned to clients if it is explicitly requested and allows it to be included in any entry regardless of the object classes that entry contains. For example, the following content could be included in a **`config/schema/98-largeattr.ldif`** file for this purpose:

   ```
   dn: cn=schema
   objectClass: top
   objectClass: ldapSubentry
   objectClass: subschema
   attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.1
     NAME 'myLargeAttr'
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
     SINGLE-VALUE
     USAGE directoryOperation )
   ```

4. Stop the server using **`bin/stop-ds`**, and then restart the server using, **`bin/start-ds`**.

5. Set the **`large-attribute-type`** property for all backends that may include entries with large attribute information. For example:

   ```
   $ bin/dsconfig set-backend-prop --backend-name userRoot \
     --set large-attribute-type:myLargeAttr --port 1389 --bindPassword secret \
     --no-prompt
   ```

6. Create the virtual attribute, **`myLargeAttr`** to appear with the rest of the entry contents when it is requested by the client.

   ```
   $ bin/dsconfig create-virtual-attribute --name "myLargeAttr Large Attribute" \
     --type large-attribute --set enabled:true --set attribute-type:myLargeAttr \
     --port 1389 --bindPassword secret --no-prompt
   ```

7. Modify an entry to add a value for the large attribute type. For example, create a file **`/tmp/my-changes.ldif`** with the following contents:

   ```
   dn: uid=user.0,ou=People,dc=example,dc=com
   changetype: modify
   replace: myLargeAttr
   myLargeAttr:This is a really big value
   ```

   Then, apply this modification to the server using the **`ldapmodify`** command:

   ```
   $ bin/ldapmodify --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
     --bindPassword secret --filename /tmp/my-changes.ldif
   ```

8. Use the **ldapsearch** tool to verify that the **myLargeAttr** attribute appears as a virtual attribute when the entry is retrieved, and that it is only retrieved when explicitly requested. For example, the **myLargeAttr** attribute should not be included when it is not requested as follows:

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --searchScope sub --baseDN dc=example,dc=com "(uid=user.0)"
dn: uid=user.0,ou=People,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
uid: user.0
givenName:User
uid: user.0
cn:User 0
```

9. Next, verify that the attribute is returned when it is explicitly requested.

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --searchScope sub \
  --baseDN dc=example,dc=com "(uid=user.0)" myLargeAttr
dn: uid=user.0,ou=People,dc=example,dc=com
myLargeAttr:This is a really big value
```

10. Finally, verify that the attribute is not returned when using the Real Attributes Only control, which prevents any virtual attributes from being returned with the entry contents. In this case, the control can be used to verify that the entry is being returned from the large attribute backend by means of a virtual attribute, rather than being directly included in the entry contents.

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --searchScope sub --baseDN dc=example,dc=com \
  --control realAttributesOnly "(uid=user.0)" myLargeAttr
dn: uid=user.0,ou=People,dc=example,dc=com
```

Running Backups for Large Attributes

Deployments using Large Attributes introduce added complexity to the backup process due to the extra time it takes to save the Large Attributes backend. The **backup** command-line tool can only save an individual backend at a time and thus cannot atomically save the userRoot and Large Attributes backends in one step.

A basic strategy to back up your Directory Server is as follows:

- Perform daily ZFS snapshots of the filesystem with the Large Attributes and the userRoot backends. Perform these backups at the same time to provide a level of backup atomiticity. Keep several days worth of ZFS snapshots. The number that you can keep will depend on how much free disk space you have and the modification rate of the databases. If you need to restore the database for any reason other than a disk failure, then you can just roll back to the latest snapshots. To do this, you need to stop the server and roll back both filesystems.

- To back up the databases to a separate filesystem (that is, on another disk array), we recommend using the **`zfs send`** and **`zfs receive`** commands to copy these snapshots from the primary systems to the backup array. The **`zfs send`** command can operate on a full filesystem (or snapshot) or the difference between two snapshots. This method provides a means to have incremental-like backups and eliminates the **`backup`** command's limitation in that it cannot store two full backups.

Note	You can consider taking snapshots even more frequently than once per day since there really is no additional overhead for snapshots that are taken between two other snapshots.

To Perform the First Backup of the Large Attributes and userRoot Backends (Solaris)

Before you begin, you should set up separate filesystems on the disk array to mirror the Large Attribute backend and the **`userRoot`** backend separately.

1. Create a ZFS snapshot of the Large Attribute backend and the filesystem with the **`userRoot`** database.
2. Run **`zfs send`** and **`zfs receive`** to copy these snapshots over to the backup array. This step could take hours depending on the size of your Large Attributes backend.
3. Take ZFS snapshots on the backup disk array.

To Perform Additional Incremental and userRoot Backends (Solaris)

1. If you have reached the limits on the number of snapshots that you can keep, delete the oldest Large Attribute backend and userRoot snapshots. This step takes about a second to run.
2. Create a ZFS snapshot of the Large Attribute backend and the filesystem with the **`userRoot`** database. This step takes about a second to run.
3. Use **`zfs send`** and **`zfs receive`** with the current snapshot and the previous snapshot to incrementally update the backup array with the contents of the current snapshot.
4. If you have reached the limits on the number of snapshots that you are keeping on the backup array, delete the oldest Large Attribute backend and userRoot snapshots. This step takes about a second to run.
5. Take ZFS snapshots on the backup disk array.

To Restore the Large Attributes and userRoot Backends from Snapshots (Solaris)

1. Stop the Directory Server.

2. Select the snapshot to which you want to roll back. Generally, you should use the more recent snapshop, then the snapshot prior to the most recent one should be used.

3. Roll back the snapshots of the userRoot and the Large Attributes backends.

4. Restart the Directory Server.

To Restore the Large Attributes and userRoot Backends from the Disk Array (ZFS)

1. Stop the Directory Server.

2. Select the snapshot to which you want to roll back. This will most likely be the most recent one.

3. Copy the Large Attribute backend database files from the backup array to the regular array. Recursively copy over the complete server root filesystem including the **`userRoot`** database over to the internal disks. By recursively copying the server root filesystem, you also copy over the replication changelog and the LDAP Change Log (i.e., **`cn=changelog`**) data.

4. Restart the Directory Server.

Working with Groups

An LDAP group is a type of entry that represents a collection of users. Groups are often used by external clients, for example, to control who has access to a particular application or features. However, they may also be used internally by the server to control its behavior. For example, groups can be used by the access control, criteria, or virtual attribute subsystems.

The specific ways in which clients create and interact with a particular group depends on the type of group being used. In general, there are three primary ways in which clients attempt to use groups:

- To determine whether a specified user is a member of a particular group.
- To determine the set of groups in which a specified user is a member.
- To determine the set of all users that are members of a particular group.

The remainder of this section provides an overview of Directory Server groups concepts and provides procedures on setting up and querying groups in the Directory Server.

Overview of Group Concepts

The Directory Server provides the following types of groups:

- **Static Groups**. A static group is an entry that contains an explicit list of **`member`** or **`unique-member`** attributes, depending on its particular structural object class. Static groups are ideal for relatively small, infrequently changing elements. Once the membership list grows, static groups become more difficult to manage as any change in a member base DN must also be changed in the group. Static groups use one of three structural object classes: **`groupOfNames`**, **`groupOfUniqueNames`**, and **`groupOfEntries`**.

  The Directory Server also supports nested groups, in which a parent group entry contains child attributes whose DNs reference another group. Nested groups are a flexible means to organize entries that provide inherited group membership and privileges. To maintain good performance throughput, a group cache is enabled by default. The cache supports both *nested static* and *nested dynamic* groups.

- **Dynamic Groups**. A dynamic group has its membership list determined by search criteria using an LDAP URL. Dynamic groups solve the scalability issues encountered for static groups and searches are efficient, constant-time operations. However, if searches range over a very large set of data, performance could be affected.

- **Virtual Static Groups**. A virtual static group is a combination of both static and dynamic groups, in which each member in a group is a virtual attribute that is dynamically generated when invoked. Virtual static groups solve the scalability issues for clients that can only support static groups and are best used when the application targets a search operation for a specific member. Virtual static groups are not good for applications that need to retrieve the entire membership list as the process for constructing the entire membership list can be expensive.

About the isMemberOf and isDirectMemberOf Virtual Attributes

The existence of static, nested, dynamic and virtual static groups can make it unnecessarily complex to work with groups in the server, particularly because the ways you interact with them are so different. And the fact that static groups can use three different structural object classes (not counting the auxiliary class for virtual static groups) does not make things any easier.

To make group operations simpler, the UnboundID Directory Server provides the ability to generate either an **`isMemberOf`** and **`isDirectMemberOf`** virtual attributes in user entries. These attributes dramatically simplify the process for making group-related determinations in a manner that is consistent across all types of groups.

The value of the **`isMemberOf`** virtual attribute is a list of DNs of all groups (including static, nested, dynamic, and virtual static groups) in which the associated user is a member. The value of the **`isDirectMemberOf`** virtual attribute is a list of DNs of all non-nested static groups of which the user is a *direct* member. While the **`isMemberOf`** virtual attribute is enabled by default, you must enable the **`isDirectMemberOf`** virtual attribute if you plan to use it. Run the following **`dsconfig`** command to enable the virtual attribute:

```
dsconfig set-virtual-attribute-prop --name isDirectMemberOf --set enabled:true
```

Because **`isMemberOf`** and **`isDirectMemberOf`** are operational attributes, only users who specifically have been granted the privilege to view operational attributes can see it. The default set of access control rules do not allow any level of access to user data. The only access that is granted is what is included in user-defined access control rules, which is generally given to a **`uid=admin`** administrator account. It is always a best practice to restrict access to operational and non-operational attributes to the minimal set of users that need to see them. The root bind DN, **`cn=Directory Manager`**, has the privilege to view operational attributes by default.

To determine whether a user is a member of a specified group using the **`isDirectMemberOf`** virtual attribute, simply perform a base-level search against the user's entry with an equality filter targeting the **`isDirectMemberOf`** attribute with a value that is the DN of the target group. The following table illustrates this simple base-level search:

Base DN	`uid=john.doe,ou=People,dc=example,dc=com`
Scope	`base`
Filter	`(isDirectMemberOf=cn=Test Group,ou=Groups,dc=exam-ple,dc=com)`
Requested Attributes	`1.1`

If this search returns an entry, then the user is a member of the specified group. If no entry is returned, then the user is not a member of the given group.

To determine the set of all groups in which a user is a member, simply retrieve the user's entry with a base-level search and include the **`isMemberOf`** attribute:

Base DN	`uid=john.doe,ou=People,dc=example,dc=com`
Scope	`base`
Filter	`(objectClass=*)`
Requested Attributes	`isMemberOf`

To determine the set of all members for a specified group, issue a subtree search with an equality filter targeting the **`isMemberOf`** attribute with a value that is the DN of the target group and requesting the attributes you wish to have for member entries:

Base DN	`dc=example,dc=com`
Scope	`sub`
Filter	`(isMemberOf=cn=Test Group,ou=Groups,dc=example,dc=com)`
Requested Attributes	`cn` `mail`

Note that if this filter targets a dynamic group using an unindexed search, then this may be an expensive operation. However, it will not be any more expensive than retrieving the target group and then issuing a search based on information contained in the member URL.

For static groups, this approach has the added benefit of using a single search to retrieve information from all user entries, whereas it would otherwise be required to retrieve the static group and then perform a separate search for each member's entry.

About Static Groups

A static group contains an explicit membership list where each member is represented as a DN-valued attribute. There are three types of static groups supported for use in the directory server:

- **groupOfNames**. A static group that is defined with the `groupOfNames` structural object class and uses the `member` attribute to hold the DNs of its members. RFC 4519 requires that the `member` attribute be required in an entry. However, the Directory Server has relaxed this restriction by making the `member` attribute optional so that the last member in the group can be removed. The following entry depicts a group defined with the `groupOfNames` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
cn: Test Group
member: uid=user.1,ou=People,dc=example,dc=com
member: uid=user.2,ou=People,dc=example,dc=com
member: uid=user.3,ou=People,dc=example,dc=com
```

- **groupOfUniqueNames**. A static group that is defined with the `groupOfUniqueNames` structural object class and uses the `uniquemember` attribute to hold the DNs of its members. RFC 4519 requires that the `uniquemember` attribute be required in an entry. However, the Directory Server has relaxed this restriction by making the `uniquemember` attribute optional so that the last member in the group can be removed. The following entry depicts a group defined with the `groupOfUniqueNames` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
cn: Test Group
uniquemember: uid=user.1,ou=People,dc=example,dc=com
uniquemember: uid=user.2,ou=People,dc=example,dc=com
uniquemember: uid=user.3,ou=People,dc=example,dc=com
```

- **groupOfEntries**. Groups defined with the `groupOfEntries` object class and that use the `member` attribute to hold the DNs of its members. This group specifies that the `member` attribute is optional to ensure that the last member can be removed from the group. Although the draft proposal (draft-findlay-ldap-groupofentries-00.txt) has expired, the Directory Server supports this implementation. The following entry depicts a group defined with the `groupOfNames` object class:

```
dn: cn=Test Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfEntries
cn: Test Group
member: uid=user.1,ou=People,dc=example,dc=com
member: uid=user.2,ou=People,dc=example,dc=com
member: uid=user.3,ou=People,dc=example,dc=com
```

Creating Static Groups

You can configure a static group by adding it using an LDIF file or from the command line. Static groups contain a membership list of explicit DNs specified by the `uniquemember` attribute.

To Create a Static Group

1. Open a text editor, and then create a group entry in LDIF. Make sure to include the **`groupOfUniquenames`** object class and **`uniquemember`** attributes. If you did not have **`ou=groups`** set up in your directory tree, you can add it in the same file. When done, save the file as **`static-group.ldif`**. The following example LDIF file creates two groups, **`cn=Development`** and **`cn=QA`**.

```
dn: ou=groups,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: groups

dn: cn=Development,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Development
ou: groups
uniquemember: uid=user.14,ou=People,dc=example,dc=com
uniquemember: uid=user.91,ou=People,dc=example,dc=com
uniquemember: uid=user.180,ou=People,dc=example,dc=com

dn: cn=QA,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: QA
ou: groups
uniquemember: uid=user.0,ou=People,dc=example,dc=com
uniquemember: uid=user.1,ou=People,dc=example,dc=com
uniquemember: uid=user.2,ou=People,dc=example,dc=com
```

2. Use **`ldapmodify`** to add the group entries to the directory.

```
$ bin/ldapmodify --defaultAdd --filename static-group.ldif
```

3. Verify the configuration by using the virtual attribute **`isDirectMemberOf`** that checks membership for a non-nested group. By default, the virtual attribute is disabled by default, but you can enable it using **`dsconfig`**.

```
$ bin/dsconfig set-virtual-attribute-prop --name isDirectMemberOf \
  --set enabled:true
```

4. Use **`ldapsearch`** to specifically search the **`isDirectMemberOf`** virtual attribute to determine if **`uid=user.14`** is a member of the **`cn=Development`** group. In this example, assume that the administrator has the privilege to view operational attributes.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isDirectMemberOf

dn: uid=user.14,ou=People,dc=example,dc=com
isDirectMemberOf: cn=Development Group,ou=groups,dc=example,dc=com
```

5. Typically, you would want to use the group as a target in access control instructions. Open a text editor, create an **`aci`** attribute in an LDIF file, and save the file as **`dev-group-aci.ldif`**. You can create a similar ACI for the QA group, which is not shown in this example.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
```

```
aci: (target ="ldap:///ou=People,dc=example,dc=com")
  (targetattr != "cn || sn || uid")
  (targetfilter ="(ou=Development)")
  (version 3.0; acl "Development Group Permissions";
    allow (write) (groupdn = "ldap:///cn=Development,ou=groups,dc=example,dc=com");)
```

6. Add the file using the **ldapmodify** tool.

```
$ bin/ldapmodify --filename dev-group-aci.ldif
```

To Add a Member to a Group

- Add a new value for the membership attribute that specifies the DN of the user to be added. The following example adds a new member, **user.4**:

```
dn: cn=QA,ou=Groups,dc=example,dc=com
changetype: modify
add: member
member: uid=user.4,ou=People,dc=example,dc=com
```

To Remove a Member from a Group

- Similarly, to remove a member from a static group, remove that user's DN from the membership attribute. The following example removes the DN of **user.1**:

```
dn: cn=QA,ou=Groups,dc=example,dc=com
changetype: modify
delete: member
member: uid=user.1,ou=People,dc=example,dc=com
```

Searching Static Groups

The following sections describe how to compose searches to determine if a user is a member of a static group, to determine all the static groups in which a user is a member, and to determine all the members of a static group.

Determining if a User is a Static Group Member

To determine whether a user is a member of a specified group, perform a base-level search to retrieve the group entry with an equality filter looking for the membership attribute of a value equal to the DN of the specified user. For best performance, you will generally want to include a specific attribute list (just "**cn**", or "**1.1**" to request that no attributes be returned) so that the entire member list is not returned. For example, to determine whether the user "**uid=john.doe,ou=People,dc=example,dc=com**" is a member of the **groupOfNames** static group "**cn=Test Group,ou=Groups,dc=example,dc=com**", issue a search with the following criteria:

TABLE 8. Search Criteria for a Single User's Membership in a Static Group

Base DN	`cn=Test Group,ou=Groups,dc=example,dc=com`
Scope	`base`

TABLE 8. Search Criteria for a Single User's Membership in a Static Group

Filter	`(uniquemember=uid=john.doe,ou=People,dc=example,dc=com)`
Requested Attributes	`1.1`

If the search returns an entry, then the user is a member of the specified group. If the search does not return an entry, then the user is not a member of the group. If you do not know the membership attribute for the specified group (it could be either a **`member`** or **`uniqueMember`** attribute), then you may want to revise the filter so that it allows either one, as follows:

```
(|(member=uid=john.doe,ou=People,dc=example,dc=com)
(uniqueMember=uid=john.doe,ou=People,dc=example,dc=com))
```

Determining the Static Groups to Which a User Belongs

To determine the set of all static groups in which a user is specified as a member, perform a subtree search based at the top of the DIT. The search filter must be configured to match any type of static group in which the specified user is a member. For example, the following criteria may be used to determine the set of all static groups in which the user, **`uid=john.doe,ou=People,dc=example,dc=com`**, is a member:

TABLE 9. Search Criteria for Determining All the Static Groups for a User

Base DN	`dc=example,dc=com`
Scope	`sub`
Filter	(|((&(objectClass=groupOfNames) (member=uid=john.doe,ou=People,dc=example,dc=com)) (&(objectClass=groupOfUniqueNames)(uniqueMember=uid=john.doe,ou=Peo- ple,dc=example,dc=com)) (&(objectClass=groupOfEntries) (member=uid=john.doe,ou=People,dc=example,dc=com)))
Requested Attributes	`1.1`

Every entry returned from the search represents a static group in which the specified user is a member. Note that the filter can be simplified if you are using only one type of group object in your deployment, such as **`groupOfUniqueNames`**.

Determining the Members of a Static Group

To determine all of the members for a static group, simply retrieve the group entry including the membership attribute. The returned entry will include the DNs of all users that are members of that group. For example, the following criteria may be used to retrieve the list of all members for the group **`cn=Test Group,ou=Groups,dc=example,dc=com`**:

TABLE 10. Search Criteria for All of a Static Group's Members

Base DN	`cn=Test Group,ou=Groups,dc=example,dc=com`
Scope	`base`
Filter	`(objectClass=*)`
Requested Attributes	`member` `uniqueMember`

If you want to retrieve additional information about the members, such as attributes from member entries, you must issue a separate search for each member to retrieve the user entry and the desired attributes.

Monitoring the Group Membership Cache

The Directory Server logs information at startup about the memory consumed by the group membership cache. This hard-coded cache contains information about all of the group memberships for internal processing, such as ACIs. The group membership cache is enabled by default.

The information about this cache is logged to the standard output log (**`server.out`**) and the standard error log. When using groups, you can use the log information to tune the server for best performance.

For example, at startup the server logs a message like the following to the **`server.out`** log:

```
[16/Aug/2011:17:14:39.462 -0500] category=JEB severity=NOTICE msgID=1887895587
msg="The database cache now holds 3419MB of data and is 32 percent full"
```

The error log will contain something like the following:

```
[16/Aug/2011:18:40:39.555 -0500] category=EXTENSIONS severity=NOTICE
msgID=1880555575 msg="'Group cache (174789 static group(s) with 7480151 total
memberships and 1000002 unique members, 0 virtual static group(s), 1 dynamic
group(s))' currently consumes 149433592 bytes and can grow to a maximum of
149433592 bytes"
```

Using the Entry Cache to Improve the Performance of Large Static Groups

The UnboundID Directory Server provides an entry cache implementation, which allows for fine-grained control over the kinds of entries that may be held in the cache. You can define filters to specify the entries included in or excluded from the cache, and you can restrict the cache so that it holds only entries with at least a specified number of values for a given set of attributes.

Under most circumstances, we recommend that the Directory Server be used *without* an entry cache. The Directory Server is designed to efficiently retrieve and decode entries from the database in most cases. The database cache is much more space-efficient than the entry cache, and heavy churn in the entry cache can adversely impact garbage collection behavior.

However, if the Directory Server contains very large static groups, such as those containing thousands or millions of members, and clients need to frequently retrieve or otherwise interact with these groups, then you may want to enable an entry cache that holds only large static groups.

In directories containing very large static groups, you can define an entry cache to hold only those large static groups. This entry cache should have an include filter that matches only group entries (for example, **`"(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfEntries))"`**). The filter contains a minimum value count so that only groups with a large number of members (such as those with at least 100

member or **uniqueMember** values) will be included. The Directory Server provides an entry cache implementation with these settings although it is disabled by default.

To Enable the Entry Cache

- Run **dsconfig** to enable the entry cache.

```
$ bin/dsconfig set-entry-cache-prop --cache-name "Static Group Entry Cache" \
--set enabled:true
```

To Configuring Your Own Entry Cache for Large Groups

- You can create your own entry cache for large groups using the **dsconfig create-entry-cache** subcommand as follows:

```
$ bin/dsconfig create-entry-cache --type fifo --set enabled:true \
--set cache-level:10 \
--set-max-entries:175000 \
--set "include-filter:(objectClass=groupOfUniqueNames)" \
--set min-cache-entry-value-count:10000 \
--set min-cache-entry-attribute:uniquemember
```

Monitoring the Entry Cache

You can monitor the memory consumed by your entry cache using the **entry-cache-info** property in the periodic stats logger. You can retrieve the monitor entry over LDAP by issuing a search on **baseDN="cn=monitor"** using **filter="(objectClass=ds-fifo-entry-cache-monitor-entry)"**. For example, the entry might appear as follows:

```
dn: cn=Static Group Entry Cache Monitor,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-fifo-entry-cache-monitor-entry
objectClass: extensibleObject
cn: Static Group Entry Cache Monitor
cacheName: Static Group Entry Cache
entryCacheHits: 6416407
entryCacheTries: 43069073
entryCacheHitRatio: 14
maxEntryCacheSize: 12723879900
currentEntryCacheCount: 1
maxEntryCacheCount: 175000
entriesAddedOrUpdated: 1
evictionsDueToMaxMemory: 0
evictionsDueToMaxEntries: 0
entriesNotAddedAlreadyPresent: 0
entriesNotAddedDueToMaxMemory: 0
entriesNotAddedDueToFilter: 36652665
entriesNotAddedDueToEntrySmallness: 0
lowMemoryOccurrences: 0
percentFullMaxEntries: 0
jvmMemoryMaxPercentThreshold: 75
jvmMemoryCurrentPercentFull: 24
jvmMemoryBelowMaxMemoryPercent: 51
isFull: false
capacityDetails: NOT FULL: The JVM is using 24% of its available memory.
Entries can be added to the cache until the overall JVM memory usage reaches the
```

```
configured limit of 75%. Cache has 174999  remaining entries before reaching
the configured limit of 175000.
```

By default, the entry cache memory is set to 75%, with a maximum of 90%.

Maintaining Referential Integrity with Static Groups

The Directory Server can automatically update references to an entry whenever that entry is removed or renamed in a process called *referential integrity*. For example, if a user entry is deleted, then the referential integrity plugin will remove that user from any static groups in which the user was a member (this is not necessary for dynamic groups, since no explicit membership is maintained). Similarly, if a modify DN operation is performed to move or rename a user entry, then the referential integrity plugin updates static groups in which that user is a member with the new user DN.

Referential integrity support is disabled by default, but may be enabled using the `dsconfig` tool as follows:

```
$ bin/dsconfig set-plugin-prop --plugin-name "Referential Integrity" \
  --set enabled:true
```

Other configuration attributes of note for this plugin include:

- **`attribute-type`**
  This attribute specifies the names or OIDs of the attribute types for which referential integrity will be maintained. By default, referential integrity is maintained for the **`member`** and **`uniqueMember`** attributes. Any attribute types specified must have a syntax of either distinguished name (**`OID "1.3.6.1.4.1.1466.115.121.1.12"`**) or name and optional UID (**`OID "1.3.6.1.4.1.1466.115.121.1.34"`**). The specified attribute types must also be indexed for equality in all backends for which referential integrity is to be maintained.

- **`base-dn`**
  This attribute specifies the subtrees for which referential integrity will be maintained. If one or more values are provided, then referential integrity processing will only be performed for entries which exist within those portions of the DIT. If no values are provided (which is the default behavior), then entries within all public naming contexts will be included.

- **`log-file`**
  This attribute specifies the path to a log file that may be used to hold information about the DNs of deleted or renamed entries. If the plugin is configured with a nonzero update interval, this log file helps ensure that appropriate referential integrity processing occurs even if the server is restarted.

- **`update-interval`**
  This attribute specifies the maximum length of time that a background thread may sleep between checks of the referential integrity log file to determine whether any referential integrity processing is required. By default, this attribute has a value of **`"0 seconds"`**, which indicates that all referential integrity processing is to be performed synchronously before a response is returned to the client. A duration greater than 0 seconds indicates that

referential integrity processing will be performed in the background and will not delay the response to the client.

In the default configuration, where referential integrity processing is performed synchronously, the throughput and response time of delete and modify DN operations may be adversely impacted because the necessary cleanup work must be completed before the response to the original operation can be returned. Changing the configuration to use a non-zero update interval alleviates this performance impact because referential integrity processing uses a separate background thread and does not significantly delay the response to delete or modify DN operations.

However, performing referential integrity processing in a background thread may introduce a race condition that may adversely impact clients that delete a user and then immediately attempt to re-add it and establish new group memberships. If referential integrity processing has not yet been completed for the delete, then newly established group memberships may be removed along with those that already existed for the previous user. Similarly, if the newly-created user is to be a member of one or more of the same groups as the previous user, then attempts by the client to re-establish those memberships may fail if referential integrity processing has not yet removed the previous membership. For this reason, we recommend that the default synchronous behavior be maintained unless the performance impact associated with it is unacceptable and clients are not expected to operate in a manner that may be adversely impacted by delayed referential integrity processing.

Note The internal operations of the referential integrity plugin are not replicated. So, in a replicated topology, you must enable the referential integrity plugin consistently on all servers in the topology to ensure that changes made by the referential integrity plugin are passed along to a replication server.

For more information about administering the referential integrity plugin, see Chapter 6, “Configuring the Directory Server”.

Tuning the Index Entry Limit for Large Groups

The Directory Server uses indexes to improve database search performance and provide consistent search rates regardless of the number of database objects stored in the DIT. You can specify an index entry limit property, which defines the maximum number of entries that are allowed to match a given index key before it is no longer maintained by the server. If the index keys have reached this limit (which is 4000 by default), then you must rebuild the indexes using the **`rebuild-index`** tool as follows:

```
bin/rebuild-index --baseDN dc=example,dc=com --index objectclass
```

In the majority of directory environments, the default index entry limit value of 4000 entries should be sufficient. However, with group-related processing, it may be necessary to increase the index entry limit. For directories containing more than 4000 groups with the same structural object class (i.e., more than 4000 **`groupOfNames`** entries, 4000 **`groupOfUniqueNames`** entries, 4000 **`groupOfEntries`** entries, or 4000 **`groupOfURLs`** entries), then you may want to increase the index entry limit for the **`objectClass`** attribute so that it has a value larger than the maximum number of group entries of each type. Set **`index-entry-limit`** property using a command line like the following:

```
bin/dsconfig set-local-db-index-prop --backend-name userRoot \
--index-name objectClass --set index-entry-limit:175000
```

As an alternative, a separate backend may be created to hold these group entries, so that an unindexed search in that backend yields primarily group entries. If you make no changes, then the internal search performed at startup to identify all groups and any user searches looking for groups of a given type may be very expensive.

For directories in which any single user may be a member of more than 4000 static groups of the same type, you may need to increase the index entry limit for the **`member`** and/or **`uniqueMember`** attribute to a value larger than the maximum number of groups in which any user is a member. If you do not increase the limit, then searches to retrieve the set of all static groups in which the user is a member may be unindexed and therefore very expensive.

Using Dynamic Groups

Dynamic groups contain a set of criteria used to identity members rather than maintaining an explicit list of group members. If a new user entry is created or if an existing entry is modified so that it matches the membership criteria, then the user will be considered a member of the dynamic group. Similarly, if a member's entry is deleted or if it is modified so that it no longer matches the group criteria, then the user will no longer be considered a member of the dynamic group.

In the Directory Server, dynamic groups include the **`groupOfURLs`** structural object class and use the **`memberurl`** attribute to provide an LDAP URL defining the membership criteria. The base, scope, and filter of the LDAP URL will be used in the process of making the determination, and any other elements present in the URL will be ignored. For example, the following entry defines a dynamic group in which all users below **`dc=example,dc=com`** with an **`employeeType`** value of **`contractor`** will be considered members of the group:

```
dn: cn=Sales Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: Sales Group
memberURL:  ldap:///dc=example,dc=com??sub?(employeeType=contractor)
```

Assuming that less than 80,000 entries have the **`employeeType`** of **`contractor`**, you need to create the following index definition to evaluate the dynamic group:

```
dsconfig create-local-db-index --backend-name userRoot \
--index-name employeeType --set index-entry-limit:80000 \
--set index-type:equality
```

Creating Dynamic Groups

You can configure a dynamic group in the same manner as static groups using an LDIF file. Dynamic groups contain a membership list of attributes determined by search filter using an LDAP URL. You must use the **`groupOfURLs`** object class and the **`memberURL`** attribute.

To Create a Dynamic Group

1. Assume for this example that **`uid=user.0`** has an **`ou=Engineering`** attribute indicating that he or she is a member of the Engineering department.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.0)" ou
dn: uid=user.0,ou=People,dc=example,dc=com
ou: Engineering
```

2. Also assume that **`uid=user.15`** is not part of any group. Use **`ldapsearch`** to verify that **`uid=user.15`** is not part of any group. In a later step, we will add the user to the dynamic group.

```
$ bin/ldapsearch -h server1.example.com -p 389 -D "uid=admin,dc=example,dc=com" \
  -w password -b dc=example,dc=com -s sub "(uid=user.15)" ou
dn: uid=user.15,ou=People,dc=example,dc=com
```

3. Open a text editor, and then create a dynamic group entry in LDIF. The LDIF defines the dynamic group to include all users who have the **`ou=Engineering`** attribute. When done, save the file as **`add-dynamic-group.ldif`**.

```
dn: cn=eng-staff,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfURLs
ou: groups
cn: eng-staff
memberURL: ldap:///ou=People,dc=example,dc=com??sub?(ou=Engineering)
```

4. Use **`ldapmodify`** to add the group entry to the directory.

```
$ bin/ldapmodify --defaultAdd --filename add-dynamic-group.ldif
```

5. Use **`ldapsearch`** to specifically search the **`isMemberOf`** virtual attribute to determine if **`uid=user.0`** is a member of the **`cn=Engineering`** group or any other group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.0)" isMemberOf
dn: uid=user.0,ou=People,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

6. If your directory tree is relatively small (under 1 million entries), you can search for the all users in the group that meet the search criteria (**`ou=Engineering`**). For very large databases, it is not practical to run a database-wide search for all users as there can be a performance hit on the directory server. The following command returns the DNs of entries that are part of the **`cn=eng-staff`** dynamic group and sorts them in ascending order by the **`sn`** attribute.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --sortOrder sn \
"(isMemberOf=cn=eng-staff,ou=groups,dc=example,dc=com)" dn
```

7. Add **`uid=user.15`** to the **`eng-staff`** group by adding an **`ou=Engineering`** attribute to the entry. This step highlights an advantage of dynamic groups: you can make a change in an entry without explicitly adding the DN to the group as you would with static groups. The entry will be automatically added to the **`eng-staff`** dynamic group.

```
$ bin/ldapmodify
dn: uid=user.15,ou=People,dc=example,dc=com
changetype: modify
```

```
add: ou
ou: Engineering
```

8. Use **ldapsearch** to check if the user is part of the **cn=eng-staff** dynamic group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com --searchScope sub "(uid=user.15)" \
isMemberOf

dn: uid=user.15,ou=People,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

Searching Dynamic Groups

The following sections describe how to compose searches to determine if a user is a member of a dynamic group, to determine all the dynamic groups in which a user is a member, and to determine all the members of a dynamic group.

Determining if a User is a Dynamic Group Member

To determine whether a user is a member of a specific dynamic group, you must verify that the user's entry is both within the scope of the member URL and that it matches the filter contained in that URL. You can verify that a user's entry is within the scope of the URL using simple client-side only processing. Evaluating the filter against the entry on the client side can be more complicated. While possible, particularly in clients that are able to perform schema-aware evaluation, a simple alternative is to simply perform a base-level search to retrieve the user's entry with the filter contained in the member URL. For example, to determine whether the user **uid=john.doe,ou=People,dc=example,dc=com** is a member of the dynamic group with the above member URL, issue a search with the following criteria:

TABLE 11. Search Criteria for a Single User's Membership in a Dynamic Group

Base DN	`uid=john.doe,ou=People,dc=example,dc=com`
Scope	`base`
Filter	`(employeeType=contractor)`
Requested Attributes	`1.1`

If the search returns an entry, then the user is a member of the specified group. If the search does not return any entries, then the user is not a member of the group.

Determining the Dynamic Groups to Which a User Belongs

To determine the set of all dynamic groups in which a user is a member, first perform a search to find all dynamic group entries defined in the server. You can do this using a subtree search with a filter of **"(objectClass=groupOfURLs)"**. You should retrieve the **memberURL** attribute so that you can use the logic described in the previous section to determine whether the specified user is a member of each of those groups. For example, to find the set of all dynamic groups defined in the `dc=example,dc=com` tree, issue the a search with the following criteria:

TABLE 12. **Search Criteria for Determining All the Dynamic Groups for a User**

Base DN	`dc=example,dc=com`
Scope	`sub`
Filter	(objectClass=groupOfURLs)
Requested Attributes	`memberURL`

Each entry returned will be a dynamic group definition. You can use the base, scope, and filter of its `memberURL` attribute to determine whether the user is a member of that dynamic group.

Determining the Members of a Dynamic Group

To determine all members of a dynamic group, issue a search using the base, scope, and filter of the member URL. The set of requested attributes should reflect the attributes desired from the member user entries, or "`1.1`" if no attributes are needed. For example, to retrieve the `cn` and `mail` attributes from the group described above use the following search:

TABLE 13. **Search Criteria for All of a Dynamic Group's Members**

Base DN	`dc=example,dc=com`
Scope	`sub`
Filter	`(employeeType=contractor)`
Requested Attributes	`cn` `mail`

Note that this search may be expensive if the associated filter is not indexed or if the group contains a large number of members.

Using Dynamic Groups for Internal Operations

You can use dynamic groups for internal operations, such as ACI or component evaluation. The directory server performs the `memberurl` parsing and internal LDAP search; however the internal search operation may not be performed with access control rules applied to it.

For example, the following dynamic group represents an organization's employees within the same department:

```
dn: cn=department 202,ou=groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: department 202
owner: uid=user.1,ou=people,dc=example,dc=com
owner: uid=user.2,ou=people,dc=example,dc=com
memberURL: ldap:///ou=People,dc=example,dc=com??sub?
  (&(employeeType=employee)(departmentNumber=202))
description: Group of employees in department 202
```

The above group could be referenced from an ACI at the `dc=example,dc=com` entry. For example:

```
aci: (targetattr="employeeType")
  (version 3.0; acl "Grant write access to employeeType" ;
   allow (all) groupdn="ldap:///cn=department 202,ou=groups,dc=example,dc=com";)
```

Any user matching the above filter can bind to the directory with their entry and modify the `employeeType` attribute within any entry under `dc=example,dc=com`.

Creating Nested Groups

The UnboundID Directory Server supports nested groups, where the DN of an entry that defines a group is included as a member in the parent entry. For example, the `uniquemember` attributes of the `cn=Engineering Group` consist of the `cn=Developers Group` and the `cn=QA Group` respectively. Each group member can consist of static and/or dynamic groups.

```
dn: cn=Engineering Group,ou=Groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Engineering Group
uniquemember: cn=Developers Group,ou=Groups,dc=example,dc=com
uniquemember: cn=QA Group,ou=Groups,dc=example,dc=com
```

In practice, nested groups are not commonly used for several reasons. LDAP specifications do not directly address the concept of nested groups, and some servers do not provide any level of support for them. Supporting nested groups in LDAP clients is not trivial, and many directory-enabled applications that can interact with groups do not provide any support for nesting. If nesting support is not needed in your environment, or if nesting support is only required for clients but is not needed for server-side evaluation (such as for groups used in access control rules, criteria, virtual attributes, or other ways that the server may need to make a membership determination), then this support should be disabled.

Nested group support is enabled by default. In order to use nested groups without the performance hit, the Directory Server uses a group cache, which is enabled by default. The group cache operates for *nested static* and *nested dynamic* groups. The Directory Server provides a new monitoring entry for the group cache, `cn=Group Cache,cn=Monitor`.

To Create Nested Groups

1. Nested groups are enabled by default. Open a text editor, and then create a group entry in LDIF. Make sure to include the `groupOfUniquenames` object class and `uniquemember` attributes. If you did not have `ou=groups` set up in your directory tree, then you can add it in the same file. When done, save the file as `nested-group.ldif`.

```
dn: ou=groups,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: groups

dn: cn=Engineering Group,ou=Groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
cn: Engineering Group
uniquemember: cn=Developers Group,ou=Groups,dc=example,dc=com
uniquemember: cn=QA Group,ou=Groups,dc=example,dc=com
```

2. Use **ldapmodify** to add the group entry to the directory.

```
$ bin/ldapmodify --defaultAdd --filename nested-group.ldif
```

3. Verify the configuration by using the **isMemberOf** virtual attribute that checks group membership for an entry. By default, the virtual attribute is enabled. Use **ldapsearch** to specifically search the **isMemberOf** virtual attribute to determine if **uid=user.14** is a member of the **cn=Development group**. In this example, assume that the administrator has the privilege to view operational attributes.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(uid=user.14)" isMemberOf

dn: uid=user.14,ou=People,dc=example,dc=com
isMemberOf: cn=Development Group,ou=groups,dc=example,dc=com
```

4. Typically, you would want to use the group as a target in access control instructions. Open a text editor, create an **aci** in LDIF, and save the file as **eng-group-aci.ldif**.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target ="ldap:///ou=People,dc=example,dc=com")
  (targetattr != "cn || sn || uid")
  (targetfilter ="(ou=Engineering Group)")
  (version 3.0; acl "Engineering Group Permissions";
    allow (write) (groupdn = "ldap:///cn=Engineering Group,ou=groups,dc=example,dc=com");)
```

5. Add the file using the **ldapmodify** tool.

```
$ bin/ldapmodify --filename eng-group-aci.ldif
```

Using Virtual Static Groups

Static groups can be easier to interact with than dynamic groups, but large static groups can be expensive to manage, and can require a large amount of memory to hold in the internal group cache. The Directory Server provides a third type of group that makes it possible to get the efficiency and ease of management of a dynamic group while allowing clients to interact with it as a static group. A virtual static group is a type of group that references another group and provides access to the members of that group as if it were a static group.

To configure a virtual static group, create an entry that has a structural object class of either **groupOfNames** or **groupOfUniqueNames** and an auxiliary class of **ds-virtual-static-group**. It should also include a **ds-target-group-dn** attribute, whose value is the dynamic group from which the virtual static group should obtain its members. For example, the following will create a virtual static group that exposes the members of the **cn=Sales Group,ou=Groups,dc=example,dc=com** dynamic group as if it were a static group:

```
dn: cn=Virtual Static Sales Group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
objectClass: ds-virtual-static-group
cn: Virtual Static Sales Group
ds-target-group-dn: cn=Sales Group,ou=Groups,dc=example,dc=com
```

Note that you must also enable a virtual attribute that allows the **member** attribute to be generated based on membership for the target group. A configuration object for this virtual attribute does exist in the server configuration, but is disabled by default. To enable it, issue the following change:

```
dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
     --set enabled:true
```

If you want to use virtual static groups with the **`groupOfUniqueNames`** object class, then you will also need to enable the **`"Virtual Static uniqueMember"`** virtual attribute in the same way.

Creating Virtual Static Groups

If your application only supports static groups but has scalability issues, then using a virtual static group is a possible solution. A virtual static group uses a virtual attribute that is dynamically generated when called, after which the operations that determine group membership are passed to another group, such as a dynamic group. You must use the **`ds-virtual-static-group`** object class and the **`ds-target-group-dn`** virtual attribute.

Virtual static groups are best used when determining if a single user is a member of a group. It is not a good solution if an application accesses the full list of group members due to the performance expense of constructing the list. If you have a small database and an application that requires that the full membership list be returned, you must also enable the **`allow-retrieving-membership`** property for the **`Virtual Static uniqueMember`** virtual attribute using the **`dsconfig`** tool.

To Create a Virtual Static Group

1. Open a text editor, and then create a group entry in LDIF. The entry contains the **`groupOfUniqueNames`** object class, but in place of the **`uniquemember`** attribute is the **`ds-target-group-dn`** virtual attribute, which is part of the **`ds-virtual-static-group`** auxiliary object class. The **`ds-target-group-dn`** refers to the **`cn=eng-staff`** dynamic group, which was created previously in “Creating Dynamic Groups” on page 206. When done, save the file as **`add-virtual-static-group.ldif`**.

```
dn: cn=virtual static,ou=groups,dc=example,dc=com
objectclass: top
objectclass: groupOfUniqueNames
objectclass: ds-virtual-static-group
ou: groups
cn: virtual static
ds-target-group-dn: cn=eng-staff,ou=groups,dc=example,dc=com
```

2. Use **`ldapmodify`** to add the virtual static group entry to the directory.

```
$ bin/ldapmodify --defaultAdd --filename add-virtual-static-group.ldif
```

3. Use **`dsconfig`** to enable the **`Virtual Static uniqueMember`** attribute, which is disabled by default.

```
$ bin/dsconfig set-virtual-attribute-prop --name "Virtual Static uniqueMember" \
  --set enabled:true
```

4. In the previous section, we set up **`uid=user.0`** to be part of the **`cn=eng-staff`** dynamic group. Use **`ldapsearch`** with the **`isMemberOf`** virtual attribute to determine if **`uid=user.0`** is part of the virtual static group.

```
$ bin/ldapsearch --baseDN dc=example,dc=com "(objectClass=*)" isMemberOf
dn: uid=user.0,ou=People,dc=example,dc=com
isMemberOf: cn=virtual static,ou=groups,dc=example,dc=com
isMemberOf: cn=eng-staff,ou=groups,dc=example,dc=com
```

5. Use **`ldapsearch`** to determine if **`uid=user.0`** is a member of the virtual static group. You should see the returned **`cn=virtualstatic`** entry if successful.

```
$ bin/ldapsearch --baseDN "cn=virtual static,ou=Groups,dc=example,dc=com" \
"(&(objectClass=groupOfUniqueNames)\
(uniqueMember=uid=user.0,ou=People,dc=example,dc=com))"
```

6. Next, try searching for a user that is not part of the **`cn=eng-staff`** dynamic group (e.g., **`uid=user.20`**). Nothing will be returned.

```
$ bin/ldapsearch --baseDN "cn=virtual static,ou=Groups,dc=example,dc=com" \
"(&(objectClass=groupOfUniqueNames)\
(uniqueMember=uid=user.20,ou=People,dc=example,dc=com))"
```

Searching Virtual Static Groups

Because virtual static groups behave like static groups, the process for determining whether a user is a member of a virtual static group is identical to the process for determining whether a user is a member of a conventional static group. Similarly, the process for determining all virtual static groups in which a user is a member is basically the same as the process for determining the set of all real static groups in which a user is a member. In fact, the query provided in the static groups discussion returns virtual static groups in addition to real static groups, because the structural object class of a virtual static group is the same as the structural object class for an actual static group.

You can also retrieve a list of all members of a virtual static group in the same way as a real static group: simply retrieve the **`member`** (or **`uniqueMember`**) attribute of the desired group. However, because virtual static groups are backed by dynamic groups and the process for retrieving member information for dynamic groups can be expensive, virtual static groups do not allow retrieving the full set of members by default. The virtual attribute used to expose membership can be updated to allow this with a configuration change such as the following:

```
dsconfig set-virtual-attribute-prop --name "Virtual Static member" \
     --set allow-retrieving-membership:true
```

Note	Because this can be an expensive operation, we recommend that the option to allow retrieving virtual static group membership be left disabled unless it is required.

Summary of Commands to Search for Group Membership

The following summary of commands show the fastest way to retrieve direct or indirect member DNs for groups.

- To retrieve direct member (non-nested) DNs of group **`"cn=group.1,ou=groups,dc=example,dc=com"`**.

```
$ bin/ldapsearch --baseDN "cn=group.1,ou=Groups,dc=example,dc=com" "(object-
Class=*)" uniqueMember member
```

- To retrieve direct member entries (non-nested) under **`"dc=example,dc=com"`** of group **`"cn=group.1,ou=groups,dc=example,dc=com"`**. This is useful when attributes from member entries are used in the filter or being returned.

```
$ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com" "(isDirectMem-
berOf=cn=group.1,ou=Groups,dc=example,dc=com)"
```

- To retrieve group DNs in which user **`"uid=user.2,ou=people,dc=example,dc=com"`** is a direct member (non-nested, static groups).

```
$ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com" "(object-
Class=*)" isDirectMemberOf
```

- To retrieve all member entries under **`ou=people,dc=example,dc=com`** of group **`"cn=group.1,ou=groups,dc=example,dc=com"`**.

```
$ bin/ldapsearch --baseDN "ou=people,dc=example,dc=com" "(isMem-
berOf=cn=group.1,ou=Groups,dc=example,dc=com)"
```

- To retrieve the group DNs in which user **`"uid=user.2,ou=people,dc=example,dc=com"`** is a member.

```
$ bin/ldapsearch --baseDN "uid=user.2,ou=people,dc=example,dc=com" "(object-
Class=*)" isMemberOf
```

Migrating Oracle DSEE Groups

You can migrate DSEE static and dynamic groups to UnboundID Directory Server groups. The following sections outline the procedures for migrating DSEE static groups to both UnboundID static groups and virtual static groups, as well as how to migrate dynamic groups. For information about the differences in access control evaluation between DSEE and UnboundID Directory Server, see "Migrating ACIs from Oracle DSEE to UnboundID Directory Server" on page 303.

Migrating DSEE Static Groups to UnboundID Static Groups

UnboundID Directory Server supports static LDAP groups with structural object classes of **`groupOfNames`**, **`groupOfUniqueNames`**, or **`groupOfEntries`**. In general, static groups may be imported without modification.

A FIFO entry cache can be enabled to cache group-to-user mappings, which improves performance when accessing very large entries, though at the expense of greater memory consumption. UnboundID Directory Server provides an out-of-the-box FIFO entry cache object for this purpose. This object must be explicitly enabled using **`dsconfig`** as described in "Monitoring the Group Membership Cache" on page 202.

To Migrate DSEE Static Groups

1. Run the **`migrate-ldap-schema`** tool to enumerate any schema differences between the DSEE deployment and the UnboundID deployment.

2. Run the **`migrate-sun-ds-config`** to enumerate any configuration differences between the DSEE deployment and the UnboundID deployment.

3. Import or configure any necessary schema and/or configuration changes recorded by the above tools.

4. Import the existing users and groups with **`import-ldif`**.

5. From the UnboundID Directory Server installation directory, open the **`sun-ds-compatibility.dsconfig`** file in the docs folder using a text editor.

6. Find the FIFO Entry Cache section and, after reading the accompanying comments, enable the corresponding **`dsconfig`** command by removing the comment character ("#").

```
dsconfig set-entry-cache-prop \
--cache-name "Static Group Entry Cache" \
--set enabled:true
```

7. Enable the Referential Integrity Plugin. This will ensure that references to an entry are automatically updated when the entry is deleted or renamed.

```
dsconfig set-plugin-prop --plugin-name "Referential Integrity \
--set enabled:true
```

   If this Directory Server is part of a replication topology, you should enable the Referential Integrity Plugin for each replica.

Migrating DSEE Static Groups to Virtual Static Groups

In many cases, electing to use virtual static groups in place of static groups can produce marked performance gains without any need to update client applications. The specifics of a migration to virtual static groups varies depending on the original DIT, but the general approach involves identifying common membership traits for all members of each group and then expressing those traits in the form of an LDAP URL.

In the following example, the common membership trait for all members of the **`All Users`** group is the parent DN **`ou=People,dc=example,dc=com`**. In other cases, a common attribute may need to be used. For example, groups based on the location of its members could use the `l` or `st` attributes.

If group membership criteria cannot be derived from existing attribute in the entry, then seamlessly migrating from static groups to virtual static groups may not be possible.

To Migrate DSEE Static Groups to Virtual Static Groups

In the following example, consider the common case of an "All Users" group, which contains all entries under the parent DN `"ou=People,dc=example,dc=com"`. When implemented as a virtual static group, this group may have a large membership set without incurring the overhead of a static group.

1. First, create a dynamic group as follows:

```
dn: cn=Dynamic All Users,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfURLs
cn: Dynamic All Users
memberURL: ldap:///ou=People,dc=example,dc=com??sub?(objectClass=person)
```

2. Next, create a virtual static group that references the dynamic group:

```
dn: cn=All Users,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
objectClass: ds-virtual-static-group
cn: All Users
ds-target-group-dn: cn=Dynamic All Users,ou=Groups,dc=example,dc=com
```

3. Finally, the `Virtual Static uniqueMember` virtual attribute must be enabled to populate the `All Users` group with `uniqueMember` virtual attributes.

```
dsconfig set-virtual-attribute-prop --name "Virtual Static uniqueMember" \
--set enabled:true
```

4. Confirm that the virtual static group is correctly configured by checking a user's membership in the group:

```
ldapsearch --baseDN "cn=All Users,ou=Groups,dc=example,dc=com" \
--searchScope base \
"(uniqueMember=uid=user.0,ou=People,dc=example,dc=com)" 1.1

dn: cn=All Users,ou=Groups,dc=example,dc=com
```

5. The ability to list all members of a virtual static group is disabled by default. You may enable this feature, but only if specifically required by a client application.

```
dsconfig set-virtual-attribute-prop --name "Virtual Static uniqueMember"
--set allow-retrieving-membership:true
```

Note	The virtual static group may also be implemented using the `groupOfNames` object class instead of `groupOfUniqueNames`. In that case, you must update the `Virtual Static member` configuration object instead of the `Virtual Static uniqueMember` configuration object.

Migrating Dynamic Groups

Like DSEE, UnboundID Directory Server supports dynamic groups with the **`groupOfURLs`** object class. In general, dynamic groups may be imported without modification.

To Migrate DSEE Dynamic Groups

1. Run the **`migrate-ldap-schema`** tool to enumerate any schema differences between the DSEE deployment and the UnboundID deployment.

2. Run the **`migrate-sun-ds-config`** tool to enumerate any configuration differences between the DSEE deployment and the UnboundID deployment.

3. Import or configure any necessary schema and/or configuration changes recorded by the above tools.

4. Import the existing users and groups with **`import-ldif`**.

Encrypting and Protecting Sensitive Data

The Directory Server provides several ways that you can protect sensitive information in the server. You can enable on-disk encryption for data in backends as well as in the changelog and the replication databases, and you can also protect sensitive attributes by limiting the ways that clients may interact with them.

Overview of Data Encryption

The Directory Server provides the ability to encrypt stored data to prevent unauthorized access. *Data encryption* itself applies only to the on-disk representation of the data, so that when clients access that data or when data is replicated between servers, it will be presented in unencrypted form. The server already supports SSL and StartTLS for encrypting communication with clients and between server instances to ensure that all access to the data is encrypted. Encryption is supported for several types of backends, including the local DB, large attribute, changelog and replication backends. Because the changelog and replication databases support encryption in addition to the regular data backends, all references to potentially sensitive data can be encrypted.

When data encryption is enabled in the server, then that encryption will be applied to the *entire* entry. Applying the encryption to the entire entry rather than only to a specified set of attributes has been shown to have a negligible impact on server performance, but it dramatically simplifies the server configuration, allows for a more compact representation of the encrypted data, and also ensures that other potential references to sensitive data (e.g., replication metadata held in the **`ds-sync-hist`** attribute in the same entry) are also be protected.

Full-entry encryption helps ensure the security of the data by eliminating concerns about forgetting to choose non-obvious attributes that might contain sensitive information.

Note While data encryption is applied to full entries, that encryption is not extended to indexes referencing content in those entries. As such, it is strongly recommended that any attributes that contain particularly sensitive data not be indexed.

About the Encryption Settings Database

The encryption settings database is a repository that the server uses to hold information for encrypting and decrypting data. The database contains any number of encryption settings definitions that specifies information about the cipher transformation and key used for encryption and decryption.

Although the encryption settings database can have multiple encryption settings definitions, only one of them can be designated as the *preferred* definition. The preferred encryption settings definition is the one that will be used for any subsequent encryption operations. Any existing data that has not yet been encrypted remains unencrypted until it is rewritten (e.g., as a result of a modify or modify DN operation, or if the data is exported to LDIF and re-imported). Similarly, if you introduce a new preferred encryption settings definition, then any existing encrypted data will continue to use the previous definition until it is rewritten. As such, if you do change the preferred encryption settings definition for the server, then it is important to retain the previous definitions until you are confident that no remaining data uses those older keys.

Supported Encryption Ciphers and Transformations

The set of encryption ciphers that are supported by the Directory Server is limited to those ciphers supported by the JVM in which the server is running. For specific reference information about the algorithms and transformations available in all compliant JVM implementations, see the following:

- Java Cryptography Architecture Reference Guide (available at http://download.oracle.com/javase/6/docs/techs/guides/security/crypto/CryptoSpec.html).
- Java Cryptography Architecture Standard Algorithm Name Documentation (available at http://download.oracle.com/javase/6/docs/techs/guides/security/StandardNames.html).

When configuring encryption, the cipher to be used must be specified using a key length (in bits) and either a cipher algorithm name (e.g., "AES") or a full cipher transformation which explicitly specifies the mode and padding to use for the encryption (e.g., "AES/CBC/PKCS5Padding"). If only a cipher algorithm is given, then the default mode and padding for that algorithm will be automatically selected.

The following cipher algorithms and key lengths have been tested using the Sun/Oracle JVM using Java 1.6.0 update 25:

TABLE 7-1.

Cipher Algorithm	Key Length (Bits)
AES	128
Blowfish	128
DES	64
DESede	192
RC4	128

Note By default, some JVM implementations may come with limited encryption strength, which may restrict the key lengths that can be used. For example, the Sun/Oracle JVM does not allow AES with 192-bit or 256-bit keys unless the unlimited encryption strength policy files are downloaded and installed (available at https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jce_policy-6-oth-JPR@CDS-CDS_Developer).

Using the encryptions-settings Tool

The **encryption-settings** tool provides a mechanism for interacting with the server's encryption settings database. It may be used to list the available definitions, create new definitions, delete existing definitions, and indicate which definition should be the preferred definition. It may also be used to export definitions to a file for backup purposes and to allow them to be imported for use in other Directory Server instances.

To List Available Encryption Settings Definitions

Use the **encryption-settings** tool with the **list** subcommand to display the set of available encryption settings definitions. This subcommand does not take any arguments. For each definition, it will include the unique identifier for the definition, as well as the cipher transformation and key length that will be used for encryption and whether it is the preferred definition.

```
$ bin/encryption-settings list
Encryption Settings Definition ID:  4D86C7922F71BB57B8B5695D2993059A26B8FC01
     Preferred for New Encryption:  false
     Cipher Transformation:  DESede
     Key Length (bits):  192

Encryption Settings Definition ID:  F635E109A8549651025D01D9A6A90F7C9017C66D
     Preferred for New Encryption:  true
     Cipher Transformation:  AES
     Key Length (bits):  128
```

Creating Encryption Settings Definitions

To create a new encryption settings definition, use the `create` subcommand. This subcommand takes the following arguments:

- **--cipher-algorithm {algorithm}**. Specifies the base cipher algorithm that should be used. This should just be the name of the algorithm (e.g., "AES", "DES", "DESede", "Blowfish", "RC4", etc.). This argument is required.
- **--cipher-transformation {transformation}**. Specifies the full cipher transformation that should be used, including the cipher mode and padding algorithms (e.g., "AES/CBC/PKCS5Padding"). This argument is optional, and if it is not provided, then the JVM-default transformation will be used for the specified cipher algorithm.
- **--key-length-bits {length}**. Specifies the length of the encryption key in bits (e.g., 128). This argument is required.
- **--set-preferred**. Indicates that the new encryption settings definition should be made the preferred definition and therefore should be used for subsequent encryption operations in the server. When creating the first definition in the encryption settings database, it will automatically be made the preferred definition.

To Create an Encryption Settings Definition

Use the `encryption-settings` tool with the `create` subcommand to specify the definition.

```
$ bin/encryption-settings create --cipher-algorithm AES
   --key-length-bits 128 --set-preferred
Successfully created a new encryption settings definition with ID
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Changing the Preferred Encryption Settings Definition

To change the preferred encryption settings definition, use the `encryption-settings` tool with the `set-preferred` subcommand. This subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be exported. This argument is required.

To Change the Preferred Encryption Settings Definition

Use the `encryption-settings` tool with the `set-preferred` subcommand to change a definition to a *preferred* definition.

```
$ bin/encryption-settings set-preferred --id 4D86C7922F71BB57B8B5695D2993059A26B8FC01
Encryption settings definition 4D86C7922F71BB57B8B5695D2993059A26B8FC01 was
successfully set as the preferred definition for subsequent encryption operations
```

Deleting an Encryption Settings Definition

To delete an encryption settings definition, use the `encryption-settings` tool with the `delete` subcommand. The subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption settings definition to be deleted. This argument is required.

Note that you should never delete an encryption settings definition unless you are certain that no data in the server is still encrypted using the settings contained in that definition. Any data still encrypted with a definition that has been removed from the database will be inaccessible to the server and will cause errors for any attempt to access it. If you wish to safely delete an encryption settings definition (e.g., because you believe the encryption key may have been compromised), then see the instructions in the "Dealing with a Compromised Encryption Key" section below.

If you merely want the server to stop using that definition for encryption and use a different definition instead, then make sure that the desired definition exists in the encryption settings database (creating it if necessary) and set it to be the preferred definition. As long as the encryption key has not been compromised, there is no harm in having old encryption settings definitions available to the server, and it is recommended that they be retained just in case they are referenced by something.

Also note that you cannot delete the preferred encryption settings definition unless it is the only one left. If you want to delete the currently-preferred definition when one or more other definitions are available, then you must first make one of the other definitions preferred as described in the previous section.

To Delete an Encryption Settings Definition

Use the `encryption-settings` command with the `delete` subcommand. Make sure to include the `--id` argument to specify the definition.

```
$ bin/encryption-settings delete --id F635E109A8549651025D01D9A6A90F7C9017C66D
Successfully deleted encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D
```

Backing Up and Restoring the Encryption Settings Definitions

If you are using data encryption in a directory server instance, it is absolutely essential that you not lose any of the encryption settings definitions that may have been used to encrypt data in the server. If an encryption settings definition is lost, then any data encrypted with that definition will be completely inaccessible. As such, it is important to ensure that you have good backups of the encryption settings definitions to prevent them from being lost.

The Directory Server provides two different mechanisms for backing up and restoring encryption settings definitions. You can export and import individual encryption settings definitions using the `encryption-settings` tool. Or, you can back up and restore the entire encryption settings database using the Directory Server's `backup` and `restore` tools.

Exporting Encryption Settings Definitions

To back up an individual definition (or to export it from one server so that you can import it into another), use the `export` subcommand to the `encryption-settings` command. The subcommand takes the following arguments:

- **--id {id}**. Specifies the ID for the encryption-settings definition to be exported. This argument is required.
- **--output-file {path}**. Specifies the path to the output file to which the encryption-settings definition will be written. This argument is required.
- **--pin-file {path}**. Specifies the path to a PIN file containing the password to use to encrypt the contents of the exported definition. If this argument is not provided, then the PIN will be interactively requested from the server.

To Export an Encryption Settings Definition

Use the `encryption-settings` tool with the `export` subcommand to export the definition to a file.

```
$ bin/encryption-settings export --id F635E109A8549651025D01D9A6A90F7C9017C66D \
  --output-file /tmp/exported-key
Enter the PIN to use to encrypt the definition:
Re-enter the encryption PIN:
Successfully exported encryption settings definition
F635E109A8549651025D01D9A6A90F7C9017C66D to file /tmp/exported-key
```

Importing Encryption Settings Definitions

To import an encryption settings definition that has been previously exported, use the `encryption-settings` tool with the `import` subcommand. The subcommand takes the following arguments:

- **--input-file {path}**. Specifies the path to the file containing the exported encryption-settings definition. This argument is required.

- **--pin-file {path}**. Specifies the path to a PIN file containing the password to use to encrypt the contents of the exported definition. If this argument is not provided, then the PIN will be interactively requested from the server.
- **--set-preferred**. Specifies that the newly-imported encryption-settings definition should be made for the preferred definition for subsequent encryption settings.

To Import an Encryption Settings Definition

Use the **`encryption-settings`** tool with the **`import`** subcommand to import the definition to a file.

```
$ bin/encryption-settings import --input-file /tmp/exported-key --set-preferred
Enter the PIN used to encrypt the definition: Successfully imported encryption settings
definition F635E109A8549651025D01D9A6A90F7C9017C66D from file /tmp/exported-key
```

Backing Up the Encryption Settings Definitions

The encryption settings database may be backed up and restored like a Directory Server back-end using the **`backup`** and **`restore`** commands.

To Back Up an Encryption Settings Definition

To back up just the encryption settings database, you could use a command like:

```
$ bin/backup --backendID encryption-settings --backupDirectory bak/encryption-settings
[13:57:32]  The console logging output is also available in '/ds/logs/tools/backup.log'
[13:57:32]  Starting backup for backend encryption-settings
[13:57:32]  The backup process completed successfully
```

Alternately, you can back up multiple backends at once using this tool, and you can even back up all backends at once with a single command.

```
$ bin/backup --backUpAll -d bak
```

Warning	Because of the importance of the encryption settings database when using data encryption in the server, it is strongly recommended that whenever you back up a component of the server that uses encrypted data, you also back up the encryption settings database.

Restoring an Encryption Settings Definition

Because of the importance of not losing any encryption settings definitions that might be referenced by encrypted data in the server, the restore process for the encryption settings database is a little different than that used when restoring other backends in the server. Rather than completely replacing any existing encryption settings database, the version of the database being restored will be merged with the existing database. The encryption settings definitions in the database (before the restore begins) will still be available after the restore has completed even if they were not in the archived version of the database. The preferred encryption settings definition will be the definition that was preferred in the archived database.

To Restore an Encryption Settings Definition

Use the **`encryption-settings`** tool with the **`restore`** subcommand to restore an encryption settings database. For example:

```
$ bin/restore -d bak/encryption-settings
[13:58:11]  The console logging output is also available in '/ds/logs/tools/
restore.log'
[13:58:11]  Backup 20101121195732Z has been successfully restored for backend encryp-
tion-settings
[13:58:11]  The restore process completed successfully
```

Enabling Data Encryption in the Server

To enable data encryption in the server, you must have at least one encryption settings definition available for use. Then, it is only necessary to set the value of the **`encrypt-data`** global configuration property to **`true`**.

Setting the global configuration property will automatically enable data encryption for all types of backends that support it (including the changelog backend) as well as for the replication server database. All subsequent write operations will cause the corresponding records written into any of these locations to be encrypted. Any existing data will remain unencrypted until it is rewritten by a write operation. If you wish to have existing data encrypted, then you will need to export that data to LDIF and re-import it. This will work for both the data backends and the changelog, but it is not an option for the replication database, so existing change records will remain unencrypted until they are purged. If this is not considered acceptable in your environment, then follow the steps in the "Dealing with a Compromised Encryption Key" to safely purge the replication database.

To Enable Data Encryption in the Server

Use **`dsconfig`** to set the global configuration property for data encryption to **`true`**.

```
$ bin/dsconfig set-global-configuration-prop --set encrypt-data:true
```

Using Data Encryption in a Replicated Environment

Data encryption is only used for the on-disk storage for data within the server. Whenever clients access that data, it is presented to them in unencrypted form (although the communication with those clients may itself be encrypted using SSL or StartTLS). The same is true for replication in that the data itself is not encrypted when it is replicated to other servers (again, replication may use SSL to protect all of its communication between the servers). Each server may apply data encryption in a completely independent manner and have different sets of encryption settings definitions. It also possible to have a replication topology containing some servers with data replication enabled and others with it disabled.

However, if you use binary copy initialization to populate the backend of one server with a backup taken from another server with data encryption enabled, then the server being initialized must have access to all encryption settings definitions that may have been used for data contained in that backend. To do this, export the appropriate definitions from the encryption settings database on the source server using the **`encryption-settings export`** command, and import them into the target server using the **`encryption-settings import`** command.

Dealing with a Compromised Encryption Key

If an encryption settings definition becomes compromised such that an unauthorized individual obtains access to the encryption key, then any data encrypted with that definition is also vulnerable because it can be decrypted using that key. It is very important that the encryption settings database be protected (e.g., using file permissions and/or filesystem ACLs) to ensure that its contents remain secure.

In the event that an encryption settings definition is compromised, then you should immediately stop using that definition. Any data encrypted with the compromised key should be re-encrypted with a new definition or purged from the server. This can be done on one server at a time in order to avoid an environment-wide downtime, but it should be completed as quickly as possible on all servers that had used that definition at any point in the past in order to minimize the risk of that data becoming exposed.

To Deal with a Compromised Encryption Key

The recommended process for responding to a compromised encryption settings definition is as follows:

1. Create a new encryption-settings definition and make it the preferred definition for new writes.
2. Ensure that client traffic is routed away from the server instance to be updated. For example, if the Directory Server is accessed through a Directory Proxy Server, then you may set the **`health-check-state`** configuration property for any LDAP external server definitions that reference that server to have a value of **`unavailable`**.
3. Ensure that external clients will not be allowed to write operations in the Directory Server instance. This may be accomplished by setting the **`writability-mode`** global configuration property to have a value of **`internal-only`**.
4. Wait for all outstanding local changes to be replicated out to other servers. This can be accomplished by looking at the monitor entries with the **`ds-replication-server-handler-monitor-entry`** object class to ensure that the value of the update-sent attribute is no longer increasing.
5. Stop the Directory Server instance.

6. Delete the replication server database by removing all files in the **`changeLogDb`** directory below the server root. As long as all local changes have been replicated out to other servers, this will not result in any data loss in the replication environment.

7. Export the contents of all local DB, large attribute, and changelog backends to LDIF. Then, re-import the data from LDIF, which will cause it to be encrypted using the new preferred encryption settings definition.

8. Export the compromised key from the encryption settings database to back it up in case it may be needed again in the future (e.g., if some remaining data was later found to have been encrypted with the key contained in that definition). Then, delete it from the encryption settings database so that it can no longer be used by that directory server instance.

9. Start the directory server instance.

10. Allow replication to bring the server back up-to-date with any changes processed while it was offline.

11. Re-allow externally-initiated write operations by changing the value of the global **`writability-mode`** configuration property back to **`enabled`**.

12. Re-configure the environment to allow client traffic to again be routed to that server instance (e.g., by changing the value of the "health-check-state" property in the corresponding LDAP external instance definitions in the Directory Proxy Server instances back to "dynamically-determined").

Configuring Sensitive Attributes

As previously mentioned, data encryption is only applied to the on-disk storage for the Directory Server instance. It does not automatically protect information accessed or replicated between servers, although the server offers other mechanisms to provide that protection (i.e., SSL, StartTLS, SASL). Ensuring that all client communication uses either SSL or StartTLS encryption and ensuring that all replication traffic uses SSL encryption ensures that the data is protected from unauthorized individuals who may be able to eavesdrop on network communication. This communication security may be enabled independently of data encryption (although if data encryption is enabled, then it is strongly recommended that secure communication be used to protect network access to that data).

However, for client data access, it may not be as simple as merely enabling secure communication. In some cases, it may be desirable to allow insecure access to some data. In other cases, it may be useful to have additional levels of protection in place to ensure that some attributes are even more carefully protected. These kinds of protection may be achieved using sensitive attribute definitions.

Each sensitive attribute definition contains a number of configuration properties, including:

- **attribute-type**. Specifies the set of attribute types whose values may be considered sensitive. At least one attribute type must be provided, and all specified attribute types must be defined in the server schema.
- **include-default-sensitive-operational-attributes**. Indicates whether the set of sensitive attributes should automatically be updated to include any operational attributes maintained by the Directory Server itself that may contain sensitive information. At present, this includes the `ds-sync-hist` operational attribute, which is used for data required for replication conflict resolution and may contain values from other attributes in the entry.
- **allow-in-filter**. Indicates whether sensitive attributes may be used in filters. This applies not only to the filter used in search requests, but also filters that may be used in other places, like the assertion and join request controls. The value of this property must be one of `allow` (allow sensitive attributes to be used in filters over both secure and insecure connections), `reject` (reject any request which includes a filter targeting one or more sensitive attributes over both secure and insecure connections), or `secure-only` (allow sensitive attributes to be used in filters over secure connections, but reject any such requests over insecure connections).
- **allow-in-add**. Indicates whether sensitive attributes may be included in entries created by LDAP add operations. The value of this property must be one of `allow` (allow sensitive attributes to be included in add requests over both secure and insecure connections), `reject` (reject any add request containing sensitive attributes over both secure and insecure connections), or `secure-only` (allow sensitive attributes to be included in add requests received over a secure connection, but reject any such requests over an insecure connection).
- **allow-in-compare**. Indicates whether sensitive attributes may be targeted by the assertion used in a compare operation. The value of this property must be one of `allow` (allow sensitive attributes to be targeted by requests over both secure and insecure connections), `reject` (reject any compare request targeting a sensitive attribute over both secure and insecure connections), or `secure-only` (allow compare requests targeting sensitive attributes over a secure connection, but reject any such requests over an insecure connection).
- **allow-in-modify**. Indicates whether sensitive attributes may be updated using modify operations. The value of this property must be one of `allow` (allow sensitive attributes to be modified by requests over both secure and insecure connections), `reject` (reject any modify request updating a sensitive attribute over both secure and insecure connections), or `secure-only` (only modify requests updating sensitive attributes over a secure connection, but reject any such request over an insecure connection).

The `allow-in-returned-entries`, `allow-in-filter`, `allow-in-add`, `allow-in-compare`, and `allow-in-modify` properties all have default values of `secure-only`, which prevents the possibility of exposing sensitive data in the clear to anyone able to observe network communication.

If a client connection policy references a sensitive attribute definition, then any restrictions imposed by that definition will be enforced for any clients associated with that client connection policy. If multiple sensitive attribute definitions are associated with a client connection policy, then the server will use the most restrictive combination of all of those sets.

Note that sensitive attribute definitions work in conjunction with other security mechanisms defined in the server and may only be used to enforce additional restrictions on clients. Sensitive attribute definitions may never be used to grant a client additional access to information that it would not have already had through other means. For example, if the **`employeeSSN`** attribute is declared to be a sensitive attribute and the **`allow-in-returned-entries`** property has a value of **`secure-only`**, then the **`employeeSSN`** attribute will only be returned to those clients that have both been granted permission by the access control rules defined in the server and are communicating with the server over a secure connection. The **`employeeSSN`** attribute will be stripped out of entries returned to clients normally authorized to see it if they are using insecure connections, and it will also be stripped out of entries for clients normally not authorized to see it even if they have established secure connections.

To Create a Sensitive Attribute

1. To create a sensitive attribute, you must first create one or more sensitive attribute definitions. For example, to create a sensitive attribute definition that will only allow access to the **`employeeSSN`** attribute by clients using secure connections, the following configuration changes may be made:

```
$ bin/dsconfig create-sensitive-attribute \
  --attribute-name "Employee Social Security Numbers" \
  --set attribute-type:employeeSSN \
  --set include-default-sensitive-operational-attributes:true \
  --set allow-in-returned-entries:secure-only \
  --set allow-in-filter:secure-only --set allow-in-add:secure-only \
  --set allow-in-compare:secure-only --set allow-in-modify:secure-only
```

2. Associate those sensitive attribute definitions with the client connection policies for which you want them to be enforced.

```
$ bin/dsconfig set-client-connection-policy-prop --policy-name default \
  --set "sensitive-attribute:Employee Social Security Numbers"
```

Configuring Global Sensitive Attributes

Administrators can assign one or more sensitive attribute definitions to a client connection policy. However, in an environment with multiple client connection policies, it could be easy to add a sensitive attribute definition to one policy but overlook it in another. The Directory Server supports the ability to define sensitive attributes as a global configuration option so that they will automatically be used across all client connection policies.

To Configure a Global Sensitive Attribute

- Run **`dsconfig`** to add a global sensitive attribute across all client connection policies. The following command adds an existing **`Employee Social SecurityNumbers`** as a global sensitive attribute, which is applied across all client connection policies.

```
$ bin/dsconfig set-global-configuration-prop \
  --add "sensitive-attribute:Employee Social Security Numbers"
```

Excluding a Global Sensitive Attribute on a Client Connection Policy

Administrators can set a global sensitive attribute across all client connection policies. However, there may be cases when a specific directory server must exclude the sensitive attribute as it may not be needed for client connection requests. For example, in most environments it would be good to declare the `userPassword` attribute type to be a sensitive attribute (called `Sensitive Password Attributes`) in a manner that prevents it from being read by external clients. However, if a Synchronization Server is installed, then it does need to be able to access passwords for synchronization purposes. In this case, the administrator can set `userPassword` to be a sensitive attribute in all client connection policies, but exclude it in a policy specifically created for use by the Synchronization Server. The Directory Server provides an `exclude-global-sensitive-attribute` property for this purpose.

To Exclude a Global Sensitive Attribute on a Client Connection Policy

1. Run `dsconfig` to add the `Sensitive Password Attributes` attribute as a global sensitive attribute, which is applied to all client connection policies. It is assumed that `Sensitive Password Attributes` attribute was already created for the `userPassword` attribute type. See “To Create a Sensitive Attribute” on page 228.

```
$ bin/dsconfig set-global-configuration-prop \
  --add "sensitive-attribute:Sensitive Password Attributes"
```

2. On a server designated to synchronize with the Synchronization Server, configure the client connection policy to exclude the global sensitive attribute. In the following example, the "Sync Only Client Connection Policy" is configured to exclude the `Sensitive Password Attributes` global sensitive attribute.

```
$ bin/dsconfig set-client-connection-policy-prop \
  --policy-name "Sync-Only Client Connection Policy" \
  --add "exclude-global-sensitive-attribute:Sensitive Password Attributes"
```

Working with the LDAP Changelog

The Directory Server provides a client-accessible LDAP changelog (based on the `Changelog Internet Draft Specification`) for the purpose of allowing other LDAP clients to retrieve changes made to the server in standard LDAP format. The LDAP changelog is typically used by external software to maintain application compatibility between client services. For example, you can install a Synchronization Server that monitors the LDAP changelog for any updates that occur on a source directory server and synchronizes these changes to a target directory or database server. The Directory Server provides an additional feature in that the LDAP changelog supports virtual attributes.

Note

The LDAP Changelog should not be confused with the Replication Changelog. The main distinction is as follows:

• The LDAP Changelog (i.e., the external changelog that clients can access) physically resides at **`<server-root>/db/changelog`**.

• The Replication Changelog Backend (i.e., the changelog that replication servers use) physically resides at **`<server-root>/changelogDB`**.

New Changelog Features

As of version 3.2, the Directory Server supports two new Changelog Backend properties that allow access control filtering and sensitive attribute evaluation for targeted entries. External client applications can change the contents of attributes they can see in the targeted entry based on the access control rules applied to the associated base DN.

- **apply-access-controls-to-changelog-entry-contents**. Indicates whether the contents of changelog entry attributes (i.e., **`changes`**, **`deletedEntryAttrs`**, **`ds-changelog-entry-key-attr-values`**, **`ds-changelog-before-values`**, and **`ds-changelog-after-values`**) are subject to access control and/or sensitive attribute evaluation to limit data that LDAP clients can see. The client must have the access control permissions to read changelog entries in order to retrieve them in any form. If this feature is enabled and the client does not have permission to read an entry at all, or if that client does not have permission to see any attributes that were targeted by the change, then the associated changelog entries targeted by those operations will be suppressed. If a client does *not* have permission to see certain attributes within the target entry, then references to those attributes in the changelog entry will also be suppressed. This property only applies to standard LDAP searches of the **`cn=changelog`** branch.

- **report-excluded-changelog-attributes**. Indicates whether to include additional information about any attributes that may have been removed due to access control filtering. This property only applies to content removed as a result of processing performed by the **`apply-access-controls-to-changelog-entry-contents`** property. Possible values are:

  - **none** - Indicates that changelog entries should *not* include any information about attributes that have been removed.

  - **attribute-counts** - Indicates that changelog entries should include a count of user and/or operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, the number of the excluded user attributes will be reported in the **`ds-changelog-num-excluded-user-attributes`** attribute of the changelog entry. If any operational attribute information was excluded from a changelog entry, then the number of the excluded operational attributes will be reported in the **`ds-changelog-num-excluded-operational-attributes`** attribute of the changelog entry. Both the **`ds-changelog-num-excluded-user-attributes`** and **`ds-changelog-num-excluded-operational-attributes`** are operational and must be explicitly requested by clients (or all operational attributes requested using "+") in order to be returned.

- **attribute-names** - Indicates that changelog entries should include the names of user and/or operational attributes that have been removed. If any user attribute information was excluded from a changelog entry, then the names of the excluded user attributes will be reported in the `ds-changelog-excluded-user-attributes` attribute of the changelog entry. If any operational attribute information was excluded from a changelog entry, then the names of the excluded operational attributes will be reported in the `ds-changelog-excluded-operational-attribute` attribute of the changelog entry. Both the `ds-changelog-excluded-user-attribute` and `ds-changelog-excluded-operational-attribute` attributes are operational and must be explicitly requested by clients (or all operational attributes requested via "+") in order to be returned.

To Enable Access Control Filtering in the LDAP Changelog

To set up access control to the LDAP Changelog, use the `dsconfig` tool to enable the properties to the Changelog Backend. Only admin users with the `bypass-acl` privilege can read the changelog.

1. Enable the `apply-access-control-to-changelog-entry-contents` property to allow LDAP clients to undergo access control filtering using standard LDAP searches of the `cn=changelog` backend.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \
  --set "apply-access-controls-to-changelog-entry-contents:true"
```

Note	Access control filtering will be applied regardless of the value of the `apply-access-controls-to-changelog-entry-contents` setting when the Changelog Backend is servicing requests from an UnboundID Synchronization Server that has the `filter-changes-by-user` Sync Pipe property set.

2. Optional. Set the `report-excluded-changelog-attributes` property to include a count of users that have been removed through access control filtering. The count appears in the `ds-changelog-num-excluded-user-attributes` attribute for users and the `ds-changelog-num-excluded-operational-attributes` attribute for operational attributes.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \
  --set "report-excluded-changelog-attributes:attribute-counts"
```

Useful Changelog Features

The Directory Server provides two useful changelog configuration properties: `changelog-max-before-after-values` and `changelog-include-key-attribute`.

- **changelog-max-before-after-values**. Setting this property to a non-zero value causes all of the old values and all of the new values (up to the specified maximum) for each changed attribute to be stored in the changelog entry. The values will be stored in the `ds-change-`

log-before-values and **ds-changelog-after-values** attributes on the changelog entry. These attributes are not present by default.

Caution	The **changelog-max-before-after-values** property can be expensive for attributes with hundreds or thousands of values, such as a group entry.

If any attribute has more than the maximum number of values, their names and number of before/after values will be stored in the '**ds-changelog-attr-exceeded-max-values-count**' attribute on the changelog entry. This is a multi-valued attribute whose format is:

```
attr=attributeName,beforeCount=100,afterCount=101
```

where "attributeName" is the name of the attribute and the "**beforeCount**" and "**afterCount**" are the total number of values for that attribute before and after the change, respectively. This attribute indicates that you need to reset the **changelog-max-before-after-values** property to a higher value. When this attribute is set, an alert will be generated.

Caution	If the number of values for an attribute exceeds the maximum value set by the **changelog-max-before-after-values** property, then those values will not be stored.

- **changelog-include-key-attribute**. This property is used for correlation attributes that need to be synchronized across servers, such as **uid**. It causes the current (after-change) value of the specified attributes to be recorded in the **ds-changelog-entry-key-attr-values** attribute on the changelog entry. This applies for all change types. On a DELETE operation, the values are from the entry before it was deleted.

  The key values will be recorded on every change and override the settings configured in **changelog-include-attribute**, **changelog-exclude-attribute**, **changelog-deleted-entry-include-attribute**, or **changelog-deleted-entry-exclude-attribute**.

Examples

After the **changelog-max-before-after-values** property is set, the before and after values of any change attribute will be recorded in the LDAP Changelog. For example, given a simple entry with two multi-valued mail attributes:

```
dn: uid=test,dc=example,dc=com
objectclass: inetorgperson
cn: test user
sn: user
description: oldDescription
mail: test@yahoo.com
mail: test@gmail.com
```

Then, apply the following changes to the entry:

```
dn: uid=test,dc=example,dc=com
changetype: modify
```

```
add: mail
mail: test@hotmail.com
-
delete: mail
mail: test@yahoo.com
-
replace: description
description: newDescription
```

The resulting changelog would record the following attribute values:

```
dn: changeNumber=1,cn=changelog
objectClass: top
objectClass: changeLogEntry
targetDN: uid=test,dc=example,dc=com
changeType: modify
changes::
YWRkOiBtYWlsCm1haWw6IHRlc3RAaG90bWFpbC5jb20KLQpkZWxldGU6IG1haWwKbWFpbDogdGVzdEB5YWh
vby5jb20KLQpyZXBsYWNlOiBkZXNjcmlwdGlvbgpkZXNjcmlwdGlvbjogbmV3RGVzY3JpcHRpb24KLQpyZX
BsYWNlOiBtb2RpZmllcnNOYW1lCm1vZGlmaWVyc05hbWU6IGNuPURpcmVjdG9yeSBNYW5hZ2VyLGNuPVJvb
3QgRE5zLGNuPWNvbmZpZwotCnJlcGxhY2U6IGRzLXVwZGF0ZS10aW1lCmRzLXVwZGF0ZS10aW1lOjogQUFB
QkxxQitIaTQ9Ci0KAA==
ds-changelog-before-values::
ZGVzY3JpcHRpb246IG9sZERlc2NyaXB0aW9uCm1haWw6IHRlc3RAeWFob28uY29tCm1haWw6IHRlc3RAZ21
haWwuY29tCmRzLXVwZGF0ZS10aW1lOjogQUFBQkxxQjdaZ1E9Cm1vZGlmaWVyc05hbWU6IGNuPURpcmVjdG
9yeSBNYW5hZ2VyLGNuPVJvb3QgRE5zLGNuPWNvbmZpZwo=
ds-changelog-after-values::
ZGVzY3JpcHRpb246IG5ld0Rlc2NyaXB0aW9uCm1haWw6IHRlc3RAZ21haWwuY29tCm1haWw6IHRlc3RAaG9
0bWFpbC5jb20KZHMtdXBkYXRlLXRpbWU6OiBBQUFCTHFCK0hpND0KbW9kaWZpZXJzTmFtZTogY249RGlyZW
N0b3J5IE1hbmFnZXIsY249Um9vdCBETnMsY249Y29uZmlnCg==
ds-changelog-entry-key-attr-values:: dWlkOiB0ZXN0Cg==
changenumber: 1
```

You can run the **bin/base64 decode -d** command-line tool to view the decoded value for the **changes**, **ds-changelog-before-values**, **ds-changelog-after-values** attributes:

After base64 decoding, the **changes** attribute reads:

```
add: mail
mail: test@hotmail.com
-
delete: mail
mail: test@yahoo.com
-
replace: description
description: newDescription
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: ds-update-time
ds-update-time:: AAABLqB+Hi4=
-
```

After base64 decoding, the **ds-changelog-before-values** attribute reads:

```
description: oldDescription
mail: test@yahoo.com
mail: test@gmail.com
ds-update-time:: AAABLqB7ZgQ=
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
```

After base64 decoding, the **ds-changelog-after-values** attribute reads:

```
description: newDescription
mail: test@gmail.com
```

```
mail: test@hotmail.com
ds-update-time:: AAABLqB+Hi4=
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
```

Viewing the LDAP Changelog Properties

You can view the LDAP changelog properties by running the **`dsconfig get-backend-prop`** command and specifying the changelog backend.

To View the LDAP Changelog Properties Using dsconfig Non-Interactive Mode

Use **`dsconfig`** to view the changelog properties on the Directory Server. To view "advanced" properties that are normally hidden, add the **`--advanced`** option when running the command. For a specific description of each property, see the *UnboundID Directory Server Configuration Reference*.

```
$ bin/dsconfig get-backend-prop --backend-name changelog --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret --advanced --no-prompt

Property                                  : Value(s)
------------------------------------------:-------------
backend-id                                : changelog
description                               : -
enabled                                   : false
writability-mode                          : disabled
base-dn                                   : cn=changelog
set-degraded-alert-when-disabled          : false
return-unavailable-when-disabled          : false
db-directory                              : db
db-directory-permissions                  : 700
changelog-maximum-age                     : 2 d
db-cache-percent                          : 1
changelog-include-attribute               : -
changelog-exclude-attribute               : -
changelog-deleted-entry-include-attribute : -
changelog-deleted-entry-exclude-attribute : -
changelog-include-key-attribute           : -
changelog-max-before-after-values         : 0
changelog-write-batch-size                : 100
changelog-purge-batch-size                : 100
changelog-write-queue-capacity            : 100
write-lastmod-attributes                  : true
use-reversible-form                       : false
```

Enabling the LDAP Changelog

By default, the LDAP changelog is disabled on the Directory Server. If you are using the **`dsconfig`** tool in interactive mode, the changelog appears in the Backend configuration as a Standard object menu item.

Note	You can enable the feature using the **`dsconfig`** tool only if required as it can significantly affect LDAP update performance.

To Enable the LDAP Changelog Using dsconfig Non-Interactive Mode

Use **`dsconfig`** to enable the changelog property on the Directory Server.

```
$ bin/dsconfig set-backend-prop --no-prompt \
  --backend-name changelog --set enabled:true \
  --host server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret
```

To Enable the LDAP Changelog Using Interactive Mode

1. Use **`dsconfig`** to enable the changelog on each server in the network. Then, authenticate to the server by entering the host name, LDAP connection, port, bindDN and bind password.

   ```
   $ bin/dsconfig
   ```

2. On the Configuration Console main menu, type o to change from the Basic object level to the Standard object level.

3. Enter the option to select the Standard Object level.

4. On the Directory Server Configuration console main menu, type the number corresponding to Backend.

5. On the Backend Management menu, enter the option to view and edit an existing backend.

6. Next, you should see a list of the accessible backends on your system. For example, you may see options for the changelog and userRoot backends. Enter the option to work with the changelog backend.

7. On the Changelog Backend properties menu, type the number corresponding to the Enabled property.

8. On the Enabled Property menu, type **`2`** to change the **`Enabled`** property to **`TRUE`**.

9. On the Backend Properties menu, type **`f`** to apply the change. If you set up the server in a server group, type **`g`** to update all of the servers in the group. Otherwise, repeat steps 1-10 on the other servers.

10. Verify that changes made to the directory data are recorded in the changelog.

Changing the LDAP Changelog Database Location

In cases where disk space issues arise, you can change the on-disk location of the LDAP Change Log database. The changelog backend supports a **`db-directory`** property that specifies the absolute or relative path (i.e., relative to the local server root) to the filesystem directory that is used to hold the Oracle Berkeley DB Java Edition database files containing the data for this backend.

If you change the changelog database location, you must stop and then restart the Directory Server for the change to take effect. If the changelog backend is already enabled, then the database files must be manually moved or copied to the new location while the server is stopped.

To Change the LDAP Changelog Location Using dsconfig Non-Interactive Mode

1. Use **`dsconfig`** to change the database location for the LDAP Changelog, which by default is at **`<server-root>/db`**. The following command sets the LDAP changelog backend to **`<server-root>/db2`**. Remember to include the LDAP connection parameters (**`hostname`**, **`port`**, **`bindDN`**, **`bindPassword`**).

```
$ bin/dsconfig set-backend-prop --backend-name changelog --set "db-directory:db2" \
  --set "enabled:true" --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

   The database files are stored under **`<server-root>/db2/changelog`**. The files for this backend are stored in a sub-directory named after the **`backend-id`** property.

2. Stop and restart the server. Since the LDAP changelog backend was previous disabled, there is not need to manually relocate any existing database files.

To Reset the LDAP Changelog Location Using dsconfig Non-Interactive Mode

1. If you have changed the LDAP Changelog location, but want to reset it to its original location, use **`dsconfig`** to reset it. The following command resets the LDAP changelog backend to **`<server-root>/db`** location. Remember to include the LDAP connection parameters (**`hostname`**, **`port`**, **`bindDN`**, **`bindPassword`**).

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --reset "db-directory" --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --no-prompt
```

2. The server attempts to use whatever it finds in the configured location when it starts. If there is nothing there, it will create an empty database. If the LDAP changelog backend at the previous location is enabled at the time, stop the server, manually copy the database files to the new LDAP changelog location, and then restart the server.

Viewing the LDAP Changelog Parameters in the Root DSE

The Root DSE is a special directory entry that holds operational information about the server. The entry provides information about the LDAP controls, extended operations, and SASL mechanisms available in the server as well as the state of the data within the changelog. For changelog parameters, the attributes of interest include:

- **firstChangeNumber**. Change number for the first (oldest) change record contained in the LDAP changelog.
- **lastChangeNumber**. Change number for the last (most recent) change record contained in the LDAP changelog.

- **lastPurgedChangeNumber**. Change number for the last change that was purged from the LDAP changelog. It can be 0 if no changes have yet been purged.
- **firstReplicaChange**. Information about the first (oldest) change record for a change received from the specified replica. This is a multi-valued attribute and should include a value for each server in the replication topology.
- **lastReplicaChange**. Information about the last (most recent) change record for a change received from the specified replica.

The `firstReplicaChange` and `lastReplicaChange` attributes use the following syntax:

`serverID:CSN:changeNumber`

where:

- **serverID**. Specifies the unique identifier for the server updating the change log.
- **CSN**. Specifies the Change Sequence Number, which is the time when the update was made to the given replica.
- **changeNumber**. Specifies the order of the change that is logged to the LDAP changelog.

The `firstReplicaChange` and `lastReplicaChange` attributes can be used to correlate information in the local LDAP Change Log with data in the LDAP Change Log of other servers in the replication topology. The order of the individual changes in the LDAP Change Log can vary between servers based on the order in which they were received from a replica.

To View the LDAP Changelog Parameters

Use `ldapsearch` to view the Root DSE.

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --baseDN "" --searchScope base "(objectclass=*)" "+"
```

Viewing the LDAP Changelog Using ldapsearch

All records in the changelog are immediate children of the `cn=changelog` entry and are named with the `changeNumber` attribute. You can view changelog entries using `ldapsearch`. Changes are represented in the form documented in the `draft-good-ldap-changelog` specification with the `targetDN` attribute providing the DN of the updated entry, the `changeType` attribute providing the type of operation (add, delete, modify, or modDN), and the `changes` attribute providing a base64-encoded representation of the attributes included in the entry (for add operations) or the changes made (for modify operations) in LDIF form. You can view the changes by decoding the encoded value using the **`base64 decode`** utility. The UnboundID LDAP SDK for Java also provides support for parsing changelog entries.

To View the LDAP Changelog Using ldapsearch

Use **`ldapsearch`** to view the changelog.

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --baseDN cn=changelog --dontWrap "(objectclass=*)"

dn: cn=changelog
objectClass: top
objectClass: untypedObject
cn: changelog

dn: changeNumber=1,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: uid=user.0,ou=People,dc=example,dc=com
changeType: modify
changes::
cmVwbGFjZTogbW9iaWxlCm1vYmlsZTogKzEgMDIwIDE1NCA5Mzk4Ci0KcmVwbGFjZTogaG9tZVBob25lCmh
vbWVQaG9uZTogKzEgMjI1IDIxNiA0OTQ5Ci0KcmVwbGFjZTogZ2l2ZW5OYW1lCmdpdmVuTmFtZTogQWFyb2
4KLQpyZXBsYWNlOiBkZXNjcmlwdGlvbgpkZXNjcmlwdGlvbjogdGhpcyBpcyB0aGUgZGVzY3JpcHRpb24gZ
m9yIEFhcm9uIEF0cC4KLQpyZXBsYWNlOiBtb2RpZmllcnNOYW1lCm1vZGlmaWVyc05hbWU6IGNuPURpcmVj
dG9yeSBNYW5hZ2VyLGNuPVJvb3QgRE5zLGNuPWNvbmZpZwotCnJlcGxhY2U6IGRzLXVwZGF0ZS10aW1lCmR
zLXVwZGF0ZS10aW1lOjogQUFBQkhQOHpUR0E9Cgo=
changenumber: 1
dn: changeNumber=2,cn=changelog
objectClass: changeLogEntry
objectClass: top
targetDN: dc=example,dc=com
changeType: modify
changes::
cmVwbGFjZTogZHMtc3luYy1zdGF0ZQpkcy1zeW5jLXN0YXRlOiAwMDAwMDExQ0ZGMzM0QzYwNDA5MzAwMDA
wMDAyCgo=
changenumber: 2
```

To View the LDAP Change Sequence Numbers

The changelog displays the server state information, which is important for failover between servers during synchronization operations. The server state information is exchanged between the servers in the network (LDAP servers and replication servers) as part of the protocol start message. It also helps the client application determine which server is most up-to-date:

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --baseDN cn=changelog --dontWrap "(objectclass=*)" "+"

   dn: cn=changelog

   dn: changeNumber=1,cn=changelog
   entry-size-bytes: 182
   targetUniqueId: 68147342-1f61-3465-8489-3de58c532130
   changeTime: 20111023002624Z
   lastReplicaCSN: 0000011D27184D9E303000000001
   replicationCSN: 0000011D27184D9E303000000001
   replicaIdentifier: 12336

   dn: changeNumber=2,cn=changelog
```

```
entry-size-bytes: 263
targetUniqueId: 4e9b7847-edcb-3791-b11b-7505f4a55af4
changeTime: 20111023002624Z
lastReplicaCSN: 0000011D27184F2E303000000002
replicationCSN: 0000011D27184F2E303000000002
replicaIdentifier: 12336
```

To View LDAP Changelog Monitoring Information

The changelog contains a monitor entry that you can access over LDAP, JConsole, the Directory Management Console, or SNMP.

Use **ldapsearch** to view the changelog monitor entry.

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --baseDN cn=changelog,cn=monitor "(objectclass=*)"

  dn: cn=changelog,cn=monitor
  objectClass: top
  objectClass: ds-monitor-entry
  objectClass: extensibleObject
  cn: changelog
  changelog: cn=changelog
  firstchangenumber: 1
  lastchangenumber: 8
  lastpurgedchangenumber: 0
  firstReplicaChange: 16225:0000011D0205237F3F6100000001:5
  firstReplicaChange: 16531:0000011CFF334C60409300000002:1
  lastReplicaChange: 16225:0000011D02054E8B3F6100000002:7
  lastReplicaChange: 16531:0000011CFF334C60409300000002:1
  oldest-change-time: 20081015063104Z
  ...(more data)...
```

Indexing the LDAP Changelog

The Directory Server supports attribute indexing in the Changelog Backend to allow Get-Changelog-Batch requests that filter results to include only those changes involving specific attributes. Using Attribute indexing, client application result in faster response times to their Get-Changelog-Batch requests when targeting the specified attributes.

Administrators can configure attribute indexing using the **index-include-attribute** and **index-exclude-attribute** properties on the Changelog Backend. The properties can accept the specific attribute name or special LDAP values "*" to specify all user attributes or "+" to specify all operational attributes. For example, if you set the **index-include-attribute** to all user attributes using "*", you can exclude any unnecessary attributes using the **index-exclude-attribute** property. To determine if the server supports this feature, administrators can view the Root DSE for the following entry: **supportedFeatures: 1.3.6.1.4.1.30221.2.12.3**

To Index a Changelog Attribute

1. Use **dsconfig** to set attribute indexing on an attribute in the Changelog Backend. The following command enables the Changelog Backend and sets the backend to include all user

attributes ("*") for ADD or MODIFY operations using the **changelog-include-attribute** property. The **changelog-deleted-entry-include-attribute** property is set to all attributes ("*") to specify all attribute types that should be included in a changelog entry for DELETE operations. Attributes specified in this list will be recorded in the **deletedEntryAttrs** attribute on the changelog entry when an entry is deleted. The attributes **displayName** and **employeeNumber** are indexed using the **index-include-attribute** property.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set "enabled:true" \
  --set "changelog-include-attribute:*" \
  --set "changelog-deleted-entry-include-attribute:*" \
  --set "index-include-attribute:displayName" \
  --set "index-include-attribute:employeeNumber"
```

2. Add another attribute to index using the **dsconfig --add** option, which adds the attribute to an existing configuration setting.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --add "index-include-attribute:cn"
```

To Index All Changelog Attributes

- Use **dsconfig** to set attribute indexing on all user attributes in the Changelog Backend. You can use the special character "*" to include all user attributes. If you want to include operational attributes, use the "+" character.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set "index-include-attribute:*"
```

To Exclude Changelog Attributes from Indexing

- Use **dsconfig** to set attribute indexing on all user attributes in the Changelog Backend. The following command includes all user attributes except the **description** and **location** attributes.

```
$ bin/dsconfig set-backend-prop --backend-name changelog \
  --set "index-include-attribute:*" \
  --set "index-exclude-attribute:description \
  --set "index-exclude-attribute:location
```

Tracking Virtual Attribute Changes in the LDAP Changelog

By default, the LDAP Changelog tracks changes to real attributes only. For client applications that require change tracking to include virtual attributes, administrators can enable the **include-virtual-attribute** property, so that real and virtual attributes are tracked within the changelog. Once the **include-virtual-attribute** property is enabled, then properties for virtual attributes that store before/after values, key attributes, and added or deleted entry attributes can be enabled.

To Track Virtual Attribute Changes in the LDAP Changelog

1. Use **`dsconfig`** to enable virtual attribute change tracking in the LDAP Changelog. The following command enables the LDAP changelog and sets **`include-virtual-attributes`** to **`add-attributes`**, which indicates that virtual attribute be included in the set of attributes listed for an add operation. The **`delete-entry-attributes`** option indicates that virtual attributes should be included in the set of deleted entry attributes listed for a delete operation. The **`before-and-after-values`** option indicates that virtual attributes should be included in the set of before and after values for attributes targeted by the changes. The **`key-attribute-values`** option indicates that virtual attributes should be included in the set of entry key attribute values.

```
$ bin/dsconfig set-backend-prop --backend-name "changelog" \
  --set "enabled:true" \
  --set "include-virtual-attributes:add-attributes" \
  --set "include-virtual-attributes:deleted-entry-attributes" \
  --set "include-virtual-attributes:before-and-after-values" \
  --set "include-virtual-attributes:key-attribute-values"
```

Working with Proxied Authorization

The Directory Server supports the Proxied Authorization Control (RFC 4370) to allow an authorized LDAP client to authenticate to the server as another user. Typically, LDAP servers are deployed as backend authentication systems that store user credentials and authorization privileges necessary to carry out an operation. Single sign-on (SSO) systems can retrieve user credentials from the directory server and then issue permissions that allow the LDAP client to request operations under the identity as another user. The use of the proxied authorization control provides a means for client applications to securely process requests without the need to bind or re-authenticate to the server for each and every operation.

The Directory Server supports the proxied authorization V1 and V2 request controls. The proxied authorization V1 request control is based on early versions of the draft-weltman-ldapv3-proxy Internet draft and is available primarily for legacy systems. It is recommended that deployments use the proxied authorization V2 request control based on RFC 4370.

The proxied authorization V2 control is used to request that the associated operation be performed as if it has been requested by some other user. This control may be used in conjunction with add, delete, compare, extended, modify, modify DN, and search requests. In that case, the associated operation will be processed under the authority of the specified authorization identity rather than the identity associated with the client connection (i.e., the user as whom that connection is bound). The target authorization identity for this control is specified as an "authzid" value, which should be either "dn:" followed by the distinguished name of the target user, or "u:" followed by the username.

Note that because of the inherent security risks associated with the use of the proxied authorization control, most directory servers which support its use enforce strict restrictions on the users that are allowed to request this control. If a user attempts to use the proxied authorization

V2 request control and does not have sufficient permission to do so, then the server will return a failure response with the **AUTHORIZATION_DENIED** result code.

Configuring Proxied Authorization

Configuring proxied authorization requires a combination of access control instructions and the **proxied-auth** privilege to the entry that will perform operations as another user.

Note You cannot use the **cn=Directory Manager** root DN as a proxying DN.

To Configure Proxied Authorization

1. Open a text editor and create a user entry, such as **uid=clientApp**, which is the user entry that will request operations as another user, **uid=admin,dc=example,dc=com**. The client application entry also requires the **proxied-auth** privilege to allow it to run proxied authorization requests. Save the file as **add-user.ldif**.

```
dn: ou=Applications,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
objectClass: extensibleObject
ou: Admins
ou: Applications

dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
givenName: Client
uid: clientApp
cn: Client App
sn: App
userPassword: password
ds-privilege-name: proxied-auth
```

2. Add the file using **ldapmodify**.

```
$ bin/ldapmodify --defaultAdd --filename add-user.ldif
```

3. The client application targets a specific subtree in the Directory Information Tree (DIT) for its operations. For example, some client may need access to an accounts subtree to retrieve customer information. Another client may need access to another subtree, such as a subscriber subtree. In this example, we want the client application to target the **ou=People,dc=example,dc=com** subtree. To allow the target, open a text editor and create an LDIF file to assign an ACI to that branch so that the client app user can access it as a proxy auth user. Add the file using the **ldapmodify**.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*")(targetscope="subtree")
```

```
 (version 3.0; acl "People Proxy Access";
 allow(proxy) userdn="ldap:///uid=clientApp,ou=Applications,dc=example,dc=com";)
```

4. Run a search to test the configuration using the bind DN **`uid=clientApp`** and the **`--proxyAs`** option, which requires that you prefix "dn:" to the proxying entry or "u:" to the username. The **`uid=clientApp`** binds to the server and proxies as **`uid=admin`** to access the **`ou=People,dc=example,dc=com`** subtree.

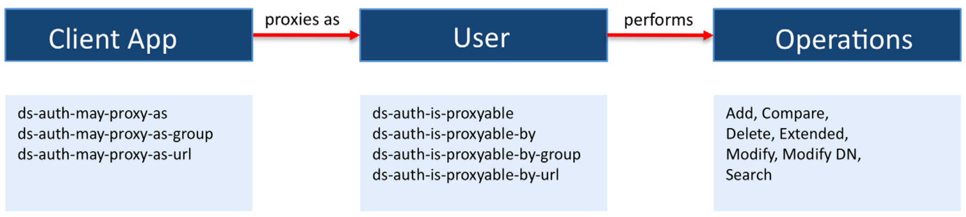
```
$ bin/ldapsearch --port 1389 \
--bindDN "uid=clientApp,ou=Applications,dc=example,dc=com" \
--bindPassword password \
--proxyAs "dn:uid=admin,dc=example,dc=com" \
--baseDN ou=People,dc=example,dc=com \
"(objectclass=*)"
```

Restricting Proxy Users

The Directory Server provides a set of operational attributes that restricts the proxied authorization capabilities of a client application and its proxyable target entry. When present in an entry, the Directory Server evaluates each operational attribute together to form a whitelist of potential users that can be proxied. If none of those attributes is present, then the user may potentially proxy as anyone.

The Directory Server supports a two-tier provision system that, when configured, can restrict specific users for proxied authorization. The first tier is a set of **`ds-auth-may-proxy-as-*`** operational attributes on the client entry that will bind to the server and carry out operations under the identity of another user. The second tier is a set of **`ds-auth-is-proxyable-*`** operational attributes on the proxied (or authorization) user entry that defines whether access is allowed, prohibited, or required by means of proxied authorization. If allowed or required, the attributes define which client entries can proxy as the user.

FIGURE 7-1. **Proxying Operational Attributes**

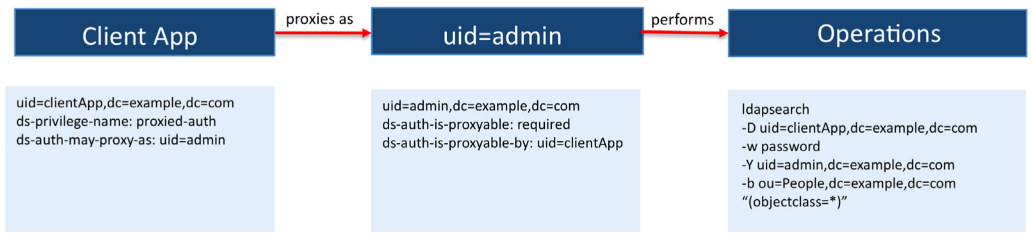

Client App
proxies as
User
performs
Operations
ds-auth-may-proxy-as
ds-auth-may-proxy-as-group
ds-auth-may-proxy-as-url
ds-auth-is-proxyable
ds-auth-is-proxyable-by
ds-auth-is-proxyable-by-group
ds-auth-is-proxyable-by-url
Add, Compare, Delete, Extended, Modify, Modify DN, Search

For example, if a client application, such as **`uid=clientApp`**, is requesting to search the **`ou=People,dc=example,dc=com`** branch as the user **`uid=admin`**, the command would look like this:

```
ldapsearch --bindDN uid=clientApp,dc=example,dc=com --bindPassword password
--proxyAs uid=admin,dc=example,dc=com --baseDN ou=People,dc=example,dc=com "(object-
class=*)
```

At bind, the Directory Server evaluates the list of users in the **`uid=clientApp`** entry based on the presence of any **`ds-auth-may-proxy-as-*`** attributes. In Figure 7-2, the **`uid=clientApp`** entry has a **`ds-auth-may-proxy-as`** attribute with a value, **`uid=admin`**, which means that the client app user may proxy only as the **`uid=admin`** account. Next, the server confirms that **`uid=admin`** is in the list of proxyable users and then evaluates the **`ds-auth-is-proxyable-*`** attributes present in the **`uid=admin`** entry. These attributes determine the list of restricted users that either are allowed, prohibited, or required to proxy as the **`uid=admin`** entry. In this case, the **`uid=admin`** entry has the **`ds-auth-is-proxyable`** attribute with a value of "**`required`**", which indicates that the entry can only be accessed by means of proxied authorization. The **`uid=admin`** entry also has the **`ds-auth-is-proxyable-by`** attribute with a value of **`uid=clientApp`**, which indicates it can only be requested by the **`uid=clientApp`** entry. Once both sets of attributes have been confirmed, the **`uid=clientApp`** can bind to the server as the authenticated user. From this point, the Directory Server performs ACI evaluation on the branch to determine if the requested user has access rights to the branch. If the branch is accessible by the **`uid=clientApp`** entry, and then the search request is processed.

FIGURE 7-2. **Proxying Attributes Example**


Client App
proxies as
uid=admin
performs
Operations
uid=clientApp,dc=example,dc=com
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as: uid=admin
uid=admin,dc=example,dc=com
ds-auth-is-proxyable: required
ds-auth-is-proxyable-by: uid=clientApp
ldapsearch
-D uid=clientApp,dc=example,dc=com
-w password
-Y uid=admin,dc=example,dc=com
-b ou=People,dc=example,dc=com
"(objectclass=*)"

About the ds-auth-may-proxy-as-* Operational Attributes

At bind, the Directory Server first evaluates the list of users that can be proxied for the authenticated user based on the presence of the **`ds-auth-may-*`** operational attributes in the entry. These operational attributes are multi-valued and are evaluated together if all are present in an entry:

- **ds-auth-may-proxy-as**. Specifies the user DNs that the associated user is allowed to proxy as. For instance, based on the previous example, you could specify in the **`uid=clientApp`** entry that it can proxy operations as **`uid=admin`** and **`uid=agent1`**.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as: uid=admin,dc=example,dc=com
ds-auth-may-proxy-as: uid=agent1,ou=admins,dc=example,dc=com
```

- **ds-auth-may-proxy-as-group**. Specifies the group DNs and its group members that the associated user is allowed to proxy as. For instance, you could specify that the potential users that the **`uid=clientApp`** entry can proxy as are those members who are present in the group **`cn=Agents,ou=Groups,dc=example,dc=com`**. This attribute is multi-valued, so that more than one group can be specified. Nested static or dynamic groups are also supported.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
```

```
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as: cn=Agents,ou=Groups,dc=example,dc=com
```

- **ds-auth-may-proxy-as-url**. Specifies the DNs that are returned based on the criteria defined in an LDAP URL that the associated user is allowed to proxy as. For instance, the attribute specifies that the client can proxy as those entries that match the criteria in the LDAP URL. This attribute is multi-valued, so that more than one LDAP URL can be specified.

```
dn: uid=clientApp,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
ds-auth-may-proxy-as: ldap:///ou=People,dc=example,dc=com??sub?(l=austin)
```

About the ds-auth-is-proxyable-* Operational Attributes

After the Directory Server has evaluated the list of users that the authenticated user can proxy as, the server checks to see if the requested authorized user is in the list. If the requested authorized user is present in the list, then the server continues processing the proxable attributes in the entry. If the requested authorized user is not present in the list, the bind will fail.

The operational attributes on the proxying entry are as follows:

- **ds-auth-is-proxyable**. Specifies whether the entry is proxyable or not. Possible values are: "`allowed`" (operation may be proxied as this user), "`prohibited`" (operations may not be proxied as this user), "`required`" (indicates that the account will not be allowed to authenticate directly but may only be accessed by some form of proxied authorization).

- **ds-auth-is-proxyable-as**. Specifies any users allowed to use this entry as a target of proxied authorization.

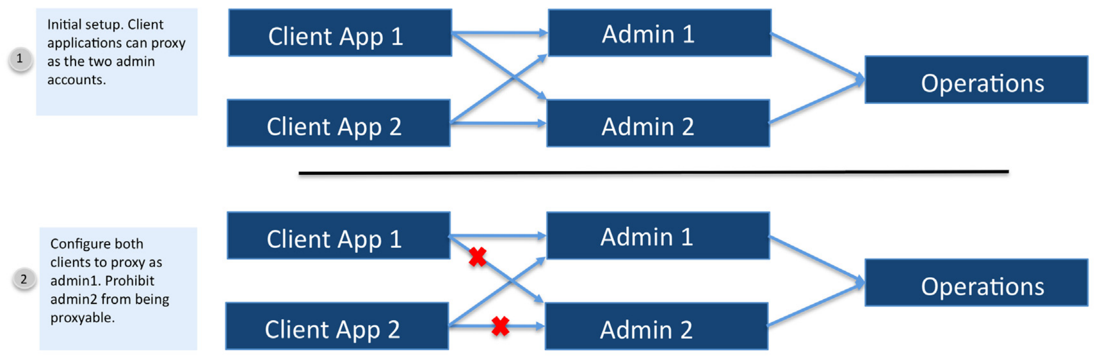
- **ds-auth-is-proxyable-as-group**. Specifies any groups allowed to use this entry as a target of proxied authorization. Nested static and dynamic groups are also supported.

- **ds-auth-is-proxyable-as-url**. Specifies the LDAP URLs that are used to determine any users that are allowed to use this entry as a target of proxied authorization.

Restricting Proxied Authorization for Specific Users

To illustrate how the proxied authorization operational attributes work, it is best to set up a simple example where two LDAP clients, `uid=clientApp1` and `uid=clientApp2` can freely proxy two administrator accounts, `uid=admin1` and `uid=admin2`. We will add the `ds-auth-may-proxy-as-*` and the `ds-auth-is-proxyable-*` attributes to these entries to restrict how each account can use proxied authorization. For example, the two client applications will continue to proxy the **uid=admin1** account but the **uid=admin2** account will no longer be able to be used as a proxied entry.

FIGURE 7-3. **Restricting Proxy Users Example Scenario**


1
Initial setup. Client applications can proxy as the two admin accounts.
Client App 1
Client App 2
Admin 1
Admin 2
Operations
2
Configure both clients to proxy as admin1. Prohibit admin2 from being proxyable.
Client App 1
Client App 2
Admin 1
Admin 2
Operations

To Restrict Proxied Authorization for a Specific User

1. For this example, set up two user entries, **uid=clientApp1** and **uid=clientApp2**, which will be proxying the **uid=admin1** and **uid=admin2** accounts to access the **ou=People,dc=example,dc=com** subtree. Both entries have the **proxied-auth** privilege assigned to it. Open a text editor and create an LDIF file. Add the file using the **ldapmodify** tool.

```
dn: uid=clientApp1,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth

dn: uid=clientApp2,ou=Applications,dc=example,dc=com
objectClass: top
...
ds-privilege-name: proxied-auth
```

2. Next, assign the ACI for each client application to the subtree, **ou=People,dc=example,dc=com**.

```
dn: ou=People,dc=example,dc=com
aci: (targetattr="*")(targetscope="subtree")
     (version 3.0; acl "People Proxy Access"; allow(proxy)
      userdn="ldap:///uid=clientApp1,ou=Applications,dc=example,dc=com";)
aci: (targetattr="*")(targetscope="subtree")
     (version 3.0; acl "People Proxy Access"; allow(proxy)
      userdn="ldap:///uid=clientApp2,ou=Applications,dc=example,dc=com";)
```

3. Run a search for each entry. In this example, assume that there are two admin accounts: **admin1** and **admin2** that have full access rights to user attributes. You should be able to proxy as the **uid=admin1** and **uid=admin2** entries to access the subtree for both clients.

```
$ bin/ldapsearch --port 1389 \
--bindDN "uid=clientApp1,ou=Applications,dc=example,dc=com" \
--bindPassword password \
--proxyAs "dn:uid=admin1,dc=example,dc=com" \
--baseDN ou=People,dc=example,dc=com \
"(objectclass=*)"

$ bin/ldapsearch --port 1389 \
--bindDN "uid=clientApp2,ou=Applications,dc=example,dc=com" \
--bindPassword password \
--proxyAs "dn:uid=admin2,dc=example,dc=com" \
--baseDN ou=People,dc=example,dc=com \
"(objectclass=*)"
```

4. Next, limit the proxied authorization capabilities for each client application. Update the `uid=clientApp1` entry to add the `ds-auth-may-proxy-as` attribute. In this example, the ds-auth-may-proxy-as attribute specifies that `uid=clientApp1` can proxy as the `uid=admin1` entry. Open a text editor, create the following LDIF file, save it, and add it using `ldapmodify`. Note that `ds-auth-may-proxy-as` is multi-valued:

```
dn: uid=clientApp1,ou=Applications,dc=example,dc=com
changetype: modify
add: ds-auth-may-proxy-as
ds-auth-may-proxy-as: uid=admin1,dc=example,dc=com
```

5. Repeat the previous step for the `uid=clientApp2` entry, except specify the `ds-auth-may-proxy-as-url`. The client entry may proxy as any DN that matches the LDAP URL.

```
dn: uid=clientApp2,ou=Applications,dc=example,dc=com
changetype: modify
add: ds-auth-may-proxy-as-url
ds-auth-may-proxy-as-url: ldap:///dc=example,dc=com??sub?(uid=admin*)
```

6. Next, we want to create a group of client applications that has `uid=clientApp1` and `uid=clientApp2` as its uniquemembers to illustrate the use of the `ds-auth-proxyable-by-group` attribute. In this example, set up a static group using the `groupOfUniqueNames` object class.

```
dn: ou=Groups,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: groups

dn: cn=Client Applications,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfUniqueNames
cn: Client Applications
ou: groups
uniquemember: uid=clientApp1,ou=Applications,dc=example,dc=com
uniquemember: uid=clientApp2,ou=Applications,dc=example,dc=com
```

7. Update the `uid=admin1` entry to provide the DN that it may be proxied as. Add the `ds-auth-is-proxyable` and the `ds-auth-is-proxyable-by` attributes. For instance, we make the `uid=admin1` a *required* proxyable entry, which means that it can only be accessed by some form of proxied authorization. Then, specify each DN that can proxy as `uid=admin1` using the `ds-auth-is-proxyable-by`. Open a text editor, create the following LDIF file, save it, and add it using `ldapmodify`.

   Note that the example includes all three types of `ds-auth-is-proxable-by-*` attributes as an illustration, but, in an actual deployment, only one type of attribute is necessary if they all target the same entries.

```
dn: uid=admin1,dc=example,dc=com
changetype: modify
add: ds-auth-is-proxyable
ds-auth-is-proxyable: required
-
add: ds-auth-is-proxyable-by
ds-auth-is-proxyable-by: ou=clientApp1,ou=Applications,dc=example,dc=com
ds-auth-is-proxyable-by: ou=clientApp2,ou=Applications,dc=example,dc=com
-
add: ds-auth-is-proxyable-by-group
ds-auth-is-proxyable-by-group: cn=Client Applications,ou=Groups,dc=example,dc=com
-
add: ds-auth-is-proxyable-by-url
ds-auth-is-proxyable-by-url: ldap:///ou=Applications,dc=exam-
ple,dc=com??sub?(uid=clientApp*)
```

8. Next, prohibit proxying for the **`uid=admin2`** entry by setting the **`ds-auth-is-proxyable`** to **`prohibited`**. Open a text editor, create the following LDIF file, save it, and add it using **`ldapmodify`**.

```
dn: uid=admin2,dc=example,dc=com
changetype: modify
add: ds-auth-is-proxyable
ds-auth-is-proxyable: prohibited
```

9. Run a search using the proxied account. For example, run a search first with **`uid=clientApp1`** or **`uid=clientApp2`** that proxies as **`uid=admin1`** to return a successful operation. However, if you run a search for **`uid=clientApp1`** that proxies as **`uid=admin2`**, as seen below, you will see an "authorization denied" message due to **`uid=admin2`** not matching the list of potential entries that can be proxied. The **`ds-auth-may-proxy-as-*`** attributes specify that the client can only proxy as **`uid=admin1`**:

```
$ bin/ldapsearch --port 1389 \
--bindDN "uid=clientApp1,ou=Applications,dc=example,dc=com" \
--bindPassword password \
--proxyAs "dn:uid=admin2,dc=example,dc=com" \
--baseDN ou=People,dc=example,dc=com \
"(objectclass=*)"

One of the operational attributes (ds-auth-may-proxy-as, ds-auth-may-proxy-as-
group, ds-auth-may-proxy-as-url) in user entry 'uid=clientApp1,ou=Applica-
tions,dc=example,dc=com' does not allow that user to be proxied as user
'uid=admin2,dc=example,dc=com'

Result Code:  123 (Authorization Denied)

Diagnostic Message:  One of the operational attributes (ds-auth-may-proxy-as, ds-
auth-may-proxy-as-group, ds-auth-may-proxy-as-url) in user entry
'uid=clientApp1,ou=Applications,dc=example,dc=com' does not allow that user to be
proxied as user 'uid=admin2,dc=example,dc=com'
```

10. Run another search using **`uid=clientApp2`**, which attempts to proxy as **`uid=admin2`**. You will see an "authorization denied" message due to the presence of the **`ds-auth-is-proxy-able:prohibited`** operational attribute, which states that **`uid=admin2`** is not available for proxied authorization.

```
$ bin/ldapsearch --port 1389 \
--bindDN "uid=clientApp2,ou=Applications,dc=example,dc=com" \
--bindPassword password \
--proxyAs "dn:uid=admin2,dc=example,dc=com" \
--baseDN ou=People,dc=example,dc=com \
"(objectclass=*)"

The 'ds-auth-is-proxyable' operational attribute in user entry 'uid=admin2,dc=exam-
ple,dc=com' indicates that user may not be accessed via proxied authorization

Result Code:  123 (Authorization Denied)

Diagnostic Message:  The 'ds-auth-is-proxyable' operational attribute in user entry
'uid=admin2,dc=example,dc=com' indicates that user may not be accessed via proxied
authorization
```

8 Managing Security

Overview

The UnboundID Directory Server provides a full suite of security features to secure communication between the client and the server, to establish trust between components (for example, for replication and administration), and to secure data. Internally, the Directory Server uses cryptographic mechanisms that leverage the Java 6 JRE's Java Secure Sockets Extension (JSSE) implementation of the SSL protocol using Key Manager and Trust Manager providers for secure connection integrity and confidentiality, and the Java Cryptography Architecture (JCA) for data encryption.

This chapter presents procedures to configure security and covers the following topics:

Overview of the Directory Server Security Features

The UnboundID Directory Server supports a strong set of cryptographic mechanisms to secure communication and data. The following security-related features are available:

- **SSL/StartTLS Support**. The UnboundID Directory Server supports the use of SSL and StartTLS to encrypt communication between the client and the server. Administrators can configure different certificates for each connection handler, or use the same certificate for all connection handlers. SSL or StartTLS can also be configured to secure communication between server components, like replicas in a replication topology. Additionally, the Directory Server allows for more fine-grained control of the key material used in connecting peers in SSL handshakes and trust material for storing certificates.

- **Message Digest/Encryption Algorithms**. The UnboundID Directory Server supports the use of a number of one-way message digests (e.g., CRYPT, 128-bit MD5, 160-bit SHA-1, and 256-bit, 384-bit, and 512-bit SHA-2 digests with or without salt) as well as a number of reversible encryption algorithms (BASE64, 3DES, AES, RC4, and Blowfish) for storing passwords. Note that even if passwords are encoded using reversible encryption, that encryption is intended for use only within the Directory Server itself, and the passwords will not be made available to administrators in unencrypted form. It is generally recommended that encrypted password storage only be used if you anticipate using an authentication mechanism that requires the server to have access to the clear-text representation of passwords, like CRAM-MD5 or DIGEST-MD5.

- **SASL Mechanism Support**. The UnboundID Directory Server supports a number of SASL mechanisms, including ANONYMOUS, CRAM-MD5, DIGEST-MD5, EXTERNAL, PLAIN, and GSSAPI. In some vendors' directory servers, the use of CRAM-MD5 and DIGEST-MD5 requires that the server have access to the clear-text password for a user. The Directory Server supports reversible encryption to store the passwords in a more secure encoding than clear text.

- **Password Policy Support**. The UnboundID Directory Server provides extensive password policy support including features, like customizable password attributes, maximum password age, maximum password reset age, multiple default password storage schemes, account expiration, idle account lockout and others. The Directory Server also supports a number of password storage schemes, like one-way digests (CRYPT, MD5, SMD5, SHA, SSHA, SSHA256, SSHA384, SSHA512) and reversible encryption (BASE64, 3DES, AES, RC4, BLOWFISH). Administrators can also use a number of password validators, like maximum password length, similarity to current password and the set of characters used. See "Managing Password Policies" on page 353 for more information.

- **Full-Featured Access Control System**. The UnboundID Directory Server provides a full-featured access control subsystem that determines whether a given operation is allowable based on a wide range of criteria. The access control system allows administrators to grant or restrict access to data, restrict the use of specific types of controls and extended operations and provides strong validation for access control rules before accepting them. See "Managing Access Control" on page 291 for more information.

- **Backup Protection**. The UnboundID Directory Server provides the ability to protect the integrity of backup contents using cryptographic digests and encryption. When generating a backup, the administrator has an option to generate a cryptographic digest of the backup contents and also optionally to digitally sign that digest. The Directory Server also has options to compress and/or encrypt the contents of the backup. When restoring the backup, the server can verify that the digest matches the content of the backup and generates an error if the backup has been changed from when it was initially written, making it tamper-evident. The server also provides the ability to verify the integrity of a backup without actually restoring it. See "Backing Up and Restoring Data" on page 159 for more information.

Performing Data Security Audits

The Directory Server provides an **audit-data-security** command-line tool that allows the administrator to assess any potential security risks in the directory data. The tool invokes a configurable set of data security auditors to identity the audit type, current account, password, and privilege settings. Each data security auditor generates a time-stamped report that the administrator can review and take action on. Reports from consecutive runs may be compared to see if changes made in the Directory Server resolved one or more identified issues.

The **audit-data-security** tool also provides refined auditing capabilities by allowing the administrator to specify the backends to audit as well as the specific severity and verbosity levels for each event. The tool can execute on backends from one of the following supported types: Local DB, LDIF, and Configuration. Note that only one data security audit can run on any Local DB backend at any given time. If multiple backends are audited (for example, Local DB and Configuration), the Data Security Auditors scan these backends simultaneously. The tool can be executed immediately or scheduled to run at a later time. If scheduled for a specific time, the tool invokes a Directory Server task that runs one or more Data Security Auditors on the selected backends.

Due to the sensitive nature of this process, only administrators with the **audit-data-security** privilege can execute data security audits.

Reports

Each Data Security Auditor generates a detailed report file named after the type of auditor used. By default, the security audit report files are saved in LDIF format to the **<server-root>/reports/audit-data-security/<timestamp>** directory. In the top-level report directory, there is a **summary.ldif** file that provides an overview of the audit results. Subdirectories for each backend store the detailed report files for that particular backend.

To View a Data Security Report

- Open a report file using a text editor. The following command opens the **entries-with-acis.ldif** report for the **userRoot** backend.

```
$ cat /UnboundID-DS/reports/audit-data-security/20110811201049Z/userRoot/entries-
with-acis.ldif

dn: dc=example,dc=com
objectClass: ds-audit-report-aci-entry
objectClass: extensibleObject
ds-audit-severity: notice
ds-audit-reason: presence of access control information
aci: (targetattr!="userPassword")(version 3.0; acl "Allow anonymous read access for
anyone"; allow (read,search,compare) userdn="ldap:///anyone";)
aci: (targetattr="*")(version 3.0; acl "Allow users to update their own entries";
allow (write) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Grant full access for the admin user"; allow
(all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

Data Security Auditors

The following table shows the available Data Security Auditors in the Directory Server. Each auditor is enabled by default but can be disabled using the **`dsconfig`** command:

TABLE 8-1. Data Security Auditors

Auditor	Description
ACCESS-CONTROL	Reports all entries with access control information.
DISABLED-ACCOUNT	Reports all disabled accounts.
EXPIRED-PASSWORD	Reports all accounts with expired passwords, accounts with passwords about to expire as well as accounts with passwords exceeding a specified age.
LOCKED-ACCOUNT	Reports locked accounts including the reason for locking.
MULTIPLE-PASSWORD	Reports entries with multiple password values. It is possible to configure the auditor to only report those entries that have multiple passwords using different password storage schemes.
PRIVILEGE	Reports entries with privileges. The report distinguishes between directly assigned privileges and privileges assigned by a virtual attribute.
WEAKLY-ENCODED-PASSWORD	Reports all entries that use one of the specified weak password storage schemes.

Configuring the Data Security Auditors

The Data Security Auditors can be configured using the **`dsconfig`** command-line utility. Each Data Security Auditor may be independently enabled or disabled. They may also be configured to include one or more attributes from the audited entry in the detailed report.

For more detailed information, each Data Security Auditor can be configured to report events at a specific severity and verbosity level. The three possible severity values are: Error, Warning, and Verbose. If the Warning level is selected, then both error and warning events are included in the reports. Similarly, if the Verbose level is selected, then the report will include events with any severity. By default, all Data Security Auditors are configured with the Warning audit severity.

To Configure Data Security Auditor

1. Run the **`dsconfig`** command and enter the connection parameters for your system.
2. Change to the Advanced menu option by entering "`o`" (the letter) and selecting **`Advanced`**.
3. On the UnboundID Directory Server configuration console main menu, enter the number corresponding to **Data Security Auditor**.
4. On the Data Security Auditor management menu, enter the number corresponding to **View and edit an existing Data Auditor.**
5. From the displayed list, enter the number corresponding to a Data Auditor that you want to modify.

6. For the specific Data Auditor, add or change any properties that you want to include in your security audit. Then, enter **`f`** to apply the changes.

The audit-data-security Tool

The **`audit-data-security`** tool runs in either interactive mode or non-interactive mode. By default, the tool executes security audits on all enabled Data Security Auditors on all supported backends (i.e., Local DB, LDIF, and Configuration). Administrators can adjust these settings using the **`dsconfig`** tool.

In non-interactive mode, the administrator can specify which Data Security Auditors should be executed during the audit, which backends the audit should scan, what entries should be included in the audit as well as specify an alternate output directory for the reports.

To Run the audit-data-security Tool

1. Run the **`audit-data-security`** tool to perform a full audit of your system.

```
$ bin/audit-data-security
```

2. Open the **`summary.ldif`** to view the summary audit report, which will be in the time-stamped directory created by the audit.

```
$ cat UnboundID-DS/reports/audit-data-security/20110811201049Z/summary.ldif
```

3. To view a specific audit report, open the report in the backend subdirectory. For example, the following command opens an auditor report for **`entries-with-acis.ldif`** for the **`userRoot`** backend.

```
$ cat UnboundID-DS/reports/audit-data-security/20110811201049Z/userRoot/entries-
with-acis.ldif
```

To Run a Security Audit on a Subset of Entries

- Run the **`audit-data-security`** tool to perform a security audit on a subset of entries in the userRoot backend but do not report on entries having privileges:

```
$ bin/audit-data-security --backendID userRoot --excludeAuditor PRIVILEGE \
  --reportFilter "(employeeType=contactor)"
```

UnboundID Directory Server SSL and StartTLS Support

The UnboundID Directory Server supports the use of SSL and/or StartTLS to secure communication with clients and other components of the directory environment.

Note	Although the term "SSL" (Secure Sockets Layer) has been superceded by "TLS" (Transport-Layer Security), the older term "SSL" will continue to be used in this document in order to make it easier to distinguish between the use of TLS as a general mechanism for securing communication and the specific use of the StartTLS extended operation.

LDAP-over-SSL (LDAPS)

The Directory Server provides the option of using dedicated connection handlers for LDAPS connections. LDAPS differs from LDAP in that upon connect, the client and server establish an SSL session before any LDAP messages are transferred. LDAPS connection handlers with SSL enabled only may only be used for secure communication, and connections must be closed when the SSL session is shut down.

StartTLS Support

The StartTLS extended operation provides a means to add SSL encryption to an existing plain-text LDAP connection. The client opens an unencrypted TCP connection to the server and, after processing zero or more LDAP operations over that clear-text connection, sends a StartTLS extended request to the server to indicate that the client-server communication should be encrypted.

To require the use of SSL for client connections accepted by a connection handler, set `use-ssl` to true for that connection handler. To allows clients to use StartTLS on a connection handler, the administrator must configure that connection handler to allow StartTLS. Because SSL and StartTLS are mutually exclusive, you cannot enable both SSL and StartTLS for the same connection handler (although you can have some connection handlers configured to use SSL and others configured to use StartTLS).

Managing Certificates

You can generate and manage certificates using a variety of commonly available tools, such as the Java keytool utility, which is a key and certificate management utility provided with the Java SDK. The keytool utility can be used to create keystores, which hold key material used in the course of establishing an SSL session, and truststores, which may be consulted to determined whether a presented certificate should be trusted.

Because there are numerous ways to create or obtain certificates, the procedures in this section will only present basic steps to set up your certificates. Many companies have their own certificate authorities or have existing certificates that they use in the servers, and in such environments you should follow the guidelines specific to your company's implementation.

The UnboundID Directory Server supports three keystore types: Java Keystore (JKS), PKCS#12, and PKCS#11.

- **Java Keystore (JKS)**. In most Java SE implementations, the JKS keystore is the default and preferred keystore format. JKS keystores may be used to hold certificates for other Java-based applications, but such keystores are likely not compatible with non-Java-based applications.

- **PKCS#12**. This keystore type is a well-defined standard format for storing a certificate or certificate chain, and may be used to hold certificates already in use for other types of servers. Most other servers that provide a proprietary format for storing certificates provide a mechanism for converting those certificates to PKCS#12.

- **PKCS#11**. Also, known as Cryptoki (pronounced "crypto-key") is a format for cryptographic token interfaces for devices, such as cryptographic smart cards, hardware accelerators, and high performance software libraries. PKCS#11 tokens may also offer a higher level of security than other types of keystores, and many of them have been FIPS 140-2 certified and may be tamper-evident or tamper-resistant.

Authentication Using Certificates

The Directory Server supports two different mechanisms for certificate-based authentication:

- **Client Certificate Validation**. The Directory Server can request the client to present its own certificate for client authentication during the SSL or StartTLS negotiation process. If the client presents a certificate, then the server will use the trust manager provider configured for the associated connection handler in order to determine whether to continue the process of establishing the SSL or StartTLS session. If the client certificate is not accepted by the trust manager provider, then the server will terminate the connection. Note that even if the client provides its own certificate to the server during the process of establishing an SSL or StartTLS session, the underlying LDAP connection may remain unauthenticated until the client sends an LDAP bind request over that connection.

- **SASL EXTERNAL Certificate Authentication**. The SASL EXTERNAL mechanism is used to allow a client to authenticate itself to the Directory Server using information provided outside of LDAP communication. In the Directory Server, that information must come in the form of a client certificate presented during the course of SSL or StartTLS negotiation. Once the client has established a secure connection to the server in which it provided its own client certificate, it may send a SASL EXTERNAL bind request to the server in order to request that the server attempt to identify the client based on information contained in that certificate. The server will then use a certificate mapper to identify exactly one user entry that corresponds to the provided client certificate, and it may optionally perform additional verification (e.g., requiring the certificate the client presented to be present in the `userCertificate` attribute of the user's entry). If the certificate mapper cannot identify exactly one user entry for that certificate, or if its additional validation is not satisfied, then the bind attempt will fail and the client connection will remain unauthenticated.

Creating Server Certificates using Keytool

You can generate and manage certificates using the *keytool* utility, which is available in the Java 1.6 SDK. The keytool utility is a key and certificate management utility that allows users to manage their public/private key pairs, x509 certificate chains and trusted certificates. The utility also stores the keys and certificates in a keystore, which is a password-protected file with a default format of JKS although other formats like PKCS#12 are available. Each key and trusted certificate in the keystore is accessed by its unique alias.

To Create a Server Certificate using Keytool

1. Change to the directory where the certificates will be stored.

```
$ cd /ds/UnboundID-DS/config
```

2. Use the **keytool** utility to create a private/public key pair and a keystore. The keytool utility is part of the Java 1.6 SDK. If you cannot access the utility, make sure to change your path to include the Java 1.6 SDK (**${JAVA_HOME}/bin**) directory.

   The following command creates a keystore named "keystore", generates a public/private key pair and creates a self-signed certificate based on the key pair. This certificate can be used as the server certificate or it can be replaced by a CA-signed certificate chain if subsequent keytool commands are executed. Self-signed certificates are only convenient for testing purposes, they are *not* recommended for use in deployments in which the set of clients is not well-defined and carefully controlled. If clients are configured to blindly trust any server certificate, then they may be vulnerable to man-in-the-middle attacks.

   The **-dname** option is used to specify the certificate's subject, which generally includes a **CN** attribute with a value equal to the fully-qualified name that clients will use to communicate with the Directory Server. Some clients may refuse to establish an SSL or StartTLS session with the server if the certificate subject contains a **CN** value which does not match the address that the client is trying to use, so this should be chosen carefully. If the **-dname** option is omitted, you will be prompted for input. The certificate will be valid for 180 days.

```
$ keytool -genkeypair \
  -dname "CN=server.example.com,ou=Directory Server Certificate,O=Example Company,C=US" \
  -alias server-cert -keyalg rsa \
  -keystore keystore -keypass changeit -storepass changeit \
  -storetype JKS -validity 180 -noprompt
```

Note	The **-keypass** and **-storepass** arguments can be omitted to cause the tool to interactively prompt for the password. Also, the key password should match the keystore password.

3. View the keystore. Notice the entry type is **`privateKeyEntry`** which indicates that the entry has a private key associated with it, which is stored in a protected format to prevent unauthorized access. Also note that the Owner and the Issuer are the same, indicating that this certificate is self-signed.

```
$ keytool -list -v -keystore keystore -storepass changeit

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: server-cert
Creation date: Sep 30, 2011
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=server.example.com, OU=Directory Server Certificate, O=Example Company, C=US
Issuer: CN=server.example.com, OU=Directory Server Certificate, O=Example Company, C=US
Serial number: 4ac3695f
Valid from: Wed Sep 30 09:21:19 CDT 2011 until: Mon Mar 29 09:21:19 CDT 2012
Certificate fingerprints:
 MD5:  3C:7B:99:BA:95:A8:41:3B:08:85:11:91:1B:E1:18:00
 SHA1: E9:7E:38:0F:1C:68:29:29:C0:B4:8C:08:2B:7C:DA:14:BF:41:DE:F5
 Signature algorithm name: SHA1withRSA
 Version: 3
```

4. If you are going to have your certificate signed by a Certificate Authority, skip to step 7. Otherwise export the self-signed certificate. Then examine the certificate.

```
$ keytool -export -alias server-cert -keystore keystore -rfc -file server.crt
Enter keystore password:
Certificate stored in file <server.crt>

$ cat server.crt
-----BEGIN CERTIFICATE-----
MIICVTCCAb6gAwIBAgIESsNpXzANBgkqhkiG9w0BAQUFADBvMQswCQYDVQQGEwJVUzEYMBYGA1UE
ChMPRXhhbXBsZSBDb21wYW55MS8wLQYDVQQLEyZVbmJvdW5kaWQgRGlyZWN0b3J5IFNlcnZlciBD
ZXJ0aWZpY2F0ZTEVMBMGA1UEAxMMMTcyLjE2LjE5My4xMB4XDTA5MDkzMDE0MjExOVoXDTEwMDMy
OTE0MjExOVowbzELMAkGA1UEBhMCVVMxGDAWBgNVBAoTD0V4YW1wbGUgQ29tcGFueTEvMC0GA1UE
CxMmVW5ib3VuZGlkIERpcmVjdG9yeSBRZXJ2ZXIgQ2VydGlmaWNhdGUxFTATBgNVBAMTDDE3Mi4x
Ni4xOTMuMTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAmRBpSeRcqur4XP8PjJWcGDVR31wE
cItmMImbjpf0rTq+KG8Ssp8+se+LjLHLaeNg3itR3xMBwp7mQ4E42i2PBIIZ0PwOKBRPxZDxpsIT
sSy3o9anTsopIVg1pUpST2iHGBQ+j+VY33cdcc5EoJwYykZ4d1iu45yc834VByXjiKUCAwEAATAN
BgkqhkiG9w0BAQUFAAOBgQCJIZfsfQuUig4F0kPC/0fFbhW96TrLTOi6AMIOTork1SuJlkxp/nT+
eD8eGoE+zshyJWTfVnzMDIlFMJwDIIVvnYmyeR1vlCchyJE6JyFiLpBWs6RuLD8iuHydYEwK8NkE
FYvVb/UIKqJlZ8H8+1Ippt0bENRnGD7zMwJv5ZE49w==
-----END CERTIFICATE-----
```

5. Import the self-signed certificate into a truststore, and then type **`yes`** to trust the certificate.

```
$ keytool -importcert -alias server-cert -file server.crt -keystore truststore \
  -storepass changeit

Owner: CN=172.13.201.1, OU=Directory Server Certificate, O=Example Company, C=US
Issuer: CN=172.13.201.1, OU=Directory Server Certificate, O=Example Company, C=US
Serial number: 4ac3695f
Valid from: Wed Sep 30 09:21:19 CDT 2011 until: Mon Mar 29 09:21:19 CDT 2012
Certificate fingerprints:
 MD5:  3C:7B:99:BA:95:A8:41:3B:08:85:11:91:1B:E1:18:00

 SHA1: E9:7E:38:0F:1C:68:29:29:C0:B4:8C:08:2B:7C:DA:14:BF:41:DE:F5
 Signature algorithm name: SHA1withRSA
 Version: 3
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

6. View the truststore with the self-signed certificate. If you intend to use this self-signed certificate as your server certificate, you are done. Note that the entry type of **`trustedCertEntry`** indicates that the keystore owner trusts that the public key in the certificate belongs to the entity identified by the owner of the certificate.

```
$ keytool -list -v -keystore truststore -storepass changeit

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: server-cert
Creation date: Sep 30, 2011
Entry type: trustedCertEntry

Owner: CN=server.example.com, OU=Directory Server Certificate, O=Example Company, C=US
Issuer: CN=server.example.com, OU=Directory Server Certificate, O=Example Company, C=US
Serial number: 4ac3695f
Valid from: Wed Sep 30 09:21:19 CDT 2011 until: Mon Mar 29 09:21:19 CDT 2012
Certificate fingerprints:
 MD5:  3C:7B:99:BA:95:A8:41:3B:08:85:11:91:1B:E1:18:00
 SHA1: E9:7E:38:0F:1C:68:29:29:C0:B4:8C:08:2B:7C:DA:14:BF:41:DE:F5
 Signature algorithm name: SHA1withRSA
 Version: 3
```

7. Create the Certificate Signing Request (CSR) by writing to the file **`server.csr`**. Follow the instructions of the third-party Certificate Authority (CA), and submit the file to a CA. The CA authenticates you and then returns a certificate reply, which you can save as **`signed.crt`**.

```
$ keytool -certreq -v -alias server-cert -keystore keystore \
  -storepass changeit -file server.csr
Certification request stored in file <server.csr>
Submit this to your CA
```

8. If you are working with a third-party CA or if your company has your own CA server, then both the key and trust stores should include information about the CA's root certificate as well as any intermediate certificates used to sign the Directory Server certificate. Obtain the CA root and any intermediate certificates to set up a chain of trust in your keystore. View the trusted CA and intermediate certificates to check that the displayed certificate fingerprints match the expected ones.

```
$ keytool -v -printcert -file root.crt
$ keytool -v -printcert -file intermediate.crt
```

9. Import the CA's root certificate in the keystore and truststore. If there are other intermediate certificates, then import them using the same commands, giving them each different aliases in the key and trust stores.

```
$ keytool -importcert -v -trustcacerts -alias cacert -keystore keystore -storepass
changeit -file root.crt

$ keytool -importcert -v -trustcacerts -alias cacert -keystore truststore -storepass
changeit -file root.crt
```

10. Import the Directory Server certificate signed by the CA into your keystore, which will replace the existing self-signed certificate created when the private key was first generated.

```
$ keytool -importcert -v -trustcacerts -alias server-cert \
  -keystore keystore -storepass changeit -file signed.crt

Owner: CN=server.example.com, OU=Directory Server Certificate, O=Example Company, C=US
Issuer: EMAILADDRESS=whatever@example.com, CN=Cert Auth, OU=My Certificate Authority, O=Example
Company, L=Austin, ST=Texas, C=US
Serial number: e19cb2838441dbb6
Valid from: Wed Sep 30 10:10:30 CDT 2011 until: Thu Sep 30 10:10:30 CDT 2012
Certificate fingerprints:
 MD5:  E0:C5:F7:CF:0D:13:F5:FC:2D:A6:A4:87:FD:4C:36:1A
 SHA1: E4:15:0B:ED:99:1C:13:47:29:66:76:A0:3B:E3:4D:60:33:F1:F8:21
 Signature algorithm name: SHA1withRSA
 Version: 1
Trust this certificate? [no]:  yes
Certificate was added to keystore
[Storing changeit]
```

11. Add the certificate to the truststore.

```
$ keytool -importcert -v -trustcacerts -alias server-cert -keystore truststore -sto-
repass changeit -file signed.crt
```

Creating Client Certificates using Keytool

The process for creating client certificates is nearly identical in the steps used to create server certificates. However, there are two important considerations for client certificates.

- If a client presents its own certificate to the server, then the server must trust that certificate. This generally means that self-signed client certificates are not acceptable for anything but testing purposes or cases in which there are very small number of clients that will be presenting their own certificates. Otherwise, it is not feasible to configure the server to trust every client certificate.

- If the client certificates will be used for LDAP authentication via SASL EXTERNAL, then the certificate must contain enough information to allow the Directory Server to associate it with exactly one user entry. The requirements for this are dependent upon the certificate mapper configured for use in the server, but this may impose constraints on the certificate (for example, the format of the certificate's subject).

Creating PKCS#12 Certificates

PKCS#12 is an industry standard format for deploying X.509 certificates (or certificate chains) and a private key as a single file. PKCS#12 is part of the family of standards called the Public-Key Cryptography Standard (PKCS) developed by RSA Laboratories.

To Generate PKCS#12 Certificates using Keytool

- In order to create a new certificate in PKCS#12 format, follow the same procedures as in "Creating Server Certificates using Keytool" on page 258, except use the **`--storetype pkcs12`** argument. For example, to create a PKCS#12 self-signed certificate and keystore, use the following command:

```
$ keytool -genkeypair -dname "CN=server.example.com,ou=Directory Server Certifi-
cate,O=Example Company,C=US" -alias server-cert -keyalg rsa -keystore keystore.p12
-keypass changeit -storepass changeit -storetype pkcs12 -validity 180 -noprompt
```

To Export a Certificate from an NSS Database in PKCS#12 Format

- Some directory servers, including the Oracle DSEE server, use the Network Security Services (NSS) library to manage certificates. If you have such a directory server and wish to migrate its certificates for use with the UnboundID Directory Server, then PKCS#12 can be used to accomplish this task.Use the **`pk12util`** NSS command-line utility to export a certificate from an NSS certificate database in PKCS12 format. You can use the PKCS#12 certificate when using QuickSetup or setting up SSL.

```
$ pk12util -o server.p12 -n server-cert -k /tmp/pwdfile -w /tmp/pwdfile \
  -d . -P "ds-"
nss-pk12util: PKCS12 EXPORT SUCCESSFUL
```

Working with PKCS#11 Tokens

The Cryptographic Token Interface Standard, PKCS#11, defines the native programming interfaces to cryptographic tokens, like Smartcards and hardware cryptographic accelerators. A security token provides cryptographic services. PKCS#11 provides an interface to cryptographic devices via "slots". Each slot, which corresponds to a physical reader or other device interface, may contain a token. A token is typically a PKCS#11 hardware token implemented in physical devices, such as hardware accelerators or smart cards. A software token is a PKCS#11 token implemented entirely in software.

Note	Because different types of PKCS#11 tokens have different mechanisms for creating, importing, and managing certificates, it may or may not be possible to achieve this using common utilities like keytool. In some cases (particularly for devices with strict FIPS 140-2 compliance), it may be necessary to use custom tools specific to that PKCS#11 token for managing its certificates. Consult the documentation for your PKCS#11 token for information about how to configure certificates for use with that token.

Configuring the Key and Trust Manager Providers

Java uses key managers to get access to certificates to use for SSL and StartTLS communication. Administrators use the Directory Server's key manager providers to provide access to keystore contents. There are three types of key manager providers:

- **JKS Key Manager Provider**. Provides access to certificates stored in keystores using the Java-default JKS format.

- **PKCS#11 Key Manager Provider**. Provides access to certificates maintained in PKCS#11 tokens.

- **PKCS#12 Key Manager Provider**. Provides access to certificates in PKCS#12 files.

Trust manager providers are used to determine whether to trust any client certificate that may be presented during the process of SSL or StartTLS negotiation. The available trust manager provider types include:

- **Blind Trust Manager Provider**. Automatically trusts any client certificate presented to the server. This should only be used for testing purposes. Never use it for production environments, because it can be used to allow users to generate their own certificates to impersonate other users in the server.

- **JKS Trust Manager Provider**. Attempts to determine whether to trust a client certificate, or the certificate of any of its issuers, is contained in a JKS-formatted file.

- **PKCS#12 Trust Manager Provider**. Attempts to determine whether to trust a client certificate, or the certificate of any of its issuers, is contained in a PKCS#12 file.

Configuring the JKS Key and Trust Manager Provider

The following procedures are identical to those in the previous section except that the **`dsconfig`** tool in non-interactive mode commands are presented from the command line.

To Configure the JKS Key Manager Provider

1. Change to the directory server install root.

```
$ cd /ds/UnboundID-DS
```

2. Create a text file containing the password for the certificate keystore. It is recommended that file permissions (or filesystem ACLs) be configured so that the file is only readable by the Directory Server user.

```
$ echo 'changeit' > config/keystore.pin
$ chmod 0400 keystore.pin
```

3. Use the **`dsconfig`** tool to enable the key manager provider.

```
$ bin/dsconfig set-key-manager-provider-prop \
  --provider-name JKS --set enabled:true \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

4. Use **`dsconfig`** to enable the trust manager provider.

```
$ bin/dsconfig set-trust-manager-provider-prop \
  --provider-name JKS --set enabled:true \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

5. Use **`dsconfig`** to enable the LDAPS connection handler. Port 636 is typically reserved for LDAPS, but if your server is using the port, you should specify another port, like 1636. If the certificate alias differs from the default "server-cert", use the **`--set ssl-cert-nickname:<aliasname>`** to set it, or you can let the server decide by using the **`--reset ssl-cert-nickname`** option. For example, if the server certificate has an alias of "server," add the option **`--set ssl-cert-nickname:server`** to the command.

```
 $ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAPS Connection Handler" \
  --set listen-port:1636 --set enabled:true \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

6. Test the listener port for SSL-based client connection on port 1636 to return the Root DSE. Type **yes** to trust the certificate.

```
$ bin/ldapsearch --port 1636 --useSSL --baseDN "" --searchScope base \
  "(objectclass=*)"
The server is using the following certificate:
    Subject DN:  CN=179.13.201.1, OU=Directory Server
    Certificate, O=Example Company, L=Austin, ST=Texas, C=US
    Issuer DN:  EMAILADDRESS=whatever@example.com, CN=Cert Auth,
    OU=My Certificate Authority, O=Example Company, L=Austin,
    ST=Texas, C=US
    Validity:  Fri Sep 25 15:21:10 CDT 2011 through Sat Sep 25 15:21:10 CDT 2012
Do you wish to trust this certificate and continue connecting to the server?
Please enter 'yes' or 'no':yes
```

7. If desired, you may disable the LDAP Connection Handler so that communication can only go through SSL.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAP Connection Handler" \
  --set enabled:false --port 1389 \
  --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

Configuring the PKCS#12 Key Manager Provider

PKCS#12 (sometimes referred to as the Personal Information Exchange Syntax Standard) is a standard file format used to store private keys with its accompanying public key certificates, protected with a password-based symmetric key.

To Configure the PKCS#12 Key Manager Provider

1. Change to the directory server root.

```
$ cd /ds/UnboundID-DS
```

2. Create a text file containing the password for the certificate keystore. It is recommended that file permissions (or filesystem ACLs) be configured so that the file is only readable by the Directory Server user.

```
$ echo ’changeit’ > config/keystore.pin
$ chmod 0400 keystore.pin
```

3. Use the **dsconfig** tool to configure and enable the PKCS#12 key manager provider.

```
$ bin/dsconfig --no-prompt set-key-manager-provider-prop \
  --provider-name PKCS12 \
  --set enabled:true \
  --set key-store-file:/config/keystore.p12 \
  --set key-store-type:PKCS12 \
  --set key-store-pin-file:/config/keystore.pin
```

4. Use the **dsconfig** tool to configure and enable the PKCS#12 trust manager provider.

```
$ bin/dsconfig --no-prompt set-trust-manager-provider-prop \
  --provider-name PKCS12 \
  --set enabled:true \
  --set trust-store-file:/config/truststore.p12
```

5. Use **`dsconfig`** to enable the LDAPS connection handler. Port 636 is typically reserved for LDAPS, but if your server is using the port, you should specify another port, like 1636. If the certificate alias differs from the default "server-cert", use the **`--set ssl-cert-nickname:<aliasname>`** to set it, or you can let the server decide by using the **`--reset ssl-cert-nickname`** option. For example, if the server certificate has an alias of "server," add the option **`--set ssl-cert-nickname:server`** to the command.

```
$ bin/dsconfig --no-prompt set-connection-handler-prop \
  --handler-name "LDAPS Connection Handler" \
  --set enabled:true \
  --set listen-port:2636 \
  --set ssl-cert-nickname:1 \
  --set key-manager-provider:PKCS12 \
  --set trust-manager-provider:PKCS12
```

Configuring the PKCS#11 Key Manager Provider

The Cryptographic Token Interface (Cryptoki), or PKCS#11, format defines a generic interface for cryptographic tokens used in single sign-on or smartcard systems. The Directory Server supports PKCS#11 keystores.

Note	The Directory Server does not provide a PKCS#11 trust manager, so an alternate trust manager provider (for example, JKS or PKCS#12) should be configured if the PKCS#11 key manager provider is to be used.

To Configure the PKCS#11 Key Manager Provider

1. Change to the directory server root.

```
$ cd /ds/UnboundID-DS
```

2. Create a text file containing the password for the certificate keystore. It is recommended that file permissions (or filesystem ACLs) be configured so that the file is only readable by the Directory Server user.

```
$ echo 'changeit' > config/keystore.pin
$ chmod 0400 keystore.pin
```

3. Use the **`dsconfig`** tool to configure and enable the PKCS#11 key manager provider.

```
$ bin/dsconfig --no-prompt set-key-manager-provider-prop \
  --provider-name PKCS11 \
  --set enabled:true \
  --set key-store-type:PKCS11 \
  --set key-store-pin-file:/config/keystore.pin
```

4. Use the **`dsconfig`** tool to enable the trust manager provider. Since there is no PKCS#11 trust manager provider, then you must use one of the other truststore provider types (for example, JKS or PKCS#12).

```
$ bin/dsconfig --no-prompt set-trust-manager-provider-prop \
  --provider-name JKS \
  --set enabled:true \
  --set trust-store-file:/config/truststore.jks
```

5. Use **dsconfig** to enable the LDAPS connection handler. Port 636 is typically reserved for LDAPS, but if your server is using the port, you should specify another port, like 1636. If the certificate alias differs from the default "server-cert", use the **--set ssl-cert-nickname:<aliasname>** to set it, or you can let the server decide by using the **--reset ssl-cert-nickname** option. For example, if the server certificate has an alias of "server," add the option **--set ssl-cert-nickname:server** to the command.

```
$ bin/dsconfig --no-prompt set-connection-handler-prop \
  --handler-name "LDAPS Connection Handler" \
  --set enabled:true \
  --set listen-port:1636 \
  --set ssl-cert-nickname:1 \
  --set key-manager-provider:PKCS11 \
  --set trust-manager-provider:JKS
```

Configuring the Blind Trust Manager Provider

The Blind Trust Manager provider accepts any peer certificate presented to it and is provided for testing purposes only. This trust manager should not be used in production environments , because it can allow any client to generate a certificate that could be used to impersonate any user in the server.

To Configure the Blind Trust Manager Provider

1. Change to the directory server install root.

```
$ cd /ds/UnboundID-DS
```

2. Use the **dsconfig** tool to enable the blind trust manager provider.

```
$ bin/dsconfig --no-prompt set-trust-manager-provider-prop \
  --provider-name "Blind Trust" --set enabled:true \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

3. Use **dsconfig** to enable the LDAPS connection handler. Port 636 is typically reserved for LDAPS, but if your server is using the port, you should specify another port, like 1636. If the certificate alias differs from the default "server-cert", use the **--set ssl-cert-nickname:<aliasname>** to set it, or you can let the server decide by using the **--reset ssl-cert-nickname** option. For example, if the server certificate has an alias of "server," add the option **--set ssl-cert-nickname:server** to the command.

```
$ bin/dsconfig --no-prompt set-connection-handler-prop \
  --handler-name "LDAPS Connection Handler" \
  --set enabled:true \
  --set listen-port:1636 \
  --set ssl-cert-nickname:1 \
  --set "trust-manager-provider:Blind Trust"
```

Enabling SSL in the Directory Server

The UnboundID Directory Server provides a means to enable SSL and/or StartTLS at installation time, using either an existing certificate or by automatically generating a self-signed certificate. However, if SSL was not configured at install time, then it may be enabled at any time using the following process. These instructions assume that the certificate is available in a JKS-formatted keystore, but a similar process may be used for certificates available through other mechanisms like a PKCS#12 file or a PKCS#11 token.

To Enable SSL in the Directory Server

1. Change to the installation directory.

   ```
   $ cd /ds/UnboundID-DS
   ```

2. Create a text file containing the password for the certificate keystore. It is recommended that file permissions (or filesystem ACLs) be configured so that the file is only readable by the Directory Server user.

   ```
   $ echo 'changeit' > config/keystore.pin
   $ chmod 0400 config/keystore.pin
   ```

3. Run the **dsconfig** command with no arguments in order to launch the **dsconfig** tool in interactive mode. Enter the connection parameters when prompted.

4. On the Directory Server Configuration Console main menu, enter "o" (lowercase letter "o") to change the complexity of the configuration objects menu. Select the option to show objects at the Standard menu.

5. On the Directory Server Configuration Console main menu, enter the number corresponding to the Key Manager Provider.

6. On the Key Manager Provider management menu, select the option to view and edit an existing key manager.

7. On the Key Manager Provider menu, enter the option for JKS. You will see other options, like Null, PKCS11, and PKCS12.

8. Make any necessary changes to the JKS key manager provider for the keystore that you will be using. The **enabled** property must have a value of **TRUE**, the **key-store-file** property must reflect the path to the keystore file containing the server certificate, and the **key-store-pin-file** property should reflect the path to a file containing the password to use to access the keystore contents.

9. On the Enabled Property menu, enter the option to change the value to **TRUE**.

10. On the File Based Key Manager Provider, type "f" to save and apply the changes.

11. Return to the **`dsconfig`** main menu, and enter the number corresponding to Trust Manager Provider.

12. On the Trust Manager Provider management menu, enter the option to view and edit an existing trust manager provider.

13. On the Trust Manager Provider menu, enter the option for JKS. You will see other options for Blind Trust (accepts any certificate) and PKCS12 reads information about trusted certificates from a PKCS#12 file.

14. Ensure that the JKS trust manager provider is enabled and that the **`trust-store-file`** property has a value that reflects the path to the truststore file to consult when deciding whether to trust any presented certificates.

15. On the File Based Trust Manager Provider menu, type "f" to save and apply the changes.

16. Return to the **`dsconfig`** main menu, enter the number corresponding to Connection Handler.

17. On the Connection Handler management menu, enter the option to view and edit and existing connection handler.

18. On the Connection Handler menu, enter the option for LDAPS Connection Handler. You will see other options for JMX Connection Handler and LDAP Connection Handler.

19. On the LDAP Connection Handler menu, ensure that the connection handler has an appropriate configuration for use. The **`enabled`** property should have a value of **`TRUE`**, the **`listen-port`** property should reflect the port on which to listen for SSL-based connections, and the **`ssl-cert-nickname`** property should reflect the alias for the target certificate in the selected keystore. Finally, when completing the changes, type "f" to save and apply the changes.

20. Verify that the server is properly configured to accept SSL-based client connections using an LDAP-based tool like **`ldapsearch`**. For example:

```
$ bin/ldapsearch --port 1636 --useSSL --baseDN "" --searchScope base \
  "(objectclass=*)"
The server is using the following certificate:
    Subject DN:  CN=179.13.201.1, OU=Directory Server
    Certificate, O=Example Company, L=Austin, ST=Texas, C=US
    Issuer DN:  EMAILADDRESS=whatever@example.com, CN=Cert Auth,
    OU=My Certificate Authority, O=Example Company, L=Austin,
    ST=Texas, C=US
    Validity:  Fri Sep 25 15:21:10 CDT 2011 through Sat Sep 25 15:21:10 CDT 2012
Do you wish to trust this certificate and continue connecting to the server?
Please enter 'yes' or 'no':yes
```

21. If desired, you may disable the LDAP connection handler so only the LDAPS connection handler will be enabled and the server will only accept SSL-based connections.

Configuring StartTLS

The StartTLS extended operation is used to initiate a TLS-secured communication channel over a clear-text connection, such as an insecure LDAP connection. The main advantage of StartTLS is that it provides a way to use a single connection handler capable of both secure and insecure communication rather than requiring a dedicated connection handler for secure communication.

To Configure StartTLS

1. Use **`dsconfig`** to configure the Connection Handler to allow StartTLS. The **`allow-start-tls`** property cannot be set if SSL is enabled. The connection handler must also be configured with a key manager provider and a trust manager provider.

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "LDAP Connection Handler" \
  --set allow-start-tls:true \
  --set key-manager-provider:JKS \
  --set trust-manager-provider:JKS\
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. Use **`ldapsearch`** to test StartTLS.

```
$ bin/ldapsearch -p 1389 --useStartTLS -b "" -s base "(objectclass=*)"

The server is using the following certificate:
    Subject DN:  CN=Server Cert, OU=Directory Server Certificate, O=Example Company, L=Austin,
ST=Texas, C=US
    Issuer DN:  EMAILADDRESS=whatever@example.com, CN=Cert Auth, OU=My Certificate Authority,
O=Example Company, L=Austin, ST=Texas, C=US
    Validity:  Thu Oct 29 10:29:59 CDT 2011 through Fri Oct 29 10:29:59 CDT 2012

Do you wish to trust this certificate and continue connecting to the server?
Please enter 'yes' or 'no':yes

dn:
objectClass: ds-root-dse
objectClass: top
startupUUID: 6fa8f196-d112-40b4-b8d8-93d6d44d59ea
```

Authentication Mechanisms

The UnboundID Directory Server supports the use of both simple and Simple Authentication and Security Layer (SASL) authentication.

Simple Authentication

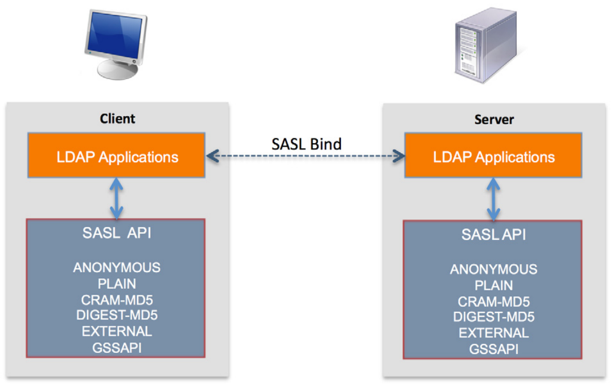
Simple authentication allows a client to identify itself to the Directory Server using the DN and password of the target user. Because the password is provided in the clear, simple authentication is inherently insecure unless the client communication is encrypted using a mechanism like SSL or StartTLS.

If both the DN and password of a simple bind request are empty (i.e., zero-length strings), then the server will process it as an anonymous bind. This will have no effect if the client is not already authenticated, but it can be used to destroy any previous authentication session and revert the connection to an unauthenticated state as if no bind had ever been performed on that connection.

Working with SASL Authentication

SASL (Simple Authentication and Security Layer, defined in RFC 4422) provides an extensible framework that can be used to add suport for a range of authentication and authorization mechanisms. The UnboundID Directory Server provides support for a number of common SASL mechanisms.

FIGURE 8-1. **Simple Authentication and Security Layer**



Working with the SASL ANONYMOUS Mechanism

The ANONYMOUS SASL mechanism does not actually perform any authentication or authorization, but it can be used to destroy an existing authentication session. It also provides an option to allow the client to include a trace string, which can be used to identify the purpose of the connection. Because there is no authentication, the content of the trace string cannot be trusted.

The SASL ANONYMOUS mechanism is disabled by default but can be enabled if desired using the **`dsconfig`** tool. The SASL configuration options are available as an Advanced menu option using **`dsconfig`** in interactive mode.

The LDAP client tools provided with the Directory Server support the use of SASL ANONYMOUS. The optional "trace" SASL option may be used to specify the trace string to include in the bind request.

To Configure SASL ANONYMOUS

1. Use **`dsconfig`** to enable the SASL ANONYMOUS mechanism.

```
$ bin/dsconfig set-sasl-mechanism-handler-prop \
  --handler-name ANONYMOUS --set enabled:true \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. Use **`ldapsearch`** to view the root DSE and enter a trace string in the access log.

```
$ bin/ldapsearch --port 1389 --saslOption mech=ANONYMOUS \
  --saslOption "trace=debug trace string" --baseDN "" \
  --searchScope base "(objectclass=*)"
dn:
objectClass: ds-root-dse
objectClass: top
startupUUID: 59bab79d-4429-49c8-8a88-c74a86792f26
```

3. View the access log using a text editor in **`/ds/UnboundID-DS/logs`** folder.

```
[26/Oct/2011:16:06:33 -0500] BIND RESULT conn=2 op=0 msgID=1 resultCode=0 addition-
alInfo="trace='debug trace string'" etime=345.663 clientConnectionPolicy="default"
```

Working with the SASL PLAIN Mechanism

SASL PLAIN is a password-based authentication mechanism which uses the following information:

- **Authentication ID**. Used to identify the target user to the server. It should be either "dn:" followed by the DN of the user or "u:" followed by a username. If the "u:"-style syntax is used, then an identify mapper will be used to map the specified username to a user entry. An authentication ID of "dn:" that is not actually followed by a DN may be used to request an anonymous bind.

- **Clear-text Password**. Specifies the password for the user targeted by the authentication ID. If the given authentication ID was "dn:", then this should be an empty string.

- **Optional Authorization ID**. Used to request that operations processed by the client be evaluated as if they had been requested by the user specified by the authorization ID rather than the authentication ID. It can allow one user to issue requests as if he/she had authenticated as another user. The use of an alternate authorization identity will only be allowed for clients with the **`proxied-auth`** privilege and the **`proxy`** access control permission.

Because the bind request includes the clear-text password, SASL PLAIN bind requests are as insecure as simple authentication. In order to avoid an observer from capturing passwords sent over the network, it is recommended that SASL PLAIN binds be issued over secure connections.

By default, the SASL PLAIN mechanism uses an Exact Match Identity Mapper that expects the provided username to exactly match the value of a specified attribute in exactly one entry (for example, the provided user name must match the value of the **`uid`** attribute). However, an alternate identity mapper may be configured for this purpose which can identify the user in other ways (for example, transforming the provided user name with a regular expression before attempt to find a user entry with that transformed value).

LDAP clients provided with the Directory can use SASL PLAIN with the following SASL options:

- authID. Specifies the authentication ID to use for the bind. This must be provided.
- authzID. Specifies an optional alternate authorization ID to use for the bind.

To Configure SASL PLAIN

1. Use **`dsconfig`** to enable the SASL PLAIN mechanism.

```
$ bin/dsconfig set-sasl-mechanism-handler-prop \
  --handler-name PLAIN --set enabled:true \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. Use **`ldapsearch`** to view the root DSE using the authentication ID (authid) with the username jdoe. The authid option is required. Enter a password for the authentication ID.

```
$ bin/ldapsearch --port 1389 --saslOption mech=PLAIN \
  --saslOption "authid=u:jdoe" --baseDN "" \
  --searchScope base "(objectclass=*)"
Password for user 'u:jdoe':
dn:
objectClass: ds-root-dse
objectClass: top
startupUUID: 59bab79d-4429-49c8-8a88-c74a86792f26
```

Note You can also specify the fully DN of the user when using the SASL PLAIN option:

```
$ bin/ldapsearch --port 1389 --saslOption mech=PLAIN \
  --saslOption "authid=dn:uid=jdoe,ou=People,dc=example,dc=com" \
  --baseDN "" --searchScope base "(objectclass=*)"
Password for user 'dn:uid=jdoe,ou=People,dc=example,dc=com':
```

Working with the SASL CRAM-MD5 Mechanism

CRAM-MD5 is a password-based SASL mechanism that prevents exposure of the clear-text password by authenticating through the use of an MD5 digest generated from a number of elements, including the clear-text password, the provided authentication ID, and a challenge comprised of randomly-generated data. This ensures that the clear-text password itself is not transmitted, and the inclusion of server-generated random data protects against replay attacks.

During the CRAM-MD5 session, the client sends a bind request of type SASL CRAM-MD5. The Directory Server sends a response with a SASL "Bind in Progress" result code plus credential information that includes a randomly generated challenge string to the LDAP client. The client combines that challenge with other information, including the authentication ID and clear-text password and uses that to generate an MD5 digest to be included in the SASL credentials, along with a clear-text version of the authentication ID. When the Directory Server receives the second request, it will receive the clear-text password from the target user's entry and generate the same digest. If the digest that the server generates matches what the client provided, then the client will have successfully demonstrated that it knows the correct password.

Note that although CRAM-MD5 does offer some level of protection for the password, so that it is not transferred in the clear, the MD5 digest that it uses is not as secure as the encryption used by SSL or StartTLS. As such, authentication mechanisms that use a clear-text password are more secure communication channel. However, the security that CRAM-MD5 offers may be sufficient for cases in which the performance overhead that SSL/StartTLS can incur. It is available for use in the UnboundID Directory Server because some clients may require it.

Also note that in order to successfully perform CRAM-MD5 authentication, the Directory Server must be able to obtain the clear-text password for the target user. By default, the Directory Server encodes passwords using a cryptographically secure one-way digest that does not allow it to determine the clear-text representation of the password. As such, if CRAM-MD5 may be used, then the password storage schemes for any users that may authenticate in this manner should be updated so that they will use a password storage scheme which supports reversible encryption, and it will be necessary for any existing users to change their passwords so that those passwords will be stored in reversible form. The reversible storage schemes supported by the Directory Server include:

- 3DES
- AES
- BASE64
- BLOWFISH
- CLEAR
- RC4

CRAM-MD5 uses an authentication ID to identify the user as whom to authenticate. The format of that authentication ID may be either "dn:" followed by the DN of the target user (or just "dn:" to perform an anonymouse bind), or "u:" followed by a username. If the "u:"-style syntax is chosen, then an identity mapper will be used to identify the target user based on that username. The **dsconfig** tool may be used to configure the identify mapper to use CRAM-MD5 authentication.

The LDAP client tools provided with the Directory Server support the use of CRAM-MD5 authentication. The "authID" SASL option should be used to specify the authentication ID for the target user.

To Configure SASL CRAM-MD5

1. Use **dsconfig** to enable the SASL CRAM-MD5 mechanism if it is disabled. By default, the CRAM-MD5 mechanism is enabled.

```
$ bin/dsconfig set-sasl-mechanism-handler-prop \
  --handler-name CRAM-MD5 --set enabled:true \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. For this example, create a Password Policy for CRAM-MD5 using a reversible password storage scheme, like 3DES.

```
$ bin/dsconfig create-password-policy \
  --policy-name "Test UserPassword Policy" \
  --set password-attribute:userpassword \
  --set default-password-storage-scheme:3DES \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

3. Use **ldapmodify** to add the **ds-pwp-password-policy-dn** attribute to an entry to indicate the Test Password Policy should be used for that entry.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Test Password Policy,cn=Password Policies,cn=config
<press CNTL-D>
Processing MODIFY request for uid=jdoe,ou=People,dc=example,dc=com
MODIFY operation successful for DN uid=jdoe,ou=People,dc=example,dc=com
```

4. Use **ldapmodify** to change the **userPassword** to a reversible password storage scheme. The password storage scheme is specified in the user's Password Policy.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: secret

Processing MODIFY request for uid=jdoe,ou=People,dc=example,dc=com
MODIFY operation successful for DN uid=jdoe,ou=People,dc=example,dc=com
```

Note	An alternate method to change the **userPassword** attribute password storage scheme is to deprecate the old scheme and then bind as the user using simple authentication or SASL PLAIN. This action will cause any existing password encoding using a deprecated scheme to be re-encoded with the existing scheme.

5. Use **ldapsearch** to view the root DSE using the authentication ID (**authid**) option with the username **jdoe**. The authid option is required. Enter a password for the user.

```
$ bin/ldapsearch --port 1389 --saslOption mech=CRAM-MD5 \
  --saslOption "authid=u:jdoe" --baseDN "" --searchScope base "(objectclass=*)"
Password for user 'u:jdoe':
dn:
objectClass: ds-root-dse
objectClass: top
startupUUID: 50567aa3-acd2-4106-a077-37a092275363
```

Working with the SASL DIGEST-MD5 Mechanism

The Directory Server supports the SASL DIGEST-MD5 mechanism, which is a stronger mechanism than SASL CRAM-MD5. Like the SASL CRAM-MD5 mechanism, the client authenticates to the Directory Server using a stronger digest of the authentication ID plus other information without exposing its clear-text password over the network.

During the DIGEST-MD5 session, the client sends a bind request of type SASL DIGEST-MD5. The Directory Server sends a response with a "Bind in Progress" message plus credential information that includes a random challenge string to the LDAP client. The client responds by sending a bind response that includes a digest of the server's random string, a separately generated client string, the authentication ID, the authorization ID if supplied, the user's clear-text password and some other information. The client then sends its second bind request. The Directory Server also calculates the digest of the client's credential. The Directory Server validates the digest and retrieves the client's password. Upon completion, the server sends a success message to the client.

The SASL DIGEST-MD5 mechanism has some security flaws, similar to CRAM-MD5 but is supported by the Directory Server for applications that use it. Like SASL CRAM-MD5, the client and the server must know the clear-text password for the user. Administrators must change the default one-way storage scheme (SSHA) to a reversible storage scheme in the user's password policy. One-way storage schemes encode a password and stores it but does not later decode it when accessed or compared with another value. Two-way storage schemes are encoded, stored, and later decoded when requested. The following password storage schemes are reversible:

- 3DES
- AES
- BASE64
- BLOWFISH
- CLEAR
- RC4

SASL DIGEST-MD5 uses the default Exact Match Identity Mapper that returns a results if the authorization ID exactly matches the username. Administrators can also configure the Identity Mapper to use regular expressions using the **`dsconfig`** tool. In many ways, the DIGEST-MD5 SASL mechanism is very similar to the CRAM-MD5 mechanism. It avoids exposing the clear-text password through the use of an MD5 digest generated from the password and other information. DIGEST-MD5 is considered stronger than CRAM-MD5 because it includes additional information in the digest (for example, randomly-generated data from the client in addition to the server-provided challenge as well as other elements like a realm and digest URI). DIGEST-MD5 also offers the ability to use an alternate authorization ID to indicate that operations should be processed under the authority of another user.

Because DIGEST-MD5 requires that the server have access to the target user's clear-text password, the passwords for users must be stored with a reversible password storage scheme that provides the ability to obtain that clear-text password.

Also, like CRAM-MD5, DIGEST-MD5 provides better security than sending the clear-text password over an unencrypted channel, but it is weaker than sending a clear-text password over an encrypted connection. Requiring that the server store passwords in a reversible form is an additional security risk, and as such, *it is generally recommended that CRAM-MD5 and DIGEST-MD5 be avoided unless required by clients*.

The LDAP client tools provided with the Directory Server provide the ability to use DIGEST-MD5 authentication using the following properties:

- **authID**. Specifies the authentication ID for the target user, in either the "dn:" or "u:" forms. This property is required.
- **authzID**. Specifies an optional authorization ID that should be used for operations processed on the connection.
- **realm**. The realm in which the authentication should be processed. This may or may not be required, based on the server configuration.
- **digest-uri**. The digest URI that should be used for the bind. It should generally be "ldap/" followed by the fully-qualified address for the Directory Server. If this is not provided, then a value will be generated.
- **qop**. The quality of protection to use for the bind request. At present, only "auth" is supported (indicating that the DIGEST-MD5 bind should only be used for authentication and should not provide any subsequent integrity or confidentiality protection for the connection), and if no value is provided then "auth" will be assumed.

To Configure SASL DIGEST-MD5

1. Use `dsconfig` to enable the SASL DIGEST-MD5 mechanism if it is disabled. By default, the DIGEST-MD5 mechanism is enabled.

```
$ bin/dsconfig set-sasl-mechanism-handler-prop \
  --handler-name DIGEST-MD5 --set enabled:true \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

2. Set up a reversible password storage scheme as outlined “Working with the SASL CRAM-MD5 Mechanism” on page 272, steps 2–5, which is also required for DIGEST-MD5.

3. Use `ldapsearch` to view the root DSE using the authentication ID with the username `jdoe`. The authid option is required. Enter a password for the authentication ID.

```
$ bin/ldapsearch --port 1389 --saslOption mech=DIGEST-MD5 \
  --saslOption "authid=u:proxy" --baseDN "" \
  --searchScope base "(objectclass=*)"
Password for user 'u:proxy':
dn:
objectClass: ds-root-dse
objectClass: top
startupUUID: 2188e4d4-c2bb-4ab9-8e1c-848e0168c9de
```

4. Use `ldapsearch` with the `authid` (required) and `authzid` option to test SASL DIGEST-MD5.

```
$ bin/ldapsearch --port 1389 --saslOption mech=DIGEST-MD5 \
  --saslOption authid=u:abergin \
  --saslOption authzid=dn:uid=proxy,ou=People,dc=example,dc=com \
  --base "" --searchScope base "(objectclass=*)"
Password for user 'u:abergin':
```

```
dn:
objectClass: ds-root-dse
objectClass: top
startupUUID: 2188e4d4-c2bb-4ab9-8e1c-848e0168c9de
```

Working with the SASL EXTERNAL Mechanism

The SASL EXTERNAL mechanism allows a client to authenticate using information about the client which is available to the server but that is not directly provided over LDAP. In the UnboundID Directory Server, SASL EXTERNAL requires the use of a client certificate provided during SSL or StartTLS negotiation. This is a very secure authentication mechanism that does not require the use of passwords, although its use on a broad scale is generally only feasible in environments with a PKI deployment.

Prior to the SASL EXTERNAL session exchange, the client should have successfully established a secure communication channel using SSL or StartTLS, and the client must have presented its own certificate to the server in the process. The SASL EXTERNAL bind request itself does not contain any credentials, and the server will use only the information contained in the provided client certificate to identify the target user.

The Directory Server's configuration settings for SASL EXTERNAL includes three important properties necessary for its successful operation:

- **certificate-validation-policy**. Indicates whether to check to see if the certificate presented by the client is present in the target user's entry. Possible values are:
  - **always** - Always require the peer certificate to be present in the user's entry. Authentication will fail if the user's entry does not contain any certificates, or if it contains one or more certificates and the certificate presented by the client is not included in the user's entry.
  - **ifpresent** - (Default) If the user's entry contains one or more certificates, require that one of them match the peer certificate. Authentication will be allowed to succeed if the user's entry does not have any certificates, but it will fail if the user's entry has one or more certificates and the certificate provided by the client is not included in the user's entry.
  - **never** - Do not look for the peer certificate to be present in the user's entry. Authentication may succeed if the user's entry does not contain any client certificates, or if the user's entry contains one or more certificates regardless of whether the provided certificate is included in that set.

- **certificate-attribute**. Specifies the name of the attribute that holds user certificates to be examined if the `ds-cfg-certificate-validation-policy` attribute has a value of `ifpresent` or `always`. This property must specify the name of a valid attribute type defined in the server schema. Default value is `userCertificate`. Note that LDAP generally requires certificate values to use the ";binary" attribute modifier, so certificates should actually be stored in user entries using the attribute "`userCertificate;binary`" rather than just "`userCertificate`".

- **certificate-mapper**. Specifies the certificate mapper that will be used to identify the target user based on the certificate presented by the client. For more information on certificate mappers, see “Configuring Certificate Mappers” on page 284. The LDAP client tools provided with the Directory Server support the use of SASL EXTERNAL authentication. This mechanism does not require any specific SASL options to be provided (other than "mech=EXTERNAL" to indicate that SASL EXTERNAL should be used). However, additional arguments are required in order to use SSL or StartTLS, and to provide a keystore so that a client certificate will be available.

To Configure SASL EXTERNAL

1. Change to the installation directory.

```
$ cd /ds/UnboundID-DS
```

2. Determine the `certificate-validation-policy` property. If you do not need to store the DER-encoded representation of the client’s certificate in the user’s entry, skip to the next step.

   If you select Always, you must ensure that the user’s entry has the `userCertificate` attribute present with a value. If you select `ifpresent`, you can optionally have the `userCertificate` attribute present. You can store the client’s certificate in the user entry using `ldapmodify.`

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
add: userCertificate;binary
userCertificate;binary:<file:///path/to/client.der
```

3. If you have an attribute other than `userCertificate`, than specify it using the `certificate-attribute` property. You may need to update your schema to support the attribute.

4. Determine the `certificate-mapper` property. For more information on certificate mappers, see “Configuring Certificate Mappers” on page 284.

5. Use `dsconfig` to enable the SASL EXTERNAL mechanism if it is disabled. By default, the SASL mechanism is enabled. For this example, set the `certificate-mapper` property to "Subject Attribute to User Attribute". All other defaults are kept.

```
$ bin/dsconfig set-sasl-mechanism-handler-prop \
  --handler-name EXTERNAL --set enabled:true \
  --set "certificate-mapper:Subject Attribute to User Attribute" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

6. Use `ldapsearch` to test SASL EXTERNAL.

```
$ bin/ldapsearch --port 1636 --useSSL \
  --keyStorePath /path/to/clientkeystore
  --keyStorePasswordFile /path/to/clientkeystore.pin \
  --trustStorePath /path/to/truststore \
  --saslOption mech=EXTERNAL --baseDN "" \
  --searchScope base "(objectClass=*)
```

Working with the GSSAPI Mechanism

The SASL GSSAPI mechanism provides the ability to authenticate LDAP clients using Kerberos V, which is a single sign-on mechanism commonly used in enterprise environments. In these environments, user credentials are stored in the Kerberos key distribution center (KDC) rather than the Directory Server. When an LDAP client attempts to authenticate to the Directory Server using GSSAPI, a three-way exchange occurs that allows the client to verify its identity to the server through the KDC.

The Directory Server's support for GSSAPI is based on the Java Authentication and Authorization Service (JAAS). By default, the server will automatically generate a JAAS configuration that should be appropriate for the most common use cases, but for more complex deployments it is possible for an administrator to supply a custom JAAS configuration that is most appropriate for that environment.

While the GSSAPI specification includes a provision for protecting client-server communication through *integrity* (in which the communication is not encrypted, but is signed so that it is possible to guarantee that it was not be altered in transit) or *confidentiality* (in which the communication is encrypted so that it cannot be examined by third-party observers), the Directory Server currently supports GSSAPI only for the purpose of authenticating clients but not for securing their communication with the server.

Preparing the Kerberos Environment for GSSAPI Authentication

To implement GSSAPI authentication in the Directory Server, it is assumed that you already have a working Kerberos V deployment in which the Directory Server and LDAP clients will participate. The process for creating such a deployment is beyond the scope of this documentation, and you should consult the documentation for your operating system to better understand how to construct a Kerberos deployment. However, there are a few things to keep in mind:

- It is recommended that the KDC be configured to use "aes128-cts" as the TKT and TGS encryption type, as this encryption type should be supported by all Java VMs. Some other encryption types may not be available by default in some Java runtime environments. In Kerberos environments using the MIT libraries, this can be achieved by ensuring that the following lines are present in the **`[libdefaults]`** section of the **`/etc/krb.conf`** configuration file on the KDC system:

  ```
  default_tkt_enctypes = aes128-cts
  default_tgs_enctypes = aes128-cts
  permitted_enctypes = aes128-cts
  ```

- When a client uses Kerberos to authenticate to a server, the addresses of the target server and the KDC are used in cryptographic operations. It is important to ensure that all systems agree on the addresses of the Directory Server and KDC systems. It is therefore *strongly* recommended that DNS be configured so that the primary addresses for the KDC and Directory Server systems are the addresses that clients will use to communicate with them.

- Kerberos authentication is time-sensitive and if system clocks are not synchronized, then authentication may fail. It is therefore *strongly* recommended that NTP or some other form of time synchronization be used for all KDC, Directory Server, and client systems.

In order to authenticate itself to the Kerberos environment, the KDC should include both host and service principals for all Directory Server systems. The host principal is in the form "host/" followed by the fully-qualified address of the server system, and the service principal should generally be "ldap/" followed by the fully-qualified address (for example, "**`host/directory.example.com`**" and "**`ldap/directory.example.com`**", respectively). In a MIT Kerberos environment, the **`kadmin`** utility may be used to create these principals, as follows:

```
# /usr/sbin/kadmin -p kws/admin
Authenticating as principal kws/admin with password.
Password for kws/admin@EXAMPLE.COM:
kadmin: add_principal -randkey host/directory.example.com
WARNING: no policy specified for host/directory.example.com@EXAMPLE.COM; defaulting to
no policy
Principal "host/directory.example.com@EXAMPLE.COM" created.
kadmin: ktadd host/directory.example.com
Entry for principal host/directory.example.com with kvno 3, encryption type AES-128 CTS
mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: add_principal -randkey ldap/directory.example.com
WARNING: no policy specified for ldap/directory.example.com@EXAMPLE.COM; defaulting to
no policy
Principal "ldap/directory.example.com@EXAMPLE.COM" created.
kadmin: quit
```

On each Directory Server system, the service principal for that instance must be exported to a keytab file, which may be accomplished using a command as follows:

```
# /usr/sbin/kadmin -p kws/admin
Authenticating as principal kws/admin with password.
Password for kws/admin@EXAMPLE.COM:
kadmin: ktadd -k /ds/UnboundID-DS/config/server.keytab ldap/directory.example.com
Entry for principal ldap/directory.example.com with kvno 4, encryption type AES- 128
CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/ds/UnboundID-DS/config/
server.keytab.
kadmin: quit
```

Because this file contains the credentials that the Directory Server will use to authenticate to the KDC, it is *strongly* recommended that appropriate protection be taken to ensure that it is only accessible to the Directory Server itself (for example, by configuring file permissions and/or file system access controls).

Configuring the GSSAPI SASL Mechanism Handler

The GSSAPI SASL mechanism handler provides the following configuration options:

- **enabled**. Indicates whether the GSSAPI SASL mechanism handler is enabled for use in the server. By default, it is disabled.
- **kdc-address**. Specifies the address that the Directory Server should use in order to communicate with the KDC. If this is not specified, then the server will attempt to determine it from the underlying system configuration.
- **server-fqdn**. Specifies the fully-qualified domain name that clients will use to communicate with the Directory Server. If this is not specified, the server will attempt to determine it from the underlying system configuration.

- **realm**. Specifies the Kerberos realm that clients will use. If this is not specified, the server will attempt to determine it from the underlying system configuration.
- **kerberos-service-principal**. Specifies the service principal that the Directory Server wil use to authenticate itself to the KDC. If this is not specified, the service principal will be "`ldap/`" followed by the fully-qualified server address (for example, `ldap/directory.example.com`).
- **keytab**. Specifies the path to the keytab file that holds the credentials for the Kerberos service principal that the Directory Server will use to authenticate itself to the KDC. If this is not specified, the server will use the system-wide keytab.
- **identify-mapper**. Specifies the identify mapper that the Directory Server will use to map a client's Kerberos principal to the entry of the corresponding user account in the server. In the default configuration, the server will use a regular expression identity mapper that will look for an entry with a `uid` value equal to the username portion of the Kerberos principal For example, for a Kerberos principal of `jdoe@EXAMPLE.COM`, the identity mapper will perform an internal search with a filter of `(uid=jdoe)`.
- **enable-debug**. Indicates whether the Directory Server should write debugging information about Kerberos-related processing (including JAAS processing) that the server performs. If enabled, this information will be written to standard error, which will appear in the `logs/server.out` log file.
- **jaas-config file**. Specifies the path to a JAAS configuration file that the server should use. If this is not specified, the server will generate a JAAS configuration file based on the values of the other configuration properties. It is recommended that this only be used in extraordinary circumstances in which the server-generated JAAS configuration is not acceptable.

Testing GSSAPI Authentication

Once the GSSAPI SASL mechanism handler has been enabled and configured in the Directory Server, then clients should be able to use GSSAPI to authenticate to the server using Kerberos. The `ldapsearch` tool provided with the Directory Server may be used to test this, with a command like:

```
$ bin/ldapsearch --hostname directory.example.com --port 389 \
    --saslOption mech=GSSAPI --saslOption authID=jdoe@EXAMPLE.COM \
    --baseDN "" --searchScope base "(objectClass=*)"
```

If the client already has a valid Kerberos session authenticated with a principal of `jdoe@EXAMPLE.COM`, then this command should make use of that existing session and proceed without requiring any further credentials. If there is no existing Kerberos session, then the `ldapsearch` command will prompt for the Kerberos password for that user (or it may be supplied using either the `--bindPassword` or `--bindPasswordFile` arguments).

The `--saslOption` command-line argument may be used to specify a number of properties related to SASL authentication, with values to that option be given in "name=value" format. When using SASL authentication, the `mech` property must always be used to specify the SASL mechanism to use, and `--saslOption mech=GSSAPI` indicates that the GSSAPI mechanism will be used. When the GSSAPI mechanism has been selected, then the following additional SASL options are available for use:

- **authid**. Specifies the authentication ID, which is the Kerberos principal for the user authenticating to the server. This option must always be provided when using GSSAPI.
- **authzID**. Specifies the authorization ID that should be used. At present, the Directory Server does not support the use of an alternate authorization identity, so this should either be omitted or identical to the value of the `authID` property.
- **kdc**. Specifies the address of the KDC that the client should use during the authentication processing. If this is not provided, the client will attempt to determine it from the system's Kerberos configuration.
- **realm**. Specifies the Kerberos realm that should be used. If this is not provided, the client will attempt to determine it from the system's Kerberos configuration.
- **protocol**. Specifies the protocol that the Directory Server uses for its service principal (i.e., the portion of the service principal that appears before the slash and fully-qualified server address). If this is not provided, a default protocol of "`ldap`" will be used.
- **useTicketCache**. Indicates whether the client should attempt to make use of a Kerberos ticket cache in order to leverage an existing Kerberos session, which may allow the client to authenticate to the server without the need to supply any additional credentials. If this is not provided, or if it is provided with a value of `TRUE`, then a ticket cache will be used if available. The use of a ticket cache may be disabled by providing this option with a value of `FALSE`.
- **requireCache**. Indicates whether to require the use of a ticket cache in order to leverage an existing Kerberos session rather than allowing the use of user-supplied credentials for authentication. By default, this will be assumed to have a value of `FALSE`, but if it is provided with a value of `TRUE`, then authentication will only be successful if the user already has an existing Kerberos session. This will be ignored if the `useTicketCache` option has been provided with a value of `FALSE`.
- **ticketCache**. Specifies the path to the file to use as the Kerberos ticket cache. If this is not provided, the default ticket cache file path will be assumed. This will be ignored if the `useTicketCache` option has been provided with a value of `FALSE`.
- **renewTGT**. Indicates whether to attempt to renew the user's ticket-granting ticket when authenticating with an existing Kerberos session. If this is not provided, a default value of `FALSE` will be used.
- **debug**. Indicates whether to write debug information about the GSSAPI authentication processing to standard error. By default, no debug information will be written but that may be enabled with a value of `TRUE`.

- **configFile**. Used to specify the path to a JAAS configuration file that the client should use when performing GSSAPI processing. If this is not specified, then a default JAAS configuration file will be generated based on other properties.

These options are available for use with all tools supplied with the Directory Server which support SASL authentication.

Adding Operational Attributes that Restrict Authentication

The Directory Server provides a number of operational attributes that can be added to user entries in order to restrict the way those users can authenticate and the circumstances under which they can be used for proxied authorization. The "is-proxyable-by" values are used to create a whitelist of potential authentication IDs (authNs) for the authorization ID (authZ). If none of these attributes are present, then the authZ user can be proxied as anyone with the prerequisite rights. The "may-proxy-as" attribute values are used to create a whitelist, which is evaluated first to make sure authorization ID (authZ) is in that list. If none of those attributes are on the entry, then this evaluation will allow proxy as anyone.

The operational attributes are as follows:

- **ds-auth-allowed-address** -- Used to indicate that the user should only be allowed to authenticate from a specified set of client systems. Values should be specified as individual IP addresses, IP address patterns (using wildcards like "1.2.3.*", CIDR notation like "1.2.3.0/24", or subnet mask notation like "1.2.3.0/255.255.255.0"), individual DNS addresses, or DNS address patterns (using wildcards like "*.example.com"). If no allowed address values are present in a user entry, then no client address restrictions will be enforced for that user.

- **ds-auth-allowed-authentication-type** -- Used to indicate that the user should only be allowed to authenticate in certain ways. Allowed values include "simple" (to indicate that the user should be allowed to bind using simple authentication) or "sasl {mech}" (to indicate that the user should be allowed to bind using the specified SASL mechanism, like "sasl PLAIN"). If no authentication type values are present in a user entry, then no authentication type restrictions will be enforced for that user.

- **ds-auth-require-secure-authentication** -- Used to specify whether the user should be required to authenticate in a secure manner. If this attribute is present with a value of "true", then that user will only be allowed to authenticate over a secure connection or using a mechanism that does not expose user credentials (e.g., the CRAM-MD5, DIGEST-MD5, and GSSAPI SASL mechanisms). If this attribute is present with a value of "false", or it is not present in the user's entry, then the user will not be required to authenticate in a secure manner.

- **ds-auth-require-secure-connection** -- Used to specify whether the user should be required to communicate with the server over a secure connection. If this attribute is present in a user entry with a value of "true", then that user will only be allowed to communicate with the server over a secure connection (using SSL or StartTLS). If this attribute is present with a value of "false", or if it is not present in the user's entry, then the user will not be required to use a secure connection.

- **ds-auth-is-proxyable** -- Used to indicate whether the user can be used as the target of proxied authorization (using the proxied authorization v1 or v2 control, the intermediate client control, or a SASL mechanism that allows specifying an alternate authorization identity). If this attribute is present in a user entry with a value of "required", then that user will not be allowed to authenticate directly to the server but instead will only be allowed to be referenced by proxied authorization. If this attribute is present with a value of "prohibited", then that user will not be allowed to be the target of proxied authorization but may only authenticate directly to the server. If this attribute is present with a value of "allowed", or if it is not present in the user's entry, then the user may authenticate directly against the server or be the target of proxied authorization.

- **ds-auth-is-proxyable-by** -- Used to restrict the set of accounts that may target the user for proxied authorization. If this attribute is present in a user's entry, then its values must be the DNs of the users that can target the user for proxied authorization (as long as those users have sufficient rights to use proxied authorization). If it is absent from the user's entry, then any account with appropriate rights may target the user via proxied authorization.

- **ds-auth-is-proxyable-by-group**. Used to restrict the set of accounts that target a list of the group DNs whose members will be allowed to proxy operations on behalf of the associated user.

- **ds-auth-is-proxyable-by-url**. Used to restrict the set of accounts that target a list of the LDAP URLs that will be allowed to proxy operations on behalf of the associated user.

- **ds-auth-may-proxy-as**. Used to restrict the set of accounts that target a list of the DNs of the users that the associated user will be allowed to proxy operations as.

- **ds-auth-may-proxy-as-group**. Used to restrict the set of accounts that target a list of the group DNs whose members the associated user will be allowed to proxy operations as.

- **ds-auth-may-proxy-as-url**. Used to restrict the set of accounts that target a list of the LDAP URLs the associated user will be allowed.

Configuring Certificate Mappers

SASL EXTERNAL requires that a certificate mapper be configured in the server. The certificate mapper is used to identify the entry for the user to whom the certificate belongs. The Directory Server supports a number of certificate mapping options including:

- **Subject Equals DN**. The Subject Equals DN mapper expects the subject of the certificate to exactly match the DN of the associated user entry. This option is not often practical as certificate subjects (e.g., `cn=jdoe,ou=Client Cert,o=Example Company,c=Aus-`

tin,st=Texas,c=US) are not typically in the same form as a directory entry (e.g., **cn=jdoe,ou=People,o=Example Company**, or **uid=jdoe,ou=People,dc=example,dc=com**).

- **Fingerprint**. The Fingerprint mapper expects the user's entry to contain an attribute (**ds-certficate-fingerprint** by default, although this is configurable), whose values are the SHA-1 or MD5 fingerprints of the certificate(s) that they can use to authenticate. This attribute must be indexed for equality.
- **Subject Attribute to User Attribute**. The Subject Attribute to User Attribute mapper can be used to build a search filter to find the appropriate user entry based on information contained in the certificate subject. For example the default configuration expects the "cn" value from the certificate subject to match the "cn" value of the user's entry, and the "e" value from the certificate subject to match the "mail" value of the user's entry.
- **Subject DN to User Attribute**. The Subject DN to User Attribute mapper expects the user's entry to contain an attribute (**ds-certificate-subject-dn** by default, although this is configurable), whose values are the subjects of the certificate(s) that they can use to authenticate. This multi-valued attribute can contain the subjects of multiple certificates. The attribute must be indexed for equality.

Configuring the Subject Equals DN Certificate Mapper

The Subject Equals DN Certificate Mapper is the default mapping option for the SASL EXTERNAL mechanism. The mapper requires that the subject of the client certificate exactly match the distinguished name (DN) of the corresponding user entry. The mapper, however, is only practical if the certificate subject has the same format as your Directory Server’s entries.

To Configure the Subject Equals DN Certificate Mapper

- Change the certificate mapper for the SASL EXTERNAL mechanism.

```
$ bin/dsconfig --no-prompt set-sasl-mechanism-handler-prop \
  --handler-name EXTERNAL \
  --set "certificate-mapper:Subject Equals DN" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

Configuring the Fingerprint Certificate Mapper

The Fingerprint Mapper causes the server to compute an MD5 or SHA-1 fingerprint of the certificate presented by the client and performs a search to find that fingerprint value in a user’s entry (**ds-certificate-fingerprint** by default). The **ds-certificate-fingerprint** attribute can be added to the user’s entry together with the **ds-certificate-user** auxiliary object class. For multiple certificates, the attribute can have separate values for each of the acceptable certificates. If you decide to use this attribute, you must index the attribute as it is not indexed by default.

The following example will use this certificate:

```
Alias name: client-cert
Creation date: Oct 29, 2011
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=jdoe, OU=Client Cert, O=Example Company, L=Austin, ST=Texas, C=US
Issuer: EMAILADDRESS=whatever@example.com, CN=Cert Auth, OU=My Certificate Authority,
O=Example Company, L=Austin, ST=Texas, C=US
Serial number: e19cb2838441dbcd
Valid from: Thu Oct 29 13:07:10 CDT 2011 until: Fri Oct 29 13:07:10 CDT 2012
Certificate fingerprints:
     MD5:  40:73:7C:EF:1B:4A:3F:F4:9B:09:C3:50:2B:26:4A:EB
     SHA1: 2A:89:71:06:1A:F5:DA:FF:51:7B:3D:2D:07:2E:33:BE:C6:5D:97:13
     Signature algorithm name: SHA1withRSA
     Version: 1
```

To Configure the Fingerprint Certificate Mapper

1. Create an LDIF file to hold a modification that adds the **`ds-certificate-user`** object class and **`ds-certificate-fingerprint`** attribute to the target user’s entry.

   ```
   dn: uid=jdoe,ou=People,dc=example,dc=com
   changetype: modify
   add: objectClass
   objectClass: ds-certificate-user
   -
   add: ds-certificate-fingerprint
   ds-certificate-fingerprint: 40:73:7C:EF:1B:4A:3F:F4:9B:09:C3:50:2B:26:4A:EB
   ```

   Then, apply the change to the entry using **`ldapmodify`**:

   ```
   $ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
     --bindPassword secret --filename add-cert-attr.ldif

   Processing MODIFY request for uid=jdoe,ou=People,dc=example,dc=com
   MODIFY operation successful for DN uid=jdoe,ou=People,dc=example,dc=com
   ```

2. Check that the attribute was added to the entry using **`ldapsearch`**.

   ```
   $ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
     --bindPassword secret --baseDN dc=example,dc=com "(uid=jdoe)" \
     ds-certificate-fingerprint

   dn: uid=jdoe,ou=People,dc=example,dc=com
   ds-certificate-fingerprint:40:73:7C:EF:1B:4A:3F:F4:9B:09:C3:50:2B:26:4A:EB
   ```

3. Create an index for the **`ds-certificate-fingerprint`** attribute. If the server is configured with multiple data backends, then the attribute should be indexed in each of those backends.

   ```
   $ bin/dsconfig --no-prompt create-local-db-index --backend-name userRoot \
     --index-name ds-certificate-fingerprint --set index-type:equality \
     --port 1389 --bindDN "cn=Directory Manager" \
     --bindPassword secret
   ```

4. Use the **`rebuild-index`** tool to cause an index to be generated for this attribute.

```
$ bin/rebuild-index --task --port 1389 --bindDN "cn=Directory Manager" --bindPass-
word secret --baseDN dc=example,dc=com --index ds-certificate-fingerprint

[14:56:28]  The console logging output is also available in '/ds/UnboundID-DS/logs/
tools/rebuild-index.log'
[14:56:29]  Due to changes in the configuration, index dc_example_dc_com_ds-certif-
icate-fingerprint.equality is currently operating in a degraded state and must be
rebuilt before it can used
[14:56:29]  Rebuild of index(es) ds-certificate-fingerprint started with 161 total
records to process
[14:56:29]  Rebuild complete. Processed 161 records in 0 seconds (average rate
1125.9/sec)
```

5. Change the certificate mapper for the SASL EXTERNAL mechanism.

```
$ bin/dsconfig --no-prompt set-sasl-mechanism-handler-prop \
  --handler-name EXTERNAL \
  --set "certificate-mapper:Fingerprint Mapper" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

Configuring the Subject Attribute to User Attribute Certificate Mapper

The Subject Attribute to User Attribute Certificate Mapper maps common attributes from the subject of the client certificate to the user's entry. The generated search filter must match exactly one entry within the scope of the base DN(s) for the mapper. If no match is returned or if multiple machines entries are found, the mapping fails.

Given the subject of the client certificate:

```
Owner: CN=John Doe, OU=Client Cert, O=Example Company, L=Austin, ST=Texas, C=US
```

We want to match to the following user entry:

```
dn: uid=jdoe,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: jdoe
givenName: John
sn: Doe
cn: John Doe
mail: jdoe@example.com
```

To Configure the Subject Attribute to User Attribute Certificate Mapper

- Change the certificate mapper for the SASL EXTERNAL mechanism.

```
$ bin/dsconfig --no-prompt set-sasl-mechanism-handler-prop \
  --handler-name EXTERNAL \
  --set "certificate-mapper:Subject Attribute to User Attribute" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

Configuring the Subject DN to User Attribute Certificate Mapper

The Subject DN to User Attribute Certificate mapper expects the user's entry to contain an attribute (**`ds-certificate-subject-dn`** by default) whose values match the subjects of the certificates that the user can use to authenticate. The **`ds-certificate-subject-dn`** attribute can be added to the user's entry together with the **`ds-certificate-user`** auxiliary object class. The attribute is multi-valued and can contain the Subject DNs of multiple certificates. The certificate mapper must match exactly one entry, or the mapping will fail.

If you decide to use this attribute, you must add an equality index for this attribute in all data backends.

To Configure the Subject DN to User Attribute Certificate Mapper

1. Create an LDIF file to hold a modification that adds the **`ds-certificate-user`** object class and **`ds-certificate-subject-dn`** attribute to the target user's entry.

```
dn: uid=jdoe,ou=People,dc=example,dc=com
changetype: modify
add: objectClass
objectClass: ds-certificate-user
-
add: ds-certificate-subject-dn
ds-certificate-subject-dn:CN=John Doe,OU=Client Certificate,O=Example Com-
pany,L=Austin,ST=Texas,C=US
```

   Then, apply the change to the entry using **`ldapmodify`**:

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --filename add-cert-attr.ldif

Processing MODIFY request for uid=jdoe,ou=People,dc=example,dc=com
MODIFY operation successful for DN uid=jdoe,ou=People,dc=example,dc=com.
```

2. Check that the attribute was added to the entry using **`ldapsearch`**.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN dc=example,dc=com "(uid=jdoe)" \
  ds-certificate-subject-dn

dn: uid=jdoe,ou=People,dc=example,dc=com
ds-certificate-fingerprint:CN=jdoe, OU=Client Cert, O=Example Company, L=Austin,
ST=Texas, C=US
```

3. Create an index to the **`ds-certificate-subject-dn`** attribute.

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
  --index-name ds-certificate-subject-dn --set index-type:equality \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

4. Use the **`rebuild-index`** tool to ensure that the index is properly generated in all appropriate backends.

```
$ bin/rebuild-index --task --port 1389 --bindDN "cn=Directory Manager" --bindPass-
word secret --baseDN dc=example,dc=com --index ds-certificate-subject-dn
```

```
[15:39:19]  The console logging output is also available in '/ds/UnboundID-DS/logs/
tools/rebuild-index.log'
[15:39:20]  Due to changes in the configuration, index dc_example_dc_com_ds-certif-
icate-subject-dn.equality is currently operating in a degraded state and must be
rebuilt before it can used
[15:39:20]  Rebuild of index(es) ds-certificate-subject-dn started with 161 total
records to process
[15:39:20]  Rebuild complete. Processed 161 records in 0 seconds (average rate
2367.6/sec)
```

5. Change the certificate mapper for the SASL EXTERNAL mechanism.

```
$ bin/dsconfig  --no-prompt set-sasl-mechanism-handler-prop \
  --handler-name EXTERNAL \
  --set "certificate-mapper:Subject DN to User Attribute" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

9 Managing Access Control

Overview

The UnboundID Directory Server provides a fine-grained access control model to ensure that users are able to access the information they need, but are prevented from accessing information that they should not be allowed to see. It also includes a privilege subsystem that provides even greater flexibility and protection in many key areas.

This chapter presents the access control model and provides examples that illustrate the use of key access control functionality.

Working with Access Control

The access control model uses access control instructions (ACIs), which are stored in the **`aci`** operational attribute, to determine what a user or a group of users can do with a set of entries, down to the attribute level. The operational attribute can appear on any entry and will affect the entry or any subentries within that branch of the directory information tree (DIT).

Key Access Control Features

The UnboundID Directory Server provides important access control features that provide added security for the directory server's entries.

Improved Validation and Security

The Directory Server provides an access control model with strong validation to help ensure that invalid ACIs are not allowed into the server. For example, the Directory Server ensures that all access control rules added over LDAP are valid and can be fully parsed. Any operation that attempts to store one or more invalid ACIs are rejected. The same validation is applied to ACIs contained in data imported from an LDIF file. Any entry containing a malformed aci value will be rejected.

Note If you will be migrating access control rules from another server (for example, an Oracle/Sun Directory Server, which uses a very similar syntax for ACIs), then you may wish to use the **`validate-acis`** tools before attempting the migration to determine whether those access control rules will be accepted by the UnboundID Directory Server. The use of the **`validate-acis`** tool is discussed later in this chapter.

As an additional level of security, the Directory Server examines and validates all ACIs stored in the data whenever a backend is brought online. If any malformed ACIs are found in the backend, then the server generates an administrative alert to notify administrators of the problem and places itself in lockdown mode. While in lockdown mode, the server only allows requests from users who have the **`lockdown-mode`** privilege. This action allows administrators to correct the malformed ACI while ensuring that no sensitive data is inadvertently exposed due to an access control instruction not being enforced. When the problem has been corrected, the administrator can use the **`leave-lockdown-mode`** tool or restart the server to allow it to resume normal operation.

Global ACIs

Global ACIs are a set of ACIs that can apply to entries anywhere in the directory (although they can also be scoped so that they only apply to a specific set of entries). They work in conjunction with access control rules stored in user data and provide a convenient way to define ACIs that span disparate portions of the DIT.

In the UnboundID Directory Server, global ACIs are defined within the server configuration, in the **`global-aci`** property of configuration object for the access control handler. They can be viewed and managed using configuration tools like **`dsconfig`** and the web administration console.

The global ACIs available by default in the UnboundID Directory Server include:

- Allow anyone (including unauthenticated users) to access key attributes of the root DSE, including **`namingContexts`**, **`subschemaSubentry`**, **`supportedAuthPasswordSchemes`**, **`supportedControl`**, **`supportedExtension`**, **`supportedFeatures`**, **`supportedLDAPVersion`**, **`supportedSASLMechanisms`**, **`vendorName`**, and **`vendorVersion`**.

- Allow anyone (including unauthenticated users) to access key attributes of the subschema subentry, including **`attributeTypes`**, **`dITContentRules`**, **`dITStructureRules`**, **`ldapSyntaxes`**, **`matchingRules`**, **`matchingRuleUse`**, **`nameForms`**, and **`objectClasses`**.

- Allow anyone (including unauthenticated users) to include the following controls in requests made to the server: authorization identity request, manage DSA IT, password policy, real attributes only, and virtual attributes only.
- Allow anyone (including unauthenticated users) to request the following extended operations: get symmetric key, password modify request, password policy state, StartTLS, and Who Am I?

Access Controls for Public or Private Backends

The UnboundID Directory Server classifies backends as either public or private, depending on their intended purpose. A *private backend* is one whose content is generated by the Directory Server itself (for example, the root DSE, monitor, and backup backends), is used in the operation of the server (for example, the configuration, schema, task, and trust store backends), or whose content is maintained by the server (for example, the LDAP changelog backend). A *public backend* is intended to hold user-defined content, such as user accounts, groups, application data, and device data.

The UnboundID Directory Server access control model also supports the distinction between public backends and private backends. Many private backends do not allow writes of any kind from clients, and some of the private backends that do allow writes only allow changes to a specific set of attributes. As a result, any access control instruction intended to permit or restrict access to information in private backends should be defined as global ACIs, rather than attempting to add those instructions to the data for that private backend.

General Format of the Access Control Rules

The general format for an access control instruction is the following syntax:

aci: (*target*) (*name*; *permissions*; *bind rule*)

where

- *target* specifies the keywords that indicate the set of entries and/or operations to which an access control rule applies.
- *name* specifies the name of the ACI in the form `version 3.0; aci "name of this aci"`
- *permission* specifies the keywords that indicate the type of operations to which an access control rule might apply.
- *bind rule* specifies the keywords that indicate whether an access control rule should apply to a given requester.

Summary of Access Control Keywords

This section provides an overview of the keywords supported for use in the UnboundID Directory Server access control implementation.

Target Keywords

The following keywords are supported for use in the target portion of ACIs to indicate the set of entries and/or operations to which an access control rule applies:

- **extop**. Specifies the OIDs for any extended operations to which the access control rule should apply.
- **target**. Specifies the set of entries (identified using LDAP URLs) to which the access control rule applies.
- **targattrfilters**. Identifies specific attribute values (based on filters) that may be added to or removed from entries to which the access control rule applies.
- **targetattr**. Specifies the set of attributes to which the access control rule should apply.
- **targetcontrol**. Specifies the OIDs for any request controls to which the access control rule should apply.
- **targetfilter**. Specifies one or more search filters that may be used to indicate the set of entries to which the access control should apply.
- **targetscope**. Specifies the scope of entries to which the access control rule should apply.

Permissions

The following keywords are supported for use in the permissions portion of ACIs to indicate the type of operations to which an access control rule might apply:

- **add**. Indicates that the access control should apply to add operations.
- **compare**. Indicates that the access control should apply to compare operations, as well as to search operations with a base-level scope that targets a single entry.
- **delete**. Indicates that the access control should apply to delete operations.
- **proxy**. Indicates that the access control rule should apply to operations that attempt to use an alternate authorization identity (for example, operations that include a proxied authorization request control, an intermediate client request control with an alternate authorization identity, or a client that has authenticated with a SASL mechanism that allows an alternate authorization identify to be specified).

- **read**. Indicates that the access control rule should apply to search result entries returned by the server.
- **search**. Indicates that the access control rule should apply to search operations with a non-base scope.
- **selfwrite**. Indicates that the access control rule should apply to operations in which a user attempts to add or remove his or her own DN to the values for an attribute (for example, whether users may add or remove themselves from groups).
- **write**. Indicates that the access control rule should apply to modify and modify DN operations.
- **all**. An aggregate permission that includes all other permissions except “proxy”. This is equivalent to providing a permission of “add, compare, delete, read, search, selfwrite, write”.

Bind Rule Keywords

The following keywords are supported for use in the bind rule portion of ACIs to indicate whether an access control rule should apply to a given requester:

- **authmethod**. Indicates that the requester’s authentication method should be taken into account when determining whether the access control rule should apply to an operation. The keyword’s syntax is as follows:

```
authmethod = "method"
where:
  method = none, simple, ssl, sasl {sasl_mechanism}
  Wildcards are not allowed in this expression.
```

- **dayofweek**. Indicates that the day of the week should be taken into account when determining whether the access control rule should apply to an operation. The keyword’s syntax is as follows:

```
dayofweek = "day"
where:
  day = sun, mon, tues, wed, thu, fri, sat
  Wildcards are not allowed in this expression. Multiple day of week values may be
  separated by commas.
```

- **dns**. Indicates that the requester’s DNS-resolvable host name should be taken into account when determining whether the access control rule should apply to an operation. The keyword’s syntax is as follows:

```
dns = "dns-host-name"
  Wildcards are allowed in this expression. Multiple DNS patterns may be separated
  by commas.
```

- **groupdn**. Indicates that the requester’s group membership should be taken into account when determining whether the access control rule should apply to any operation. The keyword’s syntax is as follows:

```
groupdn [ = || != ] "ldap:///groupdn [ || ldap:///groupdn ] ..."
  Wildcards are not allowed in this expression.
```

- **ip**. Indicates that the requester's IP address should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:

```
ip [ = || != ] ipAddressList
where:
  ipAddressList is one of the following representations:
  - A specific IPv4 address: 127.0.0.1
  - An IPv4 address with wildcards to specify a subnetwork: 127.0.0.*
  - An IPv4 address or subnetwork with subnetwork mask: 123.4.5.*+255.255.255.192
  - An IPv4 address range using CIDR notation: 123.4.5.0/24
  - An IPv6 address as defined by RFC 2373. Multiple ID address patterns may be
    separated by commas.
```

- **timeofday**. Indicates that the time of day should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:

```
timeofday [ = || != || > || >= || < ] time
where:
  time = 4-digit 24-hour time format (0 to 2359)

  Wildcards are not allowed in this expression.
```

- **userattr**. Indicates that the requester's relation to the value of the specified attribute should be taken into account when determining whether the access control rule should apply to an operation. A `bindType` value of `USERDN` indicates that the target attribute should have a value which matches the DN of the authenticated user. A `bindType` value of `GROUPDN` indicates that the target attribute should have a value which matches the DN of a group in which the authenticated user is a member. A `bindType` value of `LDAPURL` indicates that the target attribute should have a value which is an LDAP URL whose criteria matches the entry for the authenticated user. Any value other than `USERDN`, `GROUPDN`, or `LDAPURL` is expected to be present in the target attribute of the authenticated user's entry. The keyword's syntax is as follows:

```
userattr = "attrName# [ bindType || attrValue ]"
where:
  attrName = name of the attribute for matching
  bindType = USERDN, GROUPDN, LDAPURL
  attrValue = an attribute value

  Wildcards are not allowed in this expression.
```

- **userdn**. Indicates that the user's DN should be taken into account when determining whether the access control rule should apply to an operation. The keyword's syntax is as follows:

```
userdn [ = || != ] "ldap:///value [ || ldap:///value ] ..."

where value may be any of the following:

- The DN of the target user.
- A value of "anyone" to match any client, including unauthenticated clients.
- A value of "all" to match any authenticated client.
- A value of "parent" to match the client authenticated as the user defined in the
  immediate parent of the target entry.
- A value of "self" to match the client authenticated as the user defined in the
  target entry.
```

If the value provided is a DN, then that DN may include wildcard characters to define patterns. A single asterisk will match any content within the associated DN component, and two consecutive asterisks may be used to match zero or more DN components.

Target Attributes

The target attributes for an access control instruction are provided using the targetattr keyword. There are three general forms that it can take in the UnboundID Directory Server:

- **`(targetattr="*")`**. Indicates that the access control rule applies to all user attributes. Operational attributes will not automatically be included in this set.
- **`(targetattr="attr1||attr2||attr3||...||attrN")`**. Indicates that the access control rule applies only to the named set of attributes.
- **`(targetattr!="attr1||attr2||attr3||...||attrN")`**. Indicates that the access control rule applies to all user attributes except the named set of attributes. It will not apply to any operational attributes.

The targeted attributes can be classified as *user attributes* and *operational attributes*. User attributes define the actual data for that entry, while operational attributes provide additional metadata about the entry that can be used for informational purposes (for example, when the entry was created or last modified and by whom) or operational purposes (for example, specifying which password policy applies to the user, or overriding default constraints like size limit, time limit, or look-through limit for that user).

The UnboundID Directory Server distinguishes between these two types of attributes in its access control implementation. The Directory Server does not automatically grant any access at all to operational attributes. For example, a clause like:

```
(targetattr="*")
```

applies only to user attributes and not to operational attributes, which is almost always the intended effect. If there is a legitimate need for a user to be able to access an operational attribute, then it can be specifically included in the values of the targetattr clause, as follows:

```
(targetattr="ds-rlim-size-limit")
```

Or as follows:

```
targetattr="*||ds-rlim-size-limit")
```

You can use the "+" symbol to indicate that the rule should apply to all operational attributes, as follows:

```
(targetattr="+")
```

Or as follows:

```
(targetattr="*||+")
```

The implications of the Directory Server not distinguishing between user attributes and operational attributes can be outlined in an example. It can be easy to inadvertently create an access control instruction that grants far more capabilities to a user than originally intended. Consider the following example:

```
aci: (targetattr!="uid||employeeNumber")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)
```

This instruction is intended to allow a user to update any attribute in his or her own entry with the exception of **`uid`** and **`employeeNumber`**. This ACI is a very common type of rule and seems relatively harmless on the surface, but it has very serious consequences for a directory server that does not distinguish between user attributes and operational attributes. It allows users to update operational attributes in their own entries, and could be used for a number of malicious purposes, including:

- A user could alter password policy state attributes to become exempt from password policy restrictions.
- A user could alter resource limit attributes and bypass size limit, time limit, and look-through-limit constraints.
- A user could add access control rules to his or her own entry, which could allow them to make their entry completely invisible to all other users (including administrators granted full rights by access control rules, but excluding users with the **`bypass-acl`** privilege), allow them to edit any other attributes in their own entry (including those excluded by rules like **`uid`** and **`employeeNumber`** in the example above), or add, modify, or delete any entries below his or her own entry. Because the UnboundID Directory Server does not automatically include operational attributes in the target attribute list, these kinds of ACIs do not present a security risk for it.

Target Scope

The **`targetscope`** keyword can be used to restrict the scope of an access control rule. By default, ACIs use a subtree scope, which means that they are applied to the target entry (either as defined by the **`target`** clause of the ACI, or the entry in which the ACI is defined, if it does not include a **`target`**) and all entries below it. However, adding the **`targetscope`** element into an access control rule can restrict the set of entries to which it applies. The following **`targetscope`** keyword values are allowed:

- **base**. Indicates that the access control rule should apply only to the target entry and not to any of its subordinate.
- **onelevel**. Indicates that the access control rule should apply only to entries that are the immediate children of the target entry and not to the target entry itself.

- **subtree**. Indicates that the access control rule should apply to the target entry and all of its subordinates. This is the default behavior if no targetscope is specified.
- **subordinate**. Indicates that the access control rule should apply to all entries below the target entry but not the target entry itself.

Authentication Method

The **`authmethod`** keyword can be used to indicate that an access control rule should be applied based on the way in which the user authenticated to the server. The **`authmethod`** keyword allows the following values:

- **none**. Indicates that the rule should apply to clients that have not attempted to authenticate, that performed an anonymous bind (for example, using simple authentication with a zero-length DN and password), or whose last bind attempt was not successful.
- **simple**. Indicates that the rule should apply to clients that authenticated using simple authentication with a non-empty DN and password.
- **ssl**. Indicates that the rule should apply to clients that authenticated using a client certificate via the SASL EXTERNAL mechanism.
- **sasl <sasl_mechanism>**. Indicates that the rule should apply to clients that authenticated using any SASL mechanism. You can restrict the SASL authentication to a particular mechanism by including its name. For example, “sasl DIGEST-MD5” indicates that the rule should only apply to clients that authenticated using the DIGEST-MD5 SASL mechanism.

Use of Controls and Extended Operations

The UnboundID Directory Server adds new keywords that support access control to request controls and extended operations. The **`targetcontrol`** keyword can be used to indicate whether a given request control may be used. The **`extop`** keyword can be used to indicate whether a given extended operation can be used. For both keywords, the value is the object identifier for the associated control or extended operation. Multiple OIDs can be provided by separating them with the two pipe characters (optionally surrounded by spaces). For example:

- **`(targetcontrol="2.16.840.1.113730.3.4.2")`**
- **`(extop="1.3.6.1.4.1.4203.1.11.1 || 1.3.6.1.4.1.1466.20037")`**

Examples of Common Access Control Rules

This section provides a set of examples that demonstrate access controls that are commonly used in a directory environment. Note that to be able to alter access control definitions in the server, a user must have the **`modify-acl`** privilege as discussed later in this chapter.

Example 1

The following ACI can be used to grant any member of the "**cn=admins,ou=groups,dc=example,dc=com**" group full read and write access to all user attributes for any data in the server (if specified as a global ACI) or within the tree containing the ACI:

```
aci: (targetattr="*")
  (version 3.0; acl "Full access for administrators";
    allow (all) groupdn="ldap:///cn=admins,ou=groups,dc=example,dc=com";)
```

Example 2

The following ACI can be used to allow any authenticated user to see all user attributes except **userPassword**, in entries containing the **person** object class:

```
aci: (targetattr!="userPassword")
  (targetfilter="(objectClass=person)")
  (version 3.0; acl "Allow users to see other user entries";
    allow (read,search,compare) userdn="ldap:///all";)
```

Example 3

The following ACI can be used to allow an employee's manager to edit the value of the employee's **telephoneNumber** attribute. You can use the **userattr** keyword with a **bindType** of USERDN, which indicates that the target entry's manager attribute must have a value equal to the DN of the authenticated user:

```
aci: (targetattr="telephoneNumber")
  (version 3.0; acl "Allow managers to update telephone numbers of their direct reports";
    allow (write) userattr="manager#USERDN";)
```

Example 4

The following ACIs can be used to allow the user "**uid=proxy,dc=example,dc=com**" to use the proxied authorization v2 control to request that operations be performed under the authority of another user. The first ACI uses the **targetcontrol** element to indicate that the user is allowed to submit requests containing the proxied authorization v2 control (identified by OID), while the second allows that user to send requests that will be evaluated using an alternate authorization identity. The user will also be required to have the **proxied-auth** privilege as discussed later in this chapter:

```
aci: (targetattr="*")
  (targetcontrol="2.16.840.1.113730.3.4.18")
  (version 3.0; acl "Allow the proxy user to submit requests containing the proxied auth v2 control";
    allow (read) userdn="ldap:///uid=proxy,dc=example,dc=com";)

aci: (targetattr="*")
  (version 3.0;acl "Allow the proxy user to submit requests that will be evaluated
    using an alternate authorization identity";
    allow(proxy) userdn="ldap:///uid=proxy,dc=example,dc=com";)
```

Validating ACIs Before Migrating Data

Many directory servers allow for less restrictive application of their access control instructions, so that they accept invalid ACIs. For example, if Oracle DSEE encounters an access control rule that it cannot parse, then it will simply ignore it without any warning, and the server may not offer the intended access protection. Rather than unexpectedly exposing sensitive data, the UnboundID Directory Server rejects any ACIs that it cannot interpret, which ensures data access is properly limited as intended, but it can cause problems when migrating data with existing access control rules to an UnboundID Directory Server.

Validating ACIs from a File

To validate an access control instruction, the UnboundID Directory Server provides a **`validate-acis`** tool in the **`bin`** directory (UNIX or Linux systems) or **`bat`** directory (Windows systems) that identifies any ACI syntax problems before migrating data. The tool can examine access control rules contained in either an LDIF file or an LDAP directory and will write its result in LDIF with comments providing information about any problems that were identified. Each entry in the output will contain only a single ACI, so if an entry in the input contains multiple ACIs, then it may be present multiple times in the output, each time with a different ACI value. The entries contained in the output will contain only ACI values, and all other attributes will be ignored.

To Validate ACIs from a File

The **`validate-acis`** tool can process data contained in an LDIF file. It will ignore all attributes except **`aci`**, and will ignore all entries that do not contain the **`aci`** attribute, so any existing LDIF file that contains access control rules may be used.

1. Run the **`bin/validate-acis`** tool (UNIX or Linux systems) or **`bat\validate-acis`** (Windows systems) by specifying the input file and output file. If the output file already exists, the existing contents will be re-written. If no output file is specified, then the results will be written to standard output.

```
$ bin/validate-acis --ldifFile test-acis.ldif --outputFile validated-acis.ldif

# Processing complete
# Total entries examined:  1
# Entries found with ACIs:  1
# Total ACI values found:  3
# Malformed ACI values found:  0
# Other processing errors encountered:  0
```

2. Review the results by opening the output file. For example, the **`validated-acis.ldif`** file that was generated in the previous step reads as follows:

```
# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr!="userPassword")
  (version 3.0; acl "Allow anonymous read access for anyone";
    allow (read,search,compare) userdn="ldap:///anyone";)
```

```
# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

3. If the input file has any malformed ACIs, then the generated output file will show what was incorrectly entered. For example, remove the quotation marks around `userPassword` in the original `test-acis.ldif` file, and re-run the command. The following command uses the `--onlyReportErrors` option to write any error messages to the output file only if a malformed aci syntax is encountered.

```
$ bin/validate-acis --ldifFIle test-acis.ldif --outputFile validated-acis.ldif \
  --onlyReportErrors

# Processing complete
# Total entries examined:  1
# Entries found with ACIs:  1
# Total ACI values found:  3
# Malformed ACI values found:  0
# Other processing errors encountered:  0
```

The output file shows the following message:

```
# The following access control rule is malformed or contains an unsupported
# syntax:  The provided string '(targetattr!=userPassword)(version 3.0; acl
# "Allow anonymous read access for anyone"; allow (read,search,compare)
# userdn="ldap:///anyone";)' could not be parsed as a valid Access Control
# Instruction (ACI) because it failed general ACI syntax evaluation
dn: dc=example,dc=com
aci: (targetattr!=userPassword)
  (version 3.0; acl "Allow anonymous read access for anyone";
    allow (read,search,compare) userdn="ldap:///anyone";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Allow users to update their own entries";
    allow (write) userdn="ldap:///self";)

# The following access control rule is valid
dn: dc=example,dc=com
aci: (targetattr="*")
  (version 3.0; acl "Grant full access for the admin user";
    allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

To Validate ACIs in a Directory

The `validate-acis` tool also provides the ability to examine ACIs in data that exists in another directory server that you are planning to migrate to the UnboundID Directory Server in order to determine whether the UnboundID Server accepts those ACIs. To use it in this manner, provide arguments that specify the address and port of the target directory server, credentials to use to bind, and the base DN of the subtree containing the ACIs to validate.

```
$ bin/validate-acis --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN dc=example,dc=com

# Processing complete
# Total entries examined:  1
# Entries found with ACIs:  1
# Total ACI values found:  3
# Malformed ACI values found:  0
# Other processing errors encountered:  0
```

Migrating ACIs from Oracle DSEE to UnboundID Directory Server

This section describes the most important differences in access control evaluation between the Oracle Directory Server Enterprise Edition (DSEE) and the UnboundID Directory Server.

Support for Macro ACIs

DSEE provides support for macros ACIs, making it possible to define a single ACI that can be used to apply the same access restrictions to multiple branches in the same basic structure. Macros ACIs are infrequently used and can cause severe performance degradation, so support for macros ACIs is not included in the UnboundID Directory Server. However, you can achieve the same result by simply creating the same ACIs in each branch.

Support for the roleDN Bind Rule

DSEE roles are a proprietary, non-standard grouping mechanism that provide little value over standard grouping mechanisms. The UnboundID Directory Server does not support DSEE roles, and does not support the use of the `roleDN` ACI bind rule. However, the same behavior can be achieved by converting the DSEE roles to standard groups and using the `groupDN` ACI bind rule.

Targeting Operational Attributes

The DSEE access control model does not differentiate between user attributes and operational attributes. In DSEE, using `targetattr="*"` will automatically target both user and operational attributes. Using an exclusion list like `targetattr!="userPassword"` will automatically target all operational attributes in addition to all user attributes except `userPassword`. This behavior is responsible for several significant security holes in which users are unintentionally given access to operational attributes. In some cases, it allows users to do things like exempt themselves from password policy restrictions.

In the UnboundID Directory Server, operational attributes are treated differently from user attributes and operational attributes are never automatically included. As such, `targetattr="*"` will target all user attributes but no operational attributes, and `targetattr!="userPassword"` will target all users attributes except `userPassword`, but no operational attributes. Specific operational attributes can be targeted by including the names in the list, like `targetattr="creatorsName||modifiersName"`. All operational attributes can be targeted using the `"+"` character. So, `targetattr="+"` targets all operational attributes but no user attributes and `targetattr="*||+"` targets all user and operational attributes.

Specification of Global ACIs

Both DSEE and UnboundID Directory Server support global ACIs, which can be used to define ACIs that apply throughout the server. In servers with multiple naming contexts, this feature allows you to define a rule once as a global ACI, rather than needing to maintain an identical rule in each naming context.

In DSEE, global ACIs are created by modifying the root DSE entry to add values of the `"aci"` attribute. In the UnboundID Directory Server, global ACIs are managed with `dsconfig`, referenced in the `global-aci` property of the Access Control Handler.

Defining ACIs for Non-User Content

In DSEE, you can write to the configuration, monitor, changelog, and tasks backends to define ACIs. In the UnboundID Directory Server, access control for private backends, like configuration, monitor, schema, changelog, tasks, encryption settings, backups, and alerts, should be defined as global ACIs.

Limiting Access to Controls and Extended Operations

DSEE offers limited support for restricting access to controls and extended operations. To the extent that it is possible to control such access with ACIs, DSEE defines entries with a DN such as `"oid={oid},cn=features,cn=config"` where `{oid}` is the OID of the associated control or extended operation. For example, the following DSEE entry defines ACIs for the persistent search control:

```
"oid=2.16.840.1.113730.3.4.3,cn=features,cn=config"
```

In the UnboundID Directory Server, the `"targetcontrol"` keyword can be used to define ACIs that grant or deny access to controls. The `"extop"` keyword can be used to define ACIs that grant or deny access to extended operation requests.

Tolerance for Malformed ACI Values

In DSEE, if the server encounters a malformed access control rule, it simply ignores that rule without any warning. If this occurs, then the server will be running with less than the intended set of ACIs, which may prevent access to data that should have been allowed or, worse yet, may grant access to data that should have been restricted.

The UnboundID Directory Server is much more strict about the access control rules that it will accept. When performing an LDIF import, any entry containing a malformed or unsupported access control rule will be rejected. Similarly, any add or modify request that attempts to create an invalid ACI will be rejected. In the unlikely event that a malformed ACI does make it into the data, then the server immediately places itself in lockdown mode, in which the server terminates connections and rejects requests from users without the **`lockdown-mode`** privilege. Lockdown mode allows an administrator to correct the problem without risking exposure to user data.

Note	Consider running **`import-ldif`** with the **`"--rejectFile"`** option so that you can review any rejected ACIs.

About the Privilege Subsystem

In DSEE, only the root user is exempt from access control evaluation. While administrators can create ACIs that give "normal" users full access to any content, they can also create ACIs that would make some portion of the data inaccessible even to those users. In addition, some tasks can only be accomplished by the root user and you can not restrict the capabilities assigned to that root user.

The UnboundID Directory Server offers a privilege subsystem that makes it possible to control the capabilities available to various users. Non-root users can be granted limited access to certain administrative capabilities, and restrictions can be enforced on root users. In addition, certain particularly risky actions (such as the ability to interact with the server configuration, change another user's password, impersonate another user, or shutdown and restart the server) require that the requester have certain privileges in addition to sufficient access control rights to process the operation.

Identifying Unsupported ACIs

The UnboundID Directory Server provides a **`validate-acis`** tool that can be used to examine content in an LDIF file or data in another directory server (such as a DSEE instance) to determine whether the access control rules contained in that data are suitable for use in the UnboundID Directory Server instance. When migrating data from a DSEE deployment into an UnboundID Directory Server instance, the **`validate-acis`** tool should first be used to determine whether ACIs contained in the data are acceptable. If any problems are identified, then the data should be updated to correct or redefine the ACIs so that they are suitable for use in the UnboundID Directory Server.

For more information about using this tool, see "Validating ACIs Before Migrating Data" on page 301.

Working with Privileges

In addition to the access control implementation, the UnboundID Directory Server includes a privilege subsystem that can also be used to control what users are allowed to do. The privilege subsystem works in conjunction with the access control subsystem so that privileged operations are only allowed if they are allowed by the access control configuration and the user has all of the necessary privileges.

Privileges can be used to grant normal users the ability to perform certain tasks that, in most other directories, would only be allowed for the root user. In fact, the capabilities extended to root users in the UnboundID Directory Server are all granted through privileges, so you can create a normal user account with the ability to perform some or all of the same actions as root users.

Administrators can also remove privileges from root users so that they are unable to perform certain types of operations. Multiple root users can be defined in the server with different sets of privileges so that the capabilities that they have are restricted to only the tasks that they need to be able to perform.

Available Privileges

The following privileges are defined in the UnboundID Directory Server.

TABLE 9-1. Summary of Privileges

Privilege	Description
audit-data-security	This privilege is required to initiate a data security audit on the server, which is invoked by the **`audit-data-security`** tool.
backend-backup	This privilege is required to initiate an online backup through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
backend-restore	This privilege is required to initiate an online restore through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
bypass-acl	This privilege allows a user to bypass access control evaluation. For a user with this privilege, any access control determination made by the server immediately returns that the operation is allowed. Note, however, that this does not bypass privilege evaluation, so the user must have the appropriate set of additional privileges to be able to perform any privileged operation (for example, a user with the **`bypass-acl`** privilege but without the config-read privilege is not allowed to access the server configuration).
bypass-pw-policy	This privilege allows a user entry to bypass password policy evaluation. This privilege is intended for cases where external synchronization might require passwords that violate the password validation rules. The privilege is not evaluated for bind operations so that password policy evaluation will still occur when binding as a user with this privilege.
bypass-read-acl	This privilege allows the associated user to bypass access control checks performed by the server for bind, search, and compare operations. Access control evaluation may still be enforced for other types of operations.

TABLE 9-1. Summary of Privileges

Privilege	Description
config-read	This privilege is required for a user to access the server configuration. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to see.
config-write	This privilege is required for a user to alter the server configuration. The user is also required to have the `config-read` privilege. Access control evaluation is still performed and can be used to restrict the set of configuration objects that the user is allowed to alter.
disconnect-client	This privilege is required for a user to request that an existing client connection be terminated. The connection is terminated through the disconnect client task. The server's access control configuration must also allow the user to add the corresponding entry to the tasks backend.
jmx-notify	This privilege is required for a user to subscribe to JMX notifications generated by the directory server. The user is also required to have the `jmx-read` privilege.
jmx-read	This privilege is required for a user to access any information provided by the Directory Server via the Java Management Extensions (JMX).
jmx-write	This privilege is required for a user to update any information exposed by the Directory Server via the Java Management Extensions (JMX). The user is also required to have the `jmx-read` privilege. Note that currently all of the information exposed by the server over JMX is read-only.
ldif-export	This privilege is required to initiate an online LDIF export through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
ldif-import	This privilege is required to initiate an online LDIF import through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
lockdown-mode	This privilege allows the associated user to request that the server enter or leave lockdown mode, or to perform operations while the server is in lockdown mode.
modify-acl	This privilege is required for a user to add, modify, or remove access control rules defined in the server. The server's access control configuration must also allow the user to make the corresponding change to the `aci` operational attribute.
password-reset	This privilege is required for one user to be allowed to change another user's password. This privilege is not required for a user to be allowed to change his or her own password. The user must also have the access control instruction privilege to write the `userPassword` attribute to the target entry.
privilege-change	This privilege is required for a user to change the set of privileges assigned to a user, including the set of privileges, which are automatically granted to root users. The server's access control configuration must also allow the user to make the corresponding change to the `ds-privilege-name` operational attribute.
proxied-auth	This privilege is required for a user to request that an operation be performed with an alternate authorization identity. This privilege applies to operations that include the proxied authorization v1 or v2 control operations that include the intermediate client request control with a value set for the client identity field, or for SASL bind requests that can include an authorization identity different from the authentication identity.
server-restart	This privilege is required to initiate a server restart through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.
server-shutdown	This privilege is required to initiate a server shutdown through the tasks interface. The server's access control configuration must also allow the user to add the corresponding entry in the tasks backend.

TABLE 9-1. **Summary of Privileges**

Privilege	Description
stream-values	This privilege is required for a user to perform a stream values extended operation, which obtains all entry DNs and/or all values for one or more attributes for a specified portion of the DIT.
unindexed-search	This privilege is required for a user to be able to perform a search operation in which a reasonable set of candidate entries cannot be determined using the defined index and instead, a significant portion of the database needs to be traversed to identify matching entries. The server's access control configuration must also allow the user to request the search.
update-schema	This privilege is required for a user to modify the server schema. The server's access control configuration must allow the user to update the operational attributes that contain the schema elements.

Privileges Automatically Granted to Root Users

The special abilities that root users have are granted through privileges. Privileges can be assigned to root users in two ways:

- By default, root users may be granted a specified set of privileges. Note that it is possible to create root users which are not automatically granted these privileges by including the **`ds-cfg-inherit-default-root-privileges`** attribute with a value of **`FALSE`** in the entries for those root users.

- Individual root users can have additional privileges granted to them, and/or some automatically-granted privileges may be removed from that user.

The set of privileges that are automatically granted to root users is controlled by the **`default-root-privilege-name`** property of the **`Root DN`** configuration object. By default, this set of privileges includes:

- audit-date-security
- backend-backup
- backend-restore
- bypass-acl
- config-read
- config-write
- disconnect-client
- ldif-export
- ldif-import
- lockdown-mode
- modify-acl
- password-reset
- privilege-change
- server-restart
- server-shutdown
- stream-values
- unindexed-search
- update-schema

The privileges not granted to root users by default includes:

- bypass-read-acl
- jmx-read
- jmx-write
- jmx-notify
- proxied-auth
- bypass-pw-policy

The set of default root privileges can be altered to add or remove values as necessary. Doing so will require the **`config-read`**, **`config-write`**, and **`privilege-change`** privileges, as well as either the **`bypass-acl`** privilege or sufficient permission granted by the access control configuration to make the change to the server's configuration.

Assigning Privileges to Normal Users and Individual Root Users

Privileges can be granted to normal users on an individual basis. This can be accomplished by adding the **`ds-privilege-name`** operational attribute to that user's entry with the names of the desired privileges. For example, the following change will grant the **`proxied-auth`** privilege to the **`uid=proxy,dc=example,dc=com`** account:

```
dn: uid=proxy,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
```

The user making this change will be required to have the **`privilege-change`** privilege, and the server's access control configuration must also allow the requester to write to the **`ds-privilege-name`** attribute in the target user's entry.

This same method can be used to grant privileges to root users that they would not otherwise have through the set of default root privileges. You can also remove default root privileges from root users by prefixing the name of the privilege to remove with a minus sign. For example, the following change grants a root user the **`jmx-read`** privilege in addition to the set of default root privileges, and removes the **`server-restart`** and **`server-shutdown`** privileges:

```
dn: cn=Sync Root User,cn=Root DNs,cn=config
changetype: modify
add: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: -server-restart
ds-privilege-name: -server-shutdown
```

Note that because root user entries exist in the configuration, this update requires the **`config-read`** and **`config-write`** privileges in addition to the **`privilege-change`** privilege.

Disabling Privileges

Although the privilege subsystem in the UnboundID Directory Server is a very powerful feature, it might break some applications if they expect to perform some operation that requires a privilege that they do not have. In the vast majority of these cases, you can work around the problem by simply assigning the necessary privilege manually to the account used by that application. However, if this workaround is not sufficient, or if you need to remove a particular privilege (for example, to allow anyone to access information via JMX without requiring the **`jmx-read`** privilege), then privileges can be disabled on an individual basis.

The set of disabled privileges is controlled by the **`disabled-privilege`** property in the global configuration object. By default, no privileges are disabled. If a privilege is disabled, then the server behaves as if all users have that privilege.

10 Managing the Schema

Overview

This chapter presents a basic summary of the supported schema components on the UnboundID Directory Server and procedures to extend the schema with new element definitions. The chapter presents the following topics:

About the Schema

A schema is the set of directory server rules that define the structures, contents, and constraints of a Directory Information Tree (DIT). The schema guarantees that any new data entries or modifications meet and conform to these predetermined set of definitions. It also reduces redundant data definitions and provides a uniform method for clients or applications to access its directory objects.

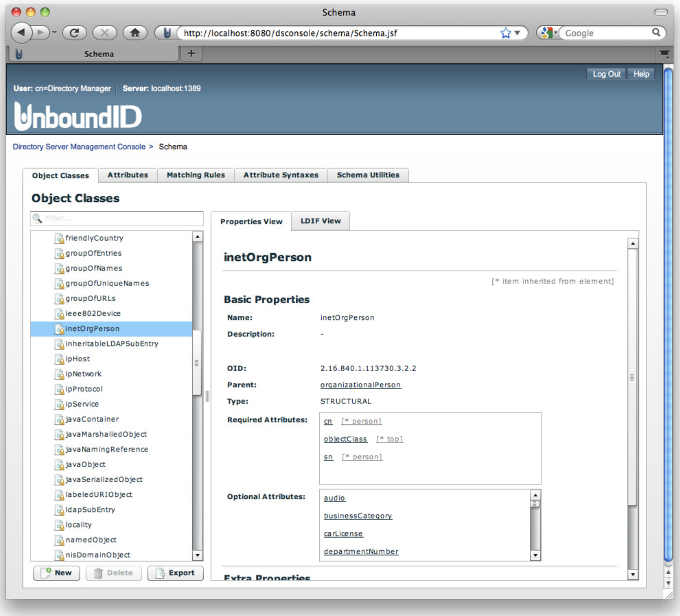
The UnboundID Directory Server ships with a default set of read-only schema files that define the core properties for the Directory Server. The Directory Management Console provides a

Schema Editor that administrators can use to view existing schema definitions and add new custom schema elements to their DIT. Any attempt to alter a schema element defined in a read-only file or add a new schema element to a read-only file will result in an "Unwilling to Perform" result. All custom schema definitions are stored in user-defined files, for example, `99-user.ldif`, which are marked as read-only through the server configuration.

About the Schema Editor

The UnboundID Directory Management Console provides a user-friendly graphical editor with tabs to manage any existing schema component related to your DIT: object classes, attributes, matching rules, attribute syntaxes, and schema utilities. The Object Classes and Attribute tabs allow you to view your existing definitions as well as add, modify, or remove custom schema elements. Read-only schema elements, such as those in the Standard folder, appear with a tiny "lock" symbol next to them. Read-write schema elements, such as those in the Custom folder, appear without a "lock" symbol.

The Matching Rules and Attribute Syntaxes tabs are read-only and provide a comprehensive listing of all of the elements necessary to define new schema elements. The Schema Utilities tab provides a schema validator that allows you to load a schema file or perform a cut-and-paste operation on the schema definition to verify that it meets the proper schema and ASN.1 formatting rules. The Utilities tab also supports schema file imports by first checking for proper syntax compliance and generating any error message if the definitions do not meet specification.



inetOrgPerson

[* item inherited from element]

Basic Properties

Name: inetOrgPerson
Description: -
OID: 2.16.840.1.113730.3.2.2
Parent: organizationalPerson
Type: STRUCTURAL
Required Attributes: cn [* person] objectClass [* top] sn [* person]
Optional Attributes: audio businessCategory carLicense departmentNumber

The Schema Editor provides two views for each definition: Properties View and LDIF View. The Properties View breaks down the schema definition by its properties and shows any inheritance relationships among the attributes. The LDIF View shows the equivalent schema definition in ASN.1 format, which includes the proper text spacing required for each schema element. Buttons to create, delete, or export any schema definitions are available under the left pane. Editable schema definitions will have an Edit button in the top right corner of a specific schema element window.

To use the Directory Management Console, you must have the console installed on your system. See “Installing the Directory Management Console” on page 52. To use the schema editor, you must have the proper privileges to view and modify the schema. See “Managing Root User Accounts” on page 102 for more information on privileges.

Default Directory Server Schema Files

The UnboundID Directory Server stores its schema as a set of LDIF files for a directory server instance in the **`<server-root>/config/schema`** directory. The directory server reads the schema files in alphanumeric order at startup, so that the **`00-core.ldif`** file is read first, then **`01-pwpolicy.ldif`**, and then the rest of the files. Custom schema files should be named so that they are loaded in last. For example, custom schema elements could be saved in a file labelled **`99-user.ldif`**, which loads after the default schema files are read at startup.

The Directory Server then uses the schema definitions to determine any violations that may occur during add, modify, or import requests. Clients and directory applications check the schema (i.e., matching rule definitions) to determine the assertion value algorithm used in comparison or search operations.

The default set of schema files are present at installation and **should not be modified**. Modifying the default schema files could result in an inoperable server.

The schema files have the following descriptions:

TABLE 10-1. Default Schema Files

Schema Files	Description
00-core.ldif	Governs the Directory Server's core functions.
01-pwpolicy.ldif	Governs password policies.
02-config.ldif	Governs the Directory Server's configuration.
03-changelog.ldif	Governs the Directory Server's change log.
03-rfc2713.ldif	Governs Java objects.
03-rfc2714.ldif	Governs Common Object Request Broker Architecture (CORBA) object references.
03-rfc2739.ldif	Governs calendar attributes for vCard.
03-rfc2926.ldif	Governs Server Location Protocol (SLP) mappings to and from LDAP schemas.
03-rfc2985.ldif	Governs PKCS #9 public-key cryptography.
03-rfc3112.ldif	Governs LDAP authentication passwords.
03-rfc3712.ldif	Governs printer services.

TABLE 10-1. Default Schema Files

Schema Files	Description
03-uddiv3.ldif	Governs web services registries of SOA components.
04-rfc2307bis.ldif	Governs mapping entities from TCP/IP and UNIX into X.500 entries.

Extending the Directory Server Schema

The UnboundID Directory Server stores its schema as LDIF files in the **`<server-root>/config/schema`** directory. At startup, the Directory Server reads the schema files once in alphanumeric order starting with **`00-core.ldif`** and ending with any custom schema definition files, such as **`99-user.ldif`** if present.

You can extend the schema to include additional customizations necessary for your directory data using one of the following methods:

- **Using the Schema Editor**. This method is the easiest and quickest way to set up a schema definition and have it validated for the correct ASN.1 formatting. The Editor lets you define your schema properties, load your custom file, or perform a cut-and-paste operation on a new schema element. If any errors exist in the file, the Schema Editor generates an error message if the schema definitions do not pass compliance.

- **Using a Custom Schema File**. You can create a custom schema file with your new definitions using a text editor, save it as **`99-user.ldif`**, and then import the file using the Schema Editor or the **`ldapmodify`** tool. You must name the custom LDIF file with a high two-digit number prefix, so that the Directory Server will read the file AFTER the core schema files are read at startup. For example, you can name the file, **`99-myschema.ldif`**, etc. See the next section, General Tips on Extending the Schema, to see the requirements for naming each file.

- **Using the Command Line**. If you have a small number of additions, you can extend the schema over LDAP and from the command line using the **`ldapmodify`** tool. The Directory Server writes the new schema changes to a file **`99-user.ldif`** in the **`<server-root>/config/schema`** directory. However, this method can be cumbersome as schema definitions require strict adherence to text spacing and white space characters.

General Tips on Extending the Schema

You should consider the following points when extending the schema:

- Never modify the default schema files as doing so could damage the Directory Server's processing capabilities.

- Define all attributes first before they can be used in an object class. If you are using the Schema Editor to add new schema elements, then you can use the Quick Add Attributes option when defining new objectclasses.

- Define the parent object class first before creating object classes that inherit from the parent.

- The schema file naming syntax requires that custom schema files must begin with exactly two digits followed by a non-digit character, followed by a zero or more characters and ending with ".ldif". Note that the two digits do not need to be followed by a dash ("-"). Any files that do not meet this criteria will be ignored and either a **NOTICE** or **SEVERE_WARNING** message will be logged.

  Any file in the **<server-root>/config/schema** directory with a name that starts with "." or with a name that ends with a tilde ('~'), ".swp", or ".tmp" will generate a **NOTICE** message indicating that temporary files will be ignored. Any other file that does not meet the naming criteria will generate a **SEVERE_WARNING** message indicating that it will be ignored.

- Define custom attributes and object classes in one file. Typically, this file will be the **99-user.ldif**. You can specify a different file name to which the Directory Server writes using the **X-SCHEMA-FILE** element and the file name in the definition. For example:

```
add: attributeTypes
attributeTypes: ( 1.3.6.1.4.1.32473.3.1.9.1
  NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'Directory Server Example'
  X-SCHEMA-FILE '99-custom.ldif' )
```

- Pay special attention to the white space characters in the schema definitions, where **WSP** means zero or more space characters, and **SP** means one or more space characters. The LDIF specification states that LDIF parsers should ignore exactly one space at the beginning of each continuation line, since continuation lines must begin with a space character. Thus, if you define a new schema definition with each keyword on a separate continuation line, you should add two spaces before an element keyword to be safe. For example, the following attribute definition has two spaces before the keywords: **NAME**, **SUP**, and **X-ORIGIN**.

```
attributeTypes: ( 2.5.4.32
  NAME 'owner'
  SUP distinguishedName
  X-ORIGIN 'RFC 4519' )
```

- In a replicated topology, any new schema additions will be replicated to other replication servers to their respective Schema backend. The additions will be written to the file specified by the **X-SCHEMA-FILE** extension or written to **99-user.ldif** if no file is specified.

Managing Attribute Types

An *attribute type* determines the important properties related to an attribute, such as specifying the matching and syntax rules used in value comparisons. An *attribute description* consists of an attribute type and a set of zero or more options. Options are short, case-insensitive text strings that differentiate between attribute descriptions. For example, the LDAPv3 specification defines only one type of option, the tagging option, which can be used to tag language options, such as **`cn;lang-de;lang-sp`** or binary data, such as **`userCertificate;binary`**. You can also extend the schema by adding your own attribute definitions.

Attributes have the following properties:

- Attributes can be user attributes that hold information for directory applications, or operational attributes that are used for administrative or server-related purposes. You can specify the purpose of the attribute by the **`USAGE`** element.

- Attributes are multi-valued by default. Multi-valued means that attributes can appear more than once in an object class instance, unless the **`SINGLE-VALUE`** element is present in the schema, which requires that it only appear once.

- Attributes can inherit properties from a parent attribute as long as they both have the same **`USAGE`**, and the child attribute has the same **`SYNTAX`** or its **`SYNTAX`** allows values which are a subset of the values allowed by the **`SYNTAX`** of the parent attribute. For example, the surname (**`sn`**) attribute is a child of the **`name`** attribute.

Attribute Type Definitions

New attribute types do not require server code extensions if the provided matching rules and attribute syntaxes are used in the definitions. Administrators can create new attributes using the Schema Editor, which stores the definition in a file in the **`<server-root>/config/schema`** directory. See “Extending the Directory Server Schema” on page 314 for more information.

The formal specification for attribute types is provided in RFC 4512, section 4.1.2 as follows:

```
AttributeTypeDescription = "(" wsp; Left parentheses followed by a white space
  numericoid                    ; Required numeric object identifier
  [ sp "NAME" sp qdescrs ]      ; Short name descriptor as alias for the OID
  [ sp "DESC" sp qdstring ]     ; Optional descriptive string
  [ sp "OBSOLETE" ]             ; Determines if the element is active
  [ sp "SUP" sp oid ]           ; Specifies the supertype
  [ sp "EQUALITY" sp oid ]      ; Specifies the equality matching rule
  [ sp "ORDERING" sp oid ]      ; Specifies ordering matching rule
  [ sp "SUBSTR" sp oid ]        ; Specifies substrings matching rule
  [ sp "SYNTAX" sp oidlen ]     ; Numeric attribute syntax with minimum upper bound length
                                ; expressed in {num}
  [ sp "SINGLE-VALUE" ]         ; Specifies if the attribute is single valued in the entry
  [ sp "COLLECTIVE" ]           ; Specifies if it is a collective attribute
  [ sp "NO-USER-MODIFICATION" ] ; Not modifiable by external clients
  [ sp "USAGE" sp usage ]       ; Application usage
  extensions wsp ")"            ; Extensions followed by a white space and ")"
usage = "userApplications" /    ; Stores user data
 "directoryOperation" /         ; Stores internal server data
 "distributedOperation" /       ; Stores operational data that must be synchronized across servers
 "dSAOperation"                 ; Stores operational data specific to a server and should not
                                ; be synchronized across servers
```

The following extensions are specific to the UnboundID Directory Server and are not defined in RFC 4512:

```
extensions = /
"X-ORIGIN" /                ; Specifies where the attribute type is defined
"X-SCHEMA-FILE" /           ; Specifies which schema file contains the definition
"X-APPROX" /                ; Specifies the approximate matching rule
"X-ALLOWED-VALUE" /         ; Explicitly specifies the set of allowed values
"X-VALUE-REGEX" /           ; Specifies the set of regular expressions to compare against
                            ;  attribute values to determine acceptance
"X-MIN-VALUE-LENGTH" /      ; Specifies the minimum character length for attribute values
"X-MAX-VALUE-LENGTH" /      ; Specifies the maximum character length for attribute values
"X-MIN-INT-VALUE" /         ; Specifies the minimum integer value for the attribute
"X-MAX-INT-VALUE" /         ; Specifies the maximum integer value for the attribute
"X-MIN-VALUE-COUNT" /       ; Specifies the minimum number of allowable values for the attribute
"X-MAX-VALUE-COUNT"         ; Specifies the maximum number of allowable values for the attribute
"X-READ-ONLY"               ; True or False. Specifies if the file that contains the schema
                            ; element is marked as read-only in the server configuration.
```

Viewing Attributes

The Schema Editor displays all of the attribute types on your directory server instance. It shows the basic properties that are required elements plus the extra properties that are allowed within the attribute definition. The **Basic Properties** section displays the standard elements in schema definition.

TABLE 10-2. **Basic Properties of Attributes**

Attributes	Description
Name	Specifies the globally unique name.
Description	Specifies an optional definition that describes the attribute and its contents. The analogous LDIF equivalent is "DESC".
OID	Specifies the object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Syntax	Specifies the attribute syntax used. For example, the `userPassword` attribute uses the User Password Syntax whereas the `authPassword` attribute uses the Authentication Password Syntax.
Parent	Specifies the schema definition's parent or supertype if any. The analogous LDIF equivalent is "SUP".
Multivalued	Specifies if the attribute can appear more than once in its containing object class.
Required By Class	Specifies any object classes that require the attribute.
Allowed By Class	Specifies any object classes that can optionally use the attribute.
Value Restrictions	Specifies any restriction on the value of the attribute.

The Extra Properties section provides additional auxiliary information associated with the attribute.

TABLE 10-3. Extra Properties of Attributes

Attributes	Description
Aliases	Specifies any shortform alias names, if any. In theory, you could have any number of shortform names as long as they are all unique. The analogous LDIF equivalent appears as the secondary element with the NAME element. For example, **`NAME ( 'sn' 'surname' )`**.
Origin	Specifies the origin of the schema definition. Typically, it could refer to a specific RFC or company.
Stored in File	Specifies the schema file that stores the definition in the **`<server-root>/config/schema`** folder.
Usage	Specifies the intended use of the attribute. Choices are the following: • userApplications • directoryOperation • distributedOperation • dSAOperation
User-Modifiable	Specifies if the attribute can be modified by an authorized user.
Obsolete	Specifies if the schema definition is obsolete or not.
Matching Rules	Specifies the associated matching rules for the attribute.

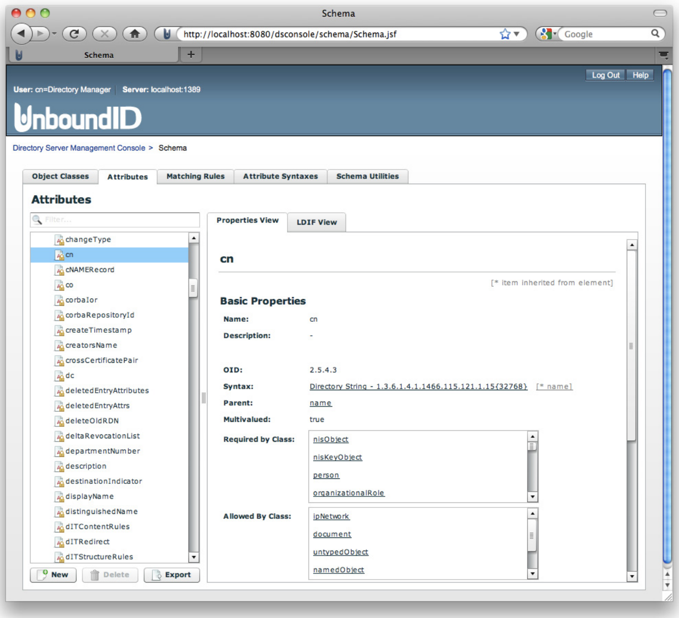
To View Attribute Types Using the Schema Editor

1. Start the Directory Management Console. Check that the directory server instance associated with the console is also running.

2. On the main menu, click Schema.

3. On the Schema Editor, click the Attributes tab, and then click the Standard folder to view the attribute definitions. The Custom folder holds user-defined schema elements if any.

4. Click a specific attribute to view its definition. In this example, click the **`sn`** attribute. In the Required by Class and Allowed By Class fields, you can click on the objectclass to view it.



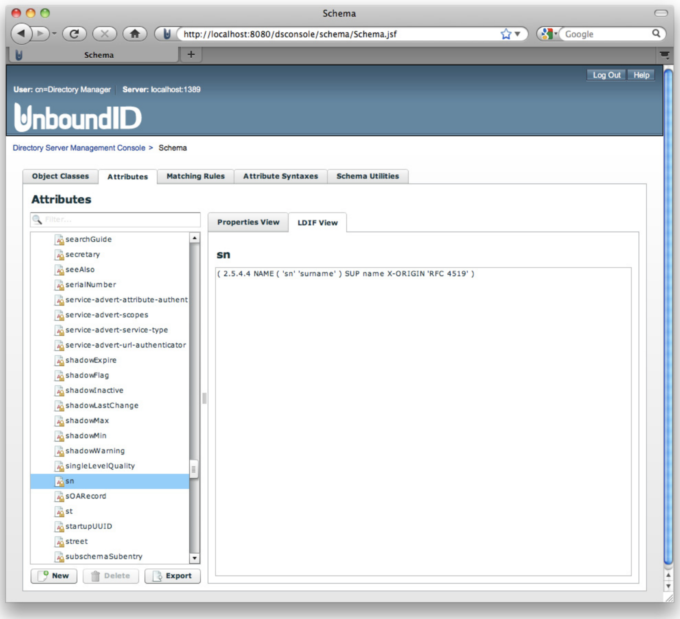
cn

[* item inherited from element]

Basic Properties

Name:	cn
Description:	-
OID:	2.5.4.3
Syntax:	Directory String - 1.3.6.1.4.1.1466.115.121.1.15{32768} [* name]
Parent:	name
Multivalued:	true
Required by Class:	nisObject nisKeyObject person organizationalRole
Allowed By Class:	ipNetwork document untypedObject namedObject

5. Click the LDIF View tab to see the equivalent attribute definition in ASN.1 format.



sn

```
( 2.5.4.4 NAME ( 'sn' 'surname' ) SUP name X-ORIGIN 'RFC 4519' )
```

To View Attribute Types over LDAP

Use **`ldapsearch`** to view a multi-valued operational attribute **`attributeTypes`**, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --baseDN cn=schema --searchScope base \
  "(objectclass=*)" attributeTypes
```

To View a Specific Attribute Type over LDAP

Use `ldapsearch` with the `--dontWrap` option and use the `grep` command to search for a specific attribute.

```
$ bin/ldapsearch --port 1389 --baseDN cn=schema \
  --searchScope base --dontWrap "(objectclass=*)" \
  attributeTypes | grep 'personalTitle'
```

Creating a New Attribute Using the Schema Editor

The Directory Management Console's Schema Editor provides the easiest and quickest way to add a new definition to your existing schema. The Schema Editor allows you to create a new attribute by its properties, load a schema file into the directory, perform a copy-and-paste operation on a definition, or define the new element from an existing attribute.

The following example shows how you can create a new attribute:

- **contractorStatus**. Defines an attribute that determines if the employee is a contractor (TRUE) or not (FALSE). The attribute is a single-valued element that uses the `booleanMatch` matching rule and the `Boolean` LDAP syntax. Because the Directory Server accepts alphanumeric OIDs, the Schema Editor uses `<Name>-OID`. The attribute is also user-modifiable and set for `userApplications`.

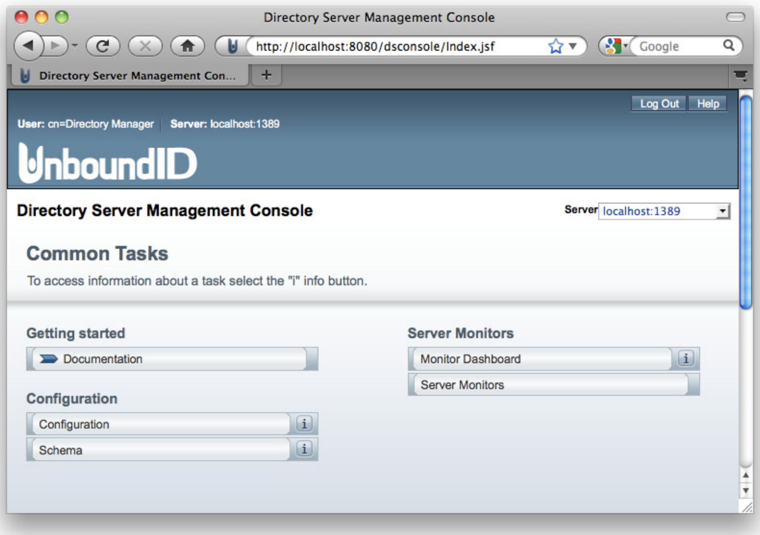
The example only adds an attribute to your directory's schema. You must add a new object class or modify an existing object class definition, so that it references this new attribute to expose it to client applications.

To extend the schema over LDAP, the user must have the `update-schema` privilege. Root users, such as `cn=Directory Manager`, have the privilege assigned by default.

To Create a New Attribute Using the Schema Editor

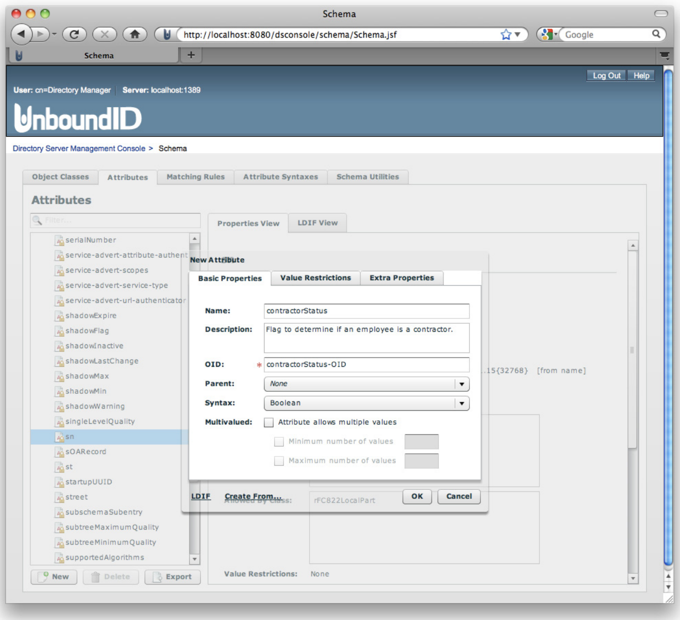
1. Start the Directory Management Console. Check the directory server instance associated with the console is also running.

2. On the main menu, click Schema under Configuration. And then, click the Attributes tab on the Schema Editor.


Directory Server Management Console
http://localhost:8080/dsconsole/index.jsf
Log Out
Help
User: cn=Directory Manager
Server: localhost:1389
UnboundID
Directory Server Management Console
Server localhost:1389
Common Tasks
To access information about a task select the "i" info button.
Getting started
Documentation
Configuration
Configuration
Schema
Server Monitors
Monitor Dashboard
Server Monitors

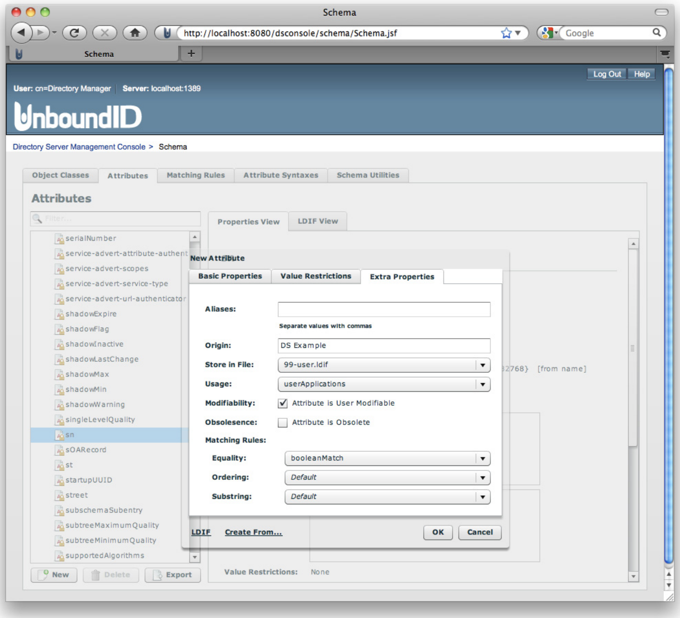
3. Click the New button below the list of attributes. The Properties dialog opens, where you can enter the individual properties.

4. Enter the properties for the new attribute. Because the Directory Server allows alphanumeric OIDs, the OID property automatically fills with the **`<name>-OID`** format when you enter a Name property.



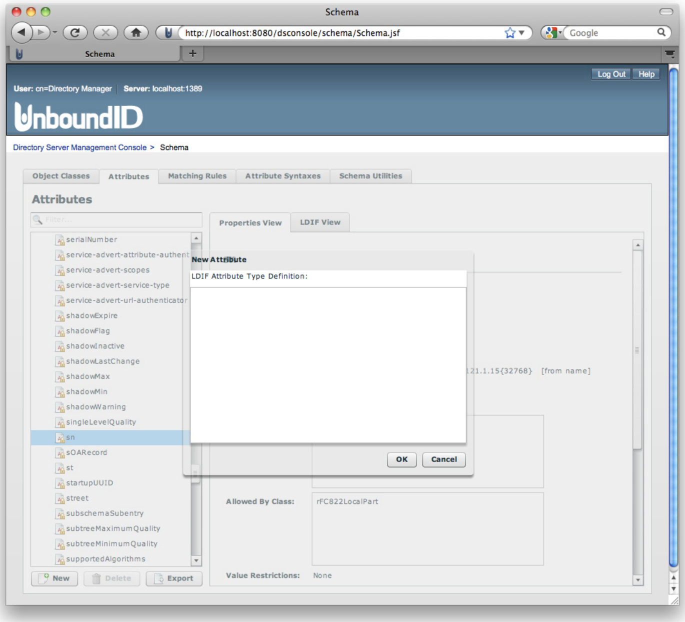
Name: contractorStatus
Description: Flag to determine if an employee is a contractor.
OID: * contractorStatus-OID
Parent: None
Syntax: Boolean
Multivalued: ☐ Attribute allows multiple values
☐ Minimum number of values
☐ Maximum number of values

5. Click Extra Properties to define additional information for the new attribute, and then click OK to create the attribute. In this example, enter the Origin (’**`DS Example`**’), select **`userApplications`**, and select **`Attribute is user modifiable`**.



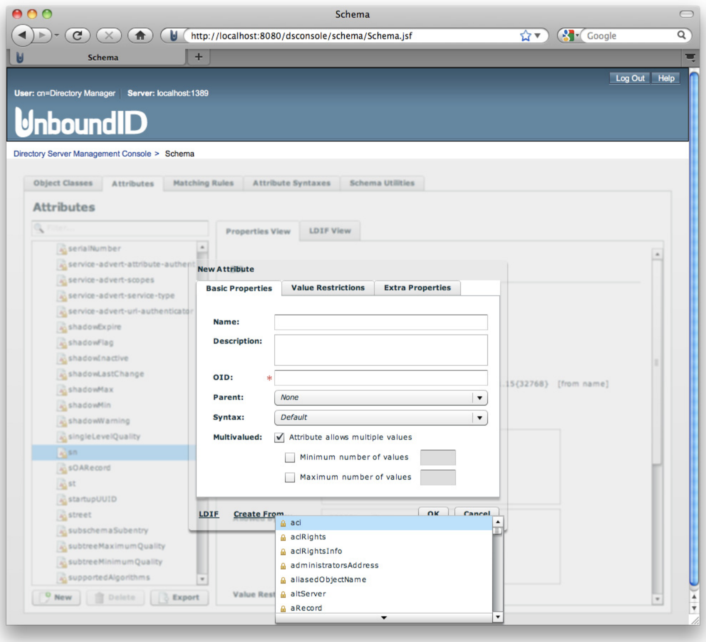
Aliases:
Separate values with commas
Origin: DS Example
Store in File: 99-user.ldif
Usage: userApplications
Modifiability: ☑ Attribute is User Modifiable
Obsolescence: ☐ Attribute is Obsolete
Matching Rules:
Equality: booleanMatch
Ordering: Default
Substring: Default

6. If you selected the LDIF option, enter the new attribute definition in the dialog window, and then click OK. You can type in the attribute definition or perform a cut-and-paste operation on an attribute. Pay special attention to the text spacings.



LDIF Attribute Type Definition:

7. If you selected Create From, you can create a new attribute definition from an existing attribute, which serves as a template. Modify any properties to fit your new attribute.



Creating a New Attribute over LDAP

The following section shows how you can add the schema element from the previous section over LDAP. You can create your own schema file or type the schema from the command line. In either case, you must pay special attention to text spacing and ASN.1 formatting.

To Add a New Attribute to the Schema over LDAP

1. Create an LDIF file with the new attribute definition using a text editor. Save the file as **myschema.ldif**.

```
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( contractorStatus-OID NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ALLOWED-VALUES ’Y’ ’N’ ’y’ ’n’
  X-ORIGIN 'UnboundID Directory Server Example' )
```

2. Use **ldapmodify** to add the attribute.

```
$ bin/ldapmodify --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --filename myschema.ldif
```

3. Verify the addition by displaying the attribute using **`ldapsearch`**.

```
$ bin/ldapsearch --port 1389 --baseDN cn=schema --searchScope base \
  --dontwrap "(objectclass=*)" attributeTypes | grep 'contractorStatus'
```

4. You can view the custom schema file at **`<server-root>/config/schema/99-user.ldif`**. You should see the following:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( contractorStatus-OID
  NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'UnboundID Directory Server Example' )
```

To Add Constraints to Attribute Types

The Directory Server provides attribute type extensions that constrain the values for the associated attribute using the **`DirectoryString`** attribute syntax. The following schema definition includes two **`attributeType`** definitions for **`myAttr1`** and **`myAttr2`**. The first definition constrains the values for the attribute **`myAttr1`** to ’foo’, ’bar’, ’baz’. The second definition constrains the minimum allowable length for **`myAttr2`** to 1 and the maximum allowable length to 5.

```
attributeTypes: (1.2.3.4
  NAME ’myAttr1’
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-ALLOWED-VALUES ( ’foo’ ’bar’ ’baz’ ))
attributeTypes: ( 1.2.3.5
  NAME ’myAttr2’
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-MIN-VALUE-LENGTH ’1’
  X-MAX-VALUE-LENGTH ’5’ )
```

Managing Object Classes

Object classes are sets of related information objects that form entries in a Directory Information Tree (DIT). The Directory Server uses the schema to define these entries, to specify the position of the entries in a DIT, and to control the operation of the server. You can also extend the schema by adding your own schema definitions.

Object classes have the following general properties:

- Object classes must have a globally unique name or identifier.

- Object classes specify the required and allowed attributes in an entry.
- Object classes can inherit the properties and the set of allowed attributes from its parent object classes, which may also be part of a hierarchical chain derived from the **`top`** abstract object class.
- Object classes that are defined in the UnboundID Directory Server can be searched using the **`objectClasses`** operational attribute. The Directory Server also has a special entry called the subschema subentry, which provides information about the available schema elements on the server.

Object Class Types

Based on RFC 4512, object classes can be a combination of three different types:

- **Abstract** object classes are used as the base object class, from which structural or auxiliary classes inherit its properties. This inheritance is a one-way relationship as abstract object classes cannot derive from structural or auxiliary classes. The most common abstract object class is **`top`**, which defines the highest level object class in a hierarchical chain of object classes.
- **Structural** object classes define the basic attributes in an entry and define where an entry can be placed in a DIT. All entries in a DIT belong to one structural object class. Structural object classes can inherit properties from other structural object classes and from abstract object classes to form a chain of inherited classes. For example, the **`inetOrgPerson`** structural object class inherits properties from the **`organizationalPerson`** structural class, which inherits from another object class, **`person`**.
- **Auxiliary** object classes are used together with structural object classes to define additional sets of attributes required in an entry. The auxiliary object class cannot form an entry alone but must be present with a structural object class. Auxiliary object classes cannot derive from structural object classes or vice-versa. They can inherit properties from other auxiliary classes and from abstract classes.

Object Class Definition

New object classes can be specified with existing schema components and do not require additional server code extensions for their implementation. Administrators can create new object classes using the Schema Editor, which manages schema in the **`<server-root>/config/schema`** directory. See “Extending the Directory Server Schema” on page 314 for more information.

The object class definition is defined in RFC 4512, section 4.1.1, as follows:

```
ObjectClassDescription = "(" wsp; Left parenthesis followed by a white space
 numericoid                ; Required numeric object identifier
 [ sp "NAME" sp qdescrs ]  ; Short name descriptor as alias for the OID
 [ sp "DESC" sp qdstring ] ; Optional descriptive string
 [ sp "OBSOLETE" sp whsp ] ; Determines if the element is inactive
 [ sp "SUP" sp oids ]      ; Specifies the direct superior object class
 [ sp kind ]               ; abstract,structural (default), auxiliary
 [ sp "MUST" sp oids ]     ; Required attribute types
 [ sp "MAY" sp oids ]      ; Allowed attribute types
extensions wsp ")"         ; Extensions, white space, right parenthesis
```

The following extensions are specific to the UnboundID Directory Server and are not defined in RFC 4512:

```
extensions = "X-ORIGIN" /  ; Specifies where the object class is defined
"X-SCHEMA-FILE"            ; Specifies which schema file contains the definition
"X-READ-ONLY"              ; True or False. Specifies if the file that
                           ;  contains the schema element is marked as
                           ;  read-only in the server configuration.
```

Note	Although RFC 4512 allows multiple superior object classes, the `UnboundID` Directory Server allows at most one superior object class, which is defined by the SUP element in the definition.

Viewing Object Classes

You can view the object classes on your directory server by using the Schema Editor on the Directory Management Console, over LDAP using the `ldapsearch` tool, or some third party tool.

The Schema Editor displays all of the object classes on your directory server instance. It shows the basic properties that are required elements and the extra properties that are allowed within the object class. The **Basic Properties** section displays the standard elements in the schema definition.

TABLE 10-4. **Basic Properties of Object Classes**

Basic Properties: Object Classes	Description
Name	Specifies the globally unique name.
Description	Specifies an optional definition that describes the object class and its contents. The analogous LDIF equivalent is "DESC".
OID	Specifies the object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Parent	Specifies the schema definition’s hierarchical parent or superior object class if any. An object class can have one parent. The analogous LDIF equivalent is "SUP".
Type	Specifies the type of schema definition: abstract, structural, or auxiliary. The analogous LDIF equivalent is "ABSTRACT", "STRUCTURAL", or "AUXILIARY".
Required Attributes	Specifies any required attributes with the object class. The Schema Editor also marks any inherited attributes from another object class. You can double-click an attribute value to take you to the Properties View for that particular attribute. The analogous LDIF equivalent is "MUST".
Optional Attributes	Specifies any optional attributes that could be used with the object class. The Schema Editor also marks any inherited attributes from another object class. You can double-click an attribute value to take you to the Properties View for that particular attribute. The analogous LDIF equivalent is "MAY".

The Extra Properties section provides additional auxiliary information associated with the object class.

TABLE 10-5. Extra Properties of Object Classes

Extra Properties	Description
Aliases	Specifies any shortform alias names, if any. In theory, you could have any number of shortform names as long as they are all unique. The analogous LDIF equivalent appears as the secondary element with the NAME element although most object classes do not have aliases.
Origin	Specifies the origin of the schema definition. Typically, it could refer to a specific RFC or company.
Obsolete	Specifies if the schema definition is obsolete or not.
Stored in File	Specifies the schema file that stores the definition in the **`<server-root>/config/schema`** folder.

To View Object Classes over LDAP

Use **`ldapsearch`** tool to view a multi-valued operational attribute, **`objectClasses`**, which publishes the object class definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --port 1389 --baseDN cn=schema --searchScope base \
  --dontWrap "(objectclass=*)" objectClasses

dn: cn=schema
objectClasses: ( 2.5.6.0 NAME 'top' ABSTRACT MUST objectClass X-ORIGIN 'RFC 4512' )
objectClasses: ( 2.5.6.1 NAME 'alias' SUP top STRUCTURAL MUST aliasedObjectName X-ORI-
GIN 'RFC 4512' )
objectClasses: ( 2.5.6.2 NAME 'country' SUP top STRUCTURAL MUST c MAY ( searchGuide $
description ) X-ORIGIN 'RFC 4519' )
...(more output)...
```

Managing an Object Class over LDAP

The following section shows how you can manage an object class schema element over LDAP by adding a new attribute element to an existing object class. You can create your own schema file or type the schema from the command line. In either case, you must pay special attention to text spacing and ASN.1 formatting. The following example procedure adds an attribute, **`contractorAddress`** to the custom schema file, then adds it to the **`contractor`** objectclass.

To Manage an Object Class over LDAP

1. Assume that you have defined the **`contractorAddress`** attribute, create an LDIF file called **`contractorAddress-attr.ldif`** with the following content:

```
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes:  ( contractor-OID NAME 'contractorAddress' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE USAGE userApplications X-ORIGIN 'user
defined' X-SCHEMA-FILE '98-custom-schema.ldif' ) X-ORIGINS 'user defined' X-SCHEMA-
FILE '98-custom-schema.ldif' )
```

2. Add the attribute using **`ldapmodify`**.

```
$ bin/ldapmodify --port 389 --bindDN "cn=Directory Manager" \
  --bindPassword password  --filename contractorAddress-attr.ldif
```

3. Next, create an LDIF file to modify the **`contractor`** objectclass to allow this attribute. When doing this, you are just re-adding the updated objectClass and the Directory Server will handle the proper replacement of the existing objectClass with the new one. Create a file called **`contractor-oc.ldif`**. Make sure that the lines are not wrapped, the **`objectClasses`** line should be one continuous line.

```
dn:cn=schema
changetype: modify
add: objectClasses
objectClasses: ( contractor-OID NAME 'contractor' DESC 'Contractor status informa-
tion SUP top AUXILIARY MAY ( contractorStatus $ contractorAgency $ contractorAddress
) X-ORIGIN 'Directory Server Example' X-SCHEMA-FILE '98-custom-schema.ldif' )
```

4. Update the **`objectClass`** using **`ldapmodify`** as follows:

```
$ bin/ldapmodify --port 389 --bindDN "cn=Directory Manager" \
  --bindPassword password --filename contractor-oc.ldif
```

5. These schema changes will be replicated to all servers in the replication topology. Verify the change by looking at the **`config/schema/98-custom-schema.ldif`** file on the other servers in the replication topology to ensure that the changes are present.

6. If you need to add an index for this attribute, you can do so by using the **`dsconfig`** command-line utility. You will need to do this on each server in your topology unless you have server configuration groups set up. See "Configuring a Server Group" on page 110 for more information.

```
$ bin/dsconfig create-local-db-index --backend-name userRoot \
  --index-name contractorAddress --set index-type:equality
```

7. Rebuild the index online. This will not affect other indexes or entries since there is no currently existing data for this attribute on any entry.

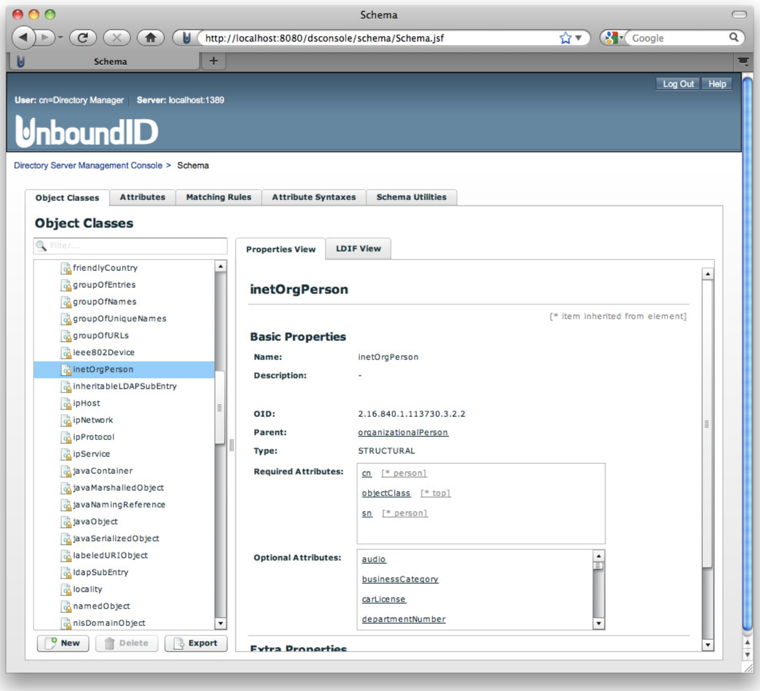
```
$ bin/rebuild-index --baseDN dc=example,dc=com --index contractorAddress \
  --port 389 --bindDN "cn=Directory Manager" --bindPassword password
```

To View Object Classes Using the Schema Editor

1. Start the Directory Server Management Console. Check that the Directory Server instance associated with the console is also running.

2. Under Object Classes, click the Standard folder, which opens the folder to display the object classes on the directory server. The Custom folder holds user-defined schema elements if any.

3. Click a specific object class to view its properties. In this example, click the **`inetOrgPerson`** objectclass. The Schema Editor opens in the Properties View and shows the Basic and Extra properties associated with the object class. If you click one of the attributes in the Required Attributes or Optional Attributes section, you will be taken to the Attributes tab, which displays the properties for that specific attribute. The "lock" symbol specifies that the objectclass cannot be modified or deleted unless that restriction is changed on the system.



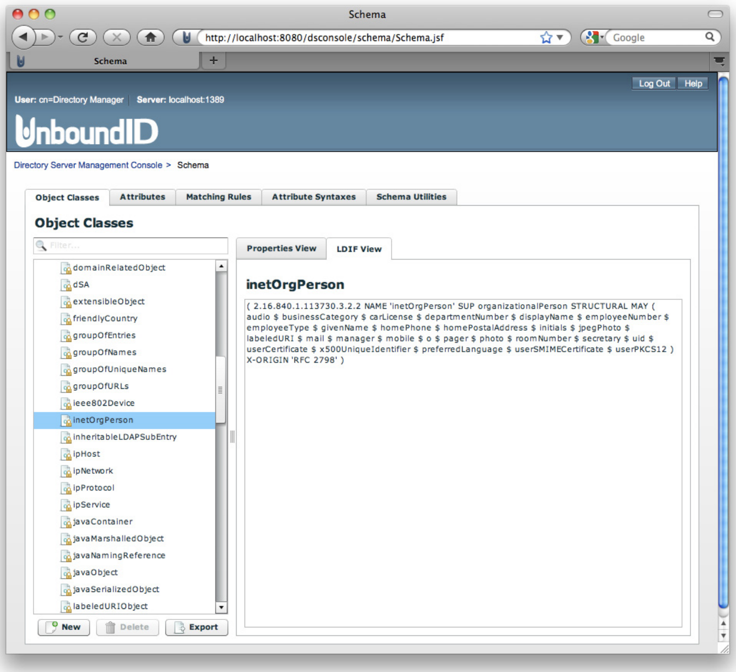
inetOrgPerson

[* item inherited from element]

Basic Properties

Name:	inetOrgPerson
Description:	-
OID:	2.16.840.1.113730.3.2.2
Parent:	organizationalPerson
Type:	STRUCTURAL
Required Attributes:	cn [* person] objectClass [* top] sn [* person]
Optional Attributes:	audio businessCategory carLicense departmentNumber

4. Click the LDIF View tab to view the equivalent objectclass definition in its ASN.1 format.



inetOrgPerson

```
( 2.16.840.1.113730.3.2.2 NAME 'inetOrgPerson' SUP organizationalPerson STRUCTURAL MAY (
audio $ businessCategory $ carLicense $ departmentNumber $ displayName $ employeeNumber $
employeeType $ givenName $ homePhone $ homePostalAddress $ initials $ jpegPhoto $
labeledURI $ mail $ manager $ mobile $ o $ pager $ photo $ roomNumber $ secretary $ uid $
userCertificate $ x500UniqueIdentifier $ preferredLanguage $ userSMIMECertificate $ userPKCS12 )
X-ORIGIN 'RFC 2798' )
```

Creating a New Object Class Using the Schema Editor

The procedures to create a new object class are similar to that of creating a new attribute with slight differences in the dialog windows. You should ensure that any attributes that are part of the new object class are defined prior to defining the object class.

The following example adds an auxiliary object class called **`contractor`** that allows two attributes, **`contractorStatus`** and **`contractorAgency`**. In a previous section, we defined the **`contractorStatus`** attribute. In this section, we will use the Schema Editor's **`QuickAdd`** function to create the **`contractorAgency`** attribute, while creating the **`contractor`** objectclass.

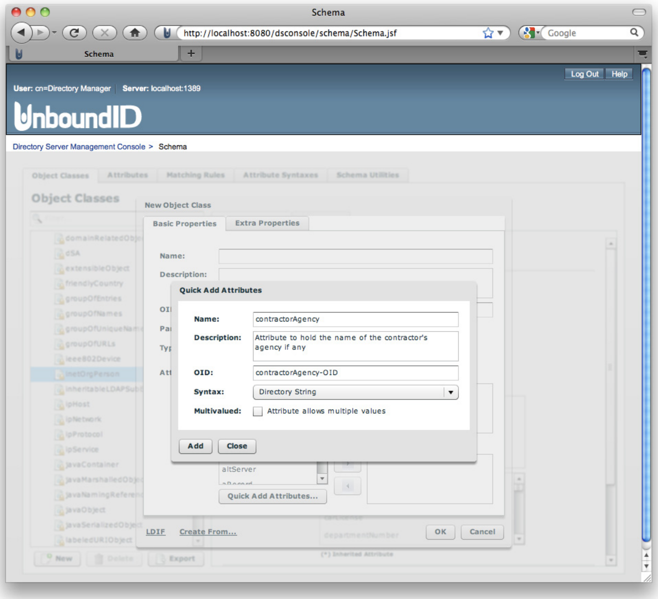
To Create a New Object Class Using the Schema Editor

1. Start the Directory Management Console. Check that the Directory Server instance associated with the console is running.

2. On the main menu, click Schema.

3. On the Schema Editor, click the Object Classes tab, and then click New located in the bottom left of the window.

4. On the New Object Class dialog, click Quick Add Attributes to add the **`contractorAgency`** attribute. The attribute uses the Directory String syntax and will be a single-valued element. When completed, click Close.



Quick Add Attributes

Name: contractorAgency

Description: Attribute to hold the name of the contractor's agency if any

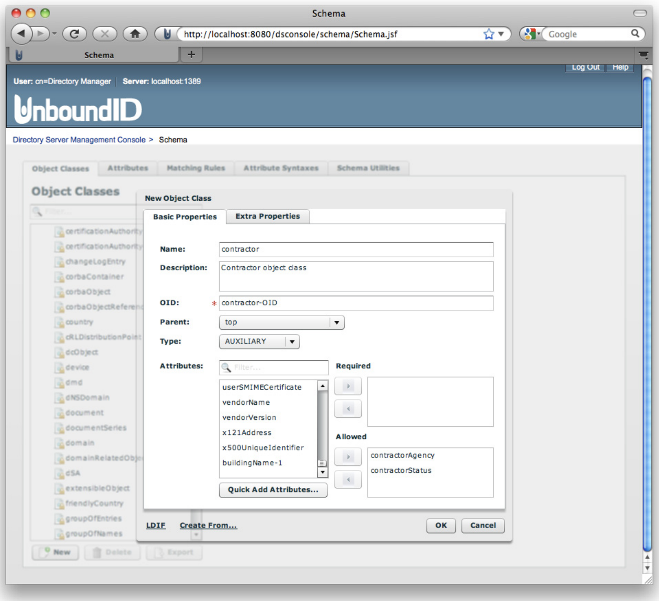
OID: contractorAgency-OID

Syntax: Directory String

Multivalued: ☐ Attribute allows multiple values

5. Enter the properties for the new objectclass. In the Attributes box, you can filter the types of attributes required for the new object class. Click the right arrow to move it into the

Required or the Allowed box. All custom attributes appear at the bottom of the list in the Attributes box.



User: cn=Directory Manager | Server: localhost:1389

Directory Server Management Console > Schema

New Object Class

Basic Properties | Extra Properties

Name: contractor

Description: Contractor object class

OID: * contractor-OID

Parent: top

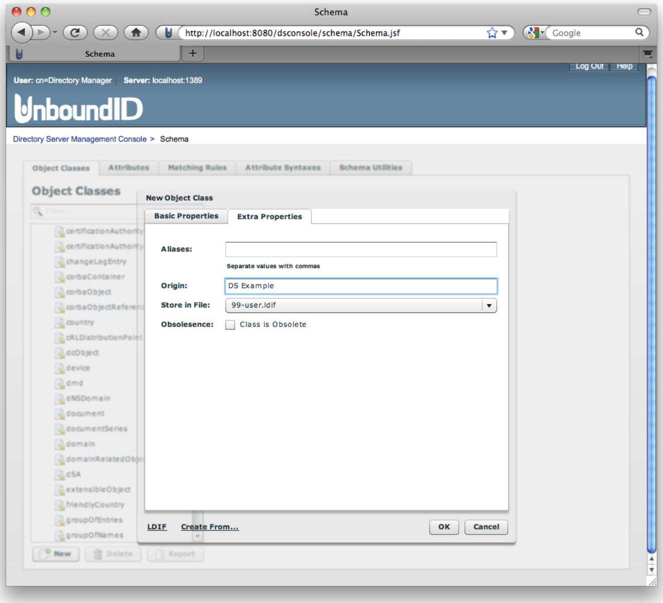
Type: AUXILIARY

Attributes: userSMIMECertificate, vendorName, vendorVersion, x121Address, x500UniqueIdentifier, buildingName-1

Required:

Allowed: contractorAgency, contractorStatus

6. Click Extra Properties tab to add the auxiliary information for the object class, and then click OK when finished.



User: cn=Directory Manager | Server: localhost:1389

Directory Server Management Console > Schema

New Object Class

Basic Properties | Extra Properties

Aliases:

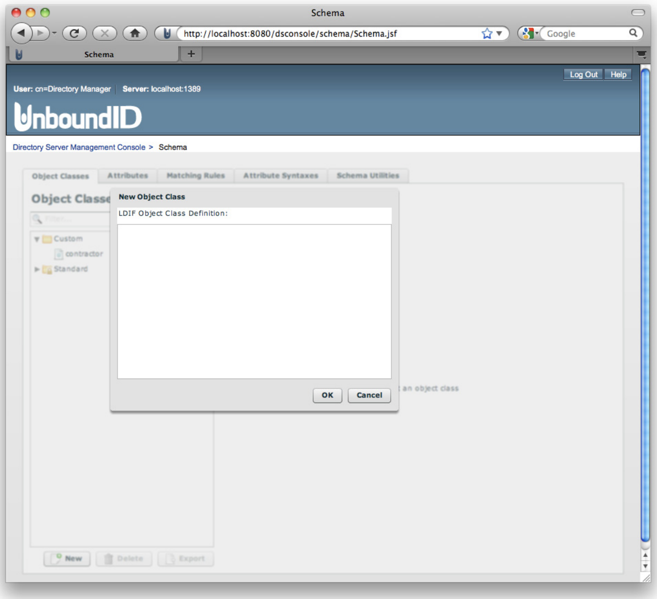
Separate values with commas

Origin: DS Example

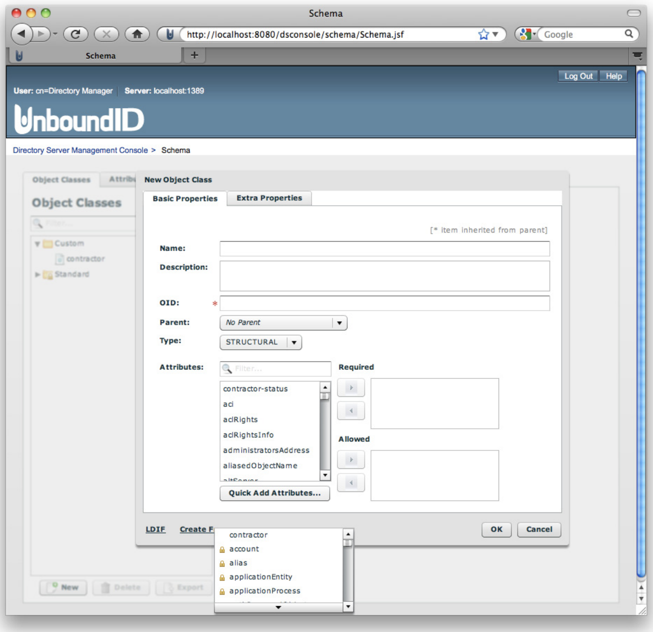
Store in File: 99-user.ldif

Obsolescence: ☐ Class is Obsolete

7. If you clicked the LDIF link on the New Object Class dialog, you can perform a cut-and-paste operation on the object class definition, and then click OK when done.



8. If you clicked the "Create From" link, you can create a new object class from an existing object class definition using it as a template.



Extending the Schema Using a Custom Schema File

You can add new attributes and object classes to your directory server schema by creating a custom schema file. You can import the file using the Schema Editor, over LDAP using the

ldapmodify tool, or from the command line. Make sure to define the attributes first, then define the object classes.

To Extend the Schema Using a Custom Schema File

1. Create an LDIF file with the new attribute extensions using a text editor.

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
attributeTypes: ( contractorStatus-OID NAME 'contractorStatus'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  SINGLE-VALUE USAGE userApplications
  X-ORIGIN 'UnboundID Directory Server Example' )
attributeTypes: ( contractorAgency-OID NAME 'contractorAgency'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{256}
  SINGLE-VALUE USAGE userApplications
  X-ORIGIN 'UnboundID Directory Server Example' )
```

2. In the same LDIF file, add a new object class definition after the attribute types. In this example, we create an auxiliary object class, **contractor**, that alone cannot be used as an entry. The object class will be used to add supplemental information to the **inetOrgPerson** structural object class. The attributes are all optional for the new object class.

```
objectClasses: ( contractor-OID NAME 'contractor'
  DESC 'Contractor status information'
  SUP top
  AUXILIARY
  MAY ( contractorStatus $ contractorAgency )
  X-ORIGIN 'UnboundID Directory Server Example' )
```

3. Save the file as **99-auxobjclass.ldif** and place it in the **<server-root>/config/schema** directory.

4. At this stage, the schema extensions are not loaded into the Directory Server yet. You have four options to load them:

   a. Create a task that loads the new extensions into the schema. We create a task labelled with the ID "**add-schema-99-auxobjclass**" and add it using **ldapmodify**. You do not need to restart the server when using this method.

   ```
   dn: ds-task-id=add-schema-99-auxobjclass,cn=Scheduled Tasks,cn=tasks
   objectClass: top
   objectClass: ds-task
   objectClass: ds-task-add-schema-file
   ds-task-id: add-schema-99-auxobjclass
   ds-task-class-name: com.unboundid.directory.server.tasks.AddSchemaFileTask
   ds-task-schema-file-name: 99-auxobjclass.ldif
   ```

   b. Import the schema file using the Schema Editor. You do not need to restart the server when using this method. See “Using the Schema Editor Utilities” on page 343.

c. Add the schema file using **ldapmodify**. You do not need to restart the server when using this method. See “Creating a New Attribute over LDAP” on page 323 to find a similar example.

d. Restart the Directory Server. The schema file is read at startup.

5. Use **ldapmodify** to make use of the new object class and one of its attributes in an existing entry.

```
$ bin/ldapmodify --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret
dn: uid=user.9,ou=People,dc=example,dc=com
changetype: modify
add: objectclass
objectclass: contractor
-
add: contractorStatus
contractorStatus: TRUE
```

6. Verify the addition by displaying the attribute using **ldapsearch**.

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --baseDN dc=example,dc=com \
  "(uid=user.9)" contractorStatus
dn: uid=user.9,ou=People,dc=example,dc=com
contractorStatus: TRUE
```

Managing Matching Rules

Matching rules determine how clients and servers compare attribute values during LDAP requests or operations. They are also used in evaluating search filter elements including distinguished names and attributes. Matching rules are defined for each attribute based on **EQUALITY** (e.g., two attributes are equal based on case, exact match, etc.), **SUBSTR** (e.g., assertion value is a substring of an attribute), and **ORDERING** (e.g., greater than or equal, less than or equal, etc.) properties.

Note	The UnboundID Directory Server supports an APPROXIMATE matching rule that compares similar attributes based on fuzzy logic. Thus, attributes that are similar or "sound-like" each other are matched. For example, "petersen" would match "peterson".

Matching Rule Definition

New matching rules require additional server code extensions to be implemented on the UnboundID Directory Server. If you need new matching rules, contact your authorized support provider for assistance.

The formal specification for matching rules is provided in RFC 4512, section 4.1.3 as follows:

```
MatchingRuleDescription = "(" wsp
  numericoid; Object identifier
  [ sp "NAME" sp qdescrs ] ; Short name descriptor
  [ sp "DESC" sp qdstring ]; Description
  [ sp "OBSOLETE" ]        ; Specifies if the rule is inactive
  sp "SYNTAX" sp numericoid; Assertion syntax
  extensions wsp ")"       ; Extensions followed by a white space and ")"
```

Default Matching Rules on the Directory Server

The UnboundID Directory Server provides a large set of matching rules, which support a variety of directory applications. The default matching rules available for the Directory Server are listed in the table below for each matching rule type: Equality, Substring, Ordering, and Approximate matches.

TABLE 10-6. **Matching Rules**

Matching Rule/OID	Attribute Syntax Name/OID	Description
uuidMatch/ 1.3.6.1.1.16.2	UUID/ 1.3.6.1.1.16.1	Compares an asserted UUID with a stored UUID attribute value for equality. RFC 4530.
uuidOrderingMatch/ 1.3.6.1.1.16.3	UUID/ 1.3.6.1.1.16.1	Compares the collation order of an asserted UUID with a stored UUID attribute value for ordering RFC 4530.
caseExactIA5Match/ 1.3.6.1.4.1.1466.109.114.1	IA5 String/ 1.3.6.1.4.1.1466.115.121.1.26	Compares an asserted value with an attribute value of International Alphabet 5 syntax. RFC 4517.
caseIgnoreIA5Match/ 1.3.6.1.4.1.1466.109.114.2	IA5 String/ 1.3.6.1.4.1.1466.115.121.1.26	Compares an asserted value with an attribute value of International Alphabet 5 syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreIA5SubstringsMatch/ 1.3.6.1.4.1.1466.109.114.3	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of IA5 string syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
authPasswordExactMatch/ 1.3.6.1.4.1.4203.1.2.2	Authentication Password Syntax/ 1.3.6.1.4.1.4203.1.1.2	Authentication password exact matching rule.
authPasswordMatch/ 1.3.6.1.4.1.4203.1.2.3	Authentication Password Syntax/ 1.3.6.1.4.1.4203.1.1.2	Authentication password matching rule.
ds-mr-double-metaphone-approx/ 1.3.6.1.4.1.30221.1.4.1	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Syntax based on the phonetic Double Metaphone algorithm for approximate matching.
ds-mr-user-password-exact/ 1.3.6.1.4.1.30221.1.4.2	User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1	User password exact matching rule.
ds-mr-user-password-equality/ 1.3.6.1.4.1.30221.1.4.3	User Password Syntax/ 1.3.6.1.4.1.30221.1.3.1	User password equality matching rule.
historicalCsnOrderingMatch/ 1.3.6.1.4.1.30221.1.4.4	1.3.6.1.4.1.30221.1.3.5	Compares the collation order of a historical change sequence number with a historical CSN attribute value.
caseExactIA5SubstringsMatch/ 1.3.6.1.4.1.30221.1.4.902	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of IA5 string syntax. RFC 4517.
compactTimestampMatch/ 1.3.6.1.4.1.30221.2.4.1	Compact Timestamp/ 1.3.6.1.4.1.30221.2.3.1	Compact Timestamp matching rule.
compactTimestampOrderingMatch/ 1.3.6.1.4.1.30221.2.4.2	Compact Timestamp/ 1.3.6.1.4.1.30221.2.3.1	Compares the collation order of a compact timestamp number with an attribute value of Compact Timestamp syntax.
objectIdentifierMatch/ 2.5.13.0	OID/ 1.3.6.1.4.1.1466.115.121.1.38	Compares an asserted value with an attribute value of OID syntax. RFC 4517.

TABLE 10-6. Matching Rules

Matching Rule/OID	Attribute Syntax Name/OID	Description
distinguishedNameMatch/ 2.5.13.1	DN/ 1.3.6.1.4.1.1466.115.121.1.12	Compares an asserted value with an attribute value of DN syntax. Spaces around commas and semicolons are ignored. Spaces around plus and equal signs around RDN components are ignored. RFC 4517.
caseIgnoreMatch/ 2.5.13.2	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with an attribute value. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreOrderingMatch/ 2.5.13.3	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares the collation order of the asserted string with an attribute value of Directory String syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
caseIgnoreSubstringsMatch/ 2.5.13.4	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring value with an attribute value of Directory String syntax. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517
caseExactMatch/ 2.5.13.5	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with an attribute value of Directory String syntax. RFC 4517.
caseExactOrderingMatch/ 2.5.13.6	Directory String 1.3.6.1.4.1.1466.115.121.1.15	Compares the collation order of the asserted string with an attribute value of Directory String syntax. RFC 3698
caseExactSubstringsMatch/ 2.5.13.7	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of Directory String syntax. RFC 3698
numericStringMatch/ 2.5.13.8	Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36	Compares an asserted value with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
numericStringOrderingMatch/ 2.5.13.9	Numeric String/ 1.3.6.1.4.1.1466.115.121.1.36	Compares the collation order of the asserted string with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
numericStringSubstringsMatch/ 2.5.13.10	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of Numeric String syntax. Spaces are ignored when performing these comparisons. RFC 4517.
caseIgnoreListMatch/ 2.5.13.11	Postal Address/ 1.3.6.1.4.1.1466.115.121.1.41	Compares an asserted value with an attribute value which is a sequence of Directory Strings. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 4517.
caseIgnoreListSubstringsMatch/ 2.5.13.12	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares the asserted substring with an attribute value, which is a sequence of Directory Strings. Case, leading, and trailing spaces are ignored. Multiple spaces are treated as a single space. RFC 3698.
booleanMatch/ 2.5.13.13	Boolean/ 1.3.6.1.4.1.1466.115.121.1.7	Compares an asserted boolean value with an attribute value of BOOLEAN syntax. Returns true if the values are both TRUE or both FALSE. RFC 3698.
integerMatch/ 2.5.13.14	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Compares an asserted value with an attribute value of INTEGER syntax. RFC 4517.
integerOrderingMatch/ 2.5.13.15	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Compares the collation order of the asserted integer with an attribute value of Integer syntax. Returns true if the attribute value is less than the asserted value. RFC 3698.
bitStringMatch/ 2.5.13.16	Bit String/ 1.3.6.1.4.1.1466.115.121.1.6	Compares an asserted Bit String value with an attribute value of Bit String syntax. RFC 4517.
octetStringMatch/ 2.5.13.17	Octet String/ 1.3.6.1.4.1.1466.115.121.1.40	Compares an asserted value with an attribute value of octet string syntax using a byte-for-byte comparison. RFC 4517.
octetStringOrderingMatch/ 2.5.13.18	Octet String/ 1.3.6.1.4.1.1466.115.121.1.40	Compares the collation order of the asserted octet string with an attribute value of Octet String syntax. Zero precedes a one bit. Shorter strings precede longer strings. RFC 3698.
octetStringSubstringsMatch/ 2.5.13.19	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted substring with an attribute value of octet string syntax using a byte-for-byte comparison. RFC 4517.
telephoneNumberMatch/ 2.5.13.20	Telephone Number/ 1.3.6.1.4.1.1466.115.121.1.50	Compares an asserted value with an attribute value of Telephone Number syntax. RFC 4517.

TABLE 10-6. Matching Rules

Matching Rule/OID	Attribute Syntax Name/OID	Description
telephoneNumberSubstringsMatch/ 2.5.13.21	Substring Assertion/ 1.3.6.1.4.1.1466.115.121.1.58	Compares an asserted value with the substrings of an attribute value of Telephone Number String syntax. RFC 4517.
presentationAddressMatch/ 2.5.13.22	Presentation Address/ 1.3.6.1.4.1.1466.115.121.1.43	Compares an asserted value with an attribute value of Presentation Address syntax. RFC 4517.
uniqueMemberMatch/ 2.5.13.23	Name and Optional UID/ 1.3.6.1.4.1.1466.115.121.1.34	Compares an asserted value with an attribute value of Unique Member syntax. RFC 4517.
protocolInformationMatch/ 2.5.13.24	Protocol Information/ 1.3.6.1.4.1.1466.115.121.1.42	Compares an asserted value with an attribute value of Protocol Information syntax. RFC 4517.
generalizedTimeMatch/ 2.5.13.27	Generalized Time/ 1.3.6.1.4.1.1466.115.121.1.24	Compares an asserted value with an attribute value of Generalized Time syntax. RFC 4517.
generalizedTimeOrderingMatch/ 2.5.13.28	Generalized Time 1.3.6.1.4.1.1466.115.121.1.24	Compares the collation order of the asserted string with an attribute value of Generalized Time String syntax and case is ignored. RFC 4517.
integerFirstComponentMatch/ 2.5.13.29	Integer/ 1.3.6.1.4.1.1466.115.121.1.27	Equality matching rules for subschema attributes between an Integer syntax and the value syntax. RFC 4517.
objectIdentifierFirstComponent-Match/ 2.5.13.30	OID/ 1.3.6.1.4.1.1466.115.121.1.38	Equality matching rules for subschema attributes between an OID syntax and the value syntax. RFC 4517.
directoryStringFirstComponent-Match/ 2.5.13.31	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted Directory String value with an attribute value of type SEQUENCE whose first component is mandatory and of type Directory String. Returns true if the attribute value has a first component whose value matches the asserted Directory String using the rules of caseIgnoreMatch. RFC 3698.
wordMatch/ 2.5.13.32	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted word with any word in the attribute value for equality RFC 3698.
keywordMatch/ 2.5.13.33	Directory String/ 1.3.6.1.4.1.1466.115.121.1.15	Compares an asserted value with any keyword in the attribute value for equality. RFC 3698.

Viewing Matching Rules

You can view the matching rules on your directory server by using the Schema Editor on the Directory Management Console, over LDAP using the `ldapsearch` tool, or some third party tool.

The Schema Editor displays all of the matching rules on your directory server instance. It shows the basic properties that are allowed within the matching rule.

The **Properties** section displays the standard elements in a matching rule schema definition.

TABLE 10-7. Properties of Matching Rules

Properties: Matching Rules	Description
Name	Specifies the descriptive and unique name of the element.
Description	Specifies an optional definition that describes the matching rule. The analogous LDIF equivalent is "DESC".
OID	Specifies the globally unique object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.

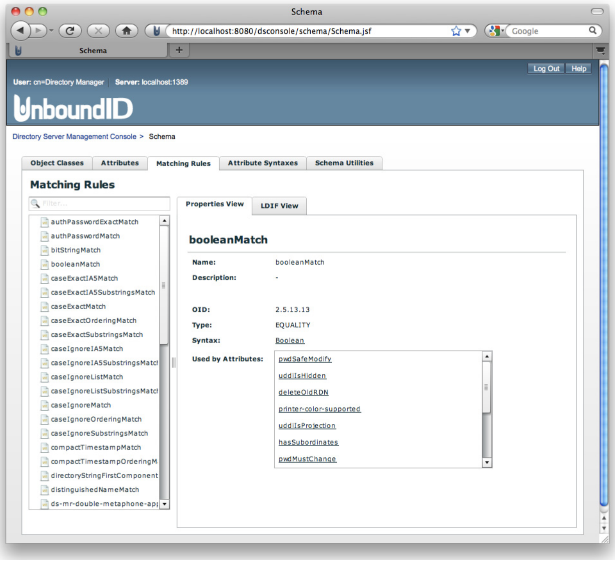
TABLE 10-7. **Properties of Matching Rules**

Properties: Matching Rules	Description
Type	Specifies the type of type of matching rule: Equality, Ordering, Substring, or Approximate.
Syntax	Specifies the matching rule syntax.
Used by Attributes	Specifies any attributes that use the corresponding matching rule.

To View Matching Rules Using the Schema Editor

1. Start the Directory Management Console, and then click Schema on the main menu. Check that the directory server instance associated with the console is also running.

2. On the Schema Editor, click the Matching Rules tab. You can also click the LDIF View tab to see the ASN.1 equivalent format.



authPasswordExactMatch
authPasswordMatch
bitStringMatch
booleanMatch
caseExactIA5Match
caseExactIA5SubstringsMatch
caseExactMatch
caseExactOrderingMatch
caseExactSubstringsMatch
caseIgnoreIA5Match
caseIgnoreIA5SubstringsMatch
caseIgnoreListMatch
caseIgnoreListSubstringsMatch
caseIgnoreMatch
caseIgnoreOrderingMatch
caseIgnoreSubstringsMatch
compactTimestampMatch
compactTimestampOrderingM
directoryStringFirstComponent
distinguishedNameMatch
ds-mr-double-metaphone-ap

booleanMatch

Name: booleanMatch
Description: -
OID: 2.5.13.13
Type: EQUALITY
Syntax: Boolean
Used by Attributes: pwdSafeModify, uddiIsHidden, deleteOldRDN, printer-color-supported, uddiIsProjection, hasSubordinates, pwdMustChange

To View Matching Rules Over LDAP

Use **ldapsearch** to view a multi-valued operational attribute, **matchingRules**, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --port 1389 --baseDN cn=schema --searchScope base "(objectclass=*)" \
  matchingRules
```

Managing Attribute Syntaxes

The attribute type definition has a **SYNTAX** element, or *attribute syntax*, that specifies how the data values for the attribute are represented. The syntax can be used to define a large range of data types necessary for client directory applications. An attribute syntax uses the Abstract Syntax Notation One (ASN.1) format for its definitions.

Attribute Syntax Definition

New attribute syntaxes require additional code to be implemented on the UnboundID Directory Server. If you need new syntax definitions, contact your authorized support provider for assistance.

The formal specification for attribute syntaxes is provided in RFC 4512, section 4.1.5 as follows:

```
SyntaxDescription = "(" wsp
  numericoid                ; Object identifier
  [ sp "DESC" sp qdstring ] ; Description
  extensions wsp ")"        ; Extensions followed by a white space and ")"
```

Default Attribute Syntaxes

The UnboundID Directory Server supports a large set of Attribute Syntax rules for directory applications. The default Attribute Syntax rules available for the directory server are listed in the table below.

TABLE 10-8. Attribute Syntaxes

LDAP Syntax	OID	Description
UUID	1.3.6.1.1.16.1	128-bit (16 octets) Universally Unique Identifier (UUID) used for Uniform Resource Names as defined in RFC 4122. For example, `a4028c1a-f36e-11da-ba1a-04112154bd1e`.
Attribute Type Description	1.3.6.1.4.1.1466.115.121.1.3	Syntax for the AttributeTypeDescription rule based on RFC 4517.
Binary	1.3.6.1.4.1.1466.115.121.1.5	Strings based on Basic Encoding Rules (BER) or Distinguished Encoding rules (DER). For example, an X.509 digital certificate or LDAP messages are BER encoded.
Bit String	1.3.6.1.4.1.1466.115.121.1.6	Sequence of binary digits based on RFC 4517. For example, '0010111'B.
Boolean	1.3.6.1.4.1.1466.115.121.1.7	TRUE or FALSE.
Certificate	1.3.6.1.4.1.1466.115.121.1.8	BER/DER-encoded octet strings based on an X.509 public key certificate as defined in RFC 4523.
Certificate List	1.3.6.1.4.1.1466.115.121.1.9	BER/DER-encoded octet string based on an X.509 certificate revocation list as defined in RFC 4523.
Certificate Pair	1.3.6.1.4.1.1466.115.121.1.10	BER/DER-encoded octet string based on an X.509 public key certificate pair as defined in RFC 4523.
Country String	1.3.6.1.4.1.1466.115.121.1.11	Two character country code specified in ISO 3166. For example, US, CA, etc.
DN	1.3.6.1.4.1.1466.115.121.1.12	Distinguished name of an entry as defined in RFC4514.
Delivery Method	1.3.6.1.4.1.1466.115.121.1.14	Sequence of services in preference order by which an entity receives messages as defined in RFC4517. For example, videotext $ telephone.
Directory String	1.3.6.1.4.1.1466.115.121.1.15	String of one or more characters from the Universal Character Set (UCS) using UCS Transformation Format 8 (UTF-8) encoding of the string.
DIT Content Rule Description	1.3.6.1.4.1.1466.115.121.1.16	DITContentRuleDescription as defined in RFC4517.

TABLE 10-8. Attribute Syntaxes

LDAP Syntax	OID	Description
DIT Structure Rule Description	1.3.6.1.4.1.1466.115.121.1.17	DITStructureRuleDesciption as defined in RFC4517.
Enhanced Guide	1.3.6.1.4.1.1466.115.121.1.21	Combination of attribute types and filter operators to be used to construct search filters as defined in RFC4517. For example, person#(sn$EQ)#oneLevel.
Facsimile Telephone Number	1.3.6.1.4.1.1466.115.121.1.22	Fax telephone number on the public switched telephone network as defined in RFC4517.
Fax	1.3.6.1.4.1.1466.115.121.1.23	Image generated using Group 3 fax process as defined in RFC4517.
Generalized Time	1.3.6.1.4.1.1466.115.121.1.24	String representing data and time as defined in RFC4517. `YYYYMMDDHHMMSS[.|,fraction][(+|-HHMM)|Z]` For example, 201103061032, 201103061032-0500, or 201103061032Z ("Z" indicates Coordinated Universal Time).
Guide	1.3.6.1.4.1.1466.115.121.1.25	Attribute types and filter operators as defined in RFC4517.
IA5 String	1.3.6.1.4.1.1466.115.121.1.26	String of zero or more characters from the International Alphabet 5 (IA5) character set as defined in RFC4517.
Integer	1.3.6.1.4.1.1466.115.121.1.27	String representations of integer values. For example, the character string "1234" represents the number 1234 as defined in RFC4517.
JPEG	1.3.6.1.4.1.1466.115.121.1.28	Image in JPEG File Interchange Format (JFIF) as defined in RFC4517.
Matching Rule Description	1.3.6.1.4.1.1466.115.121.1.30	MatchingRuleDescription as defined in RFC4512.
Matching Rule Use Description	1.3.6.1.4.1.1466.115.121.1.31	Attribute types to which a matching rule is applied in an extensibleMatch search filter (RFC4511).
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	Distinguished name and an optional unique identifier that differentiates identical DNs as defined in RFC4517. For example, `uid=jsmith,ou=People,dc=example,dc=com#'0111'B`
Name Form Description	1.3.6.1.4.1.1466.115.121.1.35	NameFormDescription as defined in RFC4512.
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	Sequence of one or more numerals and spaces as defined in RFC4517. For example, 14 848 929 102.
Object Class Description	1.3.6.1.4.1.1466.115.121.1.37	ObjectClassDescription as defined in RFC 4512.
OID	1.3.6.1.4.1.1466.115.121.1.38	Object identifier as defined in RFC 4512.
Other Mailbox	1.3.6.1.4.1.1466.115.121.1.39	Specifies an electronic mailbox as defined in RFC 4517. For example, `otherMailbox = google $ user@gmail.com`
Octet String	1.3.6.1.4.1.1466.115.121.1.40	Sequence of zero or more octets (8-bit bytes) as defined in RFC4517.
Postal Address	1.3.6.1.4.1.1466.115.121.1.41	Strings of characters that form a multi-line address in a physical mail system. Each component is separated by a "$". For example, `1234 Main St.$Austin, TX 78744$USA.`
Protocol Information	1.3.6.1.4.1.1466.115.121.1.42	Undefined.
Presentation Address	1.3.6.1.4.1.1466.115.121.1.43	String encoded OSI presentation address as defined in RFC 1278. For example, `TELEX+00728722+RFC-1006+03+10.0.0.6`
Printable String	1.3.6.1.4.1.1466.115.121.1.44	String of one or more printable ASCII alphabetic, numeric, and punctuation characters as defined in RFC 4517.
RFC3672 Subtree Specification	1.3.6.1.4.1.1466.115.121.1.45	Syntax based on subtree specification as defined as RFC 3672.
Supported Algorithm	1.3.6.1.4.1.1466.115.121.1.49	Octet string based on the LDAP-encoding for a supported algorithm value that results from the BER encoding of a SupportedAlgorithm ASN.1 value.
Telephone Number	1.3.6.1.4.1.1466.115.121.1.50	String of printable international telephone number representations in E.123 format as defined in RFC 4517. For example, `+1 512 904 5525`.
Teletex Terminal Identifier	1.3.6.1.4.1.1466.115.121.1.51	Identifier and telex terminal as defined in RFC 4517.

TABLE 10-8. Attribute Syntaxes

LDAP Syntax	OID	Description
Telex Number	1.3.6.1.4.1.1466.115.121.1.52	String representing the telex number, country code, and answerback code as defined in RFC 4517. For example, 812374, ch, ehhg.
UTC Time	1.3.6.1.4.1.1466.115.121.1.53	Character string representing the data and time in UTC Time format as defined as RFC 4517: YYMMDDHHMM[SS][(+|-HHMM)|Z], where Z is the coordinated universal time. For example, `0903051035Z`, `0903051035-0500`.
LDAP Syntax Description	1.3.6.1.4.1.1466.115.121.1.54	SyntaxDescription as defined in RFC4512.
Substring Assertion	1.3.6.1.4.1.1466.115.121.1.58	Syntax for assertion values in an extensible match as defined in RFC 4517.
Authentication Password Syntax	1.3.6.1.4.1.4203.1.1.2	Encoded password storage syntax as defined in RFC 3312. For example, the syntax specifies the storage scheme in brackets as follows: <storage -scheme>$<auth component>$<auth value> For example: `SSHA$xdEZWRqgyJk=$egDEFDXvdeeEnXUEIDPnd39dkpe=`
User Password Syntax	1.3.6.1.4.1.30221.1.3.1	Encoded password storage syntax as defined in RFC 2307. For example, the syntax specifies the storage scheme in brackets as follows: `{SSHA}XaljOF0ii3fOwCrU1klgBpWFayqSYs+5W1pMnw==`
Relative Subtree Specification	1.3.6.1.4.1.30221.1.3.2	Similar to the RFC 3672 subtree specification except it uses an LDAP search filter as the specification filter.
Absolute Subtree Specification	1.3.6.1.4.1.30221.1.3.3	Syntax for a subset of entries in a subtree based on RFC 3672.
Sun-defined Access Control Information	1.3.6.1.4.1.30221.1.3.4	Syntax for access control instructions used in Sun Directory Servers.
Compact Timestamp	1.3.6.1.4.1.30221.2.3.1	Syntax based on compact timestamp ISO 8601 format. For example, 20110306T102532.

Viewing Attribute Syntaxes

You can view the attribute syntaxes on your directory server by using the Schema Editor on the Directory Management Console, over LDAP using the **`ldapsearch`** tool, or some third party tool.

The Schema Editor displays all of the attribute syntaxes on your directory server instance. It shows the properties that are allowed within the attribute syntax.

The **Properties** section displays the standard elements in an attribute syntax.

TABLE 10-9. Properties of Attribute Syntaxes

Properties: Attribute Syntaxes	Description
Name	Indicates the descriptive and unique name of the element.
Description	Indicates an optional definition that describes the attribute syntax. The analogous LDIF equivalent is "DESC".

TABLE 10-9. **Properties of Attribute Syntaxes**

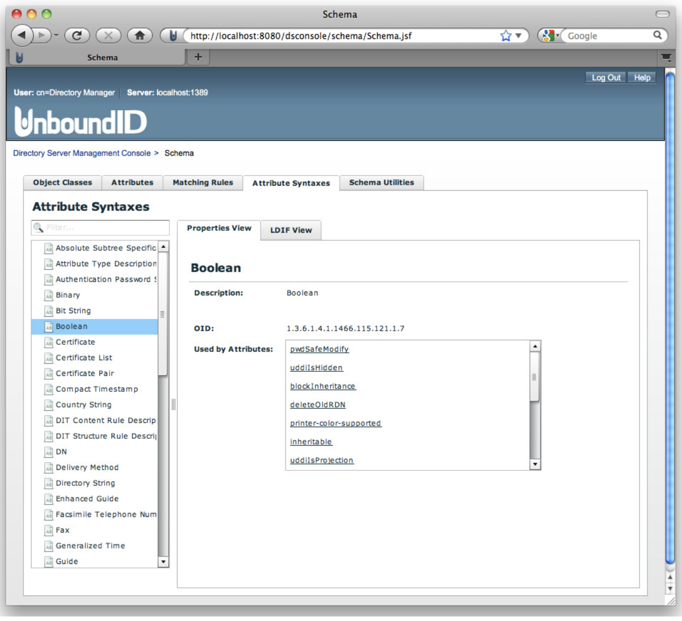
Properties: Attribute Syntaxes	Description
OID	Indicates the globally unique object identifier assigned to the schema definition. You can obtain a specific OID for your company that allows you to define your own object classes and attributes from IANA or ANSI.
Used by Attributes	Indicates any attributes that use the corresponding attribute syntax.

To View Attribute Syntaxes Using the Schema Editor

1. Start the Directory Management Console. Check that the directory server instance associated with the console is also running.

2. On the main menu, click Schema.

3. On the Schema Editor, click the Attribute Syntaxes tab. You can also click the LDIF View tab to see the ASN.1 equivalent format.



http://localhost:8080/dsconsole/schema/Schema.jsf

User: cn=Directory Manager | Server: localhost:1389

Directory Server Management Console > Schema

Attribute Syntaxes

Absolute Subtree Specific
Attribute Type Description
Authentication Password S
Binary
Bit String
Boolean
Certificate
Certificate List
Certificate Pair
Compact Timestamp
Country String
DIT Content Rule Descrip
DIT Structure Rule Descri
DN
Delivery Method
Directory String
Enhanced Guide
Facsimile Telephone Num
Fax
Generalized Time
Guide

Boolean

Description: Boolean

OID: 1.3.6.1.4.1.1466.115.121.1.7

Used by Attributes: pwdSafeModify
uddiIsHidden
blockInheritance
deleteOldRDN
printer-color-supported
inheritable
uddiIsProjection

To View Attribute Syntaxes Over LDAP

Use **ldapsearch** to view the directory server's published list of attribute syntaxes using the multi-valued operational attribute, **ldapSyntaxes**, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --port 1389 --baseDN cn=schema \
  --searchScope base "(objectclass=*)" ldapSyntaxes
```

Using the Schema Editor Utilities

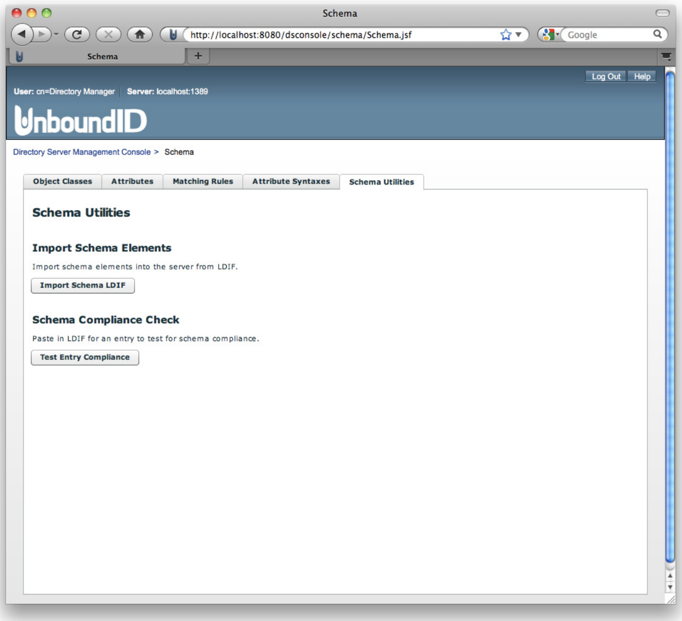
The Schema Editor provides a Utilities tab that allows you to import new schema elements from a file and to check schema compliance for new elements. If you are importing a schema file, the system automatically checks for compliance prior to the import. If your definition does not meet schema compliance, the system will display an error message. However, it is good practice to first check if your file is compliant with your schema prior to importing it.

To Check Schema Compliance Using the Schema Editor

1. Start the Directory Management Console. Check that the directory server instance associated with the console is also running.

2. On the main menu, click Schema.

3. On the Schema Editor, click the Schema Utilities tab, then click on "Test Entry Compliance".



Schema Utilities

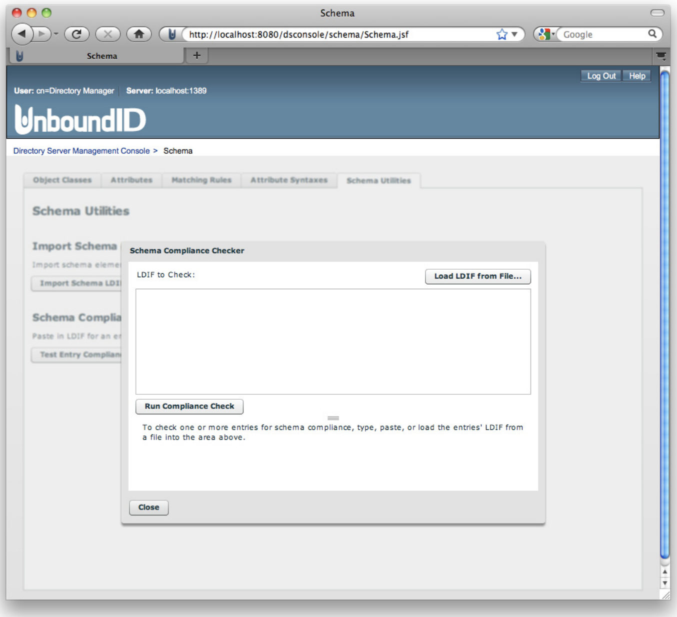
Import Schema Elements

Import schema elements into the server from LDIF.

Schema Compliance Check

Paste in LDIF for an entry to test for schema compliance.

4. Click "Load LDIF from File" to read in an LDIF file, or copy-and-paste your new schema definition, and then click "Run Compliance Check". If there is a problem, an error will be generated.



To check one or more entries for schema compliance, type, paste, or load the entries' LDIF from a file into the area above.

To Import a New Schema File Using the Schema Editor

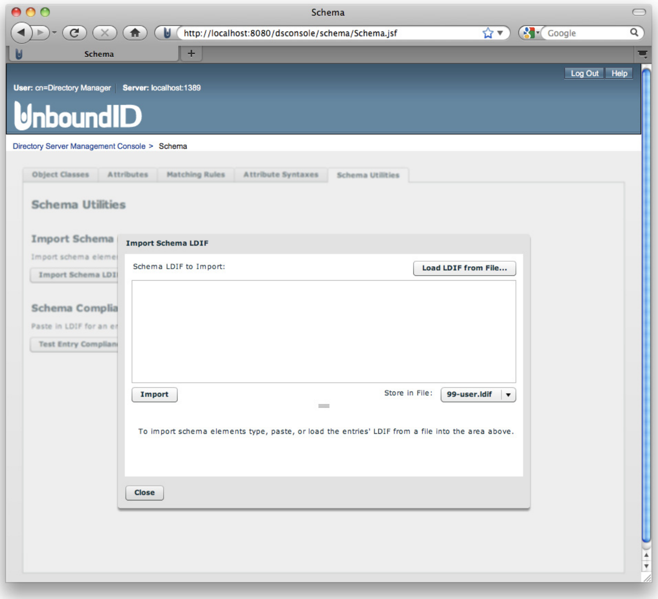
1. Start the Directory Management Console. Check that the directory server instance associated with the console is also running.

2. On the main menu, click Schema. Then, on the Schema Editor, click the Schema Utilities tab.

3. Click on Import Schema LDIF.

4. Click "Load LDIF from File" to upload an LDIF file, or copy-and-paste your new schema definition in the dialog window, and then click Import. If there is a problem, an error will

be generated. By default, you can load the file to `99-user.ldif`, but you can specify a new file by clicking the arrow and then clicking New.



To import schema elements type, paste, or load the entries' LDIF from a file into the area above.

Modifying a Schema Definition

The Directory Server only allows schema definitions that are read-write to be edited. In general, those schema definitions in the Custom folder of the Schema Editor can be modified.

To Modify a Schema Definition

1. Start the Directory Management Console. Check that the directory server instance associated with the console is running.
2. On the main menu, click Schema.
3. On the Schema Editor, click the Object Classes tab, and then click the Custom folder.
4. Select the object class that you want to modify, and then click the Edit button. The Edit dialog box appears.
5. Make your changes, and then click OK.

Deleting a Schema Definition

The Directory Server only allows schema definitions that are read-write to be deleted. In general, those schema definitions in the Custom folder of the Schema Editor can be removed from the system. You should make sure that the schema element is not currently in use.

To Delete a Schema Definition

1. Start the Directory Management Console. Check that the directory server instance associated with the console is also running.
2. On the main menu, click Schema.
3. On the Schema Editor, click the Object Classes tab, and then click the Custom folder.
4. Select the object class that you want to remove, and then click the Delete button.
5. On the Confirmation dialog, click Yes if you are sure that you want to delete the schema element.

Schema Checking

The UnboundID Directory Server provides full support for parsing all schema elements and provides access to all of its components. By default, the Directory Server enables schema checking for all operations, especially when importing data to the server or when modifying entries using the **`ldapmodify`** tool. Any schema violations will generate an error message to standard output.

To View the Schema Checking Properties

Use **`dsconfig`** to view the schema checking property.

```
$ bin/dsconfig --no-prompt get-global-configuration-prop \
  --property check-schema --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --advanced
```

To Disable Schema Checking

Although not recommended, you can use the **`dsconfig`** tool to disable the schema checking. This feature only applies to public backends. Schema checking is enforced on private backends, such as changes to the Configuration, Schema, Task, and others. An admin action alert will be generated when attempting to disable schema checking using **`dsconfig`** interactive or non-interactive mode. The alert provides alternatives to disabling schema checking.

1. Run the **dsconfig** command and specify the **set-global-configuration-prop** subcommand to disable the **check-schema** property.

   ```
   $ bin/dsconfig --no-prompt set-global-configuration-prop \
     --set check-schema:false
   ```

2. The system generates an admin action alert providing alternate options to disabling schema checking. Press **Enter** to continue the process or following one of the suggested tasks:

   ```
   One or more configuration property changes require administrative action or confir-
   mation/notification.  Those properties include:

    * check-schema:  Schema checking should only be disabled as a last resort since
   disabling schema checking harms performance and can lead to unexpected behavior in
   the server as well as the applications that access it. There are less severe options
   for addressing schema issues:

   1. Update the data to conform to the server schema.

   2. Modify the server schema to conform to the data. Contact support before modifying
   the server's default schema.

   3. Change the single-structural-objectclass-behavior property to allow entries to
   have no structural object class or multiple structural object classes.

   4. Change the invalid-attribute-syntax-behavior property to allow attribute values
   to violate their attribute syntax.

   5. Change the allow-zero-length-values property of the Directory String Attribute
   Syntax configuration to allow attributes with this syntax to have a zero length
   value.

   Continue?  Choose 'no' to return to the previous step (yes / no) [yes]:
   ```

Managing Matching Rule Uses

Matching Rule Use definitions map certain attribute types with a matching rule definition for extensible match filters. Extensible match filters allows clients to search using DN components, for example, **(ou:dn:=engineering)** or using an OID number, for example, **(cn:1.2.3.4:=Sam Carter)**. The matching rule use attribute publishes those attribute types and matching rule combinations, which can be used in extensible match assertions.

Typically, you define a matching rule use that is not normally specified in the attribute type definition. You can create new matching rule uses from the existing schema definitions by adding a custom schema file in the **<server-root>/config/schema** directory.

Matching Rule Use Definitions

Matching Rule Use can be specified with existing schema components and do not require additional code for its implementation.

The formal specification for matching rule uses is provided in RFC 4512, section 4.1.4 as follows:

```
MatchingRuleUseDescription = "(" wsp
  numericoid                ; Object identifier
  [ sp "NAME" sp qdescrs ] ; Short name descriptor
  [ sp "DESC" sp qdstring ]; Description
  [ sp "OBSOLETE" ]        ; Specifies if the rule use is inactive
  sp "APPLIES" sp oid      ; Attribute types
  extensions wsp ")"       ; Extensions followed by a white space and ")"
```

The following extensions are specific to the UnboundID Directory Server and are not defined in RFC 4512:

```
extensions = "X-ORIGIN" / ; Specifies where the object class is defined
"X-SCHEMA-FILE" /         ; Specifies which schema file contains the
                          ;  definition
"X-READ-ONLY"             ; True or False. Specifies if the file that
                          ;  contains the schema element is marked as
                          ;  read-only.
```

To View Matching Rule Uses

A matching rule use lists the attribute types that are suitable for use with an **`extensibleMatch`** search filter. Use **`ldapsearch`** to view the Directory Server's published list of matching rule uses using the multi-valued operational attribute, **`matchingRuleUse`**, which publishes the definitions on the Directory Server if any. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --port 1389 --baseDN cn=schema --searchScope base "(objectclass=*)" \
  matchingRuleUse
```

Managing DIT Content Rules

DIT Content Rules provide a way to precisely define what attributes may be present in an entry, based on its structural object class, without specifically creating a new object class definition. The DIT content rules can define the mandatory and optional attributes that entries contain, the set of auxiliary object classes that entries may be part of, and any optional attributes from the structural and auxiliary object classes that are prohibited from being present in the entries.

DIT Content Rule Definitions

DIT Content Rules can be specified with existing schema components and do not require additional code for its implementation. On the UnboundID Directory Server, only one DIT Content Rule may be defined for an entry in the structural object class.

The formal specification for DIT Content Rules is provided in RFC 4512, section 4.1.6 as follows:

```
DITContentRuleDescription = "(" wsp
  numericoid              ; Object identifier of the structural object class the rule
                          ; applies to
  [ sp "NAME" sp qdescrs ] ; Short name descriptor
  [ sp "DESC" sp qdstring ]; Description
  [ sp "OBSOLETE" ]        ; Specifies if the rule is inactive
  [ sp "AUX" sp oids ]     ; List of allowed auxiliary object classes
  [ sp "MUST" sp oids ]    ; List of required attributes
  [ sp "MAY" sp oids ]     ; List of allowed attributes in the entry
  [ sp "NOT" sp oids ]     ; List of prohibited attributes in the entry
  extensions wsp ")"       ; Extensions followed by a white space and ")"
```

The following extensions are specific to the UnboundID Directory Server and are not defined in RFC 4512:

```
extensions = "X-ORIGIN" / ; Specifies where the object class is defined
"X-SCHEMA-FILE" /         ; Specifies which schema file contains the definition
"X-READ-ONLY"             ; True or False. Specifies if the file that
                          ;  contains the schema element is marked as
                          ;  read-only.
```

To View DIT Content Rules

Use **ldapsearch** to view a multi-valued operational attribute **dITContentRules**, which publishes the definitions on the Directory Server if any. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --port 1389 --baseDN cn=schema --searchScope base "(objectclass=*)" \
  dITContentRules
```

Managing Name Forms

Name Forms define how entries can be named based on their structural object class. Specifically, name forms specify the structural object class to be named, as well as the mutually-exclusive set of required and allowed attributes to form the Relative Distinguished Names (RDNs) of the entries. Each structural object class may be associated with at most one name form definition.

Name Form Definitions

Name Forms can be specified with existing schema components and do not require additional code for its implementation. The formal specification for Name Forms is provided in RFC 4512, section 4.1.7.2 as follows:

```
NameFormDescription = "(" wsp
  numericoid               ; object identifier
  [ sp "NAME" sp qdescrs ]  ; short name descriptor
  [ sp "DESC" sp qdstring ] ; description
  [ sp "OBSOLETE" ]         ; not active
  sp "OC" sp oid            ; structural object class
  sp "MUST" SP oids         ; attribute types
  [ sp "MAY" sp oids ]      ; attribute types
  extensions wsp ")"        ; Extensions followed by a white space and ")"
```

The following extensions are specific to the UnboundID Directory Server and are not defined in RFC 4512:

```
extensions = "X-ORIGIN" /  ; Specifies where the form defined
"X-SCHEMA-FILE" /          ; Specifies which schema file contains the
                           ;  definition
"X-READ-ONLY"              ; True or False. Specifies if the file that
                           ;  contains the schema element is marked as
                           ;  read-only.
```

To View Name Forms

Use **ldapsearch** to view a multi-valued operational attribute **nameForms**, which publishes the definitions on the Directory Server. The attribute is stored in the subschema subentry.

```
$ bin/ldapsearch --port 1389 --baseDN cn=schema --searchScope base "(objectclass=*)" \
  nameForms
```

Managing DIT Structure Rules

DIT Structure Rules define which entries may be superior or subordinate to other entries in the DIT. Together with name forms, DIT Structure Rules determine how RDNs are added together to make up distinguished names (DNs). Because DITs do not have a global standard and are specific to a company's implementation, each DIT structure rule associates a name form with an object class and specifies each structure rule with an integer rule identifier, instead of an OID number. The identifier defines its relationship, either superior or subordinate, to another object class. If no superior rules are specified, then the DIT structure rule applies to the root of the subtree.

DIT Structure Rule Definition

DIT Structure Rules can be specified with existing schema components and do not require additional code for its implementation. The formal specification for DIT Structure Rules is provided in RFC 4512, section 4.1.7.1 as follows:

```
DITStructureRuleDescription = "(" wsp
  ruleid                    ; Unique integer rule identifier
  [ sp "NAME" sp qdescrs ]  ; Short name descriptor
  [ sp "DESC" sp qdstring ] ; Description
  [ sp "OBSOLETE" ]         ; Specifies if the rule is inactive
  sp "FORM" sp oid ]        ; OID or name form with which the rule is associated
  [ sp "SUP" ruleids ]      ; Superior rule IDs
  extensions wsp ")"        ; Extensions followed by a white space and ")"
```

The following extensions are specific to the UnboundID Directory Server and are not defined in RFC 4512:

```
extensions = "X-ORIGIN" / ; Specifies where the rule is defined
"X-SCHEMA-FILE" /         ; Specifies which schema file contains the
                          ;  definition
"X-READ-ONLY"             ; True or False. Specifies if the file that
```

```
                              ;  contains the schema element is marked as
                              ;  read-only.
```

To View DIT Structure Rules

Use **ldapsearch** to view a multi-valued operational attribute **dITStructureRules**, which publishes the definitions on the Directory Server. The attribute is stored in the subschema sub-entry.

```
$ bin/ldapsearch --port 1389 --baseDN cn=schema --searchScope base "(objectclass=*)" \
  dITStructureRules
```

11 Managing Password Policies

Overview

The UnboundID Directory Server provides a flexible password policy system to assign, manage, or remove password policies for root and non-root users. The password policy contains configurable properties for password expiration, failed login attempts, account lockout and other aspects of password and account maintenance on the Directory Server. The Directory Server also provides a global configuration option and a per-user privilege feature that disables parts of the password policy evaluation for production environments that do not require a password policy.

This chapter presents the following topics:

Viewing Password Policies

Password policies enforce a set of rules that ensure that access to directory data is not compromised through negligent password practices. The UnboundID Directory Server provides mechanisms to create and maintain password policies that determine whether passwords should expire, whether users are allowed to modify their own passwords, or whether too many failed authentication attempts should result in an account lockout. Many other options are available to fully configure a password policy for your directory services system.

The Directory Server provides three out-of-the-box password policies that can be used as-is or as templates for configuring custom policies. The Default Password Policy is automatically applied to all users although it is possible to use an alternate password policy on a per-user basis. The Root Password policy is enforced for the default root user, which uses a stronger password storage scheme (salted 512-bit SHA-2 rather than the salted 160-bit SHA-1 scheme used for normal users) and also requires that a root user provide his or her current password in order to select a new password.

The Secure Password Policy provides a more secure option than the default policy that makes use of a number of features, including password expiration, account lockout, last login time and last login IP address tracking, password history, and a number of password validators.

Caution	Using the Secure Password policy as-is may notably increase write load in the directory by requiring updates to password policy state attributes in user entries and/or requiring users to change passwords more frequently. In environments in which write throughput is a concern (including environments spread across multiple data centers requiring replication over a WAN), it may be useful to consider whether the policy should be updated to reduce the number of entry updates that may be required.

To View Password Policies

You can view the list of password policies configured on the Directory Server using the **`dsconfig`** tool, in either interactive or non-interactive mode, or the web administration console. The following example demonstrates the process for obtaining a list of defined password policies in non-interactive mode:

```
$ bin/dsconfig --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret \
  --no-prompt list-password-policies

Password Policy         : Type    : password-attribute : default-password-storage-scheme
------------------------:---------:--------------------:--------------------------------
Default Password Policy : generic : userpassword       : Salted SHA-1
Root Password Policy    : generic : userpassword       : Salted SHA-512
Secure Password Policy  : generic : userpassword       : CRYPT
```

To View a Specific Password Policy

Use **`dsconfig`** or the web administration console to view a specific password policy. In this example, view the Default Password Policy that applies to all uses for which no specific policy has been configured.

```
$ bin/dsconfig --no-prompt --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret get-password-policy-prop \
  --policy-name "Default Password Policy"

Property                                  : Value(s)
------------------------------------------:--------------------------
description                               : -
password-attribute                        : userpassword
default-password-storage-scheme           : Salted SHA-1
deprecated-password-storage-scheme        : -
password-validator                        : -
account-status-notification-handler       : -
allow-user-password-changes               : true
password-change-requires-current-password : false
force-change-on-add                       : false
force-change-on-reset                     : false
password-generator                        : Random Password Generator
require-secure-authentication             : false
require-secure-password-changes           : false
min-password-age                          : 0 s
max-password-age                          : 0 s
```

```
max-password-reset-age                    : 0 s
password-expiration-warning-interval      : 5 d
expire-passwords-without-warning          : false
allow-expired-password-changes            : false
grace-login-count                         : 0
lockout-failure-count                     : 0
lockout-duration                          : 0 s
lockout-failure-expiration-interval       : 0 s
require-change-by-time                    : -
last-login-time-attribute                 : ds-pwp-last-login-time
last-login-time-format                    : -
previous-last-login-time-format           : -
idle-lockout-interval                     : 0 s
password-history-count                    : 0
password-history-duration                 : 0 s
```

About the Password Policy Properties

The Directory Server provides a number of configurable properties that can be used to control password policy behavior. Some of the most notable properties include:

- **allow-user-password-changes**. Specifies whether users can change their own passwords. If a user attempts to change his/her own password, then the server will consult this property for the user's password policy, and will also ensure that the access control handler allows the user to modify the configured password attribute.

- **default-password-storage-scheme**. Specifies the names of the password storage schemes that are used to encode clear-text passwords for this password policy.

- **force-change-on-add**. Specifies whether users are required to change their passwords upon first authenticating to the Directory Server after their account has been created.

- **force-change-on-reset**. Specifies whether users are required to change their passwords after they have been reset by an administrator. An administrator is a user who has the `password-reset` privilege and the appropriate access control instruction to allow modification of other users' passwords.

- **idle-lockout-interval.** Specifies the maximum length of time that an account may remain idle (that is, the associated user does not authenticate to the server) before that user is locked out. For accounts which do not have a last login time value, the password changed time or the account creation time will be used. If that information is not available, then the user will not be allowed to authenticate. It is strongly recommended that the server be allowed to run for a period of time with last login time tracking enabled (i.e., values for both `last-login-time-attribute` and `last-login-time-format` properties) to ensure that users have a last login time before enabling idle account lockout.

- **ignore-duplicate-password-failures**. Indicates whether to ignore subsequent authentication failures using the same password as an earlier failed authentication attempt (within the time frame defined by the lockout failure expiration interval). If this option is "true", then multiple failed attempts using the same password will be considered only a single failure. This will only be applicable to simple bind attempts, or when using the password-modify extended operation with an old password. It will not be used with SASL authentication.

- **lockout-duration**. Specifies the length of time that an account is locked after too many authentication failures. The value of this attribute is an integer followed by a unit of seconds, minutes, hours, days, or weeks. A value of 0 seconds indicates that the account must remain locked until an administrator resets the password.

- **lockout-failure-count**. Specifies the maximum number of times that a user may be allowed to attempt to bind with the wrong password before that user's account becomes locked either temporarily (in which case the account will automatically be unlocked after a configurable length of time) or permanently (in which case an administrator must reset the user's password before the account may be used again). For example, if the value is set to 3, then the user is locked out on the fourth failed attempt.

- **max-password-age**. Specifies the maximum length of time that a user can continue to use the same password before he or she is required to choose a new password. The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w).

- **min-password-age**. Specifies the minimum length of time after a password change before the user is allowed to change the password again. The value of this attribute is an integer followed by a unit of seconds, minutes, hours, days, or weeks. This setting can be used to prevent users from changing their passwords repeatedly over a short period of time to flush an old password from the history so that it can be re-used.

- **password-change-requires-current-password**. Specifies whether users must include their current password when changing their password. This applies for both password changes made with the password modify extended operation as well as simple modify operations targeting the password attribute. In the latter case, if the current password is required then the password modification must remove the current value and add the desired new value (providing both the current and new passwords in the clear rather than using encoded representations).

- **password-expiration-warning-interval**. Specifies the length of time before a user's password expires that he or she receives notification about the upcoming expiration (either through the password policy or password expiring response controls). The value can be expressed in seconds (s), minutes (m), hours (h), days (d), or weeks (w).

- **password-history-count**. Specifies the maximum number of former passwords to maintain in the password history. Any newly proposed password is checked against this history in order to prevent a user from changing his or her password to a value they had already used.

- **password-history-duration**. Specifies the maximum length of time that passwords remain in the password history. When choosing a new password, the proposed password is checked to ensure that it does not match the current password, nor any other password in the history list. A value of zero seconds indicates that either no password history is to be maintained (if the password history count has a value of zero), or that there is no maximum duration for passwords in the history (if the password history count has a value greater than zero).

- **password-validator**. Specifies the names of the password validators that are used with the associated password storage scheme. The password validators are invoked when a user attempts to provide a new password, to determine whether the new password is acceptable.

- **require-secure-authentication**. Indicates whether users with the associated password policy are required to authenticate in a secure manner. This might mean either using a secure communication channel between the client and the server, or using a SASL mechanism that does not expose the credentials.
- **require-secure-password-changes**. Indicates whether users with the associated password policy are required to change their password in a secure manner that does not expose the credentials.

Note	To view a description of each of the password policy properties, see the *UnboundID Directory Server Configuration Reference* that is bundled with the UnboundID Directory Server.

Modifying an Existing Password Policy

You can modify an existing password policy to suit your company's requirements.

To Modify an Existing Password Policy

Use `dsconfig` (in interactive or non-interactive mode) or the web administration console to modify the configuration for any defined password policy. The following example sets some of the properties presented in the previous section for the default password policy using `dsconfig` in non-interactive mode:

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "max-password-age:90 days" \
  --set "password-expiration-warning-interval:14 days" \
  --set "lockout-failure-count:3" \
  --set "password-history-count:6" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --no-prompt
```

Creating a New Password Policy

You can create any number of password policies in the Directory Server using either the `dsconfig` tool (in interactive or non-interactive mode) or the web administration console.

To Create a New Password Policy

Use **dsconfig** (in interactive or non-interactive mode) or the web administration console to create a new password policy. The following example demonstrates the process for creating a new policy using **dsconfig** in non-interactive mode.

```
$ bin/dsconfig --no-prompt create-password-policy \
  --policy-name "TDemo Password Policy" \
  --set "password-attribute:userpassword" \
  --set "default-password-storage-scheme:Salted SHA-1" \
  --set "force-change-on-add:true" --set "force-change-on-reset:true" \
  --set "password-expiration-warning-interval:2 weeks" \
  --set "max-password-age:90 days" --set "lockout-duration:24 hours" \
  --set "lockout-failure-count:3" \
  --set "password-change-requires-current-password:true" \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret
```

To Assign a Password Policy to an Individual Account

In order to indicate that a user should be subject to a particular password policy (rather than automatically inheriting the default policy), include the **ds-pwp-password-policy-dn** operational attribute in that user's entry with a value equal to the DN of the desired password policy for that user. This attribute can be explicitly included in a user's entry, or it can be generated by a virtual attribute, which makes it easy to apply a custom password policy to a set of users based on a flexible set of criteria.

The following example demonstrates the process for explicitly configuring a custom password policy for a user:

1. Create a file (**assign.ldif**) with the following contents:

```
dn: uid=user.1,ou=People,dc=example,dc=com
changtype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password Policies,cn=config
```

2. Use **ldapmodify** to apply the modification to the user's entry.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --filename assign.ldif
```

To Assign a Password Policy Using a Virtual Attribute

It is possible to automatically assign a custom password policy for a set of users using a virtual attribute. The virtual attribute can be configured so that it can use a range of criteria for selecting the entries for which the virtual attribute should appear.

1. Create an LDIF file, which may be used to add a group to the server.

```
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalunit
objectClass: top
ou: Groups
```

```
dn: cn=Engineering Managers,ou=groups,dc=example,dc=com
objectClass: groupOfUniqueNames
objectClass: top
cn: Engineering Managers
uniqueMember: uid=user.0,ou=People,dc=example,dc=com
ou: groups
```

2. Use **`ldapmodify`** to add the entries to the server.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --defaultAdd --filename groups.ldif
```

3. Use **`dsconfig`** to create a virtual attribute that will add the **`ds-pwp-pasword-policy-dn`** attribute with a value of "**`cn=Demo Password Policy,cn=Password Policies,cn=config`**" to the entries for all users that are members of the "**`cn=Engineering Managers,ou=Groups,dc=example,dc=com`**" group.

```
$ bin/dsconfig --no-prompt create-virtual-attribute \
  --name "Eng Mgrs Password Policy" \
  --type user-defined \
  --set "description:Eng Mgrs Grp PWPolicy" \
  --set enabled:true \
  --set attribute-type:ds-pwp-password-policy-dn \
  --set "value:cn=Demo Password Policy,cn=Password Policies,cn=config" \
  --set "group-dn:cn=Engineering Managers,ou=Groups,dc=example,dc=com \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret
```

4. Use **`ldapsearch`** to verify that a user in the group contains the assigned password policy DN.

```
$ bin/ldapsearch --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --baseDN dc=example,dc=com "(uid=user.0)" \
  ds-pwp-password-policy-dn

dn: uid=user.0,ou=People,dc=example,dc=com
ds-pwp-password-policy-dn: cn=Demo Password Policy,cn=Password Policies,cn=config
```

To Delete a Password Policy

Custom password policies may be removed using either the **`dsconfig`** tool (in interactive or non-interactive mode) or the web administration console). For example:

```
$ bin/dsconfig delete-password-policy \
  --policy-name "Demo Password Policy" \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret
```

Note	Before removing a password policy configuration entry from the server, it is important to ensure that no users are still configured to use that policy. If a user's entry includes a **`ds-pwp-password-policy-dn`** value (whether real or virtual) which references a nonexistent policy, then that user will not be allowed to authenticate to the server.

Modifying a User's Password

There are two primary ways to change user passwords in the Directory Server:

- Perform a modify operation which replaces the value of the password attribute (often **`userPassword`**). Note that in some configurations, when a user attempts to change his or her own password it may be necessary to perform the modification by removing the password value and adding the desired new value, in order to demonstrate that the user knows the current password value.
- Use the password modify extended operation in order to change the password. Note that if a user is changing his or her own password, it may be necessary to provide the current password value. The server will allow a new password to be provided (assuming that the new password is acceptable to all configured password validators), or it may automatically generated a new password for the user.

Note that regardless of the mechanism used to change the password, all password values should be provided in clear text rather than pre-encoded, and the user will be required to have sufficient access control rights to update the password attribute in the target user's entry. Further, when one user attempts to change the password for another user, then the requester will be required to have the **`password-reset`** privilege.

To Change a User's Password using the Modify Operation

Use **`ldapmodify`** to change a user's password by replacing the **`userPassword`** attribute.

```
$ bin/ldapmodify --hostname server.example.com --port 389 \
  --bindDN uid=admin,dc=example,dc=com --bindPassword adminpw

dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: newpw
```

To Change a User's Password using the Password Modify Extended Operation

Use **`ldappasswordmodify`** to request that the Password Modify extended operation be used to modify a user's password.

```
$ ldappasswordmodify --hostname server.example.com \
  --port 389 --bindDN uid=admin,dc=example,dc=com --bindPassword adminpw \
  --authzID dn:uid=jdoe,ou=People,dc=example,dc=com --newPassword newpw
```

To Use an Automatically-Generated Password

Use **`ldappasswordmodify`** to automatically generate a new password for a user.

```
$ bin/ldappasswordmodify --port 1389 \
  --bindDN "cn=Directory Manager" \
  --bindPassword secret --authzID "u:user.1"
```

```
The LDAP password modify operation was successful
Generated Password:  fbi27oqy
```

Managing User Accounts

The Directory Server provides a user management utility, the **`manage-account`** tool, that provides a means to quickly view and manipulate a number of password and account policy properties for a user.

To Return the Password Policy State Information

Use **`manage-account`** to get information about the account's password policy.

```
$ bin/manage-account get-all \
  --targetDN uid=user.1,ou=People,dc=example,dc=com \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret

Password Policy DN:  cn=Demo Password Policy,cn=Password Policies,cn=config
Account Is Disabled:  false
Account Expiration Time:
Seconds Until Account Expiration:
Password Changed Time:  19700101000000.000Z
Password Expiration Warned Time:
Seconds Until Password Expiration:  1209600
Seconds Until Password Expiration Warning:  0
Authentication Failure Times:
Seconds Until Authentication Failure Unlock:
Remaining Authentication Failure Count:  3
Last Login Time:
Seconds Until Idle Account Lockout:
Password Is Reset:  false
Seconds Until Password Reset Lockout:
Grace Login Use Times:
Remaining Grace Login Count:  0
Password Changed by Required Time:
Seconds Until Required Change Time:
Password History:
```

To Determine Whether an Account is Disabled

Use **`manage-account`** to determine whether a user's account has been disabled.

```
$ bin/manage-account get-account-is-disabled \
  --targetDN uid=user.1,ou=People,dc=example,dc=com \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret
Account Is Disabled:  true
```

To Disable an Account

Use **`manage-account`** to disable a user's account.

```
$ bin/manage-account set-account-is-disabled \
  --operationValue true --targetDN uid=user.1,ou=People,dc=example,dc=com \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret

Account Is Disabled:  true
```

To Enable a Disabled Account

Use **manage-account** to enable an account that is in a disabled state.

```
$ bin/manage-account clear-account-is-disabled \
  --targetDN uid=user.1,ou=People,dc=example,dc=com \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret

Account Is Disabled:  false
```

Assigning the Manage-Account Access Privileges to Non-Root Users

Non-root users with admin right privileges (e.g., **uid=admin**) working with the **manage-account** tool will require access control permission to interact with certain password policy operational attributes.

For example, the presence of the **ds-pwp-account-disabled** operational attribute in an entry determines that the entry is disabled. If the non-root admin user does not have the access privilege to read or interact with the **ds-pwp-account-disabled** operational attribute, the **manage-account** tool may report that the account is still active. An account is considered active if the operational attribute does not exist in the entry or if the admin user does not have permission to see it.

To Assign the Manage-Account Access Privileges to Non-Root Users

1. Create a non-root user admin account, such as **uid=admin,dc=example,dc=com**. Grant the **password-reset** privilege to the account. See steps 1 and 4 in the Configuring Administrators section for more information.

2. Grant access control privileges to the admin account. Here you have two options: give full access rights to all user attributes and specify the **ds-pwp-account-disabled** operational attribute:

   ```
   aci: (targetattr="*||ds-pwp-account-disabled")
     (version 3.0; acl "Allow user to access all user attributes and the
      ds-pwp-account-disabled operational attribute";
      allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
   ```

   or give full access to all user and operational attributes:

   ```
   aci: (targetattr="*||+")
     (version 3.0; acl "Allow user to access all user and operational attributes";
      allow (all) userdn="ldap:///uid=admin,dc=example,dc=com";)
   ```

3. Run the **manage-account** tool to disable an account.

   ```
   $ bin/manage-account set-account-is-disabled \
     --targetDN uid=user.0,ou=People,dc=example,dc=com \
   ```

```
    --port 1389 \
    --bindDN "uid=admin,dc=example,dc=com" \
    --bindPassword password
```

4. Optional. The Directory Server comes with a default set of global ACIs that include access privileges to the pasword policy extended operation, which is automatically available to all authenticated users. If your directory services system limits access to these Global ACIs, you can grant the extended operation to all authenticated users ("ldap:///all";) or to an admin user as follows:

```
aci: (extop="1.3.6.1.4.1.30221.1.6.1")
  (version 3.0; acl "Allow the password policy extended operation";
   allow (read) userdn="ldap:///uid=admin,dc=example,dc=com";)
```

Disabling Password Policy Evaluation

The Directory Server provides a global configuration property (**`disable-password-policy-evaluation`**) that can be used to disable most password policy evaluation processing. This provides a convenience for those production environments that do not require password policy support. If the **`disable-password-policy`** property is set to true, passwords will still be encoded and evaluated, but only account expiration and account disabling will be in effect. All other password policy properties, such as password expiration, lockout, and force change on add or reset, are ignored.

The server also supports the use of a **`bypass-pw-policy`** privilege, which can be used to skip password policy evaluation for operations on a per-user basis. If a user has this privilege, then they will be allowed to perform operations on user entries that would normally be rejected by the password policy associated with the target entry. Note that this privilege will not have any effect for bind operations.

To Globally Disable Password Policy Evaluation

Use **`dsconfig`** to set the **`disable-password-policy-evaluation`** property globally for the Directory Server.

```
$ bin/dsconfig --no-prompt set-global-configuration-prop \
  --set "disable-password-policy-evaluation:false" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

To Exempt a User from Password Policy Evaluation

Use **`ldapmodify`** to add the **`bypass-pw-policy`** privilege to a user entry.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
dn: uid=user.1,ou=People,dc=example,dc=com
changetype: modify
```

```
add: ds-privilege-name
ds-privilege-name: bypass-pw-policy
```

Managing Password Validators

A password validator is a password policy component that is used to determine if a new password is acceptable. A password policy can be configured with any number of password validators. If a password policy is configured with multiple password validators, then all of them must consider a proposed new password acceptable before it will be allowed.

The UnboundID Directory Server offers a number of types of password validators, including those listed in the following table. It is also possible to use the UnboundID Server SDK to create custom password validators with whatever constraints are necessary for your environment.

TABLE 11-1. Password Validators

Password Validator	Description
Attribute Value	Ensures that the proposed password does not match the value of another attribute in the user's entry. The validator can be configured to look in all attributes or in a subset of attributes. It can perform forward and reverse mapping, and it can also reject values which are substrings of another attribute.
Character Set	Ensures that the proposed password contains a sufficient number of characters from one or more user-defined character sets. For example, the validator can ensure that passwords must have at least one lowercase letter, one uppercase letter, one digit, and one symbol.
Dictionary	Ensures that the proposed password is not present in a specified dictionary file, optionally also testing the password with all characters in reverse order. A large dictionary file is provided with the server, but the administrator can supply an alternate dictionary. In this case, then the dictionary must be a plain-text file with one word per line.
Length-Based Password Validator	Ensures that the number of characters in the proposed new password is within an acceptable range. Both a maximum and minimum number of characters may be specified.
Regular Expression Validator	Ensures that a proposed password either matches or does not match a given regular expression.
Repeated Characters	Ensures that a proposed password does not contain a substring in which the same character is repeated more than a specified number of times (for example, "aaaaa" or "aaabbb"). The validator can be configured to operated in a case-sensitive or case-insensitive manner, and you can also define custom sets of equivalent characters (for example, you could define all digits as equivalent, so the proposed password could not contain more than a specified number of consecutive digits.

TABLE 11-1. Password Validators

Password Validator	Description
Similarity-Based Password Validator	Ensures that the proposed new password is not too similar to the current password, using the Levenstein Distance algorithm, which calculates the number of characters that need to be inserted, removed, or replaced to transform one string into another. Note that for this password validator to be effective, it is necessary to have access to the user's current password. Therefore, if this password validator is to be enabled, the **`password-change-requires-current-password`** attribute in the password policy configuration must also be set to true.
Unique Characters	Ensures that the proposed password contains at least a specified minimum number of unique characters, optionally using case-insensitive validation.

Managing Password Validators

You can use the **`dsconfig`** configuration tool or web administration console to configure or modify any password validators. Once you have defined your password validators, you can add them to an existing password policy. The following example procedures show the **`dsconfig`** non-interactive commands necessary to carry out such tasks. If you use **`dsconfig`** in interactive command-line mode, you can access the Password Validator menu in the Basic Objects menu. For more details on the password validator properties, see the *UnboundID Directory Server Configuration Reference*.

To View the List of Defined Password Validators

Use **`dsconfig`** to view the set of password validators defined in the Directory Server.

```
$ bin/dsconfig --no-prompt list-password-validators \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

To Configure the Attribute Value Password Validator

1. Use **`dsconfig`** to edit the existing default configuration for the Attribute Value Password Validator. For example, the following change configures the validator to only examine a specified set of attributes.

```
$ bin/dsconfig --no-prompt set-password-validator-prop \
  --validator-name "Attribute Value" \
  --set match-attribute:cn \
  --set match-attribute:sn \
  --set match-attribute:telephonenumber \
  --set match-attribute:uid \
  --port 1389 --bindDN "cn=Directory Manager \
  --bindPassword secret
```

2. Update an existing password policy to use the Attribute Value Password Validator.

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Demo Password Policy" \
  --set "password-validator:Attribute Value" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

3. Test the Attribute Value Password Validator by submitting a password that is identical to one of the configured attributes (`cn`, `sn`, `telephonenumber`, `uid`).

```
$ bin/ldappasswordmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword user.0

The LDAP password modify operation failed with result code 53
Error Message:  The provided new password failed the validation checks defined in
the server:  The provided password was found in another attribute in the user entry
```

To Configure the Character Set Password Validator

1. Use `dsconfig` to edit the existing default configuration. In this example, we remove the requirement that special characters be required in a password, and add a requirement that at lease two digits must be included in the password. Thus, in this example, all newly created passwords must have at least one lowercase letter, one uppercase letter, and two digits.

```
$ bin/dsconfig --no-prompt set-password-validator-prop \
  --validator-name "Character Set" \
  --remove character-set:1:0123456789 \
  --remove "character-set:1:~\!@#\$\%^&*()-_=+[]{}\|;:,.<>/?" \
  --add character-set:2:0123456789 \
  --set allow-unclassified-characters:false \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

2. Update an existing password policy to use the Character Set Password Validator.

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Demo Password Policy" \
  --set "password-validator:Character Set" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

3. Test the Character Set Password Validator by submitting a password that meets the requirements (one lowercase letter, one uppercase letter, two digits). The following example should reject the given password because it does not pass the Character Set Password Validator.

```
$ bin/ldappasswordmodify --port 1389 \
  --bindDN "cn=Directory Manager" \
  --bindPassword secret \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword abab1
```

To Configure the Dictionary Password Validator

1. Use **dsconfig** to edit the existing default configuration. In this example, we enable the validator and use the default dictionary file located at **config/wordlist.txt**. If you want to specify a specific file, use the **--set dictionary-file:/path/to/file** command.

```
$ bin/dsconfig --no-prompt set-password-validator-prop \
  --validator-name "Dictionary" --set enabled:true \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

2. Update an existing password policy to use the Dictionary Password Validator.

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Demo Password Policy" \
  --set "password-validator:Dictionary" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

3. Test the Dictionary Password Validator by submitting a password that is identical to one of words in the dictionary file.

```
$ bin/ldappasswordmodify --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword Zyrian

The LDAP password modify operation failed with result code 53
Error Message:  The provided new password failed the validation checks defined in
the server:  The provided password was found in the server's dictionary
```

To Manage the Length-Based Password Validator

1. Use **dsconfig** to edit the existing default configuration. In this example, we set the required minimum number of characters in a password to five.

```
$ bin/dsconfig --no-prompt create-password-validator \
  --validator-name "Length-Based Password Validator" \
  --set max-password-length:5 --set min-password-length:5 \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

2. Update an existing password policy to use the Length-Based Password Validator.

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Temp Password Policy" \
  --set "password-validator:Length-Based Password Policy" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

3. Test the Length-Based Password Validator by submitting a password that has fewer than the minimum number of required characters.

```
$ bin/ldappasswordmodify --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword abcd
```

```
The LDAP password modify operation failed with result code 53
Error Message:  The provided new password failed the validation checks defined in
the server:  The provided password is shorter than the minimum required length of 5
characters
```

To Configure the Regular Expression Password Validator

1. Use **dsconfig** to create a Regular Expression password validator. The following password validator checks that the password contains at least one number, one lowercase letter, and one uppercase letter with no restrictions on password length. If the password matches the regular expression, then it will be accepted. When using the following command, remember to include the LDAP/LDAPS connection parameters (host name and port), bind DN, and bind password.

```
$ bin/dsconfig --no-prompt create-password-validator \
  --validator-name "Regular Expression" \
  --type regular-expression --set enabled:true \
  --set "match-pattern:^\\w*(?=\\w*\\d)(?=\\w*[a-z])(?=\\w*[A-Z])\\w*\$" \
  --set match-behavior:require-match
```

2. Update an existing password policy to use the Regular Expression validator.

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Demo Password Policy" \
  --set "password-validator:Regular Expression"
```

3. Test the Regular Expression Validator by submitting a password that meets the requirements (contains one number, one lowercase letter, and one uppercase letter), then run it again with a password that does not meet these requirements.

```
$ bin/ldappasswordmodify \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaA1
The LDAP password modify operation was successful
```

Try another password. The following password should fail, because no uppercase letter is present.

```
$ bin/ldappasswordmodify \
  --port 1389 --bindDN "cn=Directory Manager" --bindPassword secret \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" --newPassword baaa1

Error Message:  The provided new password failed the validation checks defined in
the server:  The provided password is not acceptable because it does not match reg-
ular expression pattern '^\w*(?=\w*\d)(?=\w*[a-z])(?=\w*[A-Z])\w*$'
```

To Configure the Repeated Character Password Validator

1. Use **dsconfig** to edit the existing default configuration. In this example, we set the maximum consecutive length of any character to 3. For example, the following validator rejects any passwords, such as "baaaa1" or "4eeeeb", etc.

```
$ bin/dsconfig --no-prompt set-password-validator-prop \
  --validator-name "Repeated Characters" \
  --set max-consecutive-length:3 \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

Or, you can configure the validator to reject any character from a pre-defined character set that appears more than the specified number of times in a row (2). You can also specify more than one character set. For example, the following validator defines two characters sets: [abc] and [123]. It rejects any passwords with more than two consecutive characters from a character set. Thus, "aaa", "bbb", "ccc", "abc", or "123" and so on fails, but "12a3" is okay.

```
$ bin/dsconfig --no-prompt set-password-validator-prop \
  --validator-name "Repeated Characters" \
  --set character-set:123 --set character-set:abc \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

2. Update an existing password policy to use the Repeated Character Password Validator.

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Demo Password Policy" \
  --set "password-validator:Repeated Characters" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

3. Test the Repeated Character Validator by submitting a password that has more than the maximum allowable length of consecutive characters.

```
$ bin/ldappasswordmodify --port 1389 \
  --bindDN "cn=Directory Manager" \
  --bindPassword secret \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword baaa1

The LDAP password modify operation failed with result code 53
Error Message: The provided new password failed the validation checks defined in the
server:  The provided password contained too many instances of the same character
appearing consecutively. The maximum number of times the same character may appear
consecutively in a password is 2
```

To Configure the Similarity-Based Password Validator

1. Use **dsconfig** to edit the existing default configuration. In this example, we set the minimum number of differences to 2.

```
$ bin/dsconfig --no-prompt set-password-validator-prop \
  --validator-name "Similarity-Based Password Validator" \
  --set min-password-difference:2 \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

2. Update an existing password policy to use the Similarity-Based Password Validator. The **password-change-requires-current-password** property must be set to TRUE, so that the password policy will ensure that the user's current password is available when that user is choosing a new password.

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Demo Password Policy" \
  --set "password-validator:Similarity-Based Password Validator" \
  --set password-change-requires-current-password:true \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

3. Test the Similarity-Based Password Validator by submitting a password that has fewer than the minimum number of changes (e.g., 2). The **`ldappasswordmodify`** command requires the **`--currentPassword`** option when testing the Similarity-Based Password Validator.

```
$ bin/ldappasswordmodify --port 1389 \
    --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --currentPassword abcde --newPassword abcdd

The LDAP password modify operation failed with result code 49
```

To Configure the Unique Characters Password Validator

1. Use **`dsconfig`** to edit the existing default configuration. In this example, we set the minimum number of unique characters that a password is allowed to contain to 3.

```
$ bin/dsconfig --no-prompt set-password-validator-prop \
  --validator-name "Similarity-Based" \
  --set min-unique-characters:3 \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

2. Update an existing password policy to use the Unique Characters Password Validator.

```
$ bin/dsconfig --no-prompt set-password-policy-prop \
  --policy-name "Demo Password Policy" \
  --set "password-validator:Unique Characters" \
  --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret
```

3. Test the Unique Characters Password Validator by submitting a password that has fewer than the minimum number of unique characters (e.g., 3).

```
$ bin/ldappasswordmodify --port 1389 \
  --bindDN "cn=Directory Manager" \
  --bindPassword secret \
  --authzID "uid=user.0,ou=People,dc=example,dc=com" \
  --newPassword aaaaa

The LDAP password modify operation failed with result code 53
Error Message:  The provided new password failed the validation checks defined in
the server:  The provided password does not contain enough unique characters.  The
minimum number of unique characters that may appear in a user password is 3
```

12 Managing Replication

Overview

The UnboundID Directory Server provides a robust multi-master replication system to ensure high availability and fast failover in production environments. Replication is an automated component of the Directory Server that pushes LDAP updates to other servers in the topology, ensuring that all servers end up with identical versions of the data. The replicated server environment ensures that LDAP clients can quickly fail over to another server instance without any data or transaction loss.

The Directory Server provides a number of features that allow you to configure multi-master, single master, or read-only servers that communicate over standard or encrypted communication channels over LANs, WANs, or both.

This chapter presents the architectural overview of replication, detailed configuration steps, ways to monitor replication as well as troubleshooting steps:

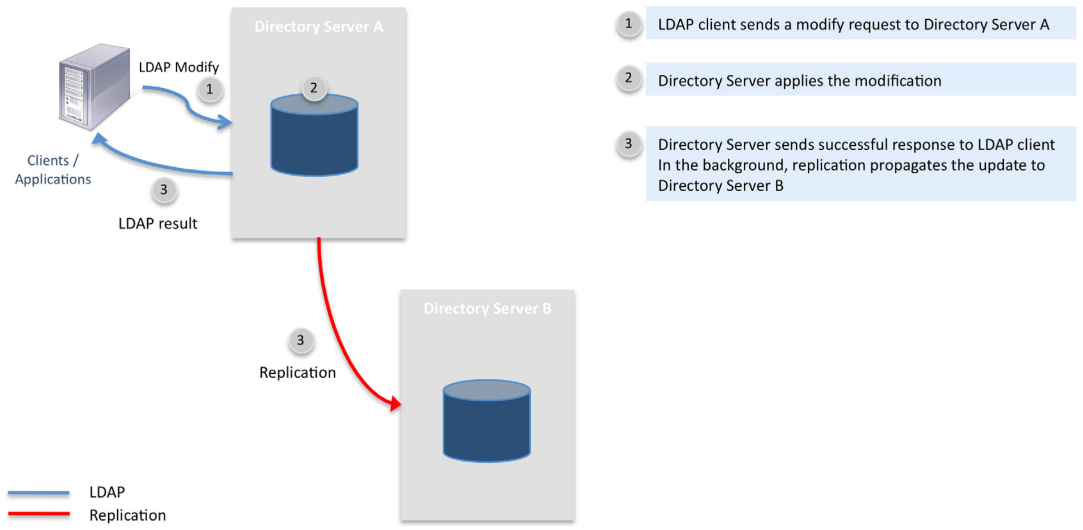
Overview of Replication

Replication is a data synchronization mechanism that ensures that updates made to a directory database are automatically replayed to other servers. Replication improves directory availability when unforeseen or planned outages occur, and improves directory search performance by allowing client requests to be distributed across multiple servers.

By default, all directory servers participating in replication are writable, so LDAP clients may perform updates at any of these Directory Server instances. The changes will be propagated to other servers automatically in the background. Replication in the UnboundID Directory Server is based on an eventual-consistency model in which all updates that propagate through the system will eventually be consistent on all servers over a period of time.

The following figure demonstrates the basic flow of replication. The replication process flow is designed to immediately propagate changes to the other replicas in the topology with little or no latency.

FIGURE 12-1. Replication Process Flow


Directory Server A
LDAP Modify
1
2
Clients / Applications
3
LDAP result
Directory Server B
3
Replication
LDAP
Replication
1 LDAP client sends a modify request to Directory Server A
2 Directory Server applies the modification
3 Directory Server sends successful response to LDAP client In the background, replication propagates the update to Directory Server B

The benefits of replication can be summarized as follows:

- **Provides High-Availability**. Replication allows multiple servers to service all types of client requests.
- **Improved Search Performance.** Search requests may be directed to any directory server participating in replication. Note that replication does not improve write throughput since updates need to be applied at all servers.
- **Synchronize Data over WAN**. The built-in compression feature in the replication protocol allows efficient propagation of updates over WAN links.

Replication is not a general purpose synchronization facility, since it creates replicas with exact copies of the replicated data. Consider using the UnboundID Synchronization Server if replication does not meet the data synchronization requirements in your deployment. The differences between replication and synchronization are as follows:

- **Replication cannot Synchronize between Different DIT Structures**. The DN of replicated entries must be the same on all servers. In some situations, it may be desirable to replicate entries with the help of DN mapping that are under different base DNs, but represent the same data, for example `uid=john.doe,ou=people,o=corp` on one server may represent the same user as `uid=john.doe,ou=people,dc=example,dc=com`. This is not supported by replication.

- **Replication cannot Map Attribute Types or Transform Attribute Values**. In some situations, it may be necessary to map attribute types or transform attribute values when synchronizing data from one server to another. Replication does not support either attribute type mappings or attribute value transformations.

- **Replication does Not Support Fractional Replication**. Replication cannot be configured to replicate a subset of the attribute types from the replicated data set.

- **Replication does Not Support Sparse Replication**. Replication cannot be configured to replicate entries with a particular object class only.

- **Replication Requires Full Control of Replicated Data**. When two servers participate in replication, both servers implicitly trust each other and apply all updates received via replication, which is considered an internal operation. The trust between servers is established using public key cryptography.

Terminology

The following replication terms in Table 12-1 are used throughout this chapter.

TABLE 12-1. Replication Terms

Term	Description
Change Number	A 64-bit number used to consistently order replication updates across the entire topology. It is composed of 3 fields: the timestamp of the update (measured in milliseconds since 00:00:00 UTC, January 1, 1970), the Replica ID, and the Sequence Number.
Conflict	LDAP client updates made at different replicas affecting the same entry may be found in conflict when the updates are replayed at another replica. The Change Number of each update allows most of these conflicts to be resolved automatically. Certain updates, such as adding an entry with different attribute values at two servers simultaneously result in conflicts that are flagged for required manual resolution.

TABLE 12-1. **Replication Terms**

Term	Description
Eventual Consistency	Since replication is a background activity, recent updates from LDAP clients are not immediately present at all servers. Out-of-sync data will be eventually synchronized and will be consistent on all servers. The network latency typically controls how long a given update takes to replicate.
Global Administrator	The administrative user with full rights to manage the replication topology. The user is present at all directory servers participating in replication. The replication command-line utility expects the user name of the Global Administrator (by default, `admin`). The user is stored in `cn=admin,cn=administrators,cn=admin data`.
Historical Data	Historical Data are records of attribute value changes as a result of an LDAP Modify Operation.
Master Replica	A replica that may be updated by LDAP clients. This is the default replica behavior.
Modify Conflict	A conflict between two LDAP Modify operations. Conflicts arising from LDAP Modify operations are automatically resolved.
Multi-Master Replication	Replication between servers where more than one server may act as a Master Replica.
Naming Conflict	Any conflict other than Modify Conflicts. Naming conflicts typically include an operation that changes the DN of the target entry, creates an entry, or deletes an entry at one Replica.
Read-Only Replica	A replica that does not accept updates from LDAP clients. Only updates received via replication are processed. A replica can be configured as read-only by setting the writability mode of the corresponding backend accordingly.
Replica	Copy of the replicated data at a particular directory server instance.
Replica ID	The unique identifier of a replica that is set automatically in the corresponding Replication Domain configuration entry at each participating server. The replica ID identifies the source of each update in the replication topology.
Replica State	A list of the most recent change numbers of replication updates that have been applied to a replica. There can be at most one change number from each replica in the state. The replica state helps the Replication Server component to determine which updates the Replica has not received yet.
Replication	An automated background component that pushes directory data changes to any number of replica.

TABLE 12-1. **Replication Terms**

Term	Description
Replication Changelog	A backend maintained by each replication server independently that records updates from each replica. This backend is distinct from the LDAP Changelog and should not be confused with the Replication Changelog. The main distinction is as follows: • The LDAP Changelog (i.e., the external changelog that clients can access) physically resides at **`<server-root>/db/changelog`**. • The Replication Changelog Backend (i.e., the changelog that replication servers use) physically resides at **`<server-root>/changelogDB`**.
Replication Domain	The directory data configured for replication as defined by the base DN. Updates to entries at and below the base DN will be replicated.
Replication Replay	When a replica locally applies the update received via replication.
Replication Server	A component within the directory server process that is responsible for propagating directory data changes to and from replicas. Each directory server instance participating in replication is also running a replication server.
Replication Server ID	The unique identifier of a replication server set automatically in its configuration object at each server in the replication topology. This identifier is used in the connection management code of the replication servers.
Replication Server State	A list of the most recent change numbers of replication updates that have been processed by a replication server. There can be at most one change number from each replica in the state. The Replication Server State is used to determine which updates need to be sent to other replication servers. Similarly, the replica can use the Replication Server State to identify the set of updates to send to the replication server.
Sequence Number	A field in the change number that indicates the sequence of the LDAP client updates at a particular replica. For every update at the replica, the number is incremented. The initial value of the sequence number is 1. The number is stored as a 32-bit integer, but only positive values are used. The sequence number can roll over.

Replication Architecture

The major elements of replication in the UnboundID Directory Server are introduced in this section.

Eventual Consistency

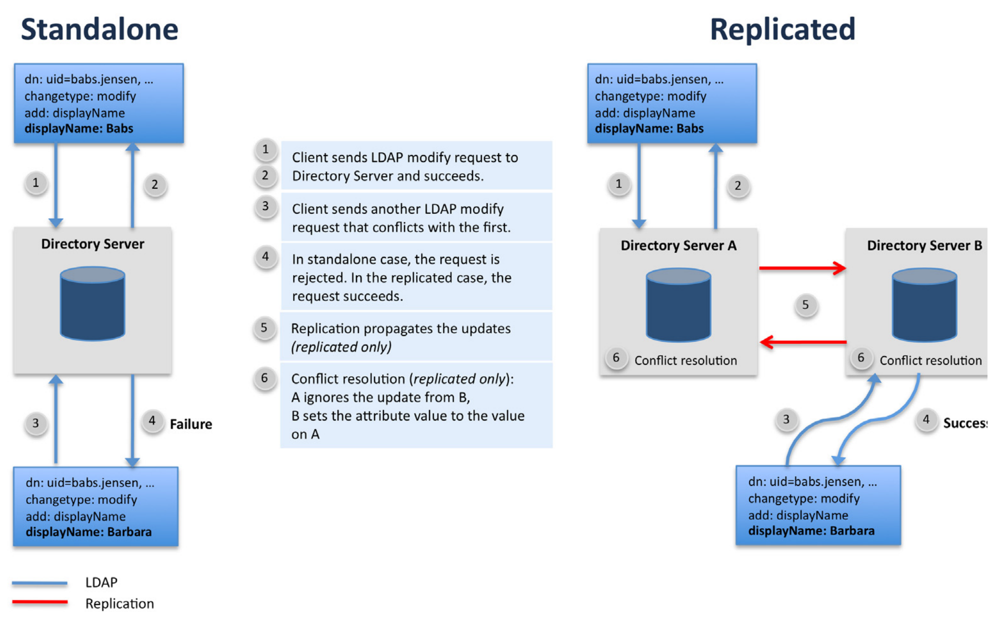
As shown in the Overview section, a client application updating the UnboundID Directory Server does not have to wait for replication to propagate the update, for example an LDAP Add, Delete, Modify, or Modify DN operation, to all other directory server instances in the topology. The Directory Server returns an LDAP response after the local processing completes. At the same time, replication propagates the update to the other directory servers. Replication does not lock target entries at other directory server instances before an update can be made locally. Replication neither enforces nor guarantees *strong* consistency of replicated data. This means that replicated directory servers may have an inconsistent view of the target entry for a short period of time. The eventual-consistency model also allows clients to complete update operations faster, since clients do not wait for replication to propagate the change. The rate of update operations also remains the same no matter how many directory servers participate in replication. This model enables servers to remain writeable in the presence of a network partition.

Conflict Resolution

The eventual-consistency model employed in replication introduces a window where conflicting updates targeting the same entry may be applied at two different directory servers. In general, two updates to the same directory server are in conflict if the update that arrived later fails. Conflict resolution, when possible, corrects conflicts introduced by clients automatically. There are some exceptions, however, when manual administrative action is required. For example, adding an entry in one replica and deleting the parent of this entry on another replica simultaneously will introduce a conflict that requires manual action. In a carefully implemented deployment, the risk of introducing conflicts that require manual action can be significantly reduced or even eliminated.

Consider the following example that results in a conflict: adding a single-valued attribute with different values to an entry concurrently at two directory servers. It is easy to see that the second operation would fail if a client attempted to add the same attribute to the same entry at the same directory server. In a replicated environment, the conflict is not immediately seen if these updates are applied concurrently at two different directory servers. The conflict is handled only after replication propagates the updates. The directory servers resolve the conflict independently. On one directory server, the entry will be updated to reflect the correct value; on the other directory server, the value will stay the same. Each directory server will independently resolve the conflict the same way based on the ordering of the updates. This example is illustrated in the Figure 12-2:

FIGURE 12-2. **Conflict Resolution**


Standalone
Replicated
dn: uid=babs.jensen, …
changetype: modify
add: displayName
displayName: Babs
1
2
Directory Server
3
4
Failure
dn: uid=babs.jensen, …
changetype: modify
add: displayName
displayName: Barbara
LDAP
Replication
1 2 Client sends LDAP modify request to Directory Server and succeeds.
3 Client sends another LDAP modify request that conflicts with the first.
4 In standalone case, the request is rejected. In the replicated case, the request succeeds.
5 Replication propagates the updates (replicated only)
6 Conflict resolution (replicated only): A ignores the update from B, B sets the attribute value to the value on A
Directory Server A
Directory Server B
5
6 Conflict resolution
6 Conflict resolution
3
4 Succes

The conflict resolution algorithm in the UnboundID Directory Server uses a mechanism that orders all updates in the replication topology. Each update in the Directory Server is assigned a unique change number. The change number is attached to each update propagated via replication and allows each directory server to order updates exactly the same way.

Change Number

A unique change number is assigned to each update operation for LDAP Add, Delete, Modify, or Modify DN operations when received from an LDAP client. The change number is composed of the following multiple fields:

- Timestamp that identifies when the update was made. The timestamp is time-zone-independent and recorded with millisecond resolution.
- Server ID that uniquely identifies the Directory Server where the update was made.
- Sequence number that defines the order of the updates received from external clients at a particular directory server.

The change number not only identifies each and every update, but it also allows ordering updates to the replicated data the same way on each directory server. Note that replication protocol sets a virtual clock that eliminates the need for synchronized time on servers. For troubleshooting purposes, however, it is still recommended to keep the system clocks synchronized.

Communication via the Replication Protocol

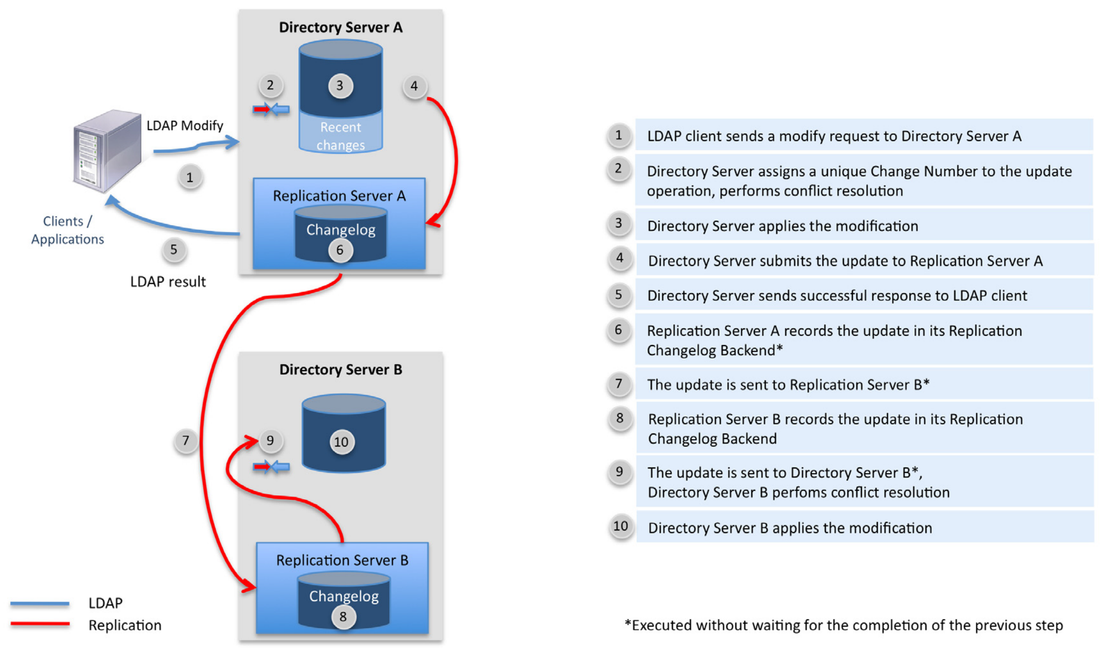
Replication communicates using a proprietary binary protocol that is implemented on top of the TCP/IP protocol. Some protocol messages are used for administrative purposes (for example, negotiation or flow control), others carry updates to replicated data. Communication may be secured using TLS. Protocol compression, which is enabled by default when directory servers are at different locations, further reduces the traffic created by replication.

Directory servers participating in replication are fully connected to each other. In a deployment with `n` servers, there are `n*(n-1)` connections in the replication topology per replicated data set. Each update is sent out on `n-1` of these connections. The fully connected mesh of directory servers allows quick propagation of updates to the replicated data and makes it easy to monitor the global status of replication. For example, the replication command line utility can display the status of replication by communicating with a single server. The replication command line utility facilitates adding a new Directory Server to an existing replication topology and configures all existing servers automatically. This way the administrator does not have to set up relationships between the new server and each server in the existing topology individually.

Directory servers keep connections open as long as possible in order to reduce the communication latency when messages are exchanged. Heartbeat messages are transmitted on a regular basis to detect a network failure or an unresponsive directory server as early as possible. Heartbeat messages also prevent idle connections from being closed by firewalls.

The following detailed communication flow will be used to describe major components of replication. This illustration is the expanded view of figure shown in the Overview section.

FIGURE 12-3. Replication Communication Flow



1	LDAP client sends a modify request to Directory Server A
2	Directory Server assigns a unique Change Number to the update operation, performs conflict resolution
3	Directory Server applies the modification
4	Directory Server submits the update to Replication Server A
5	Directory Server sends successful response to LDAP client
6	Replication Server A records the update in its Replication Changelog Backend*
7	The update is sent to Replication Server B*
8	Replication Server B records the update in its Replication Changelog Backend
9	The update is sent to Directory Server B*, Directory Server B perfoms conflict resolution
10	Directory Server B applies the modification

*Executed without waiting for the completion of the previous step

Step 1. Client sends an LDAP Modify request to Directory Server A.

Step 2. Directory Server A assigns a unique change number to the operation. Conflict resolution is executed to see if the LDAP Modify request is in conflict with the existing attribute types or values in the existing entry. The change number is assigned before the Directory Server backend is updated so that the arrival order of client requests can be preserved. Historical data in the target entry is updated to reflect the change. Note that historical data is only updated for LDAP Add and Modify operations.

Step 3. Directory Server applies the modifications in the corresponding backend. If the Directory Server process exits unexpectedly and some updates have not been passed to the Replication Server, the backend has the ability to recover the last 50,000 updates that were processed at this server, guaranteeing that these changes can be replicated when the server starts up. The Replication Server is a component within the Directory Server process responsible for propagating updates to and from the replication topology. The Directory Server itself only communicates with a single Replication Server, whereas the Replication Server component is connected to all the Replication Servers in the topology. The figure above also shows that replication protocol is used not just between Replication Servers but also between the Directory Server and the Replication Server. Authentication is also performed not just between Replication Servers but also between the Directory Server and the Replication Server.

Note that often the component in the Directory Server that handles replication outside the Replication Server is referred to as *Replica*.

Step 4. If the LDAP Modify operation successfully completes, then the Directory Server will submit the update to the Replication Server.

Step 5. The LDAP response is sent to the client. In this example, a successful response is assumed.

Step 6. The Replication Server records the update in its Changelog backend with the backend ID of `replicationChanges` and on disk with the path to `changelogDb` under the server root. The Replication Changelog backend keeps track of updates at each and every directory server in the replication topology. When a Directory Server joins the replication topology after being disconnected for some reason, updates from the Replication Changelog backend are re-sent to this Directory Server. Old records from the Replication Changelog backend are purged, which by default removes records older than 24 hours. If the backend does not contain all of the records that another directory server needed when rejoining the replication topology, then the replicated data set in the Directory Server must be re-initialized. In this case, the Directory Server enters lockdown mode and an administrative alert is sent.

Step 7. The Replication Server submits the update to the replication server component in Directory Server B. If there were more directory servers in this example, the Replication Server would submit the update to all the other replication servers.

Step 8. Just like in Step 6, the Replication Server component receiving an update inserts the change into its Replication Changelog backend.

Step 9. The update is forwarded to the Replica in Directory Server B. Conflict resolution is executed to see if the LDAP Modify request is in conflict with the existing attribute types or values in the existing entry.

Step 10. The Directory Server applies the modification in the corresponding backend. The Recent Changes database is not updated, because only updates that originated at this Directory Server are recorded in the Recent Changes database.

Authentication and Authorization

The authentication in the Replication Protocol is based on public key cryptography using client authentication via TLS. The certificate used for authentication is stored in the ads-truststore backend of the Directory Server. During replication setup, the command-line utility distributes public keys to all directory servers to establish trust between the directory servers and to enable client authentication via TLS.

The authorization model of replication is very simple: once authenticated, the remote directory server is fully authorized to exchange replication message with the local directory server. There is no other access control in place.

Logging

The access log messages in the Directory Server indicate if the update was received via replication and also include the corresponding change number. This allows the administrator to track which directory server the update originated from.

Deploying Replication

Deployments with replication require careful planning and management. Typically, deployments with replication will include the following major steps:

- **Planning**. In this stage the replication topology is designed with sufficient number of Directory Servers at the appropriate locations. UnboundID Directory Proxy Servers are also added to the topology if necessary – for example, in deployments with entry-balancing data sets. CPU, memory, disk, communication and availability requirements are all taken into account when planning the deployment. In order to minimize or avoid replication conflicts, special attention should be paid to the generation of LDAP update operations and whether load balancers are in use.

- **Initial Load**. The Directory Servers participating in replication are loaded from a source server using binary copy initialization, i.e., restoring from a backup, from a server acting as a data source. When adding a server to a replication topology, it is recommended to initialize the directory servers using the binary copy method.

- **Enabling Replication**. Replication is enabled between Directory Servers in the topology using the replication command line utility (`dsreplication`).
- **Monitoring Replication**. The status of replication (e.g. number of backlogged changes) is monitored using the replication command-line utility (`dsreplication status`). The Periodic Stats Logger plug-in may also be used to collect performance statistics from replication. For more information, see "Profiling Server Performance Using the Periodic Stats Logger" on page 484.

The rest of this section highlights important areas that should be considered when planning the deployment.

Initial Load

Replication allows backend databases to be populated in several ways. It is recommended to initialize directory servers using the binary copy method. Binary copy initialization involves backing up one or more backends at the source directory server and restoring them at the destination. It offers great advantages when working with multi-million entry backends.

Location

In multi-site deployments, it is strongly recommended to configure the Directory Servers with location information using the `dsconfig create-location` command. By default, replication will compress all traffic between Directory Servers in different locations. Location settings also drive the behavior of the fail-over mechanism both in replication and in the UnboundID Directory Proxy Server.

Uniform User Schema

Directory Servers participating in replication are required to have a uniform user schema. The replication command-line utility sets up replication for the schema backend the first time replication is enabled to ensure that future schema changes are propagated to all Directory Servers.

Disk Space

Replication increases the disk space required for the Directory Server.

The Replication Changelog backend keeps changes from all Directory Servers for 24 hours by default. After this time period, also known as the *purge delay*, the backend is trimmed automatically.

Changes older than 24 hours (by default) are removed only when a subsequent LDAP Modify updates the entry. In practice, the default 24-hour setting is hardly ever increased, because it provides plenty of time for conflicting LDAP Modify operations to propagate in the topology. The Directory Server stores the values of this attribute in a compact form to minimize the disk footprint.

The disk space impact of replication is highly dependent on the rate and size of changes in the replication topology, and the settings for the attribute history as well as Replication Changelog purge delay.

Memory

Compared to a standalone directory server, replicated directory server instances require slightly more memory. All of the items discussed in the Disk Space section have impact on the amount of memory the Directory Server is using. The additional replication overhead is typically less than 5%.

Time Synchronization

Even though replication has a built-in mechanism to compensate for the potential clock skew between hosts, it is generally recommended to keep system clocks in sync within the deployment.

Communication Ports

The replication server component in each directory server listens on a TCP/IP port for replication communication (the replication server port). This port must be accessible from all directory servers participating in replication. Even though a replica is only connected to a single replication server component, replicas poll the status of other replication servers when they start up or after losing connection to the replication server.

The communication channel is kept alive using custom protocol messages. By default, communication occurs every 10 seconds. This traffic will prevent firewalls from closing connections prematurely.

The replication command-line utility (`dsreplication`) requires access to all directory servers participating in replication. This includes the LDAP or LDAPS port of the directory servers.

Keep these communication requirements in mind when configuring firewalls.

Hardware Load Balancers

Replication allows writes to be directed to any directory server in the topology. Distributing write operations in a round-robin fashion, however, may introduce conflicts. In particular, distributing a series of LDAP Add, Delete and Modify DN operations targeting the same DN in quick succession will likely result in conflicts that require manual intervention.

If possible, consider using server affinity with the load balancer that either associates a client IP address or a client connection with a single directory server instance at a time. If server affinity is not an option, you may want to consider configuring the load balancer to direct write traffic to a single directory server only and use other directory server instances in case of a failure.

Alternatively, consider using the UnboundID Directory Proxy Server for load balancing LDAP traffic. The Server Affinity feature in the Proxy Server enables replication-friendly load balancing.

UnboundID Directory Proxy Server

In addition, to facilitate replication-friendly load balancing, the UnboundID Directory Proxy Server should be considered in every replication deployment. The Proxy Server can automatically adapt to conditions in backend directory servers using health checks and route traffic accordingly. For example, traffic can be re-routed from directory servers with large backlog of replication updates.

Replication Configuration

In a multi-master configuration, more than one master (read/write) directory server instance is configured for improved reliability and availability in the event of a server failure. Updates at any of the master servers will be propagated to the other servers by replication.

Overview

Administrators can configure replication using the `dsreplication` command-line utility, which is recommended for QA or production environments.

Replication setup involves the following basic steps:

1. **Set up the servers**. This is the basic installation steps to set up a directory server instance. See “Setting Up the Directory Server in Interactive Mode” on page 34 for information.

2. **Initialize, import, or copy the data into the replicas**. After setting up the servers, the replicas need to be populated with data from a source server.

3. **Enable replication between the servers**. This step can be done using the replication command-line tool, `bin/dsreplication`, which allows administrators to configure their replication topologies. See “Command Line Interface” on page 384 for more information.

4. **Verifying the replication topology**. Administrators can check the replication status after configuring the topology using the **`dsreplication status`** tool.

Note	The UnboundID Directory Server provides a Java-based GUI to quickly install and set up a replicated environment if your system supports graphical interfaces. This method should be used when quickly configuring replication for *evaluation* purposes with relatively small data sets. See “Setting Up a Replication Topology for Evaluation Purposes” on page 27.

Command Line Interface

Replication topologies are configured and maintained using the **`dsreplication`** command-line utility. For production environments, administrators can use the **`dsreplication`** tool in scripts using non-interactive mode. If you are running the server for the first time, we recommend using the **`dsreplication`** tool in interactive mode, which presents a command-line wizard that prompts you for the necessary input to set up replication.

The **`dsreplication`** tool has the following format including some important subcommands listed in the section “The dsreplication Command-Line Utility” on page 409:

```
dsreplication {subcommand} {connection parameters}
```

The tool prevents simultaneous updates to the replication configuration. The **`dsreplication`** tool keeps a history of invocations in the **`logs/tools/dsreplication.history`** file and keeps a log of up to 10 **`dsreplication`** sessions in the **`logs/tools`** directory.

What Happens When You Enable Replication Using the dsreplication Tool

The **`dsreplication enable`** subcommand is used to set up replication. The **`enable`** subcommand carries out the following functions:

1. If it does not already exist, the global administrator user is created. The global administrator user has all the rights and privileges to update replication-related configuration objects. Most **`dsreplication`** subcommands require the global administrator.

   The server instances are registered in the **`cn=admin data`** tree. The registration includes basic host name, port information as well as the public key used during the replication authentication process.

2. In case both servers are already participating in replication, the **`cn=admin data`** tree is merged to retain the server information from existing topologies.

3. The embedded replication server is enabled or updated in all servers. Servers already in replication will see their replication server configuration updated with the information of the new replication server. A new server added to an existing topology will get the replication server enabled.

4. The replication domain configuration for the requested base DNs are enabled or updated. In case the first base DN is enabled, the replication domains for two additional base DNs are

also enabled: **`cn=admin data`** and **`cn=schema`**. Servers already in replication will see the replication domain configuration updated with the information of the new replication server. A new server added to an existing topology will see the replication domains enabled.

5. Initialization for the **`cn=admin data`** base DN is executed. This will ensure that **`cn=admin data`** is uniform across the replication topology.

6. Initialization for the **`cn=schema`** base DN is executed. This will ensure that a uniform schema is present in the replication topology.

About Initializing Replication Data

After setting up your directory server instances, you must initialize the newly-configured replicas with data. You can follow one of the following strategies to configure your replication topology and initialize it with data when using the binary copy methods, the **`import-ldif`** tool, or the **`dsreplication initialize`** command-line tool:

- **Binary Copy**. The Binary Copy method involves copying database files from the source directory server to one or more target servers. The Directory Server provides tools necessary for backing up and restoring backends. Using **`<server-root>/bin/backup`**, create a backup of the backend containing the replicated base DN. The backup files then need to be transferred to the target server(s) and restored individually with **`<server-root>/bin/restore`**. This method is generally the fastest method for loading data onto the target replicas for datasets over 1 GB. Binary copy is also the preferred data initialization method in cases when the backend containing the replicated base DN, such as userRoot, contains no data for other base DNs, and the index settings are identical on the target server(s). Two additional points about binary copy are as follows:

    - If encryption is enabled for the backend containing the replicated base DN, then a backup must also be taken of the **`encryption-settings`** backend. Note that in order to preserve existing encryption settings, **`<server-root>/bin/restore`** will append to the **`encryption-settings`** database as opposed to replacing it.

    - When initializing a server which has been offline longer than the replication-purge-delay, the backup and restore of the **`replicationChanges`**, **`schema`**, and **`adminRoot`** backends are required.

- **export-ldif/import-ldif.** In this approach, the replicas are initialized with data while the server is offline locally, or online through remote access. This method is flexible and works well in environments where the replicas are configured differently with respect to the backend that contains the data (for example, the target replica has different indexes than the source server). Note that this process is considerably slower than the binary copy method if your directory has a large number of entries. Run the **`export-ldif`** command to export the data from the first directory server to an LDIF file, and then use **`import-ldif`** to import the data into the new replica. This method should only be performed on the backend containing the replicated Base DN, and is required when index configuration differs across servers. Using `<server-root>/bin/export-ldif`, export the data from the source Directory Server server to an LDIF file, transfer the LDIF file to the target server(s) and then use **`<server-root>/bin/import-ldif`** to import it into the appropriate backend. If a target server has been offline for longer than the replication-purge-delay, the binary copy method should be used to initialize the `replicationChanges`, `schema`, and `adminRoot` backends as well as the `encryption-settings` backend if encryption is enabled.

If you can run the command online from a remote server, you must include the connection parameters (fully-qualified host name or IP address, LDAP or LDAPS listener port, `bindDN` and password). See "About Initializing Replication Data" on page 385 for more information.

- **dsreplication initialize-all/initialize.** In this approach, the servers in the replicated topology are initialized with data while the server is online. Run the `dsreplication initialize-all` (or `dsreplication initialize` for a single replica) command using the first server as the data source to initialize data on the replicas. Special care should be used with this subcommand as it overwrites any existing data in the destination replicas. This method requires that you enable replication before running this command.

  This method is recommended for proof-of-concept evaluations, or databases with small data sets (less than 1 GB). UnboundID does not recommend this initialization method in a production environment with very large data sets due to the possible extended time it takes to load the data. See "About Initializing Replication Data" on page 385 for more information.

Basic Replication Deployment

This section describes how to set up a two-server multi-master replication topology. The procedure assumes that both servers have not participated in any previous replication topology. Also, the procedure uses the online initialization method for initializing data into the replicas, which is recommended for *small databases only (<1g)*.

Deployment Plan

The following procedures assume that both server instances are built from scratch and are solely dedicated as directory servers. The example uses the LDAP (1389, 2389) and replication (8989, 9989) ports respectively. Note that both servers could use the LDAP listener port 389 and the replication port 8989 but are using different ports for clarity. The first Directory Server will be populated with 2,000 users from the sample data. Replication will be used to initialize the data set on the second Directory Server. In this example, replication will not encrypt protocol messages.

TABLE 12-2. Replica Ports

Host Name	LDAP Port	Replication Port
server1.example.com	1389	8989
server2.example.com	2389	9989

To Configure the Basic Replication Deployment

1. Install the first directory server with 2000 sample entries.

```
$ setup --cli --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389 \
  --rootUserPassword pass --sampleData 2000 --no-prompt
```

2. Install the second directory server without any data.

```
$ setup --cli --acceptLicense --baseDN "dc=example,dc=com" --ldapPort 1389 \
  --rootUserPassword pass --no-prompt
```

3. Run the **`bin/dsreplication`** command in interactive mode to configure a replication topology. From the first server, run the command:

```
$ bin/dsreplication
```

4. From the Replication Main menu, type the number corresponding to what you want to run. In this example, enter the number to manage the topology. The command-line wizard guides you through the steps to enable replication between two servers.

```
>>>> Replication Main Menu

What do you want to do?

    1)  Display replication status
    2)  Manage the topology (add and remove servers)
    3)  Initialize replica data over the network
    4)  Initialize replica data manually
    5)  Replace existing data on all servers}

    q)  quit

Enter choice: 2
```

5. On the Manage Replication Topology menu, review the menu options, and then enter the number to enable replication on the server.

```
>>>> Manage Replication Topology

Select an operation for more information.

    1)  Enable Replication -- add or re-attach a server to the topology
    2)  Disable Replication -- permanently remove a running server from the topology
    3)  Detach -- temporarily isolate servers from the topology
    4)  Remove Defunct Server -- permanently remove an unavailable server from the
        topology
    5)  Cleanup Server -- remove replication artifacts from an offline, local server
        (allowing it to be re-added to a topology)

    b)  back
    q)  quit

Enter choice [b]: 1
```

6. On the Enable Replication menu, read the brief introduction on what will take place during the setup, and then, enter "c" to continue the enable process.

Set up the LDAP Connection Parameters for the First Replica

7. Next, enter the LDAP connection parameters for the two replicas that you are configuring. First, enter the host name or IP address of the first server.

8. Next, enter the type of LDAP connection to the first server: 1) LDAP, 2) SSL, or 3) Start-TLS.

9. Type the LDAP listener port for the first replica. If you are a root user, you will see port 389 as the default. Others will see port 1389.

10. Next, enter the Global Administrator ID or bind DN for the first replica. When you first set up a replication topology, you will be entering the directory server's root user bind DN (e.g., **`cn=Directory Manager`**) and bind DN password.

    You will be prompted later in the process to set up a global administrator and password. The global administrator is the user ID that manages the replication topology group.

Set up the LDAP Connection Parameters for the Second Replica

11. Repeat steps 6–9 for the second replica shown in the previous section.

Set up the Global Administrator and Base DNs

12. Next, the **`dsreplication`** tool checks for the base DN on both servers and asks if you want to replicate one. Replication cannot be enabled between servers that have empty (0 entry) base DNs.

    ```
    You must choose at least one base DN to be replicated.
    Replicate base DN dc=example,dc=com? (yes / no) [yes]:
    ```

Note	If you see the following message: `There are no base DNs available to enable replication between the two servers.` In most cases, one or both of the directory servers are not online, or a base DN was not set up on one of the directory servers.

13. Next, type **`no`**, since we are not setting up replication in an entry-balanced environment in this scenario. For more information, see the *UnboundID Directory Proxy Server Administration Guide*.

14. Next, enter the replication port for the first replica (default, 8989).

15. Next, press Enter or Return to accept the default (no). You have finished specifying the connection parameters for the first replica.

16. Repeat steps 13–14 for the second directory server.

17. At this time, set up the Global Administrator user ID (default is "**`admin`**") and a password for this account. The Global Administrator user ID manages the directory servers used in the replication topology.

18. Next, the **`dsreplication`** tool will test the connections to both servers and then update the registration information on the second server with information from the first server. The schema is also updated on the second server with the schema from the first server.

Check the Status of the Replication Topology

19. At this stage, replication has been successfully enabled and both servers have been registered in the replication topology. However, the data has not been initialized on the second server. To verify this, on the Manage Replication Topology menu, enter b to go back to the main menu.

20. On the Replication Main Menu, enter the number to check the status of replication. When prompted, enter the LDAP connection parameters (host name or IP address), LDAP connection type (LDAP, LDAP with SSL, LDAP with StartTLS), and the LDAP port. Then, enter the Global Administrator user ID and password.

    The following status message gives a warning indicating that there are multiple generation IDs detected on each suffix. This indicates that the data on the second replica has not been initialized from the first replica.

```
          --- Replication Servers: dc=example,dc=com ---
Server                   : Server ID : Status    : Generation ID : Security:
-------------------------:-----------:-----------:---------------:----------
server1.example.com:8989 : 20919     : Available : 2656017928    : Disabled
server2.example.com:9989 : 23728     : Available : 14458739      : Disabled


          --- Replication Status for dc=example,dc=com: Enabled ---
Server                 : Entries : Backlog : Oldest Backlog Change Age : Generation ID : Repl Svr ID
-----------------------:---------:---------:---------------------------:---------------:----------
server1.example.com:1389 : 1003    : 0       : N/A                       : 2656017928    : 20919
server2.example.com:2389 : 6       : 0       : N/A                       : 14458739      : 23728

WARNING:  Multiple Generation IDs were detected for this suffix.  Replication uses Generation
IDs to ensure consistency between replica servers by validating that each has started with the
same version of the data.  Changes made at one server are not replicated to another server if it
has a different Generation ID.  A list of servers in this topology and their Generation IDs fol-
low.  Refer to the documentation for the best way to initialize these instances to bring their
Generation IDs in sync.
```

To Initialize the Basic Replication Dataset

Since the dataset in this scenario is relatively small, the **`dsreplication`** tool can be used to initialize data over the replication protocol.

1. On the Replication Main Menu, enter the number corresponding to initializing data over the network.

2. On the Initialize Replica Data over the Network menu, select the number to initialize data on a single server, and then press c to continue.

3. Next, specify a server in the replication topology. For this example, enter the host name or IP address, LDAP connection type, LDAP port, Global Admin user ID and password of the first server.

4. Next, select the source server that is hosting the data to which the target server will be initialized. For this example, select the first server, since the sample dataset has been loaded onto this server.

5. Next, select the base DN that will be initialized. In most cases, the base DN for the root suffix will be replicated. In this example, **`dc=example,dc=com`**.

6. Next, select the second server in this example that will have its data initialized, and then enter the Global Admin user ID and password for the target server. Any data present on the target server will be over-written.

7. Press Enter to confirm that you want to initialize data on the target server. When completed, you should see "Base DN initialized successfully."

```
Initializing the contents of a base DN removes all the existing contents of that
base DN.  Do you want to remove the contents of the selected base DNs on server
server2.example.com:2389 and replace them with the contents of server server1.exam-
ple.com:1389? (yes / no) [yes]:
```

8. On the Initialize Replica Data Over Network menu, enter b to back out one level to the main menu. Then, on the Replication Main menu, enter the number to view the replication status. Here we can see that the generation IDs and the number of entries match, which indicates that the two replicas are identical and ready for replication.

```
          --- Replication Servers: dc=example,dc=com ---
Server                   : Server ID : Status    : Generation ID : Security:
-------------------------:-----------:-----------:---------------:----------
server1.example.com:8989 : 20919     : Available : 2656017928    : Disabled
server2.example.com:9989 : 23728     : Available : 2656017928    : Disabled

          --- Replication Status for dc=example,dc=com: Enabled ---
Server                   : Entries : Backlog : Oldest Backlog Change Age : Generation ID : Repl Svr ID
-------------------------:---------:---------:---------------------------:---------------:----------
server1.example.com:1389 : 1003    : 0       : N/A                       : 2656017928    : 20919
server2.example.com:2389 : 1003    : 0       : N/A                       : 2656017928    : 23728
```

Replication Deployment with Two Data Centers

This section describes how to set up a four-way multi-master replication topology in a deployment with two data centers. The procedure assumes that none of the servers participated in any previous replication topology. Also, this procedure uses the binary copy method for initializing data in the replicas. The binary copy method involves backing up databases at a source server and restoring them at the destination servers.

Deployment Plan

Distinguishing two data centers is accomplished by setting up distinct locations in the directory server instances prior to enabling replication. There will be two directory server instances present at each location. This way, each data center will provide high-availability to local clients in case of a network partition between the data centers.

In this example, LDAPS is enabled at all directory server instances, which allows communication from the `dsreplication` command-line utility to occur securely. The `dsreplication` examples will use the non-interactive mode.

Replication will encrypt protocol messages as well since data centers may be connected over public WAN links.

By default, replication compresses all communication between locations. There is not need to configure compression explicitly.

The Directory Server instances will be initialized with one million entries of sample data. The dataset will be imported from LDIF into the first server, and the other servers will be initialized using the binary copy method.

The table 12-3 summarizes the host information used in this example.

TABLE 12-3. Replica Ports

Host Name	Location	LDAP Port	LDAPS Port	Replication Port
server1-aus.example.com	Austin	1389	1636	8989
server2-aus.example.com	Austin	2389	2636	9989
server1-bud.example.com	Budapest	3389	3636	10989
server2-bud.example.com	Budapest	4389	4636	11989

To Implement the Replication Deployment with Two Data Centers

1. Install all four directory server instances. The example below shows the **setup** command-line example for the first server. To keep this example simple, a self-signed certificate is used for the server. The server instance is not started after the setup is complete. Aggressive JVM tuning is enabled and the maximum heap size is capped at 4 GB.

```
$ setup --cli --acceptLicense \
  --baseDN "dc=example,dc=com" \
  --ldapPort 1389 --ldapsPort 1636 \
  --generateSelfSignedCertificate \
  --aggressiveJVMTuning \
  --maxHeapSize 4g \
  --rootUserPassword pass \
  --doNotStart --no-prompt
```

2. Generate the sample data using the **make-ldif** tool on the first server.

```
$ bin/make-ldif --templateFile config/MakeLDIF/example-1M.template \
  --ldifFile ldif/1m.ldif
```

3. Import the LDIF into the first server.

```
$ bin/import-ldif --backendID userRoot --ldifFile ldif/1m.ldif
```

4. Since the binary copy method is used to initialize the dataset on the other servers, back up the first server. The backup will be compressed, so that the copy of the backup files to the other site will take less time and bandwidth.

```
$ bin/backup --backendID userRoot --backupDirectory bak \
  --backupID initial --compress
```

5. Copy or transport the backup files from the backup directory to the other servers, or make the backup directory accessible from the network.

6. Use the **restore** command to initialize the dataset on the other directory server instances. Assuming that the backup files are in the **bak** directory and the backup ID is set to initial, you can use the following command:

```
$ bin/restore --backupDirectory bak --backupID initial
```

7. Start all the directory server instances using **`bin/start-ds`**.

8. Configure the location, Austin, for the first and second server, and Budapest for the third and fourth servers. The example for the Austin location is as follows:

```
$ bin/dsconfig create-location --location-name Austin
$ bin/dsconfig set-global-configuration-prop --set location:Austin
```

9. Enable replication between the first and second server using the **`dsreplication enable`** subcommand (the command-line example shows this operation). Next, enable replication using the first and the third as well as using the first and fourth servers in the enable arguments. Every time **`dsreplication enable`** is executed (except the first invocation), the tool is actually adding one server into the replication topology and updates the configuration of all existing servers in the topology with the information about the new replica.

```
$ bin/dsreplication --host1 server1-aus.example.com \
  --port1 1636 --useSSL1 \
  --bindDN1 "cn=Directory Manager" \
  --bindPassword1 pass \
  --replicationPort1 8989 --secureReplication1 \
  --host2 server2-aus.example.com \
  --port2 2636 --useSSL2 \
  --bindDN2 "cn=Directory Manager" \
  --bindPassword2 pass \
  --replicationPort2 9989 --secureReplication2 \
  --baseDN "dc=example,dc=com" \
  --adminUID admin --adminPassword pass --no-prompt
```

10. Check the status of replication.

```
$ bin/dsreplication status --hostname server1-aus.example.com --port1 1636 \
  --useSSL --adminUID admin --adminPassword pass \
  --baseDN "dc=example,dc=com" --no-prompt
```

11. Set the prime method to preload the **`userRoot`** backend in each directory server instance, which loads the data in the cache prior to startup. Startup will take longer, but the response time of the Directory Server will be lower once it has started.

```
$ bin/dsconfig set-backend-prop --backend-name userRoot --set prime-method:preload
```

Advanced Configuration

The following sections are advanced configuration procedures that may be appropriate for your company's deployment.

Adding a Single Server to an Existing Topology

For databases over one gigabyte in size, it is recommended that initialization be completed using the binary copy method *before* enabling replication using the **`dsreplication enable`** command. This sequence of steps ensures that any existing data on the new server is not replicated to the existing topology.

If you are using the **dsreplication** tool in interactive mode, select the number corresponding to the "Initialize replica data manually" on the Replication Main menu, then enable replication on the new server.

To Add a Single Server to an Existing Topology for Databases Over 1 GB

1. Install the directory server instances in your replication topology in “To Configure the Basic Replication Deployment” on page 386.

2. Make sure that the new replica has the following items:

   a. Copy the required keystore and truststore files to the **<server-root>/config** directory.

   b. Copy your schema file definitions (for example, **99-user.ldif**) to the **config/schema** directory. This step is optional.

   c. Verify that the server has the correct configuration settings, indexes, password policies as the other servers. If you have a **dsconfig** batch file that has your configuration settings, you can run it again on the new server.

3. If the Directory Server participated in a previous replication topology, delete the old **replicationChanges** backend on the new server. This backend contains the changes that may have occurred in a different topology.

   ```
   $ bin/dsreplication cleanup-local-server --changelogDir replicationChanges
   ```

4. Use the **backup** tool to back up the replicated backends from the source server to the new server. The source server can be online or offline. The following example runs the backup offline and backs up the userRoot backend to a corresponding subdirectory in the **bak** directory.

   ```
   $ <source-server-root>/bin/backup --backendID userRoot -d bak/userRoot
   ```

5. Use the **backup** tool to back up the **replicationChanges** backend on the source server.

   ```
   $ <source-server-root>/bin/backup --backendID replicationChanges -d bak/replica-
   tionChanges
   ```

6. If the replication topology uses encrypted attributes, you must copy over the **encryption-settings** backend, which holds pointers to the **encryption-settings** definition files. If you are not implementing encrypted attributes, you can skip this step.

   ```
   $ <source-server-root>/bin/backup --backendID encryption-settings -d bak/encryp-
   tion-settings
   ```

7. Copy the **bak** directory to the new server.

   ```
   $ cp -r <source-server-root>/bak <new-server-root>
   ```

8. On the new server, restore the replicated the backends. The following commands are run with the server offline.

```
$ <new-server-root>/bin/restore -d bak/userRoot
```

9. Restore the `replicationChanges` backend from the source replica to the new replica.

```
$<new-server-root>/bin/restore -d bak/replicationChanges
```

10. On the new server, restore the `encryption-settings` backend if the replica implements encrypted attributes and/or sensitive attributes. If you are not implementing encrypted attributes, you can skip this step.

```
$ <new-server-root>/bin/restore -d bak/encryption-settings
```

11. Start the new server, and then run `dsreplication enable` using an existing replica in the topology as `host1` and the new server as `host2`. The host names should be the fully qualified host name or IP address of the source server and the target server that you are adding to the topology. In the following example, enable `server3` in the existing topology.

```
$ bin/dsreplication enable --host1 server1.example.com --port1 1389 \
  --bindDN1 "cn=Directory Manager" --bindPassword1 secret1 \
  --replicationPort1 8989 --host2 server3.example.com --port2 3389 \
  --bindDN2 "cn=Directory Manager" --bindPassword2 secret2 \
  --replicationPort2 10989 --adminUID admin --adminPassword password \
  --baseDN dc=example,dc=com --no-prompt
```

12. View the replication status to verify the addition.

```
$ bin/dsreplication status --hostname server3.example.com --port 3389 \
  --adminPassword password --no-prompt
```

Removing Replicas from a Topology

A directory server in a replication topology may need to be replaced for hardware or software upgrades or for other factors. The UnboundID Directory Server provides four subcommands with the `dsreplication` tool that should be used to remove a replica from a replication topology depending on its status:

- **detach**. Used when the server needs to be removed temporarily from the replication topology for maintenance or diagnosis without disabling replication. Once re-enabled, it will resume replication as if only a network partition occurred between the detached server and the rest of the replication topology. Replication will send all updates accumulated to and from the detached server as long as the replication changelog purge delay has not been exceeded. If the purge delay has been exceeded, you need to re-initialize the re-enabled replica.

- **disable**. Used when the server is accessible and available, the `disable` subcommand removes a replica from the replication topology. When the last replication domain is disabled in the replica, then the embedded replication server is disabled as well. If you use `dsreplication disable` targeting an offline server, the command will fail.

- **remove-defunct-server**. Used when the server is completely inaccessible, the `remove-defunct-server` subcommand removes replication-related artifacts from the configuration, schema and server registry, and removes the references to this server on other replicas and replication servers in the topology. See the Troubleshooting section for further discussion.

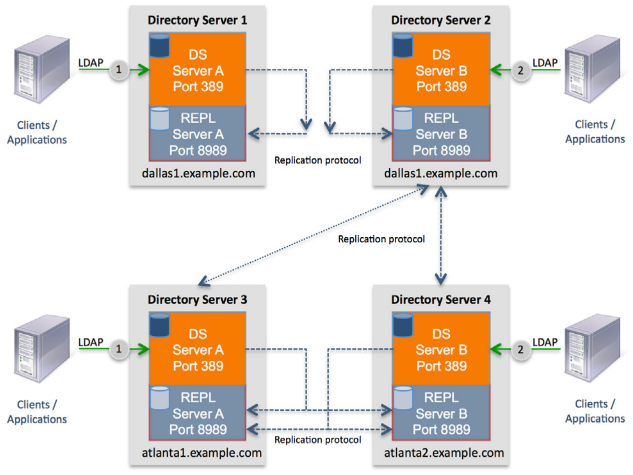
- **cleanup-local-server**. Used when the server is offline, the **`cleanup-local-server`** subcommand removes replication-related artifacts from the configuration, schema and server registry while the local server is offline. See the Troubleshooting section for further discussion.

Detaching a Replica

Administrators may encounter situations where they need to detach a single server or set of servers from the replication topology without disabling replication on the isolated server or servers. This "detach" feature is useful for diagnosing a problem temporarily without taking the server out of the topology. Once re-enabled, it receives updates from the other replication servers if the purge delay has not been exceeded. If the purge delay has been exceeded, you need to re-initialize the re-enabled replica.

For example, given four directory servers, one set (**`set1`**) consists of directory servers 1 and 2, located in Dallas; the other set consists of directory servers 3 and 4, located in Atlanta (seen in Figure 12-4), you can detach a single server from the replication topology using the **`dsreplication detach`** command. You should use the **`detach`** subcommand only if directory server will be added back to the replication topology. Otherwise, consider using the **`disable`** subcommand instead.

FIGURE 12-4. **Example of a Replica Detach Operation**



Note that directory server 1 still has replication enabled even though it no longer participates in the replication topology of servers 2 through 4. Updates made on directory server 1 will be recorded in the changelog database of the embedded replication server instance. When directory server 1 is added back to the topology, these updates will be propagated to the other servers. Changes made on directory servers 2 through 4 while directory server 1 was detached will also be applied.

To Detach a Single Server

1. To detach only server 1 from the replication topology but leave set1 connected, use the following command:

```
$ bin/dsreplication detach --hostname server1.example.com --port 1389 \
  --adminUID admin --adminPassword password
```

2. Verify that the replica was detached using **dsreplication status**.

To Detach a Replication Set

In deployments where two replication sets exist (for example, set1 and set2) and replication is taking place between the two sets, you can detach a replication set from the other. *Replication sets* are use primarily for entry-balancing proxy server deployments.

1. Use the **dsreplication** command with the **detach** subcommand to detach the directory instances in the replication set **set1** from the replication topology. After you run the command, replication no longer occurs between the two sets (**set1** and **set2**), though replication continues to take place within each of the two sets.

```
$ bin/dsreplication detach --hostname server1.example.com --port 1389 \
  --adminUID admin --adminPassword password --setname set1
```

2. Verify that the replica was detached using **dsreplication status**.

Disabling Replication on a Directory Server

You can remove a directory server from a replication topology by disabling replication on that server using the **dsreplication disable** command. The command disables replication on each of the base DNs present on the system (i.e., **dc=example,dc=com**, **cn=admin data**, and **cn=schema**). If the replica is offline, the **disable** subcommand fails.

To Disable Replication on a Server

1. Verify that the server that you want to disable is online.

2. Use **dsreplication disable** to remove a server from a replication group.

```
$ bin/dsreplication disable --no-prompt --port 2389 \
  --hostname server3.example.com --adminUID admin \
  --adminPassword password --baseDN dc=example,dc=com
```

3. Check the status of replication to see that the server is disabled.

```
$ bin/dsreplication status --hostname server3.example.com --port 3389 \
  --adminPassword password --no-prompt

          --- Replication Status for dc=example,dc=com - Enabled ---
Server                  : Entries : Replication Backlog : Age of Oldest Backlog Change : Port :
```

```
Security
------------------------:---------:--------------------:----------------------------:------
:---------
server1.example.com:1389 : 2003    : 0                  : N/A                        : 8989 :
Disabled
server2.example.com:2389 : 2003    : 0                  : N/A                        : 9989 :
Disabled

          --- Replication Status for dc=example,dc=com - Disabled ---
Server                   : Entries
-------------------------:--------
server3.example.com:3389 : 2003
```

Reloading Globally on All Replicas in an Existing Topology

If the entire set of replicas requires the re-importing of the replicated base DN, you should run the **pre-external-initialization** subcommand prior to initializing data on the first server, and then run the **post-external-initialization** subcommand to reset the generation ID on all of the replicas after all servers have been initialized.

To Reload Globally on All Replicas in an Existing Topology

1. Stop all of the replica servers.

2. Run the **dsreplication pre-external-initialization** command on one of the replicas. This command clears the generation IDs as well as the replication changelogs for the selected base DNs in the topology.

3. Reload the replicated base DNs in all replicas using, for example, the **import-ldif** command.

4. Run the **dsreplication post-external-initialization** command on one of the replicas. This command resets the generation IDs in the topology.

```
$ bin/dsreplication status --hostname server.example.com --port 1389 \
  --adminPassword password --no-prompt
```

Changing the replicationChanges DB Location

You can change the **replicationChanges** DB location if on-disk space issues arise. The replication changelog database can live outside **<server-root>** and be placed in another location on the filesystem. In that case, you must specify the absolute path of the replication changelog directory.

To Change the replicationChanges DB Location

1. Use **dsconfig** to change the database location for the replication changelog, which by default is at **<server-root>/changelogDb**. The following command sets the replication changelog backend to **<server-root>/replicationDB**. Remember to include the LDAP connection parameters (**hostname**, **port**, **bindDN**, **bindPassword**).

```
$ bin/dsconfig set-replication-server-prop \
--provider-name "Multimaster Synchronization" \
--set "replication-db-directory:/data/directory/changelogDb" \
--bindDN "cn=Directory Manager" \
--bindPassword secret --no-prompt
```

2. Stop and restart the server.

```
$ bin/stop-ds
$ mv changelogDb /data/directory
$ bin/start-ds
```

Modifying the Replication Purge Delay

The replication purge delay specifies the period after which the directory server purges changes on the replication server database. Any change that is older than the purge delay is removed from the replication server database regardless of whether the change has been applied.

Currently, the UnboundID Directory Server sets the default purge delay to one day. Administrators can change the default purge delay using the `dsconfig` tool. To ensure proper replication processing, you must have the same purge delay value set for all replication servers in the topology.

To Change the Replication Purge Delay

Use `dsconfig` to change the purge delay. The property accepts time units of seconds (`s`), minutes (`m`), hours (`h`), days (`d`), or weeks (`w`). The following command is entered on the command line and changes the purge delay from the default one day to two days.

```
$ bin/dsconfig set-replication-server-prop --provider-name "Multimaster Synchroniza-
tion" --set "replication-purge-delay:2 d"
```

Note	In `dsconfig` interactive mode, open the Advanced objects menu. Select Replication Server. Select the replication synchronization provider, and then select the option to change the replication purge delay.

Working with Replication and Large Attributes

By default, the Directory Server does not automatically replicate the Large Attributes backend if configured. If you have set up large attributes on a master server, then you must explicitly create a Large Attributes backend on the other replicas in the topology, then run a binary copy to restore the backend on those servers. All other variables will be replicated as normal. Once you have set up the backend, any subsequent changes to the Large Attribute attribute will be properly synchronized to the other replicas.

Note	Historical data is not maintained for large attributes. As a result, replication is unable to resolve conflicting LDAP modify operations involving large attributes.

Replicating Large Attributes

If a Large Attribute backend is enabled and assigned to the backend containing your replication domain, then there are several replication related aspects to know. The Large Attribute backend does not require separate replication configuration, but the Large Attribute backend must be manually copied, as an additional step, using the binary copy method, whenever initializing a replica. The binary copy of the Large Attribute backend can occur immediately before or after the initialization of the parent backend, e.g. userRoot.

To Replicate Large Attributes

1. On one of the directory server instances, configure Large Attributes on the server. See "Working with Large Attributes" on page 190 for more information.

2. Set up replication between the directory server instances. See "Replication Configuration" on page 383.

3. On the first server, run **`bin/backup`** to save a copy of the Large Attributes backend. Send the backup by FTP to the other servers or make it accessible over your network.

```
$ bin/backup --backendID Large-Attributes --backupDirectory /path/to/backups/large-
Attr
```

4. On the second replica, create the Large Attributes backend.

```
$ bin/dsconfig create-backend --backend-name Large-Attributes \
  --type large-attribute --set enabled:true \
  --hostname server2.example.com --port 2389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret --no-prompt
```

5. On the second replica, stop the directory server, restore the backend, and then restart the server.

```
$ bin/restore --backupDirectory /path/to/backups/largeAttr
```

6. Repeat steps 4–5 for the other replicas in the topology.

Configuring Locations

Location settings allow directory server instances to be labelled according to their physical location. By default, it is assumed that directory server instances at the same location are on the same LAN.

To allow replicas to select replication servers based on their locations, all replication server locations should be configured on *each* replica. The Directory Server connects to its embedded Replication Server (see the **`prefer-local-connection`** configuration property in the corresponding replication domain configuration object using **`dsconfig`**). If the local connection preference is disabled, highest preference is given to replication servers in the same location. If no replication server is available in the current location, then the servers in the preferred failover locations are contacted in the order specified in the location settings. The global configuration of each replica should be set to the appropriate location.

To Set Up Locations and Preferred Failover Locations

The replication protocol, by default, is compressed between directory server instances at different locations. This applies to both Replica-to-Replication Server as well as Replication Server-to-Replication Server connections. The example below shows how to set up a replica in the Atlanta location. This example assumes that the preferred locations for the Atlanta location are Dallas and Chicago (in this order).

1. Use the **`bin/dsconfig`** tool with the **`create-location`** subcommand. You need to first define the locations.

   ```
   $ bin/dsconfig create-location --location-name Atlanta
   $ bin/dsconfig create-location --location-name Dallas
   $ bin/dsconfig create-location --location-name Chicago
   ```

2. On the Atlanta master server, set the location property using **`bin/dsconfig`**.

   ```
   $ bin/dsconfig set-location-prop --location-name Atlanta \
     --add preferred-failover-location:Dallas \
     --add preferred-failover-location:Chicago
   ```

3. On the Atlanta master server, set the global configuration property.

   ```
   $ bin/dsconfig set-global-configuration-prop --set location:Atlanta
   ```

4. Repeat steps 1–3 for the Dallas and Chicago servers, respectively.

Configuring a Single Listener-Address for the Replication Server

The replication system requires that the fully-qualified host names be defined to resolve addresses. However, in some configurations, such as in virtualized environments, the server's host name may not be represented as such. One workaround is to configure a single listener-address for a replication server, so that it resolves to the correct address.

By default, the replication servers bind their listening ports to all local addresses using a wild card IP address. The Directory Server's bind command also can create a listening socket that allows for a specific address to be used by setting a configuration property, **`listen-on-all-addresses`**.

The replication server's configuration entry already stores a host name for itself so that it can resolve the address and specify it during the socket bind. If the server information is missing from the system, an error message will be generated with instructions on specific address binding. You can use the **`dsconfig`** tool to change the value of the **`listen-on-all-addresses`** property from **`TRUE`** (default) to **`FALSE`**.

To Configure a Replication Server to Listen on a Single Address

1. Create a new Directory Server instance with replication enabled on port `8989`.

2. Run `netstat` to see the ports bound for listening on port `8989`. Notice that `*.8989` means that it is listening on all addresses.

   ```
   $ netstat -an | grep LISTEN | grep 8989
   ```

3. Run `dsconfig` to disable listening on all addresses for the replication server.

   ```
   $ bin/dsconfig set-replication-server-prop --provider-name "Multimaster Synchroni-
   zation" --set listen-on-all-addresses:false
   ```

4. Run `netstat` again to see the ports bound for listening on port `8989`. Notice that `<address>.8989` (for example, `10.8.1.211.8989`) means that it is listening on the one address.

Managing Replication Compression

The UnboundID Directory Server supports data compression for remote networked replication traffic. Compression greatly reduces the number of transmitted bytes during replication and optimizes transmission bandwidth with a minimal performance hit on the replication rate over WAN networks. For replication compression to work, administrators must define locations for each replica to determine if the replica or replication server is local or remote. By default, a replica connects to its embedded replication server (i.e., the server that runs in the same JVM process). As a result, you will not see compression on the "replica-to-replication server" connections at the directory server instance even if compression is enabled.

Note	Readers may want to stop here if they are happy with the default settings.

The Directory Server provides two types of configuration settings for replication compression:

- **Replication Server Settings**. Sets compression on the replication server per instance.

- **Replication Domain Settings**. Sets the compression on each replication domain per instance: the userRoot DN (e.g., `dc=example,dc=com`), `cn=admin data`, and `cn=schema`.

Both types of settings are independent of the other and do not override the other configuration option. An example is presented later in this section.

Note	By default, all replication traffic between directory server instances at different locations are compressed. If this is the desired behavior, then no other configuration is necessary as long as each directory server instance has the appropriate location configured.

Compression Settings

The Directory Server's replication mechanism has three important settings that determine how a client (i.e., the replica or replication server) requests compression on a replication connection: compression criteria, compression mechanism, and compression level.

Note	Replicas are always clients in replication communication, because they initiate the connection to a replication server. Replication servers may also be clients when establishing connection to other replication servers. There is only one connection (per replication domain) between two replication servers: one of them will act as the client, the other will be the server side.

Compression Criteria

The `compression-criteria` parameter determines under what conditions compression should be used for replication. By default, the `compression-criteria` is set to "remote," which specifies that compression be used when the communicating parties are at different sites (determined by the `location` property). Administrators can use the `dsconfig` tool to set the compression criteria to one of the following values:

- **never**. Compression is disabled.
- **remote**. (Default). Compression is enabled only when the communicating parties are at different locations. Administrators must set the server locations for each replica in order to use this feature. Compression is always disabled for any replica communicating with a replication server in the same VM process.
- **always**. Compression is always enabled between remote communicating parties, except between a replica and a replication server running in the same VM process.

Note	In order for compression to be enabled between two servers, the compression criteria and the compression level must be set identically on both systems. They must also not be part of the same VM process.

Compression Mechanism

Currently, the Directory Server supports compression based on the ZIP compression algorithm. Future builds may add other compression algorithms.

Compression Parameter

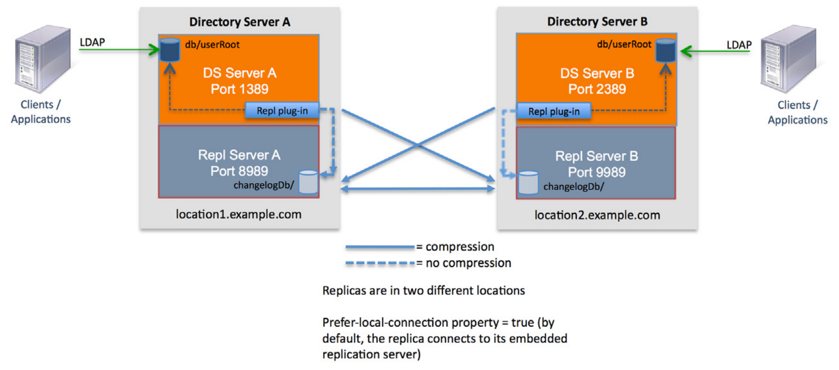
The compression parameter specifies the size of the buffer used for the compression algorithm. The compression levels can be set to one of nine possible numerical values from "1" (fastest but lowest compression) to "9" (slowest but best compression). The default compression setting level is 6.

Compression Examples

The following four use cases provide examples on using compression for replication:

1. Compress replication only if the replica or the replication server connects to a remote replication server. In this scenario, use the default settings. This scenario assumes that location is configured on each directory server in the topology.

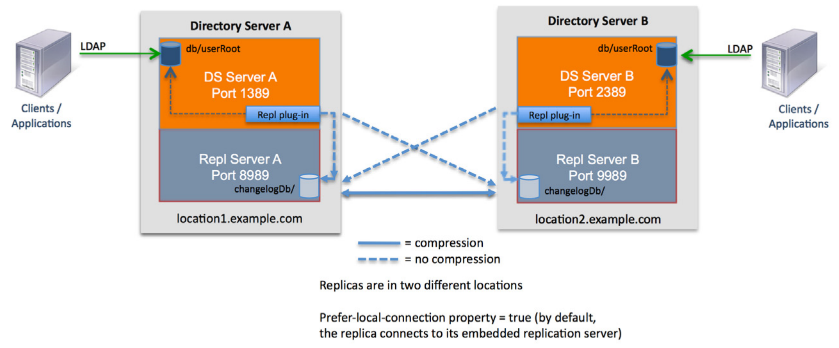
FIGURE 12-5. Default Compression Scenario



2. Compress replication communication between remote replication servers, but never compress replica-to-replication server communication. This scenario assumes that location is configured on each directory server in the topology.

   - Set the compression-criteria to "never" in every replication domain configuration entry at all servers.

FIGURE 12-6. Compression between Remote Replication Servers Only

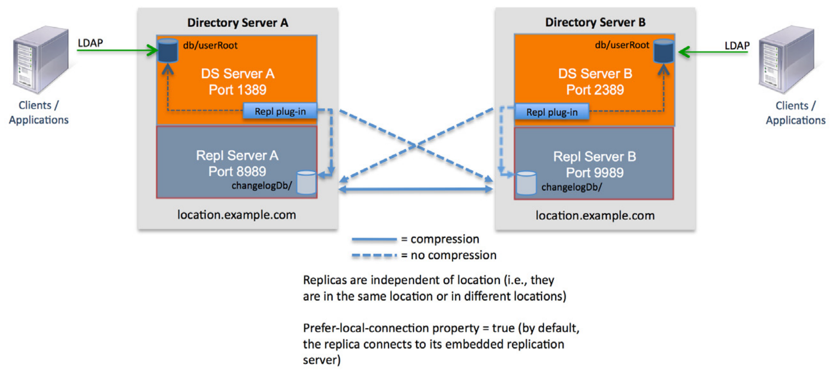


3. Always compress replication between replication servers (independent of location) but never compress replica-to-replication server communication.

   - Set the compression-criteria to "always" in the replication server configuration of each directory server.

- Set the compression-criteria to "never" in every replication domain configuration entry at all servers.

FIGURE 12-7. **Compression between Replication Servers Only (Location Independent)**

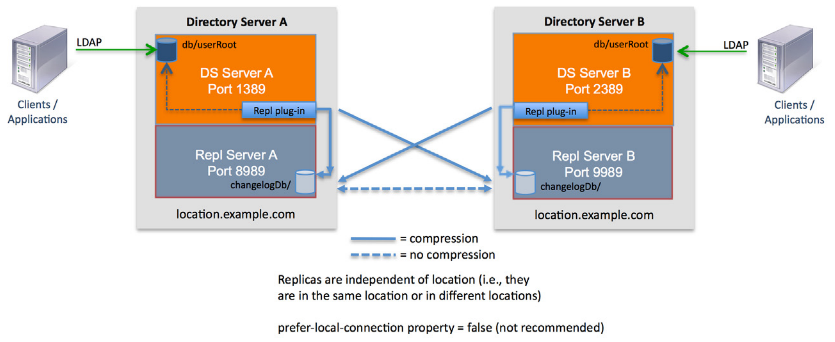


4. Never compress replication between replication servers (independent of location), but always compress replica-to-replication server communication. (This is only meaningful if the `prefer-local-connection` replication domain configuration setting is set to `FALSE`, which is not recommend due to the possible negative impact on replication flow).

   - Set the compression-criteria to "never" in the replication server configuration of each directory server.

   - Set the compression-criteria to "always" in every replication domain configuration entry at all servers.

FIGURE 12-8. **Compression between Replica to Replication Server Only (Location Independent)**



Enabling and Disabling Replication Compression

The following procedures show how to enable replication compression and disable it if necessary.

To Enable Replication Compression

1. Ensure that the two replicas are not in the same VM process.

2. Set the server location for both replicas. See “Configuring Locations” on page 399 for information.

3. Use the **dsconfig** tool to enable compression on each replication domain on your system. The following example changes the **compression-criteria** property from its default "remote" mode to "always". The **compression-parameter** property is set to "4" for each replication domain on the system. Remember to include the LDAP or LDAPS connection parameters (**hostname**, **port**, **bindDN** and **bindPassword**) with each **dsconfig** command.

   The first three commands below sets the properties on the Replication Domain for each base DN. The last command sets the Replication Server properties.

```
$ bin/dsconfig set-replication-domain-prop --provider-name "Multimaster Synchroni-
zation" --domain-name "cn_admin-data" --set compression-criteria:always --set "com-
pression-parameter:compression-level=4"

$ bin/dsconfig set-replication-domain-prop --provider-name "Multimaster Synchroni-
zation" --domain-name "cn_schema" --set compression-criteria:always --set "compres-
sion-parameter::compression-level=4"

$ bin/dsconfig set-replication-domain-prop --provider-name "Multimaster Synchroni-
zation" --domain-name "dc_example_dc_com" --set compression-criteria:always --set
"compression-parameter::compression-level=4"

$ bin/dsconfig set-replication-server-prop --provider-name "Multimaster Synchroni-
zation" --set compression-criteria:always --set "compression-parameter::compres-
sion-level=4"
```

4. Test the resulting bandwidth over your WAN networks. Compression statistics are available on the **ds-replication-server-handler-monitor-entry** objects under **cn=monitor**. The entry also provides a count of the number of messages sent and received, regardless if compression is enabled or not.

To Disable Replication Compression

1. Use **dsconfig** to set the **compression-criteria** property to "never". Remember to include the LDAP or LDAPS connection parameters (**hostname**, **port**, **bindDN** and **bindPassword**) with each **dsconfig** command.

```
$ bin/dsconfig set-replication-server-prop \
  --provider-name "Multimaster Synchronization" --set compression-criteria:never
```

2. Repeat the process on the replication domains for each base DN. Remember to include the LDAP or LDAPS connection parameters (**hostname**, **port**, **bindDN** and **bindPassword**) with each **dsconfig** command.

```
$ bin/dsconfig set-replication-domain-prop --provider-name "Multimaster Synchroni-
zation" --domain-name "cn_admin_data" --set compression-criteria:never

$ bin/dsconfig set-replication-domain-prop --provider-name "Multimaster Synchroni-
zation" --domain-name "cn_schema" --set compression-criteria:never
```

```
$ bin/dsconfig set-replication-domain-prop --provider-name "Multimaster Synchroni-
zation" --domain-name "dc_example_dc_com" --set compression-criteria:never
```

Monitoring Replication

Replication in the UnboundID Directory Server can be monitored the following ways:

- The `dsreplication status` subcommand displays basic information about the replicated base DNs, the number of entries replicated as well as the approximate size of replication backlogs at each Directory Server.
- The more detailed information about the state of replication can be obtained via the information exposed in its monitoring framework under `cn=monitor`. Directory administrators can monitor their replication topologies using several tools and protocols: SNMP, LDAP, JMX, or through the Directory Management Console. See “Managing Logging and Monitoring” on page 443.
- The Periodic Stats Logger plug-in allows collecting replication statistics for profiling server performance. See “Profiling Server Performance Using the Periodic Stats Logger” on page 484 for more information.

Monitoring Replication Using cn=monitor

The `cn=monitor` branch has a number of entries that store the replication state of a topology.

- **Direct LDAP Server <baseDN> <host name:port> <serverID>**. Defines an LDAP server that is directly connected to the replication server that you are querying. The information in this entry applies to the replication server local to the `cn=monitor` entry. For detailed information, see “Summary of the Direct LDAP Monitor Information” on page 427.
- **Indirect LDAP Server <baseDN> <serverID>**. Defines an LDAP server that is connected to another replication server in the topology. While this server is connected to the same topology, it is not connected to the replication server being queried. For detailed information, see “Summary of the Indirect LDAP Server Monitor Information” on page 429.
- **Remote Repl Server <baseDN> <host name:port> <serverID>**. Defines a remote replication server that is connected to the local replication server. Information in this entry is in respect to the Replication Server local to the `cn=monitor` branch. For detailed information, see “Summary of the Remote Replication Server Monitor Information” on page 430.
- **Replica <baseDN>**. Stores information for an instance of the replicated naming context—also known as the *replica*—with respect to the Directory Server and its communication with a replication server. The Replica information is what is responsible for sending and

receiving changes from the replication servers. For detailed information, see "Summary of the Replica Monitor Information" on page 432.

- **Replication Server <replPort> <serverID>**. Shows the information specific to the Replication Server running, for example, on the replication port <replPort> with a server ID of <serverID>. This entry defines the replication server. For detailed information, see "Summary of the Replication Server Monitor Information" on page 433.
- **Replication Server Database <baseDN> <serverID>**. Shows information for the changelog table of replica (suffix) in the replication server. As the Replication Server receives updates from the Directory Server, it records those changes in the changelog table. For detailed information, see "Summary of the Replication Server Database Monitor Information" on page 434.
- **Replication Server Database Environment <baseDN>**. Shows the information for the database environment for the replication server backend plus the total number of records added to and deleted from the database. For detailed information, see "Summary of the Replication Server Database Environment Monitor Information" on page 434.
- **Replication Summary <baseDN>**. Shows summary information on the replication topology and the state for a particular base DN. For detailed information, see "Summary of the Replication Summary Monitor Information" on page 440.
- **Replication Changes Backend**. Shows the backend information for all replication changes. For detailed information, see "Summary of the replicationChanges Backend Monitor Information" on page 440.
- **Replication Protocol Buffer**. Shows the state of the buffer (initially 4k) for protocol operations, which is stored in thread local storage. For detailed information, see "Summary of the Replication Protocol Buffer Monitor Information" on page 441.

Replication Best Practices

The following best practices tips are recommended for the various aspects of replication deployment that we have found in useful in actual production environments.

Data Initialization

General

When setting up replication from scratch, it is recommended to load the data into the Directory Servers *prior* to enabling replication using binary copy initialization if the index configuration is the same, or using **`export-ldif/import-ldif`** if the index configuration is different from the source server.

The index configuration of directory servers will be typically different when setting up a read-only replica in addition to master servers. If the deployment does not have any read-only replicas, binary copy initialization is the best method for data initialization.

Binary copy initialization involves backing up the appropriate databases at a source Directory Server, copying the backup files to the target Directory Server(s), and restoring the databases. See “Replication Deployment with Two Data Centers” on page 390 for an example.

Online Initialization

Initialization over the replication protocol (using **`dsreplication initialize/initialize-all`**) is not recommended for databases larger than 1 GB.

Empty Base DNs

The **`dsreplication`** utility will not enable replication if the base DN in both servers is empty (the base DN has no entries, not even the base DN entry).

Cache Priming

When initializing the replication topology with large number of entries (more than 10 million), it is worth considering disabling cache priming until replication is completely set up and verified that replication is working as expected. Specifically, it is recommended to omit the **`--primeDB`** option when setting up the Directory Server that will be initialized with a large dataset. Once replication is enabled and working as expected, the cache priming can be re-enabled as follows:

```
$ bin/dsconfig set-backend-prop --backend-name userRoot --set-prime-method:preload
```

Security

Global Administrator

The Global Administrator user is created when replication is enabled the first time. Authentication using this administrative user is possible at any of the Directory Servers in the topology. The Global Administrator user entry is also replicated to other servers in the topology.

It is recommended to set a non-trivial password for the Global Administrator user. The Global Administrator has important privileges that can alter the replication topology, such as updating the server configuration, schema, dataset initialization, and others.

Since the **`dsreplication`** command-line utility requires communication with other directory servers in the topology, the credentials of the Global Administrator are transmitted over the network. Unless the directory servers in the topology can be accessed over secure LDAP communication (LDAPS or LDAP with StartTLS), the credentials will be transmitted in the clear. In deployments where directory servers communicate over insecure lines (for example, unsecured WAN link), it is recommended to enable LDAPS or LDAP with StartTLS on all directory servers.

Replication Encryption

The encryption in replication is configured on a per Replication Server port basis. If the communication in replication needs to be encrypted, then encryption on all replication server ports should be enabled.

The dsreplication Command-Line Utility

Developing Scripts

The **`dsreplication`** utility maintains the history of executed **`dsreplication`** commands with the full command-line arguments in the **`logs/tools/dsreplication.history`** file. The recorded commands may be used to develop scripts to set up and configure replication.

Scripts invoking the **`dsreplication`** utility in non-interactive mode should check the return code from the **`dsreplication`** process. A non-zero return code indicates some sort of failure.

If output messages from the **`dsreplication`** utility are not desired, use the **`--quiet`** option to suppress them.

The utility, by default, fails if one or more warnings are issued during the command execution. Warnings can be suppressed using the **`--ignoreWarnings`** option. For example, this option is required when using **`dsreplication`** with non-fully-qualified hostnames (for example, localhost), otherwise **`dsreplication`** will fail. In production environments, use of this flag is strongly discouraged.

The **`dsreplication`** utility also provides an **`--ignoreLock`** option that specifies that the tool should continue processing in non-interactive mode or in scripts even if the replication topology has been locked by another tool invocation. This tool should be used with caution.

Concurrent Use

With the exception of the **`dsreplication status`** subcommand, the **`dsreplication`** subcommands cannot be executed concurrently. The command-line utility locks the replication topology at one or more servers to prevent accidental configuration changes caused by multiple **`dsreplication`** subcommands running at the same time. It is best to avoid concurrent configuration changes in general.

Status

The **`dsreplication status`** subcommand requires the Replication Servers to provide monitoring information. This can lead to a delay before the output of **`dsreplication status`** is displayed. By default, **`dsreplication`** will display the status for all replicated domains (with the exception of the special domains of the schema and the server registry).

It is recommended to select a particular base DN for **`dsreplication status`** if multiple base DNs are configured for replication.

It is also recommended to avoid invoking **`dsreplication status`** too often (more than once every 15 seconds) or from multiple locations at the same time. Some of the information displayed by **`dsreplication status`** is based on monitor information that is not refreshed every time the monitor is queried.

The **`status`** subcommand should not be used for collecting performance metrics. It is best to rely on replication-related information captured by the Periodic Stats Logger Plug-in.

Topology Management

Adding a Previously Replicating Directory Server into the Topology

If the Directory Server participated in a different replication topology, then use the **`dsreplication cleanup-local-server`** command to remove replication-related artifacts from the configuration, schema, server registry as well as to remove the replication changelog database.

```
$ bin/dsreplication cleanup-local-server --changelogDir changelogDb
```

Topology Changes in Mixed Version Environments

In general, topology changes (adding or removing a replica) in deployments where multiple versions of the Directory Server are present should be avoided. Preferably, topology changes should be performed prior to upgrading all Directory Server instances or after upgrading all directory server instances in the topology.

Replication with version 2.1.x and any other version of the directory (e.g., 2.2.x) is not supported. This means that topology changes using **`dsreplication`** and the **`disable`** subcommand are not supported across the boundary of Directory Server 2.1.x. For example, running **`dsreplication disable`** is not supported when 2.1.2 and 2.2.2 servers are in the same topology, but **`dsreplication disable`** is supported when the topology contains 2.2.2 and 3.0.1 servers. In the latter case, topology changes using **`dsreplication enable`** and **`disable`** should be performed using the **`dsreplication`** tool with the older version (i.e., on the 2.2.2 server).

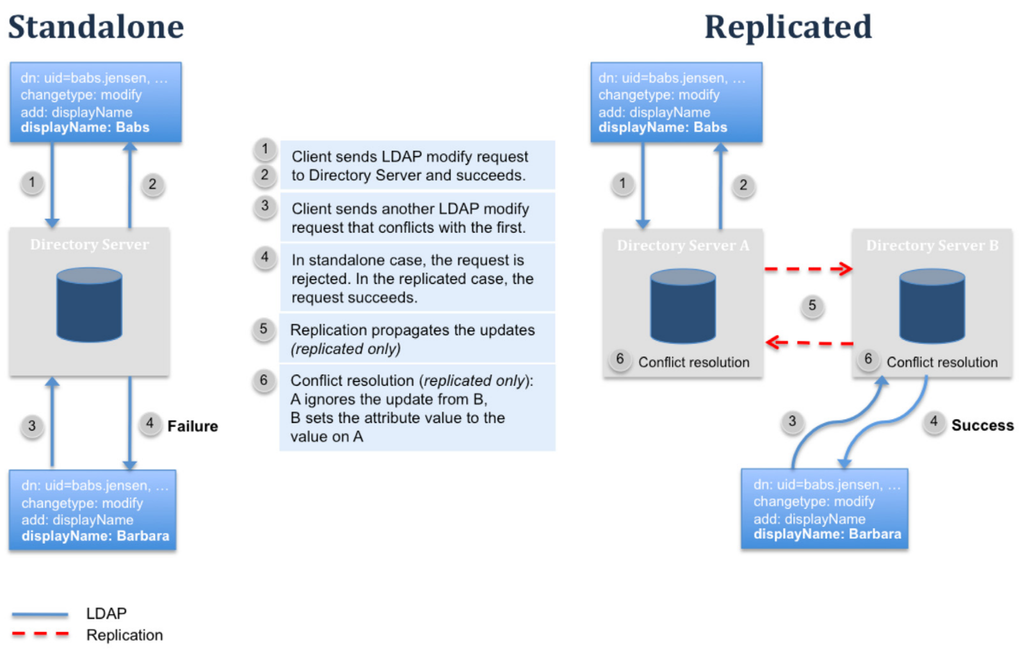
Replication Conflicts

This section provides more in-depth information on replication conflicts than presented in earlier sections, so that administrators can understand the mechanisms and possible scenarios behind these conflicts.

Updates to directory entries in a replication topology may happen independently, since replication guarantees only eventual consistency, not strong consistency. The eventual consistency model also means that conflicting changes can be applied at different directory server instances. In most cases, the Directory Server is able to resolve these conflicts automatically and in a consistent manner (i.e., all directory server instances in a replication topology will resolve each and every conflict the same way). However, in some scenarios, as seen below, manual administrative action is required. For any of these unresolved conflicts, the administrator is notified via administrative alerts.

On a high-level, the conflict resolution algorithm tries to resolve conflicts as if the operations causing the conflict in a distributed environment has been applied to a single directory server instance. For example, if the same entry is added to two different directory server instances at about the same time, then once these operations have been replicated, both Directory Servers will keep only the entry that was added first. The following figure highlights the differences between standalone versus replicated environments.

FIGURE 12-9. **Conflicting Operations in Standalone versus Replicated Environments**


Standalone
Replicated
dn: uid=babs.jensen, …
changetype: modify
add: displayName
displayName: Babs
Directory Server
Directory Server A
Directory Server B
dn: uid=babs.jensen, …
changetype: modify
add: displayName
displayName: Barbara
Failure
Success
Conflict resolution
1 2 Client sends LDAP modify request to Directory Server and succeeds.
3 Client sends another LDAP modify request that conflicts with the first.
4 In standalone case, the request is rejected. In the replicated case, the request succeeds.
5 Replication propagates the updates (replicated only)
6 Conflict resolution (replicated only): A ignores the update from B, B sets the attribute value to the value on A
LDAP
Replication

Types of Replication Conflicts

There are fundamentally two types of replication conflicts: naming and modification conflicts. Naming conflicts include operations that cause conflicts with the naming (DN) of the existing or new entries, while modification conflicts include operations that result in conflicts in the modification of attributes.

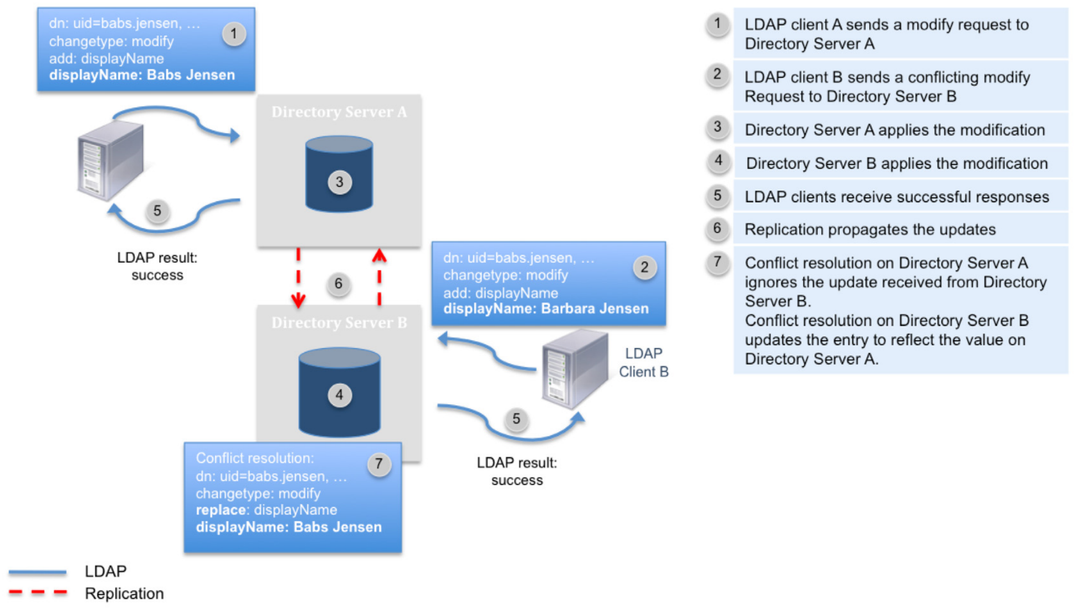
Naming Conflict Scenarios

For all of the naming conflict scenarios in the table below, assume the following:

- Update 1 was applied at Directory Server 1
- Update 2 was applied at Directory Server 2
- Update 1 occurred shortly before Update 2, so that Directory Server 2 received Update 1 **after** Update 2 was applied

The naming conflict scenario is illustrated in the following figure:

FIGURE 12-10. **Naming Conflicts**


dn: uid=babs.jensen, …
changetype: modify
add: displayName
displayName: Babs Jensen
1
Directory Server A
3
5
LDAP result: success
6
Directory Server B
4
dn: uid=babs.jensen, …
changetype: modify
add: displayName
displayName: Barbara Jensen
2
LDAP Client B
5
LDAP result: success
Conflict resolution:
dn: uid=babs.jensen, …
changetype: modify
replace: displayName
displayName: Babs Jensen
7
LDAP
Replication
1 LDAP client A sends a modify request to Directory Server A
2 LDAP client B sends a conflicting modify Request to Directory Server B
3 Directory Server A applies the modification
4 Directory Server B applies the modification
5 LDAP clients receive successful responses
6 Replication propagates the updates
7 Conflict resolution on Directory Server A ignores the update received from Directory Server B. Conflict resolution on Directory Server B updates the entry to reflect the value on Directory Server A.

The following table shows the result of a modification conflict depending on the type of updates that occurs:

TABLE 12-4. **Naming Conflict Scenarios**

Update 1	Update 2	Automatic Resolution?	Result of Conflict Resolution at Directory Server 2 When Update 1 is received
Modify	Delete	Yes	Modify is discarded.
Modify	Modify DN	Yes	New entry is located based on the entryUUID and Modify is applied to the renamed entry.
Delete	Delete	Yes	Delete operation is ignored.
Delete	Modify DN	Yes[a]	Delete operation is applied to the renamed entry.
Delete of A	Add of B under A	Yes[a]	Entry B is renamed and Entry A is deleted.
Modify DN	Delete of the same entry tar-geted by the Modify DN	Yes	Modify DN operation is ignored.
Modify DN with a new parent	Delete of parent	No	The entry targeted in the modify DN operation is marked as a conflict entry.

TABLE 12-4. **Naming Conflict Scenarios**

Update 1	Update 2	Automatic Resolution?	Result of Conflict Resolution at Directory Server 2 When Update 1 is received
Modify DN with a new parent	Modify DN of the parent	Yes	The entry will be moved under the new DN of the parent.
Modify DN of A with new DN B	Modify DN of C with new DN B	No[a]	A and B will be conflict entries.
Add A	Modify DN of the parent of A	Yes	The entry is added under the new DN of the parent.
Add A	Delete of the parent of A	No[a]	The added entry is marked as a conflict entry.
Add A	Add A with same set of attributes	Yes	The entryUUID of the incoming Add operation applied to the existing entry.
Add A	Add A with different set of attributes (or values)	No	The existing entry is marked as a conflict entry, and the incoming Add is executed.

a. The code does not compare change sequence numbers (CSNs) but applies operations in the order they were received. This may lead to inconsistent replays.

Modification Conflict Scenarios

Modification conflicts are always resolved automatically and no manual action is required. The LDAP Modify operation allows the following modification types:

1. Add of one or more values
2. Delete of one or more values or the entire attribute
3. Replacement of all values

Replication does not currently support the increment LDAP modification type.

For all of the operations in the table below, assume the following:

- LDAP Modify 1 was applied at Directory Server 1
- LDAP Modify 2 was applied at Directory Server 2
- LDAP Modify 1 occurred shortly before LDAP Modify 2, so that Directory Server 2 received LDAP Modify 1 **after** LDAP Modify 2 was applied.

The modification conflict scenario is illustrated in the figure 12-11.

FIGURE 12-11. Modification Conflicts

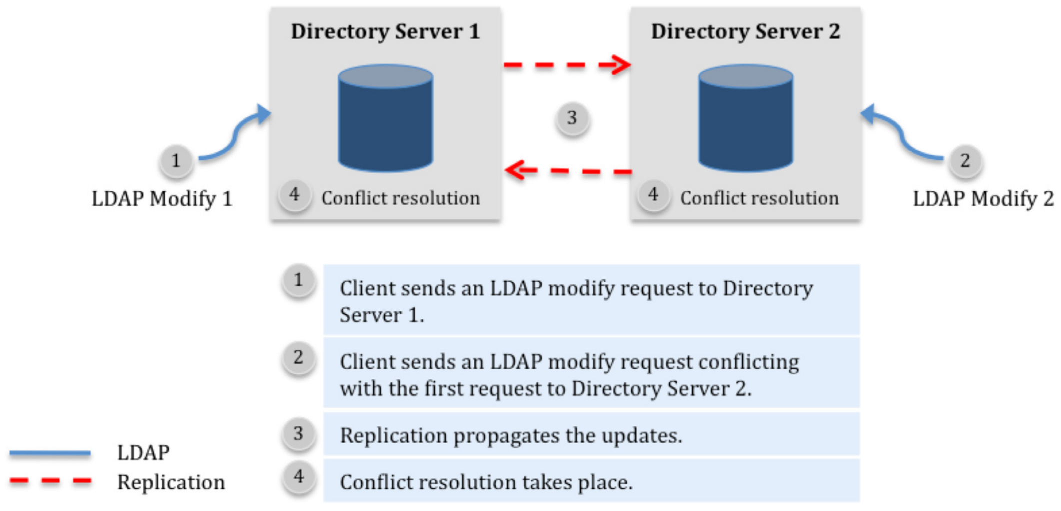

Directory Server 1
Directory Server 2
1
LDAP Modify 1
2
LDAP Modify 2
3
4 Conflict resolution
4 Conflict resolution
1 Client sends an LDAP modify request to Directory Server 1.
2 Client sends an LDAP modify request conflicting with the first request to Directory Server 2.
3 Replication propagates the updates.
4 Conflict resolution takes place.
LDAP
Replication

TABLE 12-5. Modification Conflict Scenarios

Modify 1	Modify 2	Result of Conflict Resolution at Directory Server 2 When Modify 1 is received
Add of a single-valued attribute	Delete of the same attribute	Attribute remains deleted.
Delete of a single-valued attribute	Add of the same attribute	Incoming Delete is ignored.
Add of a single-value attribute	Add of the same attribute with a different value	Existing value is replaced with the value in the incoming operation.
Delete of a single-valued attribute	Replacement of the value of the same attribute	Incoming Delete is ignored.
Replacement of a single-valued attribute	Delete of the same attribute	Incoming Replacement is ignored.
Delete some values from a multi-valued attribute	Delete of the same multi-valued attribute	Incoming Delete is ignored.
Delete a multi-valued attribute	Delete of the same multi-valued attribute	Incoming Delete is ignored.
Delete a multi-valued attribute	Add the same multi-valued attribute	Incoming Delete is ignored.
Delete value X from a multi-valued attribute	Delete value X from the same multi-valued attribute	Incoming Delete is ignored.
Delete value X from a multi-valued attribute	Add value Y to the same multi-valued attribute	Delete of value X is applied.

TABLE 12-5. **Modification Conflict Scenarios**

Modify 1	Modify 2	Result of Conflict Resolution at Directory Server 2 When Modify 1 is received
Delete value X from a multi-valued attribute	Delete value Y from the same multi-valued attribute	Delete of value X is applied.
Delete value X from a multi-valued attribute	Replace all values of the same multi-valued attribute	Incoming Delete is ignored.
Add of values X and Y to a multi-valued attribute	Delete value X from the same multi-valued attribute	Only value Y is added.
Delete value X from a multi-valued attribute	Add of values X and Y to the same multi-valued attribute	Incoming Delete is ignored.

Troubleshooting Replication

The following sections provide information to troubleshoot your replication deployment.

Recovering One Replica in an Existing Topology

If a replica becomes out-of-sync and requires re-initialization, it is not necessary to disable replication. You can perform a re-initialization with the out-of-sync replica online using `dsreplication initialize` if the database is of manageable size (under one gigabyte) and if the network throughput between servers allows the transfer within minutes. If the replica is missing changes that are older than the replication purge delay set on the rest of the servers within the topology, then a full restore is necessary based on the binary copy method. The copying and restoring of the `userRoot`, `replicationChanges`, `adminRoot`, `schema`, and possibly the `encryption-settings` backends are equivalent to the action taken by `dsreplication initialize` but is a faster restore process for very large databases.

For example, if a change has occurred to `cn=admin data` during the time the replica is offline and the time period for the replication-purge delay has passed, a recovery operation will result in an out-of-sync `cn=admin data` on the recovered replica, which will result in the server going into lockdown mode. The solution is to restore the user data, `changelog`, `schema` and `adminRoot`, and `replicationChanges` backends from another replica. If encrypted attributes are present, you will need to restore the `encryption-settings` backend. You can restore the backends using the `restore` tool.

To Recover One Replica in an Existing Topology due to an Out-of-Sync Database

1. Stop the disabled replica.

```
$ bin/stop-ds
```

2. Back up the userRoot, adminRoot, schema, replicationChanges backends from another replica using the binary copy method. The backup tool can be used to back up all backends on the source replica. Optionally, export-ldif can be used for the userRoot backend if the replica being recovered has different index settings. In this case, the import-ldif tool must be used to restore the replica's userRoot. For example, you can use the **`backup`** tool to back up the replicated backends on the source replica.

```
$ <source-server-root>/bin/backup --backendID userRoot -d bak/userRoot
$ <source-server-root>/bin/backup --backendID adminRoot -d bak/adminRoot
$ <source-server-root>/bin/backup --backendID schema -d bak/schema
$ <source-server-root>/bin/backup --backendID replicationChanges -d bak/replica-
tionChanges
$ <source-server-root>/bin/backup --backendID encryption-settings -d bak/encryp-
tion-settings  (if encrypted attributes are present)
```

3. Copy the **`bak`** directory to the new replica.

```
$ scp -r <source-server-root>/bak user@<destination-server-root>:<destination-
server-root>
```

4. Restore the backends **`userRoot`**, **`changelog`**, **`adminRoot`**, **`replicationChanges`**, and the **`encryption-settings`** backends if encrypted attributes are implemented.

```
$ <destination-server-root>/bin/restore -d bak/userRoot
$ <destination-server-root>/bin/restore -d bak/adminRoot
$ <destination-server-root>/bin/restore -d bak/schema
$ <destination-server-root>/bin/restore -d bak/replicationChanges
$ <destination-server-root>/bin/restore -d bak/encryption-settings
```

5. Start the server using **`bin/start-ds`**.

Removing an Offline Replica from a Topology

The **`dsreplication disable`** subcommand should be used when the target server is online. In case the target server is offline, the **`remove-defunct-server`** subcommand allows the offline replica to be removed from the replication topology. To prevent the offline server from connecting to the topology in case it is restarted, it is recommended to execute the **`cleanup-local-server`** on the offline instance. The example uses the following port numbers for each replica.

TABLE 12-6. **Replica Ports for Removing a Replica Example**

Replica	LDAP Port	Replication Port
replica1	1389	8989
replica2	2389	9989
badreplica	3389	10989

To Remove an Offline Replica

1. Run **`dsreplication remove-defunct-server`** to remove an offline replica.

```
$ bin/dsreplication remove-defunct-server --hostname replica1 --port 1389 \
  --defunctHost badreplica --defunctPort 3389 --defunctReplPort 10989
```

2. Use `dsreplication status` to verify that badreplica has been removed from the topology.

To Clean Up an Offline Server

1. Make sure the replica is offline.

2. On **badreplica**, run the **dsreplication cleanup-local-server**.

```
$ bin/dsreplication cleanup-local-server

Removing domain cn=schema from configuration ..... Done
Removing domain dc=example,dc=com from configuration ..... Done
Removing domain cn=admin data from configuration ..... Done
Removing replication server configuration entry ..... Done
Disabling replication changelog backend ..... Done
Removing 4 replication changelog database files ..... Done
Removing replication attributes from server schema ..... Done
Reading server registry ..... Done
Removing replication artifacts from server registry ..... Done
Creating cleanup-backends.ldif file in the logs directory ..... Done

The cleanup-backends.ldif file in the logs directory may be used to remove replica state from
the base DN of previously replicated backends. To complete the removal of replication data, his-
torical values should also be deleted. Consult the documentation to learn about the available
options for removing historical values
```

3. Check to see if the `ds-sync-state` and `ds-sync-generation-id` is present on the base DN, `dc=example,dc=com`. If yes, run `ldapmodify` with the `cleanup-backends.ldif` file to remove the replica state from the base DN of previously replicated backends.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret --fileName logs/cleanup-backends.ldif
```

4. To remove the replication history from regular suffixes, export the formerly replicated suffixes using the `-R` option of the `export-ldif` command. The resulting LDIF file can be re-imported using the `import-ldif` command. For example,

```
$ bin/export-ldif -n userRoot -R -l cleansed.ldif
$ bin/import-ldif -n userRoot -l cleansed.ldif
```

5. Remove the replication changelog if present.

```
$ rm -rf <server-root>/changelogDb
```

Removing a Defunct Replica from a Topology

The `remove-defunct-server` subcommand should be used when the server is completely inaccessible and is down due to some failure.

To Remove a Defunct Replica from a Topology

Remove the defunct server identified with LDAP port 4389, replication port 11989 and host name `deadhost` from all replicated base DNs using information in Directory Server listening on port 1389 at `server1.example.com`:

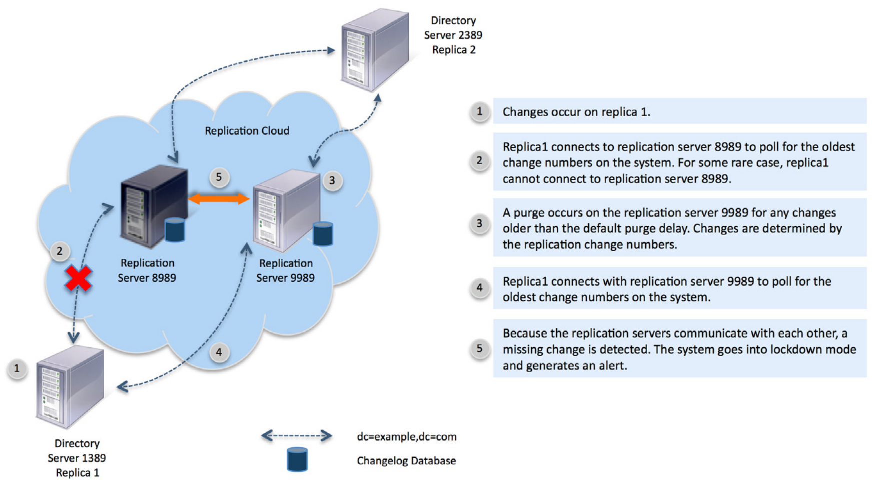
```
$ bin/dsreplication remove-defunct-server --hostname server1.example.com \
  --port 1389 --adminUID admin --adminPassword password --no-prompt \
  --defunctHost deadhost --defunctPort 4389 --defunctReplPort 11989
```

Fixing Replication Missed Changes

The Directory Server's replication system has been designed for robust performance to ensure that database replication is consistently synchronized across the topology. However, databases can have inconsistent values if conflicts arise when an attribute is modified at the same time on two different replicas or when hardware failures, such as memory or disk errors, prevent an update to be made on that server.

The illustration below shows an example scenario where a bad connection from one replica to a replication server and a purge on the other replication server might result in a missed change.

FIGURE 12-12. Example Scenario of a Possible Missed Change



1. Changes occur on replica 1.
2. Replica1 connects to replication server 8989 to poll for the oldest change numbers on the system. For some rare case, replica1 cannot connect to replication server 8989.
3. A purge occurs on the replication server 9989 for any changes older than the default purge delay. Changes are determined by the replication change numbers.
4. Replica1 connects with replication server 9989 to poll for the oldest change numbers on the system.
5. Because the replication servers communicate with each other, a missing change is detected. The system goes into lockdown mode and generates an alert.

When a missed change occurs in a replication topology, the replica enters lockdown mode only if changes are missing on all other replication servers. Lockdown mode only occurs at startup and does not occur during normal operations.

Your first step to fix any missed changes is to look at the `errors` log file to verify that the missed change occurred. Next, you have two options to fix the replica:

- **Using ldapmodify**. After you have located the differences between the two servers, you can run `bin/ldapmodify` with the Replication Repair Control OID of `1.3.6.1.4.1.30221.1.5.2` to fix the missing changes. Using the regular LDAP modify operation (that is, without the replication repair control) will be replicated to the other server instances.

- **Using dsreplication**. Re-initialize the data from another replica using the `bin/dsreplication initialize` command. If you decide to use this option, be aware that you could remove existing data from your system.

If a missed change occurred due to an unresolved naming conflict, the Directory Server automatically adds the **`ds-sync-conflict-entry`** auxiliary object class to the entry that is in conflict with an existing entry in the directory tree. Note that conflicting entries are not returned in search results by default. Therefore, you can use the **`objectclass=ds-sync-conflict-entry`** search filter component to retrieve the conflicting entries.

Fixing Missed Changes using ldapmodify

Special care must be taken when using the Replication Repair Control, because it allows you to bypass certain controls, such as no-user-modification, schema checks, and replication conflict resolution. When using the control, you should be aware that the fix can only be run on the local directory server instance. Because no replication change number is associated with the control, the operation is not published to the replication server and not recorded on historical logs. Schema checks and replication conflict resolution, such as automated administrative alerts and automatic logging (i.e., to the replication log), do not take place when using **`bin/ldapmodify`** with the control.

The Replication Repair Control also allows you to modify operational attributes generated by the Directory Server, such as **`modifiersName`**, **`modifyTimestamp`**, and **`entryuuid`**, if necessary.

To Fix Missed Changes using ldap-diff and ldapmodify

1. When a missed change is detected at server startup, the replica will go into lockdown mode by default. Check the **`logs/errors`** to view any missed change. For example, any unresolved conflict generates an administrative alert. You can view the replication log to locate the missed change.

2. Run the **`ldap-diff`** tool between the good replica and the replica with the missed change to generate a diff of the changes. The tool sends the results to an output file, **`difference.ldif`**.

```
$ bin/ldap-diff --sourceHost server1.example.com --sourcePort 1389 \
  --sourceBindDN "cn=Directory Manager" --sourceBindPassword secret1 \
  --targetHost server2.example.com --targetPort 2389 \
  --targetBindDN "cn=Directory Manager" --targetBindPassword secret2 \
  --baseDN "dc=example,dc=com" --outputLDIF difference.ldif \
  --searchFilter "(objectclass=*)"
```

3. Open the generated **`difference.ldif`** using a text editor to view the missed change.

```
dn: uid=user.999,ou=people,dc=example,dc=com
changetype: modify
add: l
l: Klamath Falls
-
delete: l
l: Klamath Fall
```

4. Run **`bin/ldapmodify`** with the Replication Repair Control to make the fix. You can use the generated **`ldap-diff`** output file to make the change.

```
$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" \
  --bindPassword secret -J "1.3.6.1.4.1.30221.1.5.2" --filename difference.ldif
Processing MODIFY request for uid=user.999,ou=People,dc=example,dc=com
MODIFY operation successful for DN uid=user.999,ou=People,dc=example,dc=com
```

Changing the Default Lockdown Mode Behavior for Missed Changes

At server startup, when a missed change occurs, a replica enters lockdown mode only if it detects that every available replication server has already deleted one or more changes that it has not received yet.

Administrators can control how lockdown mode is invoked, if necessary, by using the `dsconfig` tool and setting the `lockdown-on-missed-replication-changes` property. The property has three possible options:

- **all-servers**. (default). Enters lockdown mode if the replica detects that all available replication servers have already purged one or more changes that the replica needs.
- **any-server**. Enters lockdown mode if any of the available replication servers has already purged one or more changes that the replica needs. Note that replication servers may also miss one or more changes and data inconsistency between replicas may still occur using this mode.
- **never**. Lockdown mode will not be entered if missing changes are detected at any replication server. The administrator will still be notified using an administrative alert. Data inconsistency between replicas can occur.

To Change the Default Lockdown Mode Behavior

- First, ensure that your replicas are in a server group. Use the `dsconfig` tool on the Global Configuration option to change the `lockdown-on-missed-replication-changes` property. Remember to include the LDAP or LDAPS connection parameters (`hostname`, `port`, `bindDN` and `bindPassword`) with the `dsconfig` command. The following command changes the default replication lockdown mode on missed changes to "any-servers":

```
$ bin/dsconfig set-global-configuration-prop --set lockdown-on-missed-replication-
changes:any-servers
```

Fixing Replication Conflicts using ldapmodify

Replication conflicts can occur when an incompatible change to an entry is made on two replicas at the same time. The change is processed on one replica and then replicated to the other replica, which causes the conflict. While most conflicts are resolved automatically, some require manual action.

To Fix Replication Conflicts using ldapmodify

1. When an unresolved conflict occurs in a replication topology, the Directory Server records it in the **errors** log and generates an administrative alert. You can view the error log to locate the specific conflict. In this example, the conflicting entry is found for **uid=user.200**.

```
$ cat logs/errors
[18/Feb/2010:14:53:12 -0600] category=EXTENSIONS severity=SEVERE_ERROR
msgID=1880359005 msg="Administrative alert type=replication-unresolved-conflict
id=bbd2cbaf-90a4-42af-94a8-c1a42df32fc6 class=com.unboundid.directory.server.repli-
cation.plugin.ReplicationDomain msg='An unresolved conflict was detected for DN
uid=user.200,ou=People,dc=example,dc=com. The conflicting entry has been renamed to
entryuuid=69807e3d-ab27-43a3-8759-ec0d8d6b3107+uid=user.200,ou=People,dc=exam-
ple,dc=com'"
```

2. Verify the conflicting entry. The Directory Server prepends the **entryUUID** to the DN of the conflicting attribute and adds a **ds-sync-conflict-entry** auxiliary object class to the entry to aid in search. For example, the following command searches for any entry that has the **ds-sync-conflict-entry** objectclass and returns only the DNs that match the filter. You should see the conflicting entry for **uid=user.200**.

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --baseDN dc=example,dc=com --searchScope sub \
  "(objectclass=ds-sync-conflict-entry)" "1.1"

dn: entryuuid=69807e3d-ab27-43a3-8759-ec0d8d6b3107+uid=user.200,ou=People,dc=exam-
ple,dc=com

dn: entryuuid=523c430e-a870-4ebe-90f8-9cd811946420+uid=user.200,ou=People,dc=exam-
ple,dc=com
```

Note	Conflict entries are not returned unless the objectclass=ds-sync-conflict-entry is present in the search filter.

3. Look at both entries and then fix one of them, so that both are identical. You can modify or remove the entries, then re-add the entry to the directory server. Run **bin/ldapmodify** with the Replication Repair Control to make the fix. When making changes using the Replication Repair Control, the updates will not be propagated via replication. You should examine each and every replica one by one, and apply the necessary modifications using the control.

```
$ bin/ldapmodify --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret -J "1.3.6.1.4.1.30221.1.5.2" --filename difference.ldif
```

Fixing Mismatched Generation IDs

If a problem arises with replication other than the one listed below, then contact your authorized support provider for assistance.

Fixing Mismatched Generation IDs during Setup

If you see an error warning that multiple generation IDs were detected for a specific suffix, it indicates that one or more replicas need to be re-initialized in order to allow replication to and from these replicas. The **`dsreplication status`** command will show any generation IDs that are different on each replica as follows:

```
WARNING:  Multiple Generation IDs were detected for this suffix.  Replication uses Gen-
eration IDs to ensure consistency between replica servers by validating that each has
started with the same version of the data.  Changes made at one server are not repli-
cated to another server if it has a different Generation ID.  A list of servers in this
topology and their Generation IDs follow.  Refer to the documentation for the best way
to initialize these instances to bring their Generation IDs in sync:

    Generation ID: 2982624842
        Server: server1.example.com:1389
    Generation ID: 3211313
        Server: server2.example.com:2389
        Server: server3.example.com:3389
```

To Fix Mismatched Generation IDs

1. Ensure replication is configured between the replicas.

2. Re-load the same LDIF file onto the servers that have mismatched generation IDs using binary copy, **`import-ldif`**, **`parallel-update`**, or **`dsreplication initialize`** command-line tools.

Replication Reference

The following section shows general reference information related to replication.

Summary of the dsreplication Subcommands

A summary of the **dsreplication** subcommands and functions is presented in the table below.

TABLE 13. **dsreplication Subcommands**

dsreplication subcommand	Description

cleanup-local-server	Removes replication-related artifacts from the configuration, schema as well as the server registry while the local server is offline. The subcommand does not remove references to this server from other replicas and replication servers in the topology. Therefore, it is recommended to remove this server first from the replication topology either by using the `disable` or `remove-defunct-server` subcommands. Since this subcommand can only be executed when the server is offline, replication attributes from suffixes other than the server registry or the schema will not be removed. The tool will produce an LDIF file, `logs/cleanup-backends.ldif` that may be used to the remove replica state from the base entry of these suffixes after the server is restarted.
	
	To remove the replication history from regular suffixes, export the formerly replicated suffixes using the `--excludeReplication` option of the `export-ldif` command. The resulting LDIF file can be re-imported using the `import-ldif` command. For example,
	
	`export-ldif --backendID userRoot --excludeReplication --ldifFile cleansed.ldif`
	`import-ldif --backendID userRoot --ldifFile cleansed.ldif`
	
	Exporting using the `--excludeReplication` option of the `export-ldif` command will also remove the replica state from the output. The LDIF created by the `cleanup-local-server` subcommand does not need to be applied after the server is restarted.

disable	Disables replication on the specified server for the provided base DN and removes references to this server in the other servers with which it is replicating data.
enable	Updates the configuration of the servers to replicate the data under the specified base DN. If one of the servers is already replicating the data under the base DN with other servers, executing this subcommand will update the configuration of all the servers (so it is sufficient to execute the command line once for each server you add to the replication topology).

TABLE 13. **dsreplication Subcommands**

dsreplication subcommand	**Description**
initialize	Initializes the data under the specified base DN on the destination server with the contents on the source server. This operation performs a background export and import and should only be used for replicated base DNs whose database files are less than 1 GB in size ('initialize-all' can also be used for this purpose).
initialize-all	Initializes the data under the specified base DN on all servers in the replication topology with the contents on the specified server. This operation performs a background export and import and should only be used for replicated base DNs whose database files are less than 1 GB in size ('initialize' applied on each server can also be used for this purpose).
post-external-initialization	Resets the generation ID based on the newly-loaded data. This subcommand must be called after initializing the contents of all the replicated servers using the `import-ldif` tool or the binary copy method. You must specify the list of base DNs that have been initialized and provide the credentials of any of the servers that are being replicated. See the usage of the `pre-external-initialization` subcommand for more information. This subcommand only needs to be run on one of the replicas once.
pre-external-initialization	Clears the existing generation ID and removes all accumulated changes of the replicated suffix from the replication changelog database at each and every replication server. This subcommand should be used when globally restoring the replicas on all of the servers in a topology. You must specify the list of base DNs that will be initialized and provide the credentials of any of the servers that are being replicated. After calling this subcommand, initialize the contents of all the servers in the topology, then call the `post-external-initialization` subcommand. This subcommand only needs to be run on one of the replicas once.
remove-defunct-server	Removes an offline defunct server from the replication configuration.

TABLE 13. dsreplication Subcommands

dsreplication subcommand	Description
status	Displays the basic replication configuration information for the base DNs of the servers defined in the registration information. If no base DNs are specified as parameters, the information for all base DNs is displayed. To see an example, see “Check the Status of the Replication Topology” on page 389. The `dsreplication status` command has two sections. The first section lists the servers in the replication topology: • **Server.** Displays the server host name and port. • **ServerID**. Displays the server ID number in the system. • **GenerationID**. Displays the generation ID. • **Security**. Displays the security connection if any SSL or StartTLS. The second section shows the status of each server: • **Server.** Displays the server host name and port. • **Entries**. Displays the number of total entries in the server. • **Backlog**. Displays the number of changed entries that are in the queue to be replicated. • **Oldest Backlog Change Age**. Displays the oldest backlog change age. • **Generation ID**. Displays the generation ID of the server. • **Repl Server ID**. Displays the replication server ID.
detach	Detaches a single directory server instance or directory server instances in the same replication set from the rest of the replication topology. This will create two independent replication topologies. After the detaching process completes, replication will not propagate update messages between the removed server (or servers) and the rest of the replication topology. The `enable` subcommand is used to re-attach a server (or servers) to the replication topology.

Summary of the Replication Management Scenarios

The following table shows the sequence of the `dsreplication` subcommands necessary to configure or remove a replica using non-interactive mode. The scenarios assume that you have configured the directory server instances (for example, ds1, ds2, and in some cases, ds3). For example, the first scenario shows the order of the `dsreplication` subcommands necessary to create a replication topology from three standalone servers (ds1 has the DIT, ds2 and ds3 have base DNs only). To set up replication, you must run `dsreplication enable` for ds1 and ds2, then run the command again for ds1 and ds3. Finally, you must run `dsreplication initialize-all` to load the replication data from ds1 into replicas ds2 and ds3.

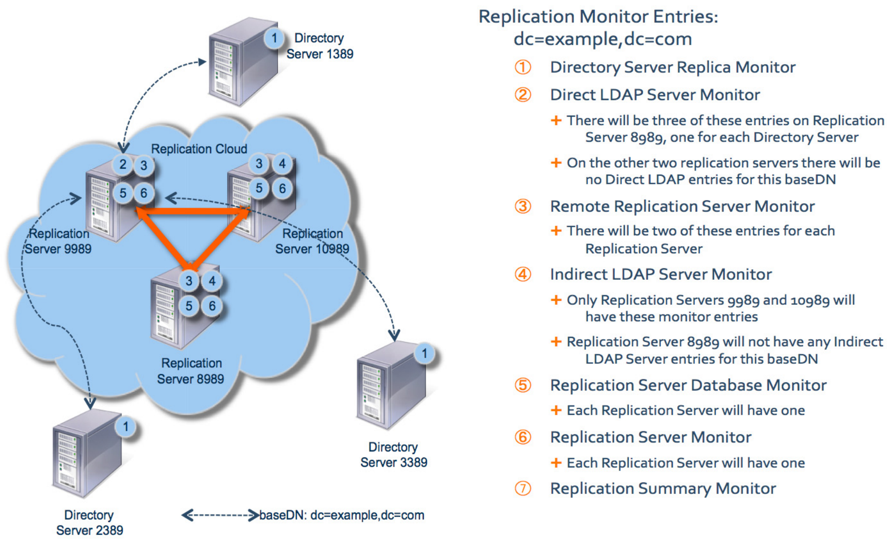
Some scenarios reference the step `[ Manual Initialize ]`, which means that data is copied from a source replica to a target replica using the binary copy method or imported onto a target replica using the `import-ldif` tool. When you manually initialize data onto a replica, it is recommended that you not use the `dsreplication initialize` command, which could result in longer initialization times for very large databases (i.e., over 1 gigabyte). See “To Recover

One Replica in an Existing Topology due to an Out-of-Sync Database” on page 415 for an example showing how to run a manual initialization using the binary copy method.

Scenario	Dsreplication Subcommand
Creating a replication topology from standalone servers (ds1, ds2, ds3) with small database	ENABLE (ds1, ds2) → ENABLE (ds1, d3) → INITIALIZE-ALL
Creating a replication topology from standalone servers (ds1, ds2, ds3) with large database	[Manual Initialize across all servers] → ENABLE (ds1, ds2) → ENABLE (ds1, d3)
Adding a server with small database to a replicated topology	ENABLE → INITIALIZE
Adding a server with large database to a replicated topology	[Manual Initialize] → ENABLE
Reinitializing a server, with small database, that is part of a replicated topology	INITIALIZE
Reinitializing a server, with large database, that is part of a replicated topology	[Manual Initialize]
Removing a server from a replication topology	DISABLE
Removing a server with large database from a topology and reinserting	DISABLE → [Manual Initialize] → ENABLE
Removing a server with small database from a topology and reinserting	DISABLE → ENABLE → INITIALIZE
Reimporting data (changing generation ID) across entire topology with large database	PRE EXTERNAL INITIALIZATION → [Manual Initialize] → POST EXTERNAL INITIALIZATION
Reimporting data (changing generation ID) across entire topology, with small database	PRE EXTERNAL INITIALIZATION → import-ldif → INITIALIZE ALL → POST EXTERNAL INITIALIZATION
Take a server out of the replication topology and cleaning all replication information from server and database	DISABLE → CLEAN SERVER → export-ldif -R → import-ldif
Remove a server from the topology when it is no longer responding	REMOVE DEFUNCT SERVER → (optional on defunct server) CLEAN SERVER
Temporarily detach a server from the topology	DETACH
Temporarily detach a set from the topology	DETACH --setName <set>

Monitoring Reference

FIGURE 12-13. **Replication Monitor Entries**



Replication Monitor Entries:
dc=example,dc=com

① Directory Server Replica Monitor

② Direct LDAP Server Monitor
- + There will be three of these entries on Replication Server 8989, one for each Directory Server
- + On the other two replication servers there will be no Direct LDAP entries for this baseDN

③ Remote Replication Server Monitor
- + There will be two of these entries for each Replication Server

④ Indirect LDAP Server Monitor
- + Only Replication Servers 9989 and 10989 will have these monitor entries
- + Replication Server 8989 will not have any Indirect LDAP Server entries for this baseDN

⑤ Replication Server Database Monitor
- + Each Replication Server will have one

⑥ Replication Server Monitor
- + Each Replication Server will have one

⑦ Replication Summary Monitor

Summary of the Direct LDAP Monitor Information

The following table provides a description of the attributes in the `cn=Direct LDAP Server` monitor entry.

TABLE 12-1. **Direct LDAP Monitor Information**

dn: cn=Direct LDAP Server <baseDN> <host name:ldapPort> <serverID>,cn=monitor	**Description**
connected-to: Replication Server <replPort> <serverID>	Replication port number and server ID of the replication server to which this LDAP Server is connected. The first number is the replication server port number and the second number is the server-id of the replication server.
replica-id:<serverID>	Replica ID number.
base-dn	Base DN.
replication-backlog	Number of changes that the replication server has not seen from the LDAP server.
approximate-delay	Difference between the time of the last change that the replication server has seen from the LDAP server and the most current time stamp on the latest change on the LDAP server.

TABLE 12-1. **Direct LDAP Monitor Information**

dn: cn=Direct LDAP Server <baseDN> <host name:ldapPort> <serverID>,cn=monitor	**Description**
approximate-catchup-time	Approximate the time the replica requires to catch up with incoming changes. The formula is based on the number of changes in replication backlog and the speed with which the replica is clearing out its backlog. If the replica is diverging (i.e., cannot receive updates as fast as updates are received from other replicas, then "n/a" will be displayed for the approximate-catchup-time attribute).
consumed-update-recent-rate	Rate that the connected Directory Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Directory Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.
produced-update-recent-rate	Rate that the connected Directory Server has sent updates to this replication server, expressed as the number of updates per second and measured over the last five seconds.
produced-update-peak-rate	Highest rate that the connected Directory Server has sent updates to this replication server, measured over a five second period since the replication server was started.
queue-size	Number of changes that are in the replication server queue to be sent to this LDAP server.
queue-size-bytes	Size in bytes of all the messages waiting in the queue.
following	When set to true, the internal buffer is large enough to hold update messages in transit between the LDAP server and the replication server. A value of true means that the LDAP server can accept changes fast enough from this replication server. When the LDAP server is unable to keep up with this replication server, and the internal queue is not large enough to hold all updates targeted for this LDAP server, then the flag is set to false.
fallen-behind-count	Number of times the connected Directory Server has fallen behind while processing updates sent by this replication server.
max-waiting-changes	Max number of changes in the queue on the replication server for each LDAP server.
update-sent	Number of changes sent to this LDAP server.

TABLE 12-1. **Direct LDAP Monitor Information**

dn: cn=Direct LDAP Server <baseDN> <host name:ldapPort> <serverID>,cn=monitor	**Description**
update-received	Number of changes received from this LDAP server.
update-waiting-acks	Not used.
ack-sent	Not used.
ack-received	Not used.
max-send-window	Maximum number of changes that can be sent to the LDAP server before requiring an ACK.
current-send-window	Current number of changes remaining to be sent to the LDAP server before requiring an ACK.
max-rcv-window	Maximum number of changes that can be received by the LDAP server before sending an ACK.
current-rcv-window	Number of changes remaining to be received by the LDAP server before sending an ACK Server.
ssl-encryption	Flag to indicate if SSL encryption is in use.
generation-id	Generation ID for this suffix on the directory server.
restricted	Boolean that indicates whether the replication domain is restricted in an Entry Balancing Configuration with Proxy Server.
messages-sent	Total number of messages sent.
messages-received	Total number of messages received.
compression	Flag to indicate if compression is in use. Values: on, off.

Summary of the Indirect LDAP Server Monitor Information

The following table provides a description of the attributes in the `cn=Indirect LDAP Server` monitor entry. These attributes provide information about a directory server that is connected to a different replication server in the topology.

TABLE 12-2. **Indirect LDAP Server Monitor Information**

dn: cn=Indirect LDAP Server <baseDN> <serverID>,cn=monitor	**Description**
replica-id: <serverID>	ID number identifying the replica.
base-dn: <baseDN>	Base DN
connected-to: Remote Repl Server <baseDN> <host name:replPort> <replID>	Replication server to which the directory server is connected.
replication-backlog	Number of changes that the replication server has not seen from the LDAP server.

TABLE 12-2. **Indirect LDAP Server Monitor Information**

dn: cn=Indirect LDAP Server <baseDN> <serverID>,cn=monitor	Description
approximate-delay	Amount of time between the last change seen by this directory server and the most recent change seen by the remote replication server. This value is the amount of time between the time stamps, not the amount of time required to synchronize the two servers.
generation-id	Generation ID for this suffix on this remote replication server.
consumed-update-recent-rate	Rate that the connected Replication Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Replication Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.

Summary of the Remote Replication Server Monitor Information

The following table provides a description of the attributes in the `cn=Remote Repl Server` monitor entry.

TABLE 12-3. **Remote Replication Server Monitor Information**

dn: cn=Remote Repl Server <baseDN> <host name:replPort> <serverID>,cn=monitor	Description
replication-server: <host name>:<repl port>	Host name and replication port number of the Replication Server.
replication-server-id:<serverID>	Server ID for the Replication Server.
consumed-update-recent-rate	Rate that the connected Directory Server is consuming updates sent by this replication server, expressed as the number of updates per second and measured over the last five seconds.
consumed-update-peak-rate	Highest rate that the connected Directory Server has consumed updates sent by this replication server, measured over a five second period since the replication server was started.
produced-update-recent-rate	Rate that the connected Directory Server has sent updates to this replication server, expressed as the number of updates per second and measured over the last five seconds.

TABLE 12-3. **Remote Replication Server Monitor Information**

dn: cn=Remote Repl Server <baseDN> <host name:replPort> <serverID>,cn=monitor	**Description**
produced-update-peak-rate	Highest rate that the connected Directory Server has sent updates to this replication server, measured over a five second period since the replication server was started.
queue-size	Number of changes that are in the replication server queue to be sent to this remote replication server.
queue-size-bytes	Size in bytes of all the messages waiting in the queue.
following	When the remote replication server is following this replication server, then the internal buffer is large enough hold update messages in transit. A value of true generally means that the remote replication server can accept changes fast enough from this replication server. When the remote replication server is unable to keep up with the changes from the replication server, and the internal queue is not large enough to hold all updates targeted for this remote replication server, then the flag is set to false.
fallen-behind-count	Number of times the connected Directory Server has fallen behind while processing updates sent by this replication server.
max-waiting-changes	Max number of changes that can be in the queue on the replication server.
update-sent	Number of changes sent to this remote replication server.
update-received	Number of changes received from this remote replication server.
update-waiting-acks	Not used.
ack-sent	Not used.
ack-received	Not used.
max-send-window	Maximum number of changes that can be sent to the remote replication server before requiring an ACK.
current-send-window	Number of changes remaining to be sent to the remote replication server before requiring an ACK.
max-rcv-window	Maximum number of changes that can be received from the remote replication server before sending an ACK.
current-rcv-window	Number of changes remaining to be received from the remote replication server before sending an ACK.
ssl-encryption	Flag to indicate if SSL encryption is in use.
generation-id	Generation ID for this suffix on this remote replication server.

TABLE 12-3. **Remote Replication Server Monitor Information**

dn: cn=Remote Repl Server <baseDN> <host name:replPort> <serverID>,cn=monitor	**Description**
messages-sent	Total number of messages sent.
messages-received	Total number of messages received.
compression	Flag to indicate if compression is in use.

Summary of the Replica Monitor Information

The following table provides a description of the attributes in the `cn=Replica` monitor entry for a specific base DN.

TABLE 12-4. **Replica Monitor Information**

dn: cn=Replica <baseDN>,cn=monitor	**Description**
base-DN: <baseDN>	Specified base DN. The monitor entries track your company's base DN (or `dc=example,dc=com`), `cn=schema`, and `cn=admin data`.
connected-to: Replication Server <replPort> <serverID>	Replication port number and server ID of the replication server to which this LDAP Server is connected. The first number is the replication server port number and the second number is the `server-id` of the replication server.
lost-connections	Number of times the Directory Server has lost connection to a replication server.
received-updates	Number of updates that the Directory Server Replica has received from the connected replication server.
sent-updates	Number of updates sent to the replication server.
pending-updates	Number of updates pending to send to the replication server.
replayed-updates	Total number of updates from the replication server that have been replayed for this replica.
replayed-updates-ok	Number of updates for this replica that have been successfully replayed with no conflicts.
replayed-update-failed	Number of updates for this replica that were successfully replayed after automatically resolving a modify conflict.
resolved-modify-conflicts	Number of updates for this replica that were successfully resolved after a modify conflict.
resolved-naming-conflicts	Number of updates for this replica that were successfully resolved after a naming conflict.
unresolved-naming-conflicts	Number of updates for this replica that could not be replayed due to an unresolvable naming conflict.

TABLE 12-4. **Replica Monitor Information**

dn: cn=Replica <baseDN>,cn=monitor	**Description**
replica-id	Server ID for this replica.
max-rcv-window	Maximum number of changes that the Directory Server Replica can receive at a time before sending an acknowledgment back to the replication server.
current-rcv-window	Current received window size for this replica.
max-send-window	Maximum number of changes that the Directory Server Replica can send at a time to the replication server before requiring an ACK.
current-rcv-window	Number of changes remaining to be received from the replication server before it must send an ACK.
max-send-window	Maximum number of changes that the Directory Server Replica can send at a time to the replication server before requiring an ACK.
current-send-window	Number of changes remaining to be sent to the replica-tion server before requiring an ACK.
ssl-encryption	Flag to indicate if SSL encryption is in use.
generation-id	Generation ID for this suffix on the directory server.
replication-backlog	Number of changes that are from this replica.

Summary of the Replication Server Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server` monitor entry.

TABLE 12-5. **Replication Server Monitor Information**

dn: cn=Replication Server <baseDN> <replServerID>,cn=monitor	**Description**
replication-server-id	Server ID for the Replication server ID.
replication-server-port	Port number on which the replication server listens for communication from other servers.
base-dn: <baseDN>	Indicates the suffix to which this replication server database applies.
Generation IDs by Base DN	List of generation IDs for each base DN on the server.
num-outgoing-replication-server-connections	Number of outgoing connections from the replication server.
num-incoming-replication-server-connections	Number of incoming connections into the replication server.
num-incoming-replica-connec-tions	Number of incoming connections to the replica.

Summary of the Replication Server Database Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server database` monitor entry.

TABLE 12-6. Replication Server Database Monitor Information

dn: cn=Replication Server database <baseDN> <replServerID>,cn=monitor	Description
database-replica-id: <replicaID>	Specifies the replication server ID.
base-dn: <baseDN>	Indicates the suffix to which this replication server database applies.
first-change	First change number that is in this replication database table for this suffix from this `server-id`. For example, an example entry looks like the following: `0000012209EA622C390D00000002 Mon Jun 22 16:41:11 CDT 2011`
last-change	Last change number that is in this replication database table for this suffix from this `server-id`. For example, an example entry looks like the following: `0000012209EA622C390D00000002 Mon Jun 22 16:41:11 CDT 2011`
queue-size	Number of changes in the replication server queue waiting to be sent to this remote replication server database.
queue-size-bytes	Size in bytes of all the messages waiting in the queue.
records-added	Displays the number of records changed or added to the DIT.
records-removed	Displays the number of records removed from the DIT.

Summary of the Replication Server Database Environment Monitor Information

The following table provides a description of the attributes in the `cn=Replication Server Database Environment` monitor entry, which includes the environment variables associated with the Oracle Berkeley Database Java Edition backend.

TABLE 12-7. Replication Server Database Environment Monitor Information

dn: cn=Replication Server Database Environment,cn=monitor	Description
je-version	Current version of the Oracle Berkeley Java Edition.
current-db-cache-size	Current DB cache size.
max-db-cache-size	Maximum DB cache size.
db-cache-percent-full	Percentage of the cache used by the Directory Server.
db-directory	Directory that holds the `changelogDb` file.

TABLE 12-7. **Replication Server Database Environment Monitor Information**

dn: cn=Replication Server Database Environment,cn=monitor	Description
db-on-disk-size	Size of the DB on disk.
cleaner-backlog	Number of log files that must be cleaned for the cleaner to meet its target utilization.
random-read-count	Number of disk reads which required repositioning the disk head more than 1MB from the previous file position.
random-write-count	Number of disk writes which required repositioning the disk head by more than 1MB from the previous file position.
sequential-read-count	Number of disk reads which did not require repositioning the disk head more than 1MB from the previous file position.
sequential-write-count	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
nodes-evicted	Accumulated number of nodes evicted.
active-transaction-count	Number of currently active transactions.
num-checkpoints	Number of checkpoints. A checkpoint is a process that writes to your log files all the internal BTree nodes and structures modified as a part of write operations to your log files to facilitate a quick recovery.
checkpoint-in-progress: false	Indicates if a checkpoint is in progress.
total-checkpoint-duration-millis	Total time in milliseconds for all checkpoints.
average-checkpoint-duration-millis	Average time in milliseconds for all checkpoints.
last-checkpoint-duration-millis	Duration in milliseconds of the last checkpoint run.
last-checkpoint-start-time	Start time of the last checkpoint.
last-checkpoint-stop-time	Stop time of the last checkpoint.
millis-since-last-checkpoint	Time in milliseconds since the last checkpoint.
read-locks-held	Total read locks currently held.
write-locks-held	Total write locks currently held.
transactions-waiting-on-locks	Total transactions waiting for locks
je-env-stat-AdminBytes	Number of bytes of JE cache used for log cleaning metadata and other administrative structure.
je-env-stat-BufferBytes	Total memory currently consumed by log buffers, in bytes.
je-env-stat-CacheDataBytes	Total memory of cache used for data.
je-env-stat-CacheTotalBytes	Total amount of JE cache in use, in bytes.

TABLE 12-7. **Replication Server Database Environment Monitor Information**

dn: cn=Replication Server Database Environment,cn=monitor	Description
je-env-stat-CleanerBacklog	Number of files to be cleaned to reach the target utilization.
je-env-stat-CursorsBins	Number of bottom internal nodes (BINs) encountered by the INCompressor that had cursors referring to them when the compressor ran. The compressor thread cleans up the internal BTree as records are deleted to ensure unused nodes are not present.
je-env-stat-DataBytes	Amount of JE cache used for holding data, keys and internal Btree nodes, in bytes.
je-env-stat-DbClosedBins	Number of bins encountered by the INCompressor that had their database closed between the time they were put on the compressor queue and when the compressor ran.
je-env-stat-EndOfLog	Location of the next entry to be written to the log.
je-env-stat--InCompQueueSize	Number of entries in the INCompressor queue when the `getStats()` call was made.
je-env-stat-LastCheckpointEnd	Location in the log of the last checkpoint end.
je-env-stat-LastCheckpointId	ID of the last checkpoint.
je-env-stat-lastCheckpointStart	Location in the log of the last checkpoint start.
je-env-stat-lockBytes	Number of bytes of JE cache used for holding locks and transactions.
je-env-stat-NBINsStripped	Number of BINS stripped by the evictor.
je-env-stat-NCacheMiss	Total number of requests for database objects which were not in memory.
je-env-stat-NCheckpoints	Total number of checkpoints run so far.
je-env-stat-NCleanerDeletions	Number of cleaner file deletions this session.
je-env-stat-NCleanerEntriesRead	Accumulated number of log entries read by the cleaner.
je-env-stat-NCleanerRuns	Number of cleaner runs this session.
je-env-stat-NClusterLNsProcessed	Accumulated number of leaf nodes (LNs) processed because they qualify for clustering.
je-env-stat-NDeltaINFlush	Accumulated number of delta internal nodes (INs) flushed to the log.
je-env-stat-NEvictPasses	Number of passes made to the evictor.
je-env-stat-NFSyncRequests	Number of fsyncs requested through the group commit manager. `Fsync()` synchronizes the filesystem after a write or a transaction.
je-env-stat-NFSyncTimeouts	Number of fsync requests submitted to the group commit manager which timed out.

TABLE 12-7. **Replication Server Database Environment Monitor Information**

dn: cn=Replication Server Database Environment,cn=monitor	Description
je-env-stat-NFSyncs	Number of fsyncs issued through the group commit manager.
je-env-stat-NFileOpens	Number of times a log file has been opened.
je-env-stat-NFullBINFlush	Accumulated number of full bottom internal nodes (BINS) flushed to the log.
je-env-stat-NFullINFlush	Accumulated number of full INs flushed to the log.
je-env-stat-NINsCleaned	Accumulated number of INs cleaned.
je-env-stat-NINsDead	Accumulated number of INs that were not found in the tree anymore (deleted).
je-env-stat-NINsMigrated	Accumulated number of INs migrated.
je-env-stat-NINsObsolete	Accumulated number of INs obsolete.
je-env-stat-NLNQueueHits	Accumulated number of LNs processed without a tree lookup.
je-env-stat-NLNsCleaned	Accumulated number of LNs cleaned.
je-env-stat-NLNsDead	Accumulated number of LNs that were not found in the tree anymore (deleted).
je-env-stat-NLNsLocked	Accumulated number of LNs encountered that were locked.
je-env-stat-NLNsMarked	Accumulated number of LNs that were marked for migration during cleaning.
je-env-stat-NLNsMigrated	Accumulated number of LNs migrated.
je-env-stat-NLNsObsolete	Accumulated number of LNs obsolete.
je-env-stat-NLogBuffers	Number of log buffers currently instantiated.
je-env-stat-NMarkedLNsPro-cessed	Accumulated number of LNs processed because they were previously marked for migration.
je-env-stat-NNodesExplicitly-Evicted	Accumulated number of nodes evicted.
je-env-stat-NNodesScanned	Accumulated number of nodes scanned in order to select the eviction set.
je-env-stat-NNodesSelected	Accumulated number of nodes selected to evict.
je-env-stat-NNotResident	Number of requests for database objects not contained within the in memory data structures.
je-env-stat-NOpenFiles	Number of files currently open in the file cache.
je-env-stat-NPendingLNsLocked	Sccumulated number of pending LNs that could not be locked for migration because of a long duration application lock.
je-env-stat-NPendingLNsPro-cessed	Accumulated number of LNs processed because they were previously locked.

TABLE 12-7. **Replication Server Database Environment Monitor Information**

dn: cn=Replication Server Database Environment,cn=monitor	Description
je-env-stat-NRandomReadBytes	Number of bytes read which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomReads	Number of disk reads which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomWriteBytes	Number of bytes written which required repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NRandomWrites	Number of disk writes which required repositioning the disk head by more than 1MB from the previous file position.
je-env-stat-NRepeatFaultReads	Number of reads which had to be repeated when faulting in an object from disk because the read chunk size controlled by je.log.faultReadSize is too small.
je-env-stat-NRepeatIteratorReads	Number of times we try to read a log entry larger than the read buffer size and can't grow the log buffer to accommodate the large object.
je-env-stat-NRootNodesEvicted	Accumulated number of database root nodes evicted.
je-env-stat-NSequentialReadBytes	Number of bytes read which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialReads	Number of disk reads which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialWriteBytes	Number of bytes written which did not require repositioning the disk head more than 1MB from the previous file position.
je-env-stat-NSequentialWrites	Number of disk writes which did not require repositioning the disk head by more than 1MB from the previous file position.
je-env-stat-NSharedCacheEnvironments	Number of environments using the shared cache.
je-env-stat-NTempBufferWrites	Number of writes which had to be completed using the temporary marshalling buffer because the fixed size log buffers specified by `je.log.totalBufferBytes` and `je.log.numBuffers` were not large enough.
je-env-stat-NToBeCleanedLNsProcessed	Accumulated number of LNs processed because they are soon to be cleaned.
je-env-stat-NonEmptyBins	Number of non-empty bins.
je-env-stat-ProcessedBins	Number of bins that were successfully processed by the INCompressor.

TABLE 12-7. **Replication Server Database Environment Monitor Information**

dn: cn=Replication Server Database Environment,cn=monitor	Description
je-env-stat-RequiredEvictBytes	Number of bytes that must be evicted in order to get within the memory budget.
je-env-stat-SharedCacheTotal-Bytes	Total amount of the shared JE cache in use, in bytes.
je-env-stat-SplitBins	Number of bins encountered by the INCompressor that were split between the time they were put on the compressor queue and when the compressor ran.
je-env-stat-TotalLogSize	Approximation of the current total log size in bytes.
je-env-stat-NOwners	Total lock owners in the lock table.
je-env-stat-NReadLocks	Total read locks currently held.
je-env-stat-NRequests	Total number of lock requests to date.
je-env-stat-NTotalLocks	Total locks currently in the lock table.
je-env-stat-NWaiters	Total transactions waiting for locks.
je-env-stat-NWaits	Total number of lock waits to date.
je-env-stat-NWriteLocks	Total write locks currently held.
je-env-stat-LastCheckpointTime	Time of the last checkpoint.
je-env-stat-LastTxnId	Last transaction ID allocated.
je-env-stat-NAborts	Number of transactions that have aborted.
je-env-stat-NActive	Number of transactions that are currently active.
je-env-stat-NBegins	Number of transactions that have begun.
je-env-stat-NCommits	Number of transactions that have committed.
je-env-stat-NXAAborts	Number of XA transactions that have aborted.
je-env-stat-NXACommits	Number of XA transactions that have committed.
je-env-stat-NXAPrepares	Number of XA transactions that have been prepared.

Summary of the Replication Summary Monitor Information

The following table provides a description of the attributes in the **`cn=Replication Summary`** monitor entry.

TABLE 12-8. **Replication Summary Monitor Information**

dn: cn=Replication Summary <baseDN>,cn=monitor	**Description**
base-dn:<baseDN>	Base DN summary.
replica: replica-id, ldap-server connected-to, generation-id, replication-backlog, recent-update-rate, peak-update-rate, age-of-oldest-backlog-change	Summary information for each replica in the topology. This entry appears for each replica in the topology with its own respective properties.
replication-server: server-id, server, generation-id, status, last-connected, last-failed, failed-attempts, attributes.	Summary information for each remote replication server only in the topology. This entry appears for each Replication Server in the topology with its own respective **`serverID`** and **`server`** properties.

Summary of the replicationChanges Backend Monitor Information

The following table provides a description of the attributes in the **`cn=replicationChanges Backend`** monitor entry.

TABLE 12-9. **replicationChanges Backend Monitor Information**

dn: cn=replicationChanges Backend,cn=monitor	**Description**
ds-backend-id	ID descriptor for the backend. Typically, this will be "replicationChanges".
ds-backend-base-dn	Base DN for the backend. Typically, this will be **`cn=replicationChanges`**.
ds-backend-is-private	Flag to indicate if the backend is private.
ds-backend-entry-count	Entry count for the backend.
ds-base-dn-entry-count	Entry count for the base DN and the specified base DN.
ds-backend-writability-mode	Flag to indicate if the backend is writable or not.

Summary of the Replication Protocol Buffer Monitor Information

The following table provides a description of the attributes in the `cn=Replication Protocol Buffer` monitor entry. The monitors provide information on the state of the buffer for protocol operations, which is kept in the local storage.

TABLE 12-10. **Replication Protocol Buffer Monitor Information**

dn: cn=Replication Protocol Buffer,cn=monitor	Description
saved-buffers	Number of protocol buffers. Initial buffer size is 4k.
reallocations	Number of times the buffers had to be reallocated due to insufficient size.
large-buffer-creates	Number of times a buffer larger than 512k was requested.
large-buffer-evictions	Number of times a buffer was removed from the thread local storage, because it has grown above 512k.

13 Managing Logging and Monitoring

Overview

The UnboundID Directory Server supports a rich set of log publishers to monitor access, debug, and error messages that occur during normal server processing. Administrators can view the standard set of default log files as well as configure custom log publishers with pre-defined filtering with its own log rotation and retention policies.

The UnboundID Directory Server also provides a flexible monitoring framework that exposes its monitoring information under the `cn=monitor` entry and provides interfaces through JMX, the UnboundID Directory Management Console, over LDAP, and SNMP.

This chapter presents the following information:

Default Directory Server Logs

The Directory Server provides a standard set of default log files to monitor the server activity. You can view this set of logs in the **`UnboundID-DS/logs`** directory. The following default log files are available on the Directory Server and are presented below.

TABLE 13-1. Directory Server Logs

Log File	Description
access	File-based Access Log that records operations processed by the Directory Server. Access log records can be used to provide information about problems during operation processing (for example, unindexed searches, failed requests, etc.), and provide information about the time required to process each operation.
change-notifications.log	Records changes to data anywhere in the server, which match one or more configured change subscriptions.
config-audit.log	Records information about changes made to the Directory Server configuration in a format that can be replayed using the **`dsconfig`** tool
errors	File-based Error Log. Provides information about warnings, errors, and significant events that occur during server processing.
expensive-ops	Expensive Operations Log. Disabled by default. Provides only those operations that took longer than 1000 milliseconds to complete.
failed-ops	Failed Operations Log. Provides information on all operations that failed in single-line log format.
replication	Records any replication errors and publishes them to the filesystem.
searches-returning-no-entries	Searches Returning No Entries Log. Disabled by default. Provides information only on operations that did not return any entries.
server.out	Records anything written to standard output or standard error, which includes startup messages. If garbage collection debugging is enabled, then the information will be written to server.out.
server.pid	Stores the server's process ID.
server.status	Stores the timestamp, a status code, and an optional message providing additional information on the server status.
setup.log	Records messages that occur during the initial configuration of a directory server with the **`setup`** tool.
tools	Directory that holds logs for long running utilities: **`import-ldif`**, **`export-ldif`**, **`backup`**, **`restore`**, **`verify-index`**. Current and previous copies of the log are present in the directory server.
update.log	Records messages that occur during a directory server update.

Types of Log Publishers

The UnboundID Directory Server provides several classes of log publishers for parsing, aggregating, and filtering information events that occur during normal processing in the server.

There are three primary types of Log Publishers: access logs, debug logs, and error logs. Each type has multiple subtypes of log files based on the log server type:

TABLE 13-2. Log Publisher Types

Log Publisher Type	Description	
Access	Provides the requests received and responses returned by the Directory Server. The information can be used to understand the operations performed by clients and to debug problems with directory-enabled applications. It can also be used for collecting usage information for performance and capacity planning purposes. There are tools described later that can analyze the access log to provide summaries of the LDAP activity and performance.	
	• File-based Audit Log	Special type of access logger that provides detailed information about changes processed within the server. Disabled by default.
	• JDBC-based Access Log	Stores access log information using a JDBC database connection. Disabled by default.
	• File-based Access Logs	Provides a character-based stream used by TextWriter Publishers as a target for outputting log records. There are six types of file-based loggers: • Admin Alert Access Log. Generates administrative alerts for any operations that match the criteria for this access logger. Disabled by default. • File-based Access Log. Publishes access log messages to the filesystem. Enabled by default. • Syslog-based Access Log. Publishes access log messages to a syslogd port. Disabled by default. • Expensive-Operations Access Log. Publishes only those access log messages of operations that take longer than 1000 milliseconds. Disabled by default. • Failed-Operations Access Log. Publishes only those access log messages of operations that failed for any reason. Enabled by default. • Successful Searches with No Entries Returned Log. Publishes only those access log messages of search operations that failed to return any entries. Disabled by default.
Debug	Provides detailed information about processing performed by the server, including any exceptions caught during processing, detailed information about data read from or written to clients, and accesses to the underlying database.	
	• File-based Debug Logs	Publishes debug messages to the filesystem. Disabled by default.
Error Log	Provides information about warnings, errors, or significant events that occur within the server.	
	• File-based Error Logs	There are two types of File-based Error Logs: • Error log. Publishes error messages to the filesystem. Enabled by default. • Replication log. Publishes replication error messages to the filesystem. Enabled by default.
	• JDBC-based Error Logs	Stores error log information using a JDBC database connection. Disabled by default.
	• Syslog-based Error Logs	Publishes error messages to a `syslogd` port.

Viewing the List of Log Publishers

You can quickly view the list of log publishers on the Directory Server using the **`dsconfig`** tool.

Note Initially, the JDBC, syslog, and Admin Alert log publishers must specifically be configured using **`dsconfig`** before they appear in the list of log publishers. Procedures to configure these types of log publishers appear later in this chapter.

To View the List of Log Publishers

Use **`dsconfig`** to view the log publishers.

```
$ bin/dsconfig list-log-publishers


Log Publisher                                 : Type              : enabled
---------------------------------------------:-------------------:--------
Expensive Operations Access Logger            : file-based-access : false
Failed Operations Access Logger               : file-based-access : true
File-Based Access Logger                      : file-based-access : true
File-Based Audit Logger                       : file-based-audit  : false
File-Based Debug Logger                       : file-based-debug  : false
File-Based Error Logger                       : file-based-error  : true
Replication Repair Logger                     : file-based-error  : true
Successful Searches with No Entries Returned : file-based-access : false
```

Enabling or Disabling a Default Log Publisher

You can enable or disable any log publisher available on the Directory Server using the **`dsconfig`** tool. By default, the following loggers are disabled and should be enabled only when troubleshooting an issue on the server:

- Expensive Operations Access Logger
- File-Based Audit Logger
- File-Based Debug Logger
- Successful Searches with No Entries Returned

To Enable a Default Access Log

Use **`dsconfig`** to enable an Access Log Publisher. In this example, enable the **`Expensive-Ops`** log, which will record only those access log messages that take 1000 milliseconds or longer.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "Expensive Operations Access Logger" \
  --set enabled:true
```

Managing Access Log Publishers

The Access Log records every request received and response returned by the Directory Server. The Access Log stores the IDs for the client connection, operation, the LDAP message involved with each client request, and the server response. The information can be used to debug any problems with a directory application by correlating the numeric operation identifier to the client request or response.

The Directory Server supports multiple classes of access log publishers depending on your logging requirements. The following types of access log publishers are available on the system:

- **File-Based Access Log Publishers**. Provides a character-based TextWriter stream for outputting log records. There are three subclasses of TextWriter access logs:
    - **File-Based Access Logs**. Enabled by default. The File-based Access Log publishes access messages to the filesystem as **`<directory-instance>/logs/access`**. The Failed-Operations Log, Expensive-Operations Log, and the Searches with No Entries Returned Log are specialized types of the File-Based Access Log and shows only specific information necessary for troubleshooting purposes.
    - **Admin-Alert Access Logs**. Disabled by default. The Admin-Alert Access Log is specialized type of logger that automatically generates administrative alerts for any operations that match a criteria for this access log publisher.
    - **Syslog-Based Access Log**s. Disabled by default. The Syslog Access Log publishes access messages to a syslogd port.
- **File-Based Audit Logs**. Disabled by default. The Audit Log provides detailed information about modifications (writes) processed within the Directory Server. The File-based Audit Log publishes access messages to the filesystem as **`<directory-instance>/logs/audit`**.
- **JDBC-Based Access Logs**. Disabled by default. The JDBC-based Access Log provides information using a JDBC database connection.

Managing File-Based Access Log Publishers

The UnboundIDDirectory Server supports a flexible and configurable Access Logging system that provides a full set of customized components to meet your system's logging requirements. The default Access Log can be configured to write information to a log file with two records per operation, one for the request received and one for the response returned. It can also be configured to write one message per operation or configured to record information about a subset of operations processed by the server. In addition to modifying existing default log files, you can create custom log publishers to monitor specific properties or connection criteria. For more information, see "Creating New Log Publishers" on page 453.

The Directory Server can be configured to use multiple access log publishers writing logs to different files using different configurations. This approach makes it possible to have fine-grained logging for various purposes (for example, a log that contains only failed operations, a log that contains only operations requested by root users, or a log that contains only operations that took longer than 20ms to complete).

The Directory Server provides an additional mechanism to filter access logs to record only a subset of messages of one or more types. The access log filtering mechanism uses the operation criteria (connection, request, result, search-entry, search-reference) to determine whether a given message should be logged based on information associated with the client connection as well as information in the operation request/response messages. For more information, see “Configuring Filtered Logging” on page 456.

Access Log Format

The Access Log has a standard format that lists various elements identifying the connection and operation occurring within the Directory Server. By default, each operation generates one access log message.

The Access Log displays the following common properties:

- **Timestamp**. Displays the date and time of the operation. Format: DD/Month/YYYY:HH:MM:SS <offset from UTC time>

- **Connection Type**. Displays the connection type requested by the client and the response by the server. Examples include the following:
  - CONNECT
  - BIND REQUEST/RESULT
  - UNBIND REQUEST
  - DISCONNECT
  - SEARCH REQUEST/RESULT
  - MODIFY REQUEST/RESPONSE
  - others include: ABANDON, ADD, COMPARE, DELETE, EXTENDED OPERATION, MODIFY, MODIFY DN

- **Connection ID**. Numeric identifier, starting incrementally with 0, that identifies the client connection that is requesting the operation. The connection ID is unique for a span of time on a single server. Values of the connection ID will be re-used when the server restarts or when it has had enough connections to cause the identifier to wrap back to zero.

- **Operation ID**. Numeric identifier, starting incrementally with 0, that identifies the operation. The operation ID is unique for a span of time on a single server. Values of the operation ID will be re-used when the server restarts or when it has serviced enough operations to cause the identifier to wrap back to zero.

- **Result Code**. LDAP result code that determines the success or failure of the operation result. Result messages include a `result` element that indicates whether the operation was successful or if failed, the general category for the failure, and an `etime` element that indicates the length of time in milliseconds that the server spent processing the operation.

The Directory Server provides a useful tool **`<server-root>/bin/ldap-result-code`** (UNIX, Linux) or **`<server-root>\bat\ldap-result-code`** (Windows), that displays all of the result codes used in the system. You can use the utility if you are not sure what a result code means. For example, use the following:

- "**`ldap-result-code --list`**" displays all of the defined result codes in the Directory Server.
- "**`ldap-result-code --int-value 16654`**" displays the name of the result code with a numeric value of 16654.
- "**`ldap-result-code --search operation`**" displays a list of all result codes whose name includes the substring "operation".

- **Elapsed Time**. Displays the elapsed time (milliseconds) during which the operation completed its processing.
- **Message ID**. Numeric identifier, starting incrementally with 1, which identifies the LDAP message used to request the operation.

Access Log Example

The following example shows output from the Access Log in **`<directory-instance>/logs/access`**:

```
[01/Jun/2011:14:48:17 -0500] CONNECT conn=0 from="10.8.1.243" to="10.8.1.243" proto-
col="LDAP"
[01/Jun/2011:14:48:17 -0500] BIND REQUEST conn=0 op=0 msgID=1 version="3"
dn="cn=Directory Manager" authType="SIMPLE"
[01/Jun/2011:14:48:17 -0500] BIND RESULT conn=0 op=0 msgID=1 resultCode=0 etime=26.357
authDN="cn=Directory Manager,cn=Root DNs,cn=config"
[01/Jun/2011:14:48:17 -0500] UNBIND REQUEST conn=0 op=1 msgID=2
[01/Jun/2011:14:48:17 -0500] DISCONNECT conn=0 reason="Client Unbind"
... (more output) ...
```

Modifying the Access Log Using dsconfig Interactive Mode

The File-Based Access Log can be modified to include or exclude all log messages of a given type using the **`dsconfig`** tool in interactive or non-interactive mode.

To Modify the Access Log

1. Use **`dsconfig`** in interactive mode to modify the access log properties.

   ```
   $ bin/dsconfig
   ```

2. Follow the prompts to specify the LDAP connection parameters for host name or IP address, connection type (LDAP, SSL, or StartTLS), port number, bind DN and password.

3. On the Directory Server Configuration Console main menu, type the number corresponding to the Log Publisher.

4. On the Log Publisher Management menu, enter the option to view and edit an existing log publisher.

5. On the Log Publisher menu, type the number corresponding to File-based Access Logger.

6. On the File-Based Access Log Publisher menu, type the number corresponding to the property that you want to change, and then follow the prompts.

7. Type **`f`** to apply the changes.

Modifying the Access Log Using dsconfig Non-Interactive Mode

You can use the **`dsconfig`** tool in non-interactive mode to quickly modify a log publisher property on the command line or in a script. For information on each property, see the *Directory Server Configuration Reference Guide*, which is an HTML document listing the various properties for each Directory Server component.

To Modify the Access Log Using dsconfig Non-Interactive Mode

Use **`dsconfig`** with the **`--no-prompt`** option with the properties that you want to modify or set for your access log. In this example, enable the properties to include the instance name and startup ID.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set include-instance-name:true \
  --set include-startup-id:true
```

Modifying the Maximum Length of Log Message Strings

By default, the Directory Server sets the maximum length of log message strings to 2000 characters. This value is configurable for any access log publisher, except the Syslog publisher, which is set to 500 characters. You can change the maximum length of log message strings by setting the **`max-string-length`** configuration property. If any string has more than the configured number of characters, then that string will be truncated and a placeholder will be appended to indicate the number of remaining characters in the original string.

To Modify the Maximum Length of Log Message Strings

Use **`dsconfig`** to set the **`max-string-length`** property for an access log. The following command configures the "File-based Access Logger" to include the instance name and the maximum length of the log message strings to 5000 characters.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set include-instance-name:true \
  --set max-string-length:5000
```

Generating Access Logs Summaries

The Directory Server provides a convenience tool `(summarize-access-log)` that generates a summary of one or more file-based access logs. The summary provides analytical metric information that could be useful for administrators. The following metrics are provided in each summary:

- Total time span covered by the log files.
- Number of connections established and the average rate of new connections per second.
- IP addresses of up to the top 20 of the clients that most frequently connect to the server, the number of connections by each address, and the percentage of connections of each.
- Total number of operations processed, the number of operations of each type, the percentage of each type out of the total number, and the average rate per second for each type of operation.
- Average processing time for all operations and for each type of operation.
- Histogram of the processing times for each type of operation.
- Up to the 20 most common result codes for each type of operation, the number of operations of that type with that result code, and the percentage of operations of that type with that result code.
- Number of unindexed searches processed by the server.
- Breakdown of the scopes used for search operations with the number of percentage of searches with each scope.
- Breakdown of the most common entry counts for search operations with the number and percentage of searches that returned that number of entries.
- Breakdown of the most commonly used filter types for searches with a scope other than "base" (that is, those searches for which the server will use es when processing the filter). These filters will be represented in a generic manner so that any distinct assertion values or substring assertion elements will be replaced with question marks and attribute names in all lowercase characters (for example, `(givenName=John)` would be represented as `(givenName=?)`).

To Generate an Access Log Summary

Use the `bin/summarize-access-log` with path to one or more access log files.

```
$ bin/summarize-access-log /path/to/logs/access

Examining access log /path/to/logs/access
Examined 500 lines in 1 file covering a total duration of 1 day, 22 hours, 57 minutes,
31 seconds

Total connections established:  69 (0.000/second)
Total disconnects:  69 (0.000/second)

Most common client addresses:
127.0.0.1:  61 (88.406)
10.8.1.209:  8 (11.594)

Total operations examined:  181
... (metric for each operation examined) ...

Average operation processing duration:  22.727ms
Average add operation processing duration:  226.600ms
Average bind operation processing duration:  5.721ms
Average delete operation processing duration:  77.692ms
Average modify operation processing duration:  35.530ms
Average search operation processing duration:  4.017ms

Count of add operations by processing time:
... (histogram for add operations) ...

Count of bind operations by processing time:

... (histogram for bind operations) ...

Count of delete operations by processing time:
... (histogram for delete operations) ...

Count of modify operations by processing time:
... (histogram for modify operations) ...

Count of search operations by processing time:
... (histogram for search operations) ...

Most common add operation result codes:
success:  11 (84.615%)
entry already exists:  2 (15.385%)

Most common bind operation result codes:
success:  4 (50.000%)
invalid credentials:  4 (50.000%)

Most common delete operation result codes:
success:  1 (100.000%)

Most common modify operation result codes:
success:  9 (69.231%)
no such object:  4 (30.769%)

Most common search operation result codes:
success:  133 (91.724%)
no such object:  12 (8.276%)

Number of unindexed searches:  0

Most common search scopes:
BASE:  114 (78.621%)
SUB:  16 (11.034%)
ONE:  15 (10.345%)

Most common search entry counts:
1:  119 (82.069%)
0:  17 (11.724%)
2:  5 (3.448%)
10:  4 (2.759%)

Most common generic filters for searches with a non-base scope:
(objectclass=?):  19 (61.290%)
(ds-backend-id=?):  12 (38.710%)
```

About Log Compression

The Directory Server supports the ability to compress log files as they are written. This feature can significantly increase the amount of data that can be stored in a given amount of space, so that log information can be kept for a longer period of time.

Because of the inherent problems with mixing compressed and uncompressed data, compression can only be enabled at the time the logger is created. Compression cannot be turned on or off once the logger is configured. Further, because of problems in trying to append to an existing compressed file, if the server encounters an existing log file at startup, it will rotate that file and begin a new one rather than attempting to append to the previous file.

Compression is performed using the standard gzip algorithm, so compressed log files can be accessed using readily-available tools. The **`summarize-access-log`** tool can also work directly on compressed log files, rather than requiring them to be uncompressed first. However, because it can be useful to have a small amount of uncompressed log data available for troubleshooting purposes, administrators using compressed logging may wish to have a second logger defined that does not use compression and has rotation and retention policies that will minimize the amount of space consumed by those logs, while still making them useful for diagnostic purposes without the need to uncompress the files before examining them.

You can configure compression by setting the **`compression-mechanism`** property to have the value of "**`gzip`**" when creating a new logger.

Creating New Log Publishers

The UnboundID Directory Server provides customization options to help you create your own log publishers with the **`dsconfig`** command.

When you create a new log publisher, you must also configure the log retention and rotation policies for each new publisher. For more information, see “Configuring Log Rotation” on page 455 and “Configuring Log Retention” on page 455.

To Create a New Log Publisher

1. Use the **`dsconfig`** command in non-interactive mode to create and configure the new log publisher. This example shows how to create a logger that only logs disconnect operations.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access --publisher-name "Disconnect Logger" \
  --set enabled:true \
  --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --set "rotation-policy:Size Limit Rotation Policy" \
  --set "retention-policy:File Count Retention Policy" \
  --set log-connects:false \
  --set log-requests:false --set log-results:false \
  --set log-file:logs/disconnect.log
```

Note	To configure compression on the logger, add the option to the previous command: `--set compression-mechanism: gzip` Compression cannot be disabled or turned off once configured for the logger. Therefore, careful planning is required to determine your logging requirements including log rotation and retention with regards to compressed logs.

2. View the Log Publishers.

```
$ bin/dsconfig list-log-publishers

Log Publisher            : Type              : enabled
-------------------------:-------------------:--------
Disconnect Logger        : file-based-access : true
File-Based Access Logger : file-based-access : true
File-Based Audit Logger  : file-based-access : false
File-Based Debug Logger  : file-based-debug  : false
File-Based Error Logger  : file-based-error  : true
Replication Repair Logger : file-based-error  : true
```

To Create a Log Publisher Using dsconfig Interactive Command-Line Mode

1. On the command line, type **`bin/dsconfig`**.
2. Authenticate to the server by following the prompts.
3. On the UnboundID Directory Server Configuration console main menu, select the option to configure the log publisher.
4. On the Log Publisher Management menu, select the option to create a new log publisher.
5. Select the Log Publisher type. In this case, select File-Based Access Log Publisher.
6. Type a name for the log publisher.
7. Enable it.
8. Type the path to the log file, relative to the directory server root. For example, **`logs/disconnect.log`**.
9. Select the rotation policy you want to use for your log publisher.
10. Select the retention policy you want to use for your log publisher.
11. On the Log Publisher Properties menu, select the option for **`log-connects:false`**, **`log-disconnects:true`**, **`log-requests:false`**, and **`log-results:false`**.
12. Type **`f`** to apply the changes.

Configuring Log Rotation

The Directory Server allows you to configure the log rotation policy for the server. When any rotation limit is reached, the Directory Server rotates the current log and starts a new log. If you create a new log publisher, you must configure at least one log rotation policy.

You can select the following properties:

- **Time Limit Rotation Policy**. Rotates the log based on the length of time since the last rotation. Default implementations are provided for rotation every 24 hours and every 7 days.
- **Fixed Time Rotation Policy**. Rotates the logs every day at a specified time (based on 24-hour time). The default time is 2359.
- **Size Limit Rotation Policy**. Rotates the logs when the file reaches the maximum size for each log. The default size limit is 100 MB.
- **Never Rotate Policy**. Used in a rare event that does not require log rotation.

To Configure the Log Rotation Policy

Use `dsconfig` to modify the log rotation policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --remove "rotation-policy:24 Hours Time Limit Rotation Policy" \
  --add "rotation-policy:7 Days Time Limit Rotation Policy"
```

Configuring Log Retention

The Directory Server allows you to configure the log retention policy for each log on the server. When any retention limit is reached, the Directory Server removes the oldest archived log prior to creating a new log. Log retention is only effective if you have a log rotation policy in place. If you create a new log publisher, you must configure at least one log retention policy.

- **File Count Retention Policy**. Sets the number of log files you want the Directory Server to retain. The default file count is 10 logs. If the file count is set to 1, then the log will continue to grow indefinitely without being rotated.
- **Free Disk Space Retention Policy**. Sets the minimum amount of free disk space. The default free disk space is 500 MBytes.
- **Size Limit Retention Policy**. Sets the maximum size of the combined archived logs. The default size limit is 500 MBytes.

- **Custom Retention Policy**. Create a new retention policy that meets your directory server's requirements. This will require developing custom code to implement the desired log retention policy.
- **Never Delete Retention Policy**. Used in a rare event that does not require log deletion.

To Configure the Log Retention Policy

Use **`dsconfig`** to modify the log retention policy for the access logger.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Access Logger" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

Configuring Filtered Logging

The UnboundID Directory Server provides a mechanism to filter access log messages based on specific criteria. The filtered log can then be used with a custom log publisher to create and to generate your own custom logs. Adding new filtered logs and associate publishers does not change the behavior of any existing logs. For example, adding a new log that only contains operations that were unsuccessful does not result in those operations being removed from the default access log.

The following example shows how to create a set of criteria that matches any operation that did not complete successfully. It then explains how to create a custom access log publisher that logs only operations matching that criteria. Note that this log does not include messages for connects or disconnects, and only a single message is logged per operation. This message contains both the request and result details.

To run log filtering based on any operation result (for example, result code, processing time, and response controls), turn off **`request logging`** and set the **`include-request-details-in-result-messages`** property to **`TRUE`**. Since filtering based on the results of an operation cannot be done until the operation completes, the server has no idea whether to log the request. Therefore, it might log request messages but not log any result messages. Instead, if you can only log result messages and include request details in the result messages, then only messages for operations that match the result criteria are logged. All pertinent information about the corresponding requests are included.

To Configure a Filtered Log Publisher

1. Use the **`dsconfig`** command in non-interactive mode to create a result criteria object set to failure-result-codes, a predefined set of result codes that indicate when an operation did not complete successfully.

   ```
   $ bin/dsconfig create-result-criteria --type simple \
     --criteria-name "Failed Operations"
     --set result-code-criteria:failure-result-codes
   ```

2. Use **dsconfig** to create the corresponding log publisher that uses the result criteria. The log rotation and retention policies are also set with this command.

```
$ bin/dsconfig create-log-publisher \
  --type file-based-access \
  --publisher-name "Filtered Failed Operations" \
  --set enabled:true \
  --set log-connects:false \
  --set log-disconnects:false \
  --set log-requests:false \
  --set "result-criteria:Failed Operations" \
  --set log-file:logs/failed-ops.log \
  --set include-request-details-in-result-messages:true \
  --set "rotation-policy:7 Days Time Limit Rotation Policy" \
  --set "retention-policy:Free Disk Space Retention Policy"
```

3. View the **failed-ops.log** in the **logs** directory. Verify that only information about failed operations is written to it.

Managing Admin Alert Access Logs

Admin Alert Access Logs are a specialized form of filtered log that automatically generates an administrative alert when criteria configured for the log publisher matches those messages in the access log.

About Access Log Criteria

Configuring an Admin Alert Access Log requires that you configure the criteria for the access log messages. Each criteria can be either a *Simple* or an *Aggregate* type. The Simple type uses the set of properties for the client connection, operation request, and the contents of any operation-specific requests or results. The Aggregate type provides criteria that contains Boolean combination of other operation-specific criteria objects. For more information, see the *Directory Server Configuration Reference*.

The criteria can be one or more of the following:

- **Connection Criteria**. Defines sets of criteria for grouping and describing client connection based on a number of properties, including protocol, client address, connection security, and authentication state.
- **Request Criteria**. Defines sets of criteria for grouping and describing operation requests based on a number of properties, including properties for the associated client connection, the type of operation, targeted entry, request controls, target attributes, and other operation-specific terms.
- **Result Criteria**. Defines sets of criteria for grouping and describing operation results based on a number of properties, including the associated client connection and operation

request, the result code, response controls, privileges missing or used, and other operation-specific terms.

- **Search Entry Criteria**. Defines sets of criteria for grouping and describing search result entries based on a number of properties, including the associated client connection and operation request, the entry location and contents, and included controls.
- **Search Reference Criteria**. Defines sets of criteria for grouping and describing search result references based on a number of properties, including the associated client connection and operation request, reference contents, and included controls.

Configuring an Admin Alert Access Log Publisher

Prior to configuring an Admin Alert Access Log, you must establish an administrative alert handler in your system. For more information, see "Working with Administrative Alert Handlers" on page 497.

To Configure an Admin Alert Access Log Publisher

1. Use **`dsconfig`** to create a criteria object for the Admin Alert Access Log. For this example, we want to log only write operations that target user entries. The following command matches any of the specified operations whose target entry matches the filter **`"(objectClass=person)"`**.

```
$ bin/dsconfig create-request-criteria --type simple \
  --criteria-name "User Updates" \
  --set operation-type:add \
  --set operation-type:delete \
  --set operation-type:modify \
  --set operation-type:modify-dn \
  --set "any-included-target-entry-filter:(objectClass=person)"
```

Note	If you are using the **`dsconfig`** tool in interactive mode, the menu items for the criteria operations are located in the Standard objects menu.

2. Use **`dsconfig`** to create a log publisher of type **`admin-alert-access`**.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "User Updates Admin Alert Access Log" \
  --type admin-alert-access \
  --set "request-criteria:User Updates" \
  --set include-request-details-in-result-messages:true \
  --set enabled:true
```

Managing Syslog-Based Access Log Publishers

The Directory Server supports access logging using the syslog protocol that has been part of Berkeley Software Distribution (BSD) operating system for many years. Syslog provides a flexible, albeit simple, means to generate, store and transfer log messages that is supported on most UN*X and Linux operating systems.

The quasi-standard syslog message format cannot exceed 1 kbytes and has three important parts:

- **PRI.** Specifies the message priority based on its facility and severity. The message facility is a numeric identifier that specifies the type of log messages, such as kernel messages, mail system messages, etc. The severity is a numeric identifier that specifies the severity level of the operation that is being reported. The following values are for Solaris systems. For Linux systems, refer to the **`syslog.h`** file for the specific numbers on their target operating system:

  - 0 (emergency level)
  - 1 (Alert: action must be taken immediately)
  - 2 (Critical: critical conditions)
  - 3 (Error: error conditions)
  - 4 (Warning: warning conditions)
  - 5 (Notice: normal but significant condition)
  - 6 (Informational: informational messages)
  - 7 (Debug: debug level messages)

  Together, the facility and the severity determine the priority of the log message indicated by angled brackets and 1-3 digit priority number. For example, "<0>", "<13>", "<103>" are valid representations of the PRI.

- **Timestamp and Host Name**. The timestamp displays the current date and time of the log. The host name or IP address displays the source of the log.

- **Message**. Displays the actual log message.

Administrators can configure syslog to handle log messages using log priorities that are based on the message's facility and severity. This feature allows users to configure the logging system in such a way that messages with high severities can be sent to a centralized repository, while lower severity messages can be stored locally on a server.

Note	Since the numeric values of the severity and facility are operating system-dependent, the central repository must only include syslog messages from compatible OS types, otherwise the meanings of the PRI field is ambiguous.

Before You Begin

You will need to identify the host name and port to which you want to connect. Because the Syslog Protocol uses User Datagram Protocol (UDP) packets, we highly recommend that you use **`localhost`** and utilize some additional logging tools, such as **`syslog-ng`**. UDP is unreliable and non-secure means to transfer data packets between hosts.

Default Access Log Severity Level

All messages are logged at the syslog severity level of 6, which is *Informational: informational messages*. Note that this value is not standard across different types of UNIX or Linux systems. Please consult your particular operating system.

Syslog Facility Properties

When using syslog, you have to specify a facility for the access log messages. As an advanced property, you can select a number that corresponds to the facility you wish to use. The default value for the syslog-facility property is 1 (one) for user-level messages. The following facility properties are available on the Directory Server. Note that this value is not standard across different types of UNIX or Linux systems. Please consult your particular operating system documentation.

TABLE 13-3. Syslog Facility Properties (Solaris)

Facility	Description
0	kernel messages
1	user-level messages (default)
2	mail system
3	system daemons
4	security/authorization messages
5	messages generated internally by syslogd
6	line printer subsystem
7	network news subsystem
8	UUCP subsystem
9	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon
16	local use 0
17	local use 1
18	local use 2
19	local use 3
20	local use 4
21	local use 5
22	local use 6
23	local use 7

Queue-Size Property

The maximum number of log records that can be stored in the asynchronous queue is determined by the **`queue-size`** property. The default queue size set is 10000, which means that the server will continuously flush messages from the queue to the log. The server does not wait for the queue to fill up before flushing to the log. Therefore, lowering this value can impact performance.

Configuring a Syslog-Based Access Log Publisher

You can configure a Syslog-based Access Log Publisher using the **`dsconfig`** tool. We recommend that you use syslog locally on **`localhost`** and use **`syslog-ng`** to send the syslog messages to remote syslog servers.

Because syslog implementations differ by vendor, please review your particular vendor's syslog configuration.

To Configure a Syslog-Based Access Log Publisher

Use **`dsconfig`** to create a log publisher of type **`syslog-based-access`**.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "syslog-access" \
  --type syslog-based-access \
  --set syslog-facility:4 \
  --set enabled:true
```

Note	If you are using the **`dsconfig`** tool in interactive mode, the menu item for Syslog Facility is an advanced property, which can be exposed by typing **`a`** (for "show advanced properties") on the Syslog-Based Access Log Publisher menu.

Managing the File-Based Audit Log Publishers

The Directory Server provides an audit log, a specialized version of the access log, for troubleshooting problems that may occur in the course of processing. The log records all changes to directory data in LDIF format so that administrators can quickly diagnose the changes an application made to the data or replay the changes to another server for testing purposes.

The audit log does not record authentication attempts but can be used in conjunction with the access log to troubleshoot security-related issues. The audit log is disabled by default because it does adversely impact the server's write performance.

Audit Log Format

The audit log uses standard LDIF format, so that administrators can quickly analyze what changes occurred to the directory data. The audit log begins logging when enabled and should be used to debug any issues that may have occurred. Some common properties are the following:

- **Timestamp**. Displays the date and time of the operation. Format: DD/Month/YYYY:HH:MM:SS <offset from UTC time>
- **Connection ID**. Numeric identifier, starting incrementally with 0, that identifies the client connection that is requesting the operation.
- **Operation ID**. Numeric identifier, starting incrementally with 0, that identifies the operation.
- **Modifiers Name**. Displays the DN of the user who made the change.
- **Update Time**. Records the `ds-update-time` operational attribute.

Audit Log Example

The following example shows output from the Audit Log in the `<directory-instance>/logs/audit`. The first entry shows when the audit log was enabled. The second entry show changes made to a user entry.

```
# 05/Jun/2011:10:29:04 -0500; conn=0; op=55
dn: cn=File-Based Audit Logger,cn=Loggers,cn=config
changetype: modify
replace: ds-cfg-enabled
ds-cfg-enabled: true
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: ds-update-time
ds-update-time:: AAABIbEJlCw=
# 05/Jun/2011:10:31:20 -0500; conn=2; op=1
dn: uid=user.996,ou=People,dc=example,dc=com
changetype: modify
replace: pager
pager: +1 115 426 4748
-
replace: homePhone
homePhone: +1 407 383 4949
-
replace: modifiersName
modifiersName: cn=Directory Manager,cn=Root DNs,cn=config
-
replace: ds-update-time
ds-update-time:: AAABIbELqd4=
```

Enabling the File-Based Audit Log Publisher

You can enable the File-Based Audit Log Publisher using the **`dsconfig`** tool. The audit log can impact the Directory Server's write performance, so enable it only when troubleshooting any issues.

To Enable the File-Based Audit Log Publisher

Use **`dsconfig`** to enable the File-Based Audit Log Publisher. For this example, the instance name and startup ID is also enabled in the audit log.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Audit Logger" \
  --set enabled:true \
  --set include-instance-name:true \
  --set include-startup-id:true
```

Managing the JDBC Access Log Publishers

The Directory Server supports the Java Database Connectivity (JDBC™) API, which allows access to SQL data stores by means of its JDBC drivers. The JDBC 4.0 API, part of the Java 6 SDK, provides a seamless method to interface with various database types in heterogeneous environments.

By easily connecting to a database, the Directory Server can be configured to implement a centralized logging system with different databases. Centralized logging simplifies log correlation and analysis tasks and provides security by storing data in a single repository. Some disadvantages of centralized logging are that data flow asymmetries may complicate synchronization or network provisioning and may unduly burden the central repository with possibly heavy loads.

Before You Begin

Before configuring the JDBC Access Log Publisher, you need to carry out two essential steps to set up the database.

- Install the database drivers in the Directory Server **`lib`** directory.
- Define the log mapping tables needed to map access log elements to the database column data. Only those elements in the log mapping table gets logged by the JDBC Log Publisher.

The following sections provide more information about these tasks.

Configuring the JDBC Drivers

The Directory Server supports a number of JDBC drivers available in the market. We highly recommend using the JDBC 4 drivers supported in the Java 6 platform. For example, for Oracle databases, you need to use the **`ojdbc6.jar`** driver for Java 6 and any associated JAR files (e.g., National Language Support jars, and others) required to connect with the particular database. The following databases are supported:

- DB2
- MySQL
- Oracle Call Interface (OCI)
- Oracle Thin
- PostgreSQL
- SQL Server

To Configure the JDBC Driver

Obtain the JAR file(s) for your particular database, and copy it into the **`<directory-instance>/lib`** directory.

Configuring the Log Field Mapping Tables

The log field mapping table associates access log fields to the database column names. You can configure the log field mapping table using the **`dsconfig`** tool, which then generates a DDL file that you can import into your database. The DDLfile is generated when you create the JDBC Log Publisher.

To uniquely identify a log record, we recommend always mapping the following fields: **`timestamp`**, **`startupid`**, **`message-type`**, **`connection-id`**, **`operation-type`**, **`instance-name`**.

The table name is not part of this mapping.

The Directory Server also provides three options that you can quickly select for creating a log field mapping table:

- Complete JDBC Log Field Mappings. Maps all 52 object properties.
- Simple JDBC Log Field Mappings. Maps a common set of object properties.
- Custom JDBC Log Field Mappings. Create a custom set of JDBC log field mappings.

To Configure the Log Field Mapping Tables

1. Use **`dsconfig`** to create a log field mapping table. On the Directory Server Configuration Console main menu, type "o" to change to the Standard Object menu, and type the number corresponding to Log Field Mapping.

2. On the Log Field Mapping management menu, enter the option to create a new Log Field Mapping.

3. On the Log Field Mapping template menu, enter the option to select a complete JDBC Access Log Field mapping to use as a template for your new field mapping.

4. Next, enter a name for the new field mapping. In this example, type "my-jdbc-test".

5. On the Access Log Field Mapping Properties menu, type a property if you want to change the value. Any property that is undefined will not be logged by the JDBC Access Log Publisher. When complete, type `f` to save and apply the changes.

6. On the Log Field Mapping Management menu, type `q` to exit the menu.

7. View the existing Log Mappings on the system.

```
$ bin/dsconfig list-log-field-mappings

Log Field Mapping                        : Type
-----------------------------------------:-------
Complete JDBC Access Log Field Mappings : access
Complete JDBC Error Log Field Mappings  : error
my-jdbc-test                            : access
Simple JDBC Access Log Field Mappings   : access
```

Configuring the JDBC Access Log Publisher using dsconfig Interactive Mode

After setting up the drivers and the log mapping table, use the **`dsconfig`** utility to configure the JDBC Access Log Publisher on the Directory Server. The following example uses **`dsconfig`** interactive mode to illustrate the steps required to configure the log publisher and the external database server.

To Configure the JDBC Access Log Publisher

1. Copy the database JAR files to the **`<directory-instance>/lib`** directory, and then restart the Directory Server.

2. Launch the **`dsconfig`** tool in interactive command-line mode.

```
$ bin/dsconfig
```

3. Next, type the connection parameters to bind to the Directory Server. Enter the directory host name or IP address, type of LDAP connection (LDAP, SSL, or StartTLS) that you are using on the Directory Server, the LDAP listener port number, the user bind DN, and the bind DN password.

4. On the Directory Server Configuration console main menu, type the number corresponding to Log Publisher.

5. On the Log Publisher management menu, enter the option to create a new Log Publisher.

6. On the Log Publisher template menu, type `n` to create a new Log Publisher.

7. On the Log Publisher Type menu, enter the option to create a new JDBC-Based Access Log Publisher.

8. Type a name for the JDBC Access Log Publisher.

9. On the Enabled Property menu, enter the option to enable the log publisher.

10. On the Server Property menu, enter the option to create a new JDBC External Server.

11. Next, type the name for the JDBC External Server. This is a symbolic name used to represent the DBMS.

12. On the JDBC Driver Type Property menu, type the number corresponding to the type of JDBC database driver type.

13. Next, type a name for the **`database-name`** property. This is the DBMS database name. The database name must contain the table referred to in the generated DDL.

14. Next, type the host name or IP address (server-host-name) of the external server.

15. Type the server listener port. In this example, type 1541.

16. Review the properties for the external server, and the type **`f`** to apply the changes.

17. If you need to supply your own JDBC URL, type **`a`** for advanced properties to open the **`jdbc-driver-url`** property and supply the appropriate URL. The example below shows how to access an Oracle Thin Client connection using a SID instead of a Service.

```
>>>> Configure the properties of the JDBC External Server

        Property          Value(s)
        ----------------------------------------------------
    1)  description       -
    2)  jdbc-driver-type  oraclethin
    3)  jdbc-driver-url   jdbc:oracle:thin@myhost:1541:my_SID
    4)  database-name     jdbc-test
    5)  server-host-name  localhost
    6)  server-port       1541
    7)  user-name         -
    8)  password          -

    ?)  help
    f)  finish - create the new JDBC External Server
    a)  hide advanced properties of the JDBC External Server
    d)  display the equivalent dsconfig arguments to create this object
    b)  back
    q)  quit

Enter choice [b]: f

JDBC External Server was created successfully
```

18. When the JDBC Log Publisher is created, the Directory Server automatically generates a DDL file of the Log Field Mappings in the **`<directory-instance>/logs/ddls/<name-of-logger>.sql`** file. Import the DDL file to your database.

Configuring the JDBC Access Log Publisher Using dsconfig Non-Interactive Mode

The following example uses **`dsconfig`** non-interactive mode to illustrate the steps to configure the log publisher and the external database server presented in the previous section.

To Configure the JDBC Access Log Publisher in Non-Interactive Mode

1. Use `dsconfig` with the `--no-prompt` option to create the JDBC external server.

```
$ bin/dsconfig --no-prompt create-external-server \
  --server-name jdbc-external \
  --type jdbc \
  --set jdbc-driver-type:oraclethin \
  --set database-name:ubid_access_log \
  --set server-host-name:localhost --set server-port:1541
```

2. Use `dsconfig` to create the log publisher.

```
$ bin/dsconfig --no-prompt create-log-publisher \
  --publisher-name jdbc-test \
  --type jdbc-based-access \
  --set enabled:true \
  --set server:jdbc-external \
  --set "log-field-mapping:Simple JDBC Access Log Field Mappings"
```

3. When the JDBC Log Publisher is created, the Directory Server automatically generates a DDL file of the Log Field Mappings in the `<directory-instance>/logs/ddls/<name-of-logger>.sql` file. Import the DDL file to your database.

Note

The procedure to configure the JDBC-Based Error Log Publisher is similar to creating a JDBC-Based Access Log Publisher. You can run the previous `dsconfig` command with the `--type jdbc-based-error` as follows:

```
$ bin/dsconfig --no-prompt create-log-publisher \
  --publisher-name jdbc-error-test \
  --type jdbc-based-error \
  --set enabled:true \
  --set server:jdbc-external \
  --set "log-field-mapping:Simple JDBC Access Log Field Map-
pings"
```

Managing the File-Based Error Log Publisher

The Error Log reports errors, warnings, and informational messages about events that occur during the course of the Directory Server's operation. Each entry in the error log records the following properties (some are disabled by default and must be enabled):

- **Time Stamp**. Displays the date and time of the operation. Format: DD/Month/YYYY:HH:MM:SS <offset from UTC time>
- **Category**. Specifies the message category that is loosely based on the server components.
- **Severity**. Specifies the message severity of the event, which defines the importance of the message in terms of major errors that need to be quickly addressed. The default severity levels are: fatal-error, notice, severe-error, severe-warning.

- **Message ID**. Specifies the numeric identifier of the message.
- **Message**. Stores the error, warning, or informational message.

Error Log Example

The following example displays the error log for the Directory Server. The log enabled by default and is accessible in the **`<directory-instance>/logs/errors`** file.

```
[05/Jun/2011:14:42:22 -0500] category=RUNTIME_INFORMATION severity=NOTICE msgID=20381715 msg="JVM
Arguments: '-Xserver', '-Xmx256m', '-Xms256m', '-XX:+UseConcMarkSweepGC', '-XX:+CMSConcurrentMTEn-
abled', '-XX:+CMSParallelRemarkEnabled', '-XX:+CMSParallelSurvivorRemarkEnabled', '-XX:ParallelC-
MSThreads=1', '-XX:CMSMaxAbortablePrecleanTime=10', '-XX:CMSInitiatingOccupancyFraction=80', '-
XX:+UseParNewGC', '-XX:+UseMembar', '-XX:+UseBiasedLocking', '-XX:+HeapDumpOnOutOfMemoryError', '-
Dcom.unboundid.directory.server.scriptName=setup'"
[05/Jun/2011:14:42:23 -0500] category=JEB severity=NOTICE msgID=8847402 msg="The database backend
userRoot using Berkeley DB Java Edition 3.3.82 and containing 2003 entries has started"
[05/Jun/2011:14:42:25 -0500] category=CORE severity=NOTICE msgID=458887 msg="The Directory Server
has started successfully"
```

To Modify the File-Based Error Logs

Use **`dsconfig`** to modify the default File-Based Error Log.

```
$ bin/dsconfig set-log-publisher-prop \
  --publisher-name "File-Based Error Logger" \
  --set include-product-name:true --set include-instance-name:true \
  --set include-startup-id:true
```

Managing the Syslog-Based Error Log Publisher

The Directory Server supports a Syslog-Based Error Log Publisher using the same mechanism as the Syslog-Based Access Log Publisher. You can easily configure the error logger using the **`dsconfig`** tool.

Syslog Error Mapping

The Directory Server automatically maps error log severities to the syslog severities. The following mappings are used:

TABLE 13-4. **Error to Syslog Severities Mappings to Syslog**

Error Log Severity	Syslog Severity
FATAL_ERROR,0	Syslog Emergency
SEVERE_ERROR,1	Syslog Alert
SEVERE_WARNING,2	Syslog Critical
MILD_ERROR,3	Syslog Error
MILD_WARNING,4	Syslog Warn

TABLE 13-4. **Error to Syslog Severities Mappings to Syslog**

Error Log Severity	Syslog Severity
NOTICE,5	Syslog Notice
INFORMATION,6	Syslog Info
DEBUG,7	Syslog Debug

Configuring a Syslog-Based Error Log Publisher

You can configure a Syslog-based Error Log Publisher using the **`dsconfig`** tool. Again, we recommend that you use syslog locally on **`localhost`** and use **`syslog-ng`** to send the data packets over the UDP protocol.

Because syslog implementations differ by vendor, please review your particular vendor's syslog configuration.

To Configure a Syslog-Based Error Log Publisher

Use **`dsconfig`** to create a log publisher of type **`syslog-based-error`**. In this example, set the syslog facility to 4 for security/authorization messages.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "syslog-error" --type syslog-based-error \
  --set syslog-facility:4 --set enabled:true
```

Creating a File-Based Debug Log Publishers

The Directory Server provides a File-Based Debug Log Publisher that can be configured when troubleshooting a problem that might occur during server processing. Because the debug data may be too large to maintain during normal operations, the Debug Log Publisher must be specifically configured and enabled. The Debug Log reports the following types of information:

- Exception data thrown by the server.
- Data read or written to network clients.
- Data read or written to the database.

Access control or password policy data made within the server.

You can use the **`dsconfig`** tool to create a debug log publisher. You should only create a debug logger when troubleshooting a problem due to the voluminous output the Directory Server generates.

To Create a File-Based Debug Log Publisher

Use **`dsconfig`** to create the debug log publisher.

```
$ bin/dsconfig create-log-publisher \
  --publisher-name "File-Based Debug Logger" \
  --type file-based-debug \
  --set enabled:true \
  --set log-file:/logs/debug
```

To Delete a File-Based Debug Log Publisher

Use **dsconfig** to create the debug log publisher.

```
$ bin/dsconfig delete-log-publisher \
  --publisher-name "File-Based Debug Logger"
```

Monitoring the Directory Server

The UnboundID Directory Server exposes its monitoring information under the **cn=monitor** entry for easy access to its information. Administrators can use various means to monitor the server's information including the Directory Management Console, JConsole, LDAP command-line tools, and through SNMP.

The following monitoring components are accessible:

TABLE 13-5. **Directory Server Monitoring Components**

Component	Description
Active Operations	Provides information about the operations currently being processed by the Directory Server. Shows the number of operations, information on each operation, and the number of active persistent searches.
Backends	Provides general information about the state of a Directory Server backend, including the backend ID, base DN(s), entry counts, entry count for the **cn=admin data**, writability mode, and whether it is a private backend. The following backend monitors are provided: • adminRoot • ads-truststore • alerts • backup • config • monitor • schema • tasks • userRoot
Berkeley DB JE Environment	Provides information about the state of the Oracle Berkeley DB Java Edition database used by the Directory Server backend. Most of the statistics are obtained directly from the Berkeley DB JE.
Client Connections	Provides information about all client connections to the Directory Server. The client connection information contains a name followed by an equal sign and a quoted value (e.g., connID="15", connectTime="20100308223038Z", etc.)

TABLE 13-5. **Directory Server Monitoring Components**

Component	Description
Connection Handlers	Provides information about the available connection handlers on the Directory Server, which includes the HTTP, LDAP and LDIF connection handlers. These handlers are used to accept client connections and to read requests and send responses to those clients.
Disk Space Usage	Provides information about the disk space available to various components of the Directory Server.
General	Provides general information about the state of the Directory Server, including product name, vendor name, server version, etc.
Index	Provides information on each index. The monitor captures the number of keys preloaded, and counters for read/write/remove/open-cursor/read-for-search. These counters provide insight into how useful an index is for a given workload.
JVM Stack Trace	Provides a stack trace of all threads processing within the JVM.
LDAP Connection Handler Statistics	Provides statistics about the interaction that the associated LDAP connection handler has had with its clients, including the number of connections established and closed, bytes read and written, LDAP messages read and written, operations initiated, completed, and abandoned, etc.
Processing Time Histogram	Categorizes operation processing times into a number of user-defined buckets of information, including the total number of operations processed, overall average response time (ms), number of processing times between 0ms and 1ms, etc.
System Information	Provides general information about the system and the JVM on which the Directory Server is running, including system host name, operation system, JVM architecture, Java home, Java version, etc.
Version	Provides information about the Directory Server version, including build ID, version, revision number, etc.
Work Queue	Provides information about the state of the Directory Server work queue, which holds requests until they can be processed by a worker thread, including the requests rejected, current work queue size, number of worker threads, number of busy worker threads, etc.

Monitoring Disk Space Usage

The disk space usage monitor provides information about the amount of usable disk space available for Directory Server components. It also provides the ability to generate administrative alerts, as well as take additional action if the amount of usable space drops below the defined thresholds.

You can configure three thresholds for this monitor:

- Low space warning threshold. This threshold is defined as either a percentage or absolute amount of usable space. If the amount of usable space drops below this threshold, then the Directory Server will generate an administrative alert but will remain fully functional. It will generate alerts at regular intervals that you configure (such as once a day) unless action is taken to increase the amount of usable space, and will also generate additional alerts as the amount of usable space is further reduced (e.g., each time the amount of usable space drops below a value 10% closer to the low space error threshold). If an administrator frees up disk space or adds additional capacity, then the server should automatically recognize this and stop generating alerts.

- Low space error threshold. This threshold is also defined as either a percentage or absolute size. Once the amount of usable space drops below this threshold, then the server will generate an alert notification and will begin rejecting all operations requested by non-root users with "UNAVAILABLE" results. The server should continue to generate alerts during this time. We may want to consider suspending access logging for rejected operations from non-root users, and we may also want to consider suspending accepting changes from replication. Once the server enters this mode, then an administrator will have to take some kind of action (e.g., running a command to invoke a task or removing a signal file) before the server will resume normal operation. This threshold must be less than or equal to the low space warning threshold, and if they are equal, then the server will begin rejecting requests from non-root users immediately upon detecting low usable disk space.

- Out of space error threshold. This threshold may also be defined as a percentage or absolute size. Once the amount of usable space drops below this threshold, then the Directory Server will generate a final administrative alert and will shut itself down. This threshold must be less than or equal to the low space error threshold, and if they are equal then the server will shut itself down rather than rejecting requests from non-root users.

The threshold values may be specified either as absolute sizes or as percentages of the total available disk space. Either all values must be specified as absolute values or the must all be specified as percentages; a mix of absolute values and percentages cannot be used. The low space warning threshold must be greater than or equal to the low space error threshold, the low space error threshold must be greater than or equal to the out of space error threshold, and the out of space error threshold must be greater than or equal to zero.

If the out of space error threshold is set to zero, then the server will not attempt to automatically shut itself down if it detects that usable disk space has become critically low. If the amount of usable space reaches zero, then the database will preserve its integrity but may enter a state in which it rejects all operations with an error and requires the server (or at least the affected backends) to be restarted. If the low space error threshold is also set to zero, then the server will generate periodic warnings about low available disk space but will remain fully functional for as long as possible. If all three threshold values are set to zero, then the server will not attempt to warn about or otherwise react to a lack of usable disk space.

Monitoring Using SNMP

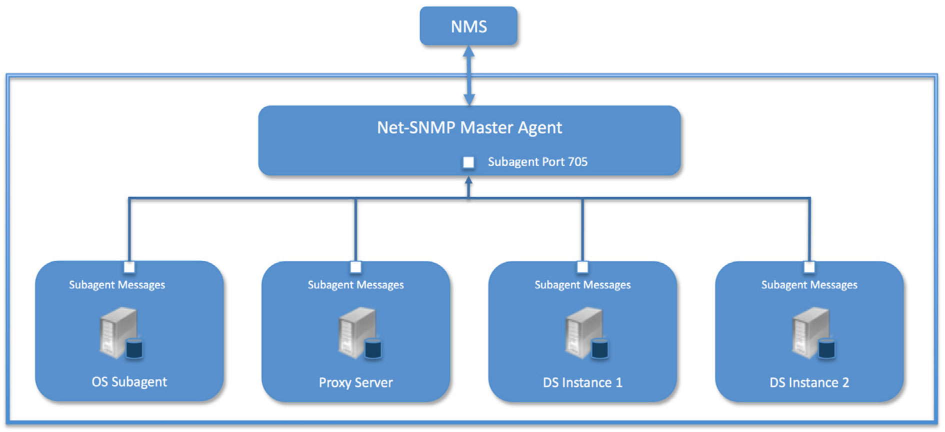
The UnboundID Directory Server supports real-time monitoring using the Simple Network Management Protocol (SNMP). The Directory Server provides an embedded SNMPv3 subagent plug-in that, when enabled, sets up the server as a managed device and exchanges monitoring information with a master agent based on the AgentX protocol.

Directory Server Implementation

In a typical SNMP deployment, many production environments use a network management system (NMS) for a unified monitoring and administrative view of all SNMP-enabled devices. The NMS communicates with a master agent, whose main responsibility is to translate the SNMP protocol messages and multiplex any request messages to the subagent on each managed device (for example, Directory Server instance, Directory Proxy Server, or OS Subagent). The master agent also processes responses or traps from the agents. Many vendors provide commercial NMS systems, such as Alcatel-Lucent (Omnivista EMS), HP (OpenView), IBM-Tivoli (Netview), Oracle-Sun (Solstice Enterprise Manager), and others. Specific discussion on integrating an SNMP deployment on an NMS system is beyond the scope of this chapter. Consult with your NMS system for specific information.

The UnboundID Directory Server contains an SNMP subagent plug-in that connects to a Net-SNMP master agent over TCP. The main configuration properties of the plug-in are the address and port of the master agent, which default to `localhost` and port `705`, respectively. When the plug-in is initialized, it creates an AgentX subagent and a managed object server, and then registers as a MIB server with the Directory Server instance. Once the plug-in's startup method is called, it starts a session thread with the master agent. Whenever the connection is lost, the subagent automatically attempts to reconnect with the master agent. The Directory Server's SNMP subagent plug-in only transmits read-only values for polling or trap purposes (set and inform operations are not supported). SNMP management applications cannot perform actions on the server on their own or by means of an NMS system.

FIGURE 13-1. Example SNMP Deployment



One important note is that the UnboundID Directory Server was designed to interface with a Net-SNMP (version 5.3.2.2 or later) master agent implementation with AgentX over TCP. Many operating systems provide their own Net-SNMP module, such as the System Management Agent (SMA) on Solaris or OpenSolaris. However, SMA disables some features present in the Net-SNMP package and only enables AgentX over UNIX Domain Sockets, which cannot be supported by Java. If your operating system has a native Net-SNMP master agent that only enables UNIX Domain Sockets, you must download and install a separate Net-SNMP binary from its web site.

Configuring SNMP

Because all server instances provide information for a common set of MIBs, each server instance provides its information under a unique SNMPv3 context name, equal to the server instance name. The server instance name is defined in the Global Configuration, and is constructed from the host name and the server LDAP port by default. Consequently, information must be requested using SNMPv3, specifying the context name that pertains to the desired server instance. This context name is limited to 30 characters or less. Any context name longer than 30 characters will result in an error message. Since the default context name is limited to 30 characters or less, and defaults to the server instance name and the LDAP port number, pay special attention to the length of the fully-qualified (DNS) hostname.

Note The Directory Server supports SNMPv3, and only SNMPv3 can access the MIBs. For systems that implement SNMP v1 and v2c, Net-SNMP provides a proxy function to route requests in one version of SNMP to an agent using a different SNMP version.

To Configure SNMP

1. Enable the Directory Server's SNMP plug-in using the **dsconfig** tool. Make sure to specify the address and port of the SNMP master agent. On each Directory Server instance, enable the SNMP subagent. Note that the SNMPv3 context name is limited to 30 bytes maximum. If the default dynamically-constructed instance name is greater than 30 bytes, there will be an error when attempting to enable the plug-in.

```
$ bin/dsconfig set-plugin-prop --plugin-name "SNMP Subagent" --set enabled:true \
  --set agentx-address:localhost --set agentx-port:705 --set session-timeout:5s \
  --set connect-retry-max-wait:10s
```

2. Enable the SNMP Subagent Alert Handler so that the sub-agent will send traps for administrative alerts generated by the server.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true
```

3. View the error log. You will see a message that the master agent is not connected, because it is not yet online.

```
The SNMP sub-agent was unable to connect to the master agent at localhost/705:
Timeout
```

4. Install Net-SNMP (version 5.3.2.2 or later) on your machine. This step will vary depending on the OS. On Linux, the SNMP install should be compiled to allow AgentX over TCP. You can verify that it is compiled with TCP by installing the **net-snmp-devel** package and running this command:

```
$ net-snmp-config --configure-options
```

The output of that command should NOT contain **--enable-agentx-dom-sock-only**.

5. Edit the SNMP agent configuration file, **snmpd.conf**, which is often located in **/usr/local/share/snmp/snmpd.conf**. Add the directive to run the agent as an AgentX master agent:

```
master agentx
agentXSocket tcp:localhost:705
```

   Note that the use of localhost means that only subagents running on the same host can connect to the master agent. This requirement is necessary since there are no security mechanisms in the AgentX protocol.

6. Add the trap directive to send SNMPv2 traps to localhost with the community name, *public* (or whatever SNMP community has been configured for your environment).

```
trap2sink localhost public
```

7. Create an SNMPv3 user.

```
$ net-snmp-config --create-snmpv3-user -A password snmpuser
```

8. Start the **snmpd** daemon and after a few seconds you should see the following message in the Directory Server error log:

```
The SNMP subagent connected successfully to the master agent at localhost:705. The
SNMP context name is host.example.com:389
```

   At this point SNMP is up and running.

9. Set up a trap client to see the alerts that are generated by the Directory Server. Create a config file in **/tmp/snmptrapd.conf** and add the directive below to it. The directive specifies that the trap client can process traps using the **public** community string, and can log and trigger executable actions.

```
authcommunity log, execute public
```

10. Then, run the trap client using the **snmptrapd** command. The following example specifies that the command should not create a new process using **fork()** from the calling shell (**-f**), do not read any configuration files (**-C**) except the one specified with the **-c** option, print to standard output (**-Lo**), and then specify that debugging output should be turned on for the User-based Security Module (**-Dusm**).

```
$ snmptrapd -f -C -c /tmp/snmptrapd.conf -Lo -Dusm
```

11. Install the MIB definitions for the Net-SNMP client tools, usually located in the **/usr/share/snmp/mibs** directory.

```
$ cp resource/mib/* /usr/share/snmp/mibs
```

12. Run the Net-SNMP client tools to test the feature. The following options are required: **-v <SNMP version>**, **-u <user name>**, **-A <user password>**, **-l <security level>**, **-n <context name (instance name)>** . The **-m all** option loads all MIBs in the default MIB directory in **/usr/share/snmp/mibs** so that MIB names can be used in place of numeric OIDs.

```
$ snmpget -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389 \
-m all localhost localDBBackendCount.0
```

```
$ snmpwalk -v 3 -u snmpuser -A password -l authNoPriv -n host.example.com:389 \
-m all localhost systemStatus
```

13. If you want alerts sent from the SNMP Subagent through the Net-SNMP master agent and onwards, you must enable the SNMP Subagent Alert Handler. The SNMP Alert Handler is used in deployments that do not enable the Subagent.

```
$ bin/dsconfig --no-prompt set-alert-handler-prop
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
  --set community-name:public
```

Configuring SNMP on AIX

Native AIX SNMP implementations do not provide support for AgentX sub-agents, which is a requirement for UnboundID Directory Servers. To implement SNMP on AIX platforms, any freely-available net-snmp package must be installed. Special care must be made to ensure that you are using the net-snmp binary packages and not the native snmp implementation. Third-party net-snmp binary packages typically install under **`/opt/freeware`** and have the following differences:

Native Daemon: /**`usr/sbin/snmpd`**
Native Configuration File: **`/etc/snmpd.conf`**, **`/etc/snmpdv3.conf`**
Native Daemon Start and Stop: **`startsrc -s snmpd`**, **`stopsrc -s snmpd`**

net-snmp Daemon: **`/opt/freeware/sbin/snmpd`**
net-snmp Configuration File: **`/opt/freeware/etc/snmp/snmpd.conf`**
net-snmp start and stop: **`/etc/rc.d/init.d/snmpd start|stop`**

When configuring an SNMP implementation on AIX, remember to check the following items so that the Directory Server is referencing the net-snmp installation:

1. The shell PATH will reference the native implementation binaries. Adjust the PATH variable or invoke the net-snmp binaries explicitly.

2. If the native daemon is not stopped, there will likely be port conflicts between the native daemon and the net-snmp daemon. Disable the native daemon or use distinct port numbers for each.

SNMP on AIX: Security Considerations

On AgentX sub-agent-compliant systems, it is recommended to use **`agentXSocket tcp:localhost:705`** to configure the net-snmp master agent to allow connections only from sub-agents located on the same host. On AIX systems, it is possible to specify an external IP network interface (for example, **`agentXSocket tcp:0.0.0.0:708`** would listen on all external IP interfaces), which would allow the UnboundID Directory Server to be located on a different host as the snmp master agent.

While it is possible to implement non-local sub-agents, administrators should understand the security risks that are involved with this configuration. Primarily, because there is no communication authentication or privacy between the UnboundID Directory Server and the master agent, an eavesdropper might be able to listen in on the monitoring data sent by the UnboundID Directory Server. Likewise, a rogue sub-agent might be able to connect to the master agent and provide false monitoring data or deny access to SNMP monitoring data.

In general, it is recommended that sub-agents be located on the same host as the master agent.

MIBS

The Directory Server provides SMIv2-compliant MIB definitions (RFC 2578, 2579, 2580) for distinct monitoring statistics. These MIB definitions are to be found in text files under **`resource/mib`** directory under the server root directory.

Each MIB provides managed object tables for each specific SNMP management information as follows:

- **LDAP Remote Server MIB.** Provides information related to the health and status of the LDAP servers that the proxy server connects to, and statistics about the operations invoked by the proxy server on those LDAP servers.
- **LDAP Statistics MIB**. Provides a collection of connection-oriented performance data that is based on a connection handler in the Directory Server. A server typically contain only one connection handler and therefore supplies only one table entry.
- **Local DB Backend MIB**. Provides key metrics related to the state of the local database backends contained in the server.
- **Processing Time MIB**. Provides a collection of key performance data related to the processing time of operations broken down by several criteria but reported as a single aggregated data set.
- **Replication MIB**. Provides key metrics related to the current state of replication, which can help diagnose how much outstanding work replication may have to do.
- **System Status MIB**. Provides a set of critical metrics for determining the status and health of the system in relation to its work load.

For information on the available monitoring statistics for each MIB available on the directory server and the directory proxy server, see the text files provided in the **`resource/mib`** directory below the server installation.

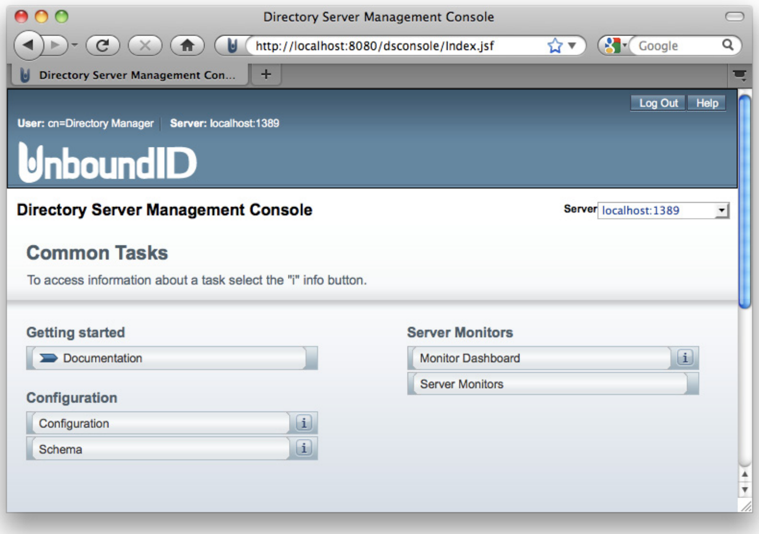
The Directory Server generates an extensive set of SNMP traps for event monitoring. The traps display the severity, description, name, OID, and summary. For information about the available alert types for event monitoring, see the **`resource/mib/UNBOUNDID-ALERT-MIB.txt`** file.

Monitoring with the Directory Management Console

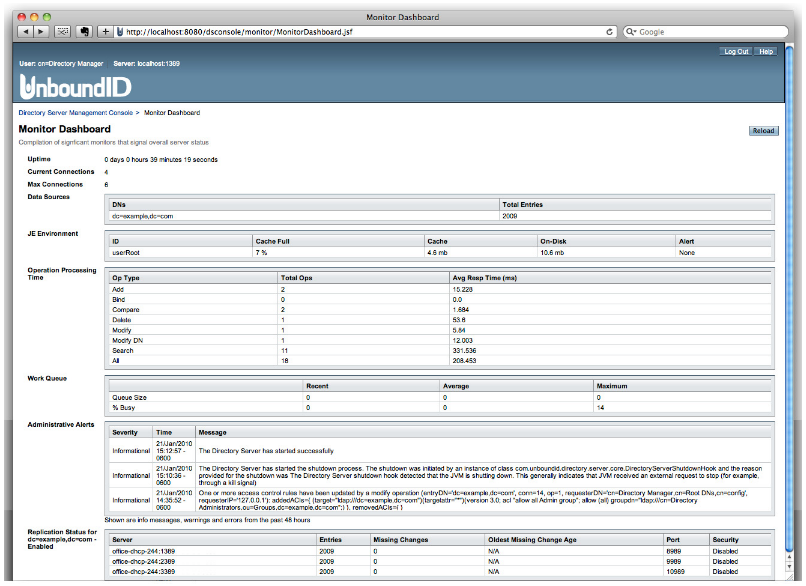
UnboundID has developed a graphical web console for administrators to configure the directory server. The console also provides a monitoring component that accesses the server's monitor content.

To View the Monitor Dashboard

1. Ensure that the Directory Server is running.
2. Open a browser to **`http://hostname:8080/dsconsole/`**. For information about installing the Directory Server Management Console, see "Installing the Directory Management Console" on page 52.
3. Type the root user DN (or any authorized administrator user name) and password, and then click Login.
4. Click Monitor Dashboard.


Directory Server Management Console
http://localhost:8080/dsconsole/Index.jsf
Google
Directory Server Management Con...
Log Out
Help
User: cn=Directory Manager
Server: localhost:1389
UnboundID
Directory Server Management Console
Server
localhost:1389
Common Tasks
To access information about a task select the "i" info button.
Getting started
Documentation
Configuration
Configuration
Schema
Server Monitors
Monitor Dashboard
Server Monitors

5. View the monitoring information on the dashboard.


Monitor Dashboard
http://localhost:8080/dsconsole/monitor/MonitorDashboard.jsf
Google
Log Out
Help

User: cn=Directory Manager Server: localhost:1389

UnboundID

Directory Server Management Console > Monitor Dashboard

Monitor Dashboard

Reload

Compilation of signficant monitors that signal overall server status

Uptime	0 days 0 hours 39 minutes 19 seconds
Current Connections	4
Max Connections	6

Data Sources

DNs	Total Entries
dc=example,dc=com	2009

JE Environment

ID	Cache Full	Cache	On-Disk	Alert
userRoot	7 %	4.6 mb	10.6 mb	None

Operation Processing Time

Op Type	Total Ops	Avg Resp Time (ms)
Add	2	15.228
Bind	0	0.0
Compare	2	1.684
Delete	1	53.6
Modify	1	5.84
Modify DN	1	12.003
Search	11	331.536
All	18	208.453

Work Queue

	Recent	Average	Maximum
Queue Size	0	0	0
% Busy	0	0	14

Administrative Alerts

Severity	Time	Message
Informational	21/Jan/2010 15:12:57 -0600	The Directory Server has started successfully
Informational	21/Jan/2010 15:10:36 -0600	The Directory Server has started the shutdown process. The shutdown was initiated by an instance of class com.unboundid.directory.server.core.DirectoryServerShutdownHook and the reason provided for the shutdown was The Directory Server shutdown hook detected that the JVM is shutting down. This generally indicates that JVM received an external request to stop (for example, through a kill signal)
Informational	21/Jan/2010 14:35:52 -0600	One or more access control rules have been updated by a modify operation (entryDN='dc=example,dc=com', conn=14, op=1, requesterDN='cn=Directory Manager,cn=Root DNs,cn=config', requesterIP='127.0.0.1'): addedACIs={ (target="ldap:///dc=example,dc=com")(targetattr="*")(version 3.0; acl "allow all Admin group"; allow (all) groupdn="ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";) }, removedACIs={ }

Shown are info messages, warnings and errors from the past 48 hours

Replication Status for dc=example,dc=com - Enabled

Server	Entries	Missing Changes	Oldest Missing Change Age	Port	Security
office-dhcp-244:1389	2009	0	N/A	8989	Disabled
office-dhcp-244:2389	2009	0	N/A	9989	Disabled
office-dhcp-244:3389	2009	0	N/A	10989	Disabled

Accessing the Processing Time Histogram

The UnboundID Directory Server provides a processing time histogram that classifies operation response time into user-defined buckets. The histogram tracks the processing on a per operation basis and as a percentage of the overall processing time for all operations. It also provides statistics for each operation type (add, bind, compare, delete, modify, modify DN, search).

To Access the Processing Time Histogram

1. On the Directory Management Console, click **Server Monitors**.

2. Click Processing Time Histogram. Other monitor entries can be accessed in similar ways.

Monitoring with JMX

The UnboundID Directory Server supports monitoring the JVM™ through a Java Management Extensions (JMX™) management agent, which can be accessed using JConsole or any other kind of JMX client. The JMX interface provides JVM performance and resource utilization information for applications running Java. You can monitor generic metrics exposed by the JVM itself, including memory pools, threads, loaded classes, and MBeans, as well as all the monitor information that the Directory Server provides. You can also subscribe to receive JMX notifications for any administrative alerts that are generated within the server.

Running JConsole

Before you can access JConsole, you must configure and enable the JMX Connection Handler for the directory server using the **`dsconfig`** tool. See "Configuring the JMX Connection Handler and Alert Handler" on page 501.

To invoke the JConsole executable, type **`jconsole`** on the command line. If **`JDK_HOME`** is not set in your path, you can access JConsole in the `bin` directory of the **`JDK_HOME`** path.

To Run JConsole

1. Use JConsole to open the Java Monitoring and Management Console. You can also run JConsole to monitor a specific process ID for your application: **`jconsole PID`**. Or you can run JConsole remotely using: **`jconsole hostname:port`**.

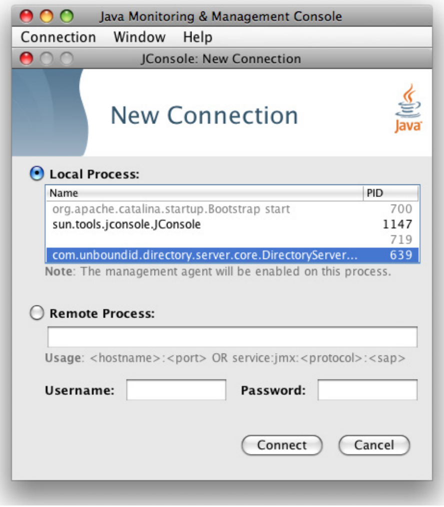
   ```
   $ jconsole
   ```

   Note

   If SSL is configured on the JMX Connection Handler, you need to specify the directory server jar file in the class path when running **`jconsole`** over SSL. For example, run the following **`jconsole`** command:

   ```
   $ jconsole \
   -J-Djavax.net.ssl.trustStore=/path/to/certStores/truststore \
   -J-Djavax.net.ssl.trustStorePassword=secret \
   -J-Djava.class.path=$SERVER_ROOT/UnboundID-DS.jar
   ```

2. On the Java Monitoring & Management Console, click Local Process, and then click the PID corresponding to the directory server.



Local Process:

Name	PID
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1147
	719
com.unboundid.directory.server.core.DirectoryServer...	639

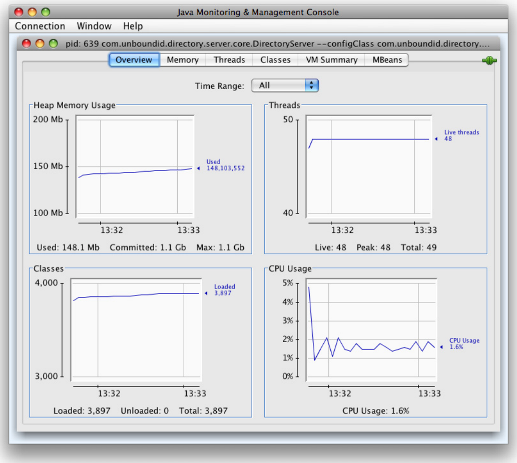
Note: The management agent will be enabled on this process.

Remote Process:

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

Username: Password:

3. Review the resource monitoring information.


Java Monitoring & Management Console
Connection Window Help
pid: 639 com.unboundid.directory.server.core.DirectoryServer --configClass com.unboundid.directory....
Overview Memory Threads Classes VM Summary MBeans
Time Range: All
Heap Memory Usage
200 Mb
150 Mb
100 Mb
Used 148,103,552
13:32 13:33
Used: 148.1 Mb Committed: 1.1 Gb Max: 1.1 Gb
Threads
50
40
Live threads 48
Live: 48 Peak: 48 Total: 49
Classes
4,000
3,000
Loaded 3,897
Loaded: 3,897 Unloaded: 0 Total: 3,897
CPU Usage
5% 4% 3% 2% 1% 0%
CPU Usage 1.6%
CPU Usage: 1.6%

Monitoring the Directory Server Using JConsole

You can set up JConsole to monitor the UnboundID Directory Server using a remote process. Make sure to enable the JMX Connection Handler and to assign at least the **`jmx-read`** privilege to a regular user account (the **`jmx-notify`** privilege is required to subscribe to receive JMX notifications). Do not use a root user account, as this would pose a security risk.

To Monitor the Directory Server using JConsole

1. Start the directory server.

```
$ bin/start-ds
```

2. Enable the JMX Connection handler using the **`dsconfig`** tool. The handler is disabled by default. Remember to include the LDAP connection parameters (**`hostname`**, **`port`**, **`bindDN`**, **`bindPassword`**).

```
$ bin/dsconfig set-connection-handler-prop \
  --handler-name "JMX Connection Handler" \
  --set enabled:true
```

3. Assign **`jmx-read`**, **`jmx-write`**, and **`jmx-notify`** (if the user receives notifications) to the user.

```
$ bin/ldapmodify --hostname server1.example.com --port 1389 \
  --bindDN "cn=Directory Manager" --bindPassword secret
dn: uid=admin,dc=example,dc=com
changetype: modify
replace: ds-privilege-name
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
ds-privilege-name: jmx-notify
```

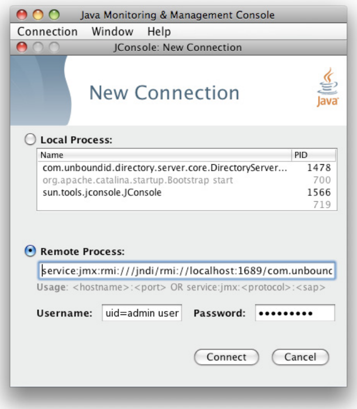
4. On the Java Monitoring & Management Console, click Remote Process, and enter the following JMX URL using the host and port of your directory.

```
service:jmx:rmi:///jndi/rmi://<host>:<port>/com.unboundid.directory.server.proto-
cols.jmx.client-unknown
```

For example, assuming your host name is **`server1.example.com`** and the JMX port is **`1689`**, use the following:

```
service:jmx:rmi:///jndi/rmi://server1.example.com:1689/com.unboundid.direc-
tory.server.protocols.jmx.client-unknown
```

5. In the Username and Password fields, type the bind DN and password for a user that has at least the **`jmx-read`** privilege. Click **Connect**.



Local Process:

Name	PID
com.unboundid.directory.server.core.DirectoryServer...	1478
org.apache.catalina.startup.Bootstrap start	700
sun.tools.jconsole.JConsole	1566
	719

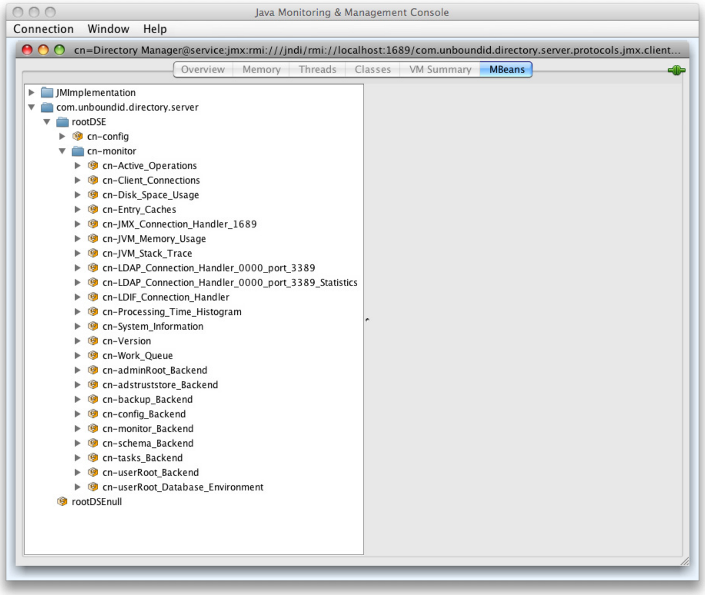
Remote Process: [✓]

service:jmx:rmi:///jndi/rmi://localhost:1689/com.unbounc

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

Username: uid=admin user Password: •••••••••

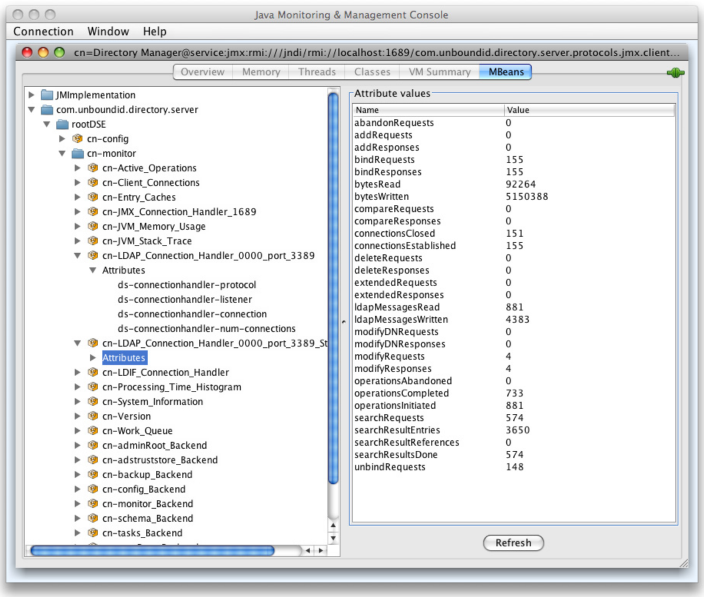
6. Click **`com.unboundid.directory.server`**, and expand the **`rootDSE`** node and the **`cn-monitor`** sub-node.



cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...

```
▶ JMImplementation
▼ com.unboundid.directory.server
  ▼ rootDSE
    ▶ cn-config
    ▼ cn-monitor
      ▶ cn-Active_Operations
      ▶ cn-Client_Connections
      ▶ cn-Disk_Space_Usage
      ▶ cn-Entry_Caches
      ▶ cn-JMX_Connection_Handler_1689
      ▶ cn-JVM_Memory_Usage
      ▶ cn-JVM_Stack_Trace
      ▶ cn-LDAP_Connection_Handler_0000_port_3389
      ▶ cn-LDAP_Connection_Handler_0000_port_3389_Statistics
      ▶ cn-LDIF_Connection_Handler
      ▶ cn-Processing_Time_Histogram
      ▶ cn-System_Information
      ▶ cn-Version
      ▶ cn-Work_Queue
      ▶ cn-adminRoot_Backend
      ▶ cn-adstruststore_Backend
      ▶ cn-backup_Backend
      ▶ cn-config_Backend
      ▶ cn-monitor_Backend
      ▶ cn-schema_Backend
      ▶ cn-tasks_Backend
      ▶ cn-userRoot_Backend
      ▶ cn-userRoot_Database_Environment
  rootDSEnull
```

7. Click a monitoring entry. In this example, click the LDAP Connection Handler entry.


Java Monitoring & Management Console
Connection
Window
Help
cn=Directory Manager@service:jmx:rmi:///jndi/rmi://localhost:1689/com.unboundid.directory.server.protocols.jmx.client...
Overview
Memory
Threads
Classes
VM Summary
MBeans
Refresh

```
JMImplementation
com.unboundid.directory.server
  rootDSE
    cn-config
    cn-monitor
      cn-Active_Operations
      cn-Client_Connections
      cn-Entry_Caches
      cn-JMX_Connection_Handler_1689
      cn-JVM_Memory_Usage
      cn-JVM_Stack_Trace
      cn-LDAP_Connection_Handler_0000_port_3389
        Attributes
          ds-connectionhandler-protocol
          ds-connectionhandler-listener
          ds-connectionhandler-connection
          ds-connectionhandler-num-connections
      cn-LDAP_Connection_Handler_0000_port_3389_St
        Attributes
      cn-LDIF_Connection_Handler
      cn-Processing_Time_Histogram
      cn-System_Information
      cn-Version
      cn-Work_Queue
      cn-adminRoot_Backend
      cn-adstruststore_Backend
      cn-backup_Backend
      cn-config_Backend
      cn-monitor_Backend
      cn-schema_Backend
      cn-tasks_Backend
```

Attribute values

Name	Value
abandonRequests	0
addRequests	0
addResponses	0
bindRequests	155
bindResponses	155
bytesRead	92264
bytesWritten	5150388
compareRequests	0
compareResponses	0
connectionsClosed	151
connectionsEstablished	155
deleteRequests	0
deleteResponses	0
extendedRequests	0
extendedResponses	0
ldapMessagesRead	881
ldapMessagesWritten	4383
modifyDNRequests	0
modifyDNResponses	0
modifyRequests	4
modifyResponses	4
operationsAbandoned	0
operationsCompleted	733
operationsInitiated	881
searchRequests	574
searchResultEntries	3650
searchResultReferences	0
searchResultsDone	574
unbindRequests	148

Monitoring Using the LDAP SDK

You can use the monitoring API to retrieve monitor entries from the directory proxy server as well as to retrieve specific types of monitor entries.

For example, you can retrieve all monitor entries published by the Directory Server and print the information contained in each using the generic API for accessing monitor entry data as follows:

```
for (MonitorEntry e : MonitorManager.getMonitorEntries(connection))
   {
     System.out.println("Monitor Name:  " + e.getMonitorName());
     System.out.println("Monitor Type:  " + e.getMonitorDisplayName());
     System.out.println("Monitor Data:");
     for (MonitorAttribute a : e.getMonitorAttributes().values())
     {
       for (Object value : a.getValues())
       {
         System.out.println("  " + a.getDisplayName() + ":  " +
                            String.valueOf(value));
       }
     }
     System.out.println();
   }
```

For more information about the LDAP SDK and the methods in this example, see the UnboundID LDAP SDK documentation.

Monitoring over LDAP

The UnboundID Directory Server exposes a majority of directory information under the **cn=monitor** entry. You can access these entries over LDAP using the **ldapsearch** tool.

```
$ bin/ldapsearch --hostname server1.example.com --port 1389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --baseDN "cn=monitor" "(objectclass=*)"
```

Profiling Server Performance Using the Periodic Stats Logger

The Directory Server ships with a built-in Periodic Stats Logger that is useful for profiling server performance for a given configuration. At a specified interval, the Periodic Stats logger writes server statistics to a log file in a comma-separated format (.csv), which can be ready by spreadsheet applications. The logger has a negligible impact on server performance unless the **log-interval** property is set to a very small value (less than 1 second).

You can control what statistics are logged and their verbosity. We recommend that you see what stats are available by experimenting with configurations or values. For example, you can make a configuration change in a test environment and see what stats are logged in the file.

To Enable the Periodic Stats Logger

By default, the Directory Server ships with the built-in 'Periodic Stats Logger' disabled. To enable it using the **dsconfig** tool or the web console, go to Plugins menu (available on the Advanced object menu), and then, select "Stats Logger."

1. Run **dsconfig** in interactive mode. Enter the LDAP or LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```

2. Enter "o" to change to the Advanced Objects menu.

3. On the Directory Server configuration console menu, enter the number for Plugins.

4. On the Plugin management menu, enter **3** to view and edit an existing plug-in.

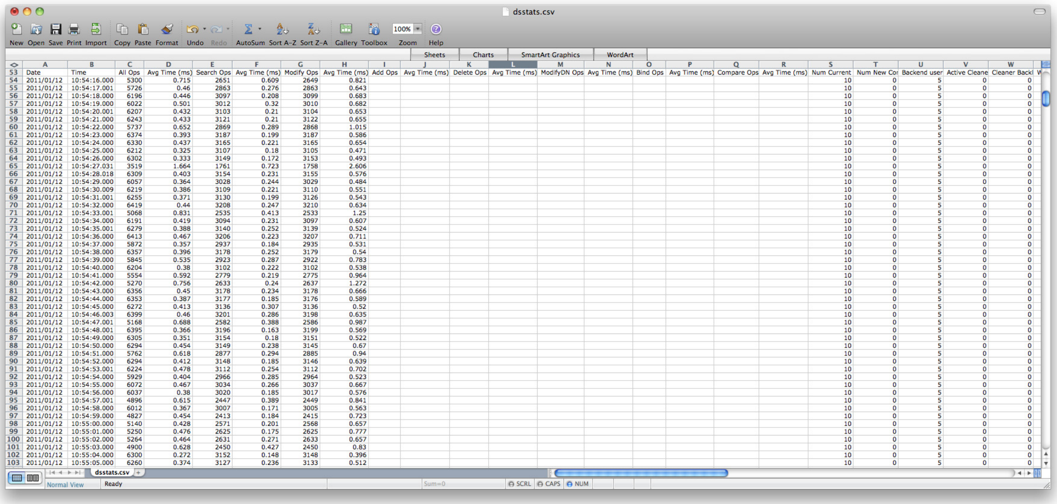
5. On the Plugin selection list, enter the number corresponding to the Stats Logger.

6. On the Periodic Stats Logger Plugin menu, enter **2** to set the **enabled** property to TRUE. When done, enter **f** to save and apply the configuration. The default logger will log information about the server every second to **<server-root>/logs/dsstats.csv**. If the server is idle, nothing will be logged, but this can be changed by setting the **suppress-if-idle** property to FALSE (**suppress-if-idle=false**).

```
>>>> Configure the properties of the Periodic Stats Logger Plugin

         Property                   Value(s)
        --------------------------------------------------------------------------------------
    1)   description                Logs performance stats to a log file periodically.
    2)   enabled                    true
    3)   log-interval               1 s
    4)   collection-interval        200 ms
    5)   suppress-if-idle           true
    6)   header-prefix-per-column   false
    7)   empty-instead-of-zero      true
    8)   lines-between-header       50
    9)   included-ldap-stat         active-operations, num-connections, op-count-and-latency,
                                    work-queue
    10)  included-resource-stat     memory-utilization
    11)  histogram-format           count
    12)  histogram-op-type          all
    13)  local-db-backend-info      basic
    14)  replication-info           basic
    15)  entry-cache-info           -
    16)  log-file                   logs/dsstats.csv
    17)  log-file-permissions       640
    18)  append                     true
    19)  rotation-policy            Fixed Time Rotation Policy, Size Limit Rotation Policy
    20)  retention-policy           File Count Retention Policy

    ?)   help
    f)   finish - apply any changes to the Periodic Stats Logger Plugin
    a)   hide advanced properties of the Periodic Stats Logger Plugin
    d)   display the equivalent dsconfig command lines to either re-create this object or only
         to apply pending changes
    b)   back
    q)   quit

Enter choice [b]:f
```

7. Run the Directory Server. For example, if you are running in a test environment, you can run the **`search-and-mod-rate`** tool to apply some searches and modifications to the server. You can run **`search-and-mod-rate --help`** to see an example command.

8. View the Stats log output at **`<server-root>/logs/dsstats.csv`**. You can open the file in a spreadsheet. The following image displays a portion of the file's output. On the actual file, you will need to scroll right for more statistics.



dsstats.csv
New Open Save Print Import Copy Paste Format Undo Redo AutoSum Sort A-Z Sort Z-A Gallery Toolbox Zoom Help
Sheets Charts SmartArt Graphics WordArt
Normal View
Ready

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
53	Date	Time	All Ops	Avg Time (ms)	Search Ops	Avg Time (ms)	Modify Ops	Avg Time (ms)	Add Ops	Avg Time (ms)	Delete Ops	Avg Time (ms)	ModifyDN Ops	Avg Time (ms)	Bind Ops	Avg Time (ms)	Compare Ops	Avg Time (ms)	Num Current	Num New Co	Backend user	Active Cleane	Cleaner Backl
54	2011/01/12	10:54:16.000	5300	0.715	2651	0.609	2649	0.821											10	0	5	0	0
55	2011/01/12	10:54:17.001	5726	0.46	2863	0.276	2863	0.643											10	0	5	0	0
56	2011/01/12	10:54:18.000	6196	0.446	3097	0.208	3099	0.683											10	0	5	0	0
57	2011/01/12	10:54:19.000	6022	0.501	3012	0.32	3010	0.682											10	0	5	0	0
58	2011/01/12	10:54:20.001	6207	0.432	3103	0.21	3104	0.653											10	0	5	0	0
59	2011/01/12	10:54:21.000	6243	0.433	3121	0.21	3122	0.655											10	0	5	0	0
60	2011/01/12	10:54:22.000	5737	0.652	2869	0.289	2868	1.015											10	0	5	0	0
61	2011/01/12	10:54:23.000	6374	0.393	3187	0.199	3187	0.586											10	0	5	0	0
62	2011/01/12	10:54:24.000	6330	0.437	3165	0.221	3165	0.654											10	0	5	0	0
63	2011/01/12	10:54:25.000	6212	0.325	3107	0.18	3105	0.471											10	0	5	0	0
64	2011/01/12	10:54:26.000	6302	0.333	3149	0.172	3153	0.493											10	0	5	0	0
65	2011/01/12	10:54:27.031	3519	1.664	1761	0.723	1758	2.606											10	0	5	0	0
66	2011/01/12	10:54:28.018	6309	0.403	3154	0.231	3155	0.576											10	0	5	0	0
67	2011/01/12	10:54:29.000	6057	0.364	3028	0.244	3029	0.484											10	0	5	0	0
68	2011/01/12	10:54:30.009	6219	0.386	3109	0.221	3110	0.551											10	0	5	0	0
69	2011/01/12	10:54:31.001	6255	0.371	3130	0.199	3126	0.543											10	0	5	0	0
70	2011/01/12	10:54:32.000	6419	0.44	3208	0.247	3210	0.634											10	0	5	0	0
71	2011/01/12	10:54:33.001	5068	0.831	2535	0.413	2533	1.25											10	0	5	0	0
72	2011/01/12	10:54:34.000	6191	0.419	3094	0.231	3097	0.607											10	0	5	0	0
73	2011/01/12	10:54:35.001	6279	0.388	3140	0.252	3139	0.524											10	0	5	0	0
74	2011/01/12	10:54:36.000	6413	0.467	3206	0.223	3207	0.711											10	0	5	0	0
75	2011/01/12	10:54:37.000	5872	0.357	2937	0.184	2935	0.531											10	0	5	0	0
76	2011/01/12	10:54:38.000	6357	0.396	3178	0.252	3179	0.54											10	0	5	0	0
77	2011/01/12	10:54:39.000	5845	0.535	2923	0.287	2922	0.783											10	0	5	0	0
78	2011/01/12	10:54:40.000	6204	0.38	3102	0.222	3102	0.538											10	0	5	0	0
79	2011/01/12	10:54:41.000	5554	0.592	2779	0.219	2775	0.964											10	0	5	0	0
80	2011/01/12	10:54:42.000	5270	0.756	2633	0.24	2637	1.272											10	0	5	0	0
81	2011/01/12	10:54:43.000	6356	0.45	3178	0.234	3178	0.666											10	0	5	0	0
82	2011/01/12	10:54:44.000	6353	0.387	3177	0.185	3176	0.589											10	0	5	0	0
83	2011/01/12	10:54:45.000	6272	0.413	3136	0.307	3136	0.52											10	0	5	0	0
84	2011/01/12	10:54:46.003	6399	0.46	3201	0.286	3198	0.635											10	0	5	0	0
85	2011/01/12	10:54:47.001	5168	0.688	2582	0.388	2586	0.987											10	0	5	0	0
86	2011/01/12	10:54:48.001	6395	0.366	3196	0.163	3199	0.569											10	0	5	0	0
87	2011/01/12	10:54:49.000	6305	0.351	3154	0.18	3151	0.522											10	0	5	0	0
88	2011/01/12	10:54:50.000	6294	0.454	3149	0.238	3145	0.67											10	0	5	0	0
89	2011/01/12	10:54:51.000	5762	0.618	2877	0.294	2885	0.94											10	0	5	0	0
90	2011/01/12	10:54:52.000	6294	0.412	3148	0.185	3146	0.639											10	0	5	0	0
91	2011/01/12	10:54:53.001	6224	0.478	3112	0.254	3112	0.702											10	0	5	0	0
92	2011/01/12	10:54:54.000	5929	0.404	2966	0.285	2964	0.523											10	0	5	0	0
93	2011/01/12	10:54:55.000	6072	0.467	3034	0.266	3037	0.667											10	0	5	0	0
94	2011/01/12	10:54:56.000	6037	0.38	3020	0.185	3017	0.576											10	0	5	0	0
95	2011/01/12	10:54:57.001	4896	0.615	2447	0.389	2449	0.841											10	0	5	0	0
96	2011/01/12	10:54:58.000	6012	0.367	3007	0.171	3005	0.563											10	0	5	0	0
97	2011/01/12	10:54:59.000	4827	0.454	2413	0.184	2415	0.723											10	0	5	0	0
98	2011/01/12	10:55:00.000	5140	0.428	2571	0.201	2568	0.657											10	0	5	0	0
99	2011/01/12	10:55:01.000	5250	0.476	2625	0.175	2625	0.777											10	0	5	0	0
100	2011/01/12	10:55:02.000	5264	0.464	2631	0.271	2633	0.657											10	0	5	0	0
101	2011/01/12	10:55:03.000	4900	0.628	2450	0.427	2450	0.83											10	0	5	0	0
102	2011/01/12	10:55:04.000	6300	0.272	3152	0.148	3148	0.396											10	0	5	0	0
103	2011/01/12	10:55:05.000	6260	0.374	3127	0.236	3133	0.512											10	0	5	0	0

To Configure Multiple Periodic Stats Loggers

Multiple Periodic Stats Loggers can be created to log different statistics or to create a log at different intervals (such as logging cumulative operations statistics every hour). To create a new log, use the existing Stats Logger as a template to get reasonable settings, including rotation and retention policy.

1. Run **`dsconfig`** by repeating steps 1–3 in “To Enable the Periodic Stats Logger” on page 484 (the previous procedure).
2. From the Plugin management menu, enter **`2`** to create a new plug-in.
3. From the How to Create a New Plugin menu, enter **`t`** to use an existing plug-in as a template.
4. Enter the number corresponding to use the existing stats logger as a template.
5. Next, enter a descriptive name for the new stats logger. For this example, type **`Stats Logger-10s`**.
6. Enter the log file path to the file. For this example, type **`logs/dsstats2.csv`**.
7. On the Periodic Stats Logger Plugin menu, make any other change to your logger. For this example, change the **`log-interval`** to **`10s`**, and the **`suppress-if-idle`** to **`false`**. When finished, enter **`f`** to save and apply the configuration.
8. You should now see two loggers **`dsstats.csv`** and **`dsstats2.csv`** in the logs directory.

Adding Custom Logged Statistics to the Periodic Stats Logger

You can add custom statistics based on any attribute in any entry under **`cn=monitor`** using the Custom Logged Stats object. This configuration object provides powerful controls for how monitor attributes are written to the log. For example, you can extract a value from a monitor attribute using a regular expression. Newly created Custom Logged Stats will automatically be included in the Periodic Stats Logger output.

Besides allowing a straight pass-through of the values using the 'raw' statistic-type, you can configure attributes to be treated as a *counter* (where the interval includes the difference in the value since the last interval), an *average*, a *minimum*, or a *maximum* value held by the attribute during the specified interval. The value of an attribute can also be scaled by a fixed value or by the value of another monitor attribute.

Note	Custom third-party server extensions that were written using the Server SDK can also expose interval statistics using the Periodic Stats Logger. The extension must first implement the SDK's MonitorProvider interface and register with the server. The monitor attributes produced by this custom MonitorProvider are then available to be referenced by a Custom Logged Stats object.

To illustrate how to configure a Custom Logged Statistics Logger, the following procedure reproduces the built-in "Consumer Total GB" column that shows up in the output when the **`included-resource-stat`** property is set to **`memory-utilization`** on the Periodic Stats Logger. The column is derived from the **`total-bytes-used-by-memory-consumers`** attribute of the **`cn=JVM Memory Usage,cn=monitor entry`** as follows:

```
dn: cn=JVM Memory Usage,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-memory-usage-monitor-entry
objectClass: extensibleObject
cn: JVM Memory Usage
...
total-bytes-used-by-memory-consumers: 3250017037
```

To Configure a Custom Logged Statistic Using dsconfig Interactive

1. Run **`dsconfig`** and enter the LDAP/LDAPS connection parameters when prompted.

   ```
   $ bin/dsconfig
   ```

2. On the Directory Server configuration main menu (Advanced Objects menu), enter the number corresponding **`to Custom Logged Stats`**.

   ```
   >>>> UnboundID Directory Server configuration console main menu

   What do you want to configure?

       1)   Access Control Handler               26)  Log Retention Policy
       2)   Account Status Notification Handler  27)  Log Rotation Policy
       3)   Alert Handler                        28)  Monitor Provider
       4)   Backend                              29)  Password Generator
       5)   Certificate Mapper                   30)  Password Policy
       6)   Change Subscription                  31)  Password Storage Scheme
       7)   Change Subscription Handler          32)  Password Validator
       8)   Cipher Stream Provider               33)  Plugin
       9)   Client Connection Policy             34)  Plugin Root
       10)  Connection Criteria                  35)  Replication Domain
       11)  Connection Handler                   36)  Replication Server
       12)  Custom Logged Stats                  37)  Request Criteria
       13)  Debug Target                         38)  Result Code Map
       14)  DN Map                               39)  Result Criteria
       15)  Extended Operation Handler           40)  Root DN
       16)  External Server                      41)  Root DSE Backend
       17)  Global Configuration                 42)  SASL Mechanism Handler
       18)  Identity Mapper                      43)  Search Entry Criteria
       19)  Key Manager Provider                 44)  Search Reference Criteria
       20)  LDAP SDK Debug Logger                45)  Sensitive Attribute
       21)  Local DB Index                       46)  Synchronization Provider
       22)  Local DB VLV Index                   47)  Trust Manager Provider
       23)  Location                             48)  Virtual Attribute
       24)  Log Field Mapping                    49)  Work Queue
       25)  Log Publisher

       o)   'Advanced' objects are shown - change this
       q)   quit

   Enter choice: 12
   ```

3. On the Custom Logged Stats menu, enter the number corresponding to Create a new Custom Logged Stats.

   ```
   >>>> Custom Logged Stats management menu

   What would you like to do?

       1)  List existing Custom Logged Stats
       2)  Create a new Custom Logged Stats
   ```

```
    3)  View and edit an existing Custom Logged Stats
    4)  Delete an existing Custom Logged Stats

    b)  back
    q)  quit

Enter choice [b]: 2
```

4. Select the Periodic Stats Logger Plugin from the list if more than one is present on the system. If you only have one stats logger, press Enter to confirm that you want to use the existing plugin.

```
>>>> There is only one Periodic Stats Logger Plugin: 'Stats Logger'. Are you sure
that this is the correct one? (yes / no) [yes]:
```

5. Enter a descriptive name for the Custom Logged Stats. For this example, enter **`Memory Usage`**.

```
>>>> Enter a name for the Custom Logged Stats that you want to create: Memory Usage
```

6. From the monitor-objectclass property menu, enter the objectclass attribute to monitor. For this example, enter **`ds-memory-usage-monitor-entry`**. You can run **`ldapsearch`** using the base DN "**`cn=JVM Memory Usage,cn=monitor`**" entry to view the entry.

```
>>>> Configuring the 'monitor-objectclass' property
 >>>> via creating 'Memory Usage' Custom Logged Stats

    The objectclass name of the monitor entries to examine for generating these
    statistics.

    Syntax:  monitor-entry-objectclass - The objectclass name of the monitor entries
    to examine for generating these statistics.

Enter a value for the 'monitor-objectclass' property: ds-memory-usage-monitor-entry
```

7. Next, specify the attributes of the monitor entry that you want to log in the stats logger. In this example, enter **`total-bytes-used-by-memory-consumers`**, and then press Enter again to continue.

```
>>>> Configuring the 'attribute-to-log' property
 >>>> via creating 'Memory Usage' Custom Logged Stats

    Specifies the attributes on the monitor entries that should be included in the
    output.

    The full column name, which appears in the header, is built from the
    header-prefix, the header-prefix-attribute, and the column name, which is
    either determined automatically from the monitor entry attribute name or is
    specified explicitly in the column-name property. Both the header-prefix, and
    the header-prefix-attribute column names are optional.

    Syntax:  monitor-entry-objectclass - The name of the attribute to log.

Enter a value for the 'attribute-to-log' property: total-bytes-used-by-memory-consumers

Enter another value for the 'attribute-to-log' property [continue]:
```

8. Next, specify the type of statistics for the monitored attribute that will appear in the log file. In this example, enter **`the option`** for **`raw`** statistics as recorded by the logger.

```
>>>> Configuring the 'statistic-type' property
 >>>> via creating 'Memory Usage' Custom Logged Stats

    Specifies the type of statistic to include in the output for each monitored
    attribute.

    The statistics listed here will be applied to each attribute listed in the
    attribute-to-log list. The statistic is computed after all other optional
    processing such as regular expression manipulation and value scaling.
```

```
     If a single monitor entry includes attributes of different statistic-types,
     then multiple Custom Logged Stats objects must be specified.

Select one or more values for the 'statistic-type' property:

    1)  average
    2)  counter
    3)  maximum
    4)  minimum
    5)  raw

    ?)  help
    c)  cancel
    q)  quit

Enter one or more choices separated by commas [c]: 5
```

9. In the Custom Logged Stats menu, review the configuration. At this point, we want to set up a column name that lists the Memory Usage. Enter the option to change the **`column-name`** property.

```
>>>> Configure the properties of the Custom Logged Stats
 >>>> via creating 'Memory Usage' Custom Logged Stats

         Property                    Value(s)
         --------------------------------------------------------------

    1)   description                 -
    2)   enabled                     true
    3)   monitor-objectclass         ds-memory-usage-monitor-entry
    4)   include-filter              -
    5)   attribute-to-log            total-bytes-used-by-memory-consumers
    6)   column-name                 -
    7)   statistic-type              raw
    8)   header-prefix               -
    9)   header-prefix-attribute     -
    10)  regex-pattern               -
    11)  regex-replacement           -
    12)  divide-value-by
    13)  divide-value-by-attribute   -
    14)  decimal-format              #.##
    15)  non-zero-implies-not-idle   false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]: 6
```

10. Next, we want to add a specific label for the column name. Enter the option to add a value, and then enter **`Memory Consumer Total (GB)`**, and press Enter again to continue.

```
>>>> Configuring the 'column-name' property
 >>>> via creating 'Memory Usage' Custom Logged Stats

    Optionally, specifies an explicit name for each column header instead of
    having these names automatically generated from the monitored attribute name.

    The full column name, which appears in the header, is built from the
    header-prefix, the header-prefix-attribute, and the column name, which is
    either determined automatically from the monitor entry attribute name or is
    specified explicitly in the column-name property. The column-name,
    header-prefix, and header-prefix-attribute properties are optional.

    The number of items in this property must line up with the attribute-to-log
    property. That is, the first value in this property will be used as the column
    name for the first value in the attribute-to-log property.

    Syntax:  STRING

Do you want to modify the 'column-name' property?

    1)  Leave undefined
    2)  Add one or more values

    ?)  help
    q)  quit

Enter choice [1]: 2

Enter a value for the 'column-name' property [continue]: Memory Consumer Total (GB)

Enter another value for the 'column-name' property [continue]:
```

11. Confirm that you want to use the **`column-name`** value that you entered in the previous step, and then press Enter to use the value.

```
>>>> Configuring the 'column-name' property (Continued)

Do you want to modify the 'column-name' property?

    1)  Use the value: Memory Consumer Total (GB)
    2)  Add one or more values
    3)  Remove one or more values
    4)  Leave undefined
    5)  Revert changes

    ?)  help
    q)  quit

Enter choice [1]:
```

12. Next, we want to scale the Memory Consumer Totals by one gigabyte. On the Custom Logged Stats menu, enter the option to change the **`divide-value-by`** property.

```
>>>> Configure the properties of the Custom Logged Stats
 >>>> via creating 'Memory Usage' Custom Logged Stats

         Property                   Value(s)
         --------------------------------------------------------------

    1)   description                -
    2)   enabled                    true
    3)   monitor-objectclass        ds-memory-usage-monitor-entry
    4)   include-filter             -
    5)   attribute-to-log           total-bytes-used-by-memory-consumers
    6)   column-name                Memory Consumer Total (GB)
    7)   statistic-type             raw
    8)   header-prefix              -
    9)   header-prefix-attribute    -
    10)  regex-pattern              -
    11)  regex-replacement          -
    12)  divide-value-by
    13)  divide-value-by-attribute  -
    14)  decimal-format             #.##
    15)  non-zero-implies-not-idle  false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]: 12
```

13. On the **`divide-value-by`** property menu, enter the option to change the value, and then enter **`1073741824`** (i.e., 1073741824 bytes = 1 gigabytes).

```
>>>> Configuring the 'divide-value-by' property
 >>>> via creating 'Memory Usage' Custom Logged Stats

    An optional floating point value that can be used to scale the resulting
    value.

    This is used to scale the resulting value by a fixed amount. This is
    typically used to reduce large values to a more manageable length. For
    instance, free disk space could be represented in GB instead of just
```

```
        bytes. If a value for this property is specified, then it will be applied
        to the monitored attribute after any regular expression replacement is
        done and before it is scaled by the divide-value-by-attribute.

        Syntax:  divide-by-amount - A floating point value used to scale the result.

    Do you want to modify the 'divide-value-by' property?

        1)  Leave undefined
        2)  Change the value

        ?)  help
        q)  quit

    Enter choice [1]: 2

    Enter a value for the 'divide-value-by' property [continue]: 1073741824
```

14. On the Custom Logged Stats menu, review your configuration. When finished, enter **f** to save and apply the settings.

```
>>>> Configure the properties of the Custom Logged Stats
 >>>> via creating 'Memory Usage' Custom Logged Stats

         Property                   Value(s)
         --------------------------------------------------------------

    1)   description                -
    2)   enabled                    true
    3)   monitor-objectclass        ds-memory-usage-monitor-entry
    4)   include-filter             -
    5)   attribute-to-log           total-bytes-used-by-memory-consumers
    6)   column-name                Memory Consumer Total (GB)
    7)   statistic-type             raw
    8)   header-prefix              -
    9)   header-prefix-attribute    -
    10)  regex-pattern              -
    11)  regex-replacement          -
    12)  divide-value-by            1073741824
    13)  divide-value-by-attribute  -
    14)  decimal-format             #.##
    15)  non-zero-implies-not-idle  false

    ?)   help
    f)   finish - create the new Custom Logged Stats
    a)   hide advanced properties of the Custom Logged Stats
    d)   display the equivalent dsconfig arguments to create this object
    b)   back
    q)   quit

Enter choice [b]:f

The Custom Logged Stats was created successfully
```

When the Custom Logged Stats configuration change is completed, the new stats value should immediately show up in the Periodic Stats Logger output file.

To Configure a Custom Periodic Stats Logger Using dsconfig Non-Interactive

Use the **dsconfig** non-interactive command-line equivalent to create your custom stats logger. The following one-line command replicates the procedure in the previous section. This command produces a column named "Memory Consumer Total (GB)" that contains the value of the of **total-bytes-used-by-memory-consumers** attribute pulled from the entry with the **ds-memory-usage-monitor-entry** objectclass. This value is scaled by 1073741824 to get to a value represented in GBs.

```
$ bin/dsconfig create-custom-logged-stats --plugin-name "Stats Logger" \
--stats-name "Memory Usage" --type custom \
--set monitor-objectclass:ds-memory-usage-monitor-entry \
--set attribute-to-log:total-bytes-used-by-memory-consumers \
--set "column-name:Memory Consumer Total (GB)" --set statistic-type:raw \
--set divide-value-by:1073741824
```

14 Managing Notifications and Alerts

Overview

The UnboundID Directory Server provides delivery mechanisms for account status notifications and administrative alerts using SMTP, JMX, or SNMP in addition to standard error logging. Alerts and events reflect state changes within the server that may be of interest to a user or monitoring service. Notifications are typically the delivery of an alert or event to a user or monitoring service. Account status notifications are only delivered to the account owner notifying a change in state in the account.

This chapter presents the following topics:

Working with Account Status Notifications

The UnboundID Directory Server supports notification handlers that can be used to notify users and/or administrators of significant changes related to password policy state for user entries. The following two notification handlers are available:

- **Error Log Account Status Notification Handler.** Enabled by default. The handlers send alerts to the error log when an account event occurs.

- **SMTP Account Status Notification Handler**. Disabled by default. You can enable the SMTP Handler with the `dsconfig` command to send notifications to designated email addresses.

The handlers send alerts when one of the account status events described in the following table occurs during password policy processing.

TABLE 14-1. **Account Status Notification Types**

Account Status Notification Type	Description
account-disabled	Generates a notification whenever a user account is disabled by an administrator.
account-enabled	Generates a notification whenever a user account is enabled by an administrator.
account-expired	Generates a notification whenever a user authentication attempt fails because the account has expired.
account-idle-locked	Generates a notification whenever a user authentication attempt fails because the account has been locked after idling for too long.
account-permanently-locked	Generates a notification whenever a user account is permanently locked (requiring administrative action to unlock the account) after too many failed attempts.
account-reset-locked	Generates a notification whenever an authentication attempt fails because the user account is locked because the user failed to change a password within the required interval that was reset by an administrator.
account-temporarily-locked	Generates a notification whenever a user account is temporarily locked after too many failed attempts.
account-unlocked	Generates a notification whenever a user account is unlocked by an administrator.
password-changed	Generates a notification whenever a user changes his or her own password.
password-expired	Generates a notification whenever a user authentication fails because the password has expired.
password-expiring	Generates a notification the first time that a password expiration warning is encountered for a user password.
password-reset	Generates a notification whenever a user's password is reset by an administrator.

Working with the Error Log Account Status Notification Handler

The Error Log Account Status Notification Handler is enabled by default and sends alerts when one of the account status events occur.

To Disable the Error Log Account Status Notification Handler

Use the **`dsconfig`** tool to disable the Error Log Handler. You can view the log at **`logs/error`**.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  ---handler-name "Error Log Handler" --set enabled:false
```

To Remove a Notification Type from the Error Log Handler

While not recommended, if you want to remove an account status notification type, use the **dsconfig** tool with the **--remove** option.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  --handler-name "Error Log Handler" \
  --remove account-status-notification-type: password-reset
```

Working with the SMTP Account Status Notification Handler

You can enable account status notifications to be sent to designated email addresses of end users, administrators, or both through an outgoing SMTP server. The email message is automatically generated from template files that contain the text to use in the message body. For example, the message subject for the **account-disabled** event is:

```
account-disabled: Your directory account has been disabled.
```

The message templates are located in the **config/messages** directory. The typical message body template is as follows:

```
Your directory account has been disabled.

For further assistance, please contact a server administrator.
```

By default, the sender address is **notifications@example.com**, but you can configure your own address.

Before you enable the SMTP Account Status Notification Handler, you must configure the Directory Server to use at least one mail server as shown below. You can configure an SMTP server using **dsconfig** and the **set-global-configuration-prop** option.

To Configure the SMTP Server

1. Use dsconfig to configure a simple mail server.

   ```
   $ bin/dsconfig create-external-server --server-name smtp1 --type smtp \
   --set server-host-name:smtp.example.com
   ```

2. Use **dsconfig** to configure an SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

   ```
   $ bin/dsconfig set-global-configuration-prop --set smtp-server:smtp1
   ```

To Configure an SSL Connection to the SMTP Server

You can configure an SSL connection to an SMTP server using the **dsconfig** tool.

1. Use **dsconfig** to create an external SMTP server using SSL.

   ```
   $ bin/dsconfig create-external-server --server-name ssl.smtp.example.com \
   --type smtp --set server-host-name:stmp.gmail.com --set server-port:465 \
   --set smtp-security:ssl --set 'username:my.name@example.com' \
   --set password:xxxxxx --set "smtp-timeout:10s"
   ```

2. Use **dsconfig** to configure an SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

   ```
   $ bin/dsconfig set-global-configuration-prop \
     --set smtp-server:ssl.smtp.example.com
   ```

To Configure a StartTLS Connection to the SMTP Server

You can configure a StartTLS connection to an SMTP server using the **dsconfig** tool.

1. Use **dsconfig** to configure a StartTLS connection to the server.

   ```
   $ bin/dsconfig create-external-server --server-name myTLSServer --type smtp \
   --set server-host-name:tls.smtp.example.com --set server-port:587 \
   --set smtp-security:starttls --set user-name:MyAccountName \
   --set "password:"password:AAD5yZ+DjvwiYkBSMer6GQ6B3szQ6gSSBjA="
   ```

2. Use **dsconfig** to configure a newly-created SMTP server. This command adds the server to the global list of mail servers that the Directory Server can use.

   ```
   $ bin/dsconfig set-global-configuration-prop \
     -set smtp-server:myTLSServer
   ```

To Enable the SMTP Account Status Notification Handler

Use **dsconfig** to enable the SMTP account status notification handler.

```
$ bin/dsconfig set-account-status-notification-handler-prop \
  --handler-name "SMTP Server" \
  --set enabled:true \
  --set "recipient-address:admin@example.com" \
  --set "sender-address:acct-status-notifications@example.com
```

To View the Account Status Notification Handlers

After you have enabled the SMTP server, view the list of account status notification handlers using **dsconfig**.

```
$ bin/dsconfig list-account-status-notification-handlers


Account Status Notification Handler : Type      : enabled
------------------------------------:-----------:--------
Error Log Handler                   : error-log : true
SMTP Handler                        : smtp      : true
```

Associating Account Status Notification Handlers with Password Policies

In order to generate notifications whenever appropriate password policy state changes occur in the server, the password policy that governs the entry being updated must be configured to use one or more account status notification handlers. By default, password policies are not configured with any such handlers, and therefore, no account status notifications will be generated.

The set of account status notification handlers that should be in use for a password policy is controlled by the `account-status-notification-handler` property for that password policy. It can be configured using `dsconfig` or the web administration console. For example, the following change updates the default password policy, so that the error log account status notification handler will be invoked for any appropriate password policy state changes for entries governed by the default password policy:

```
$ bin/dsconfig set-password-policy-prop \
  --policy-name "Default Password Policy" \
  --set "account-status-notification-handler:Error Log Handler"
```

Working with Administrative Alert Handlers

The UnboundID Directory Server provides mechanisms to send alert notifications to administrators when significant problems or events occur during processing, such as problems during server startup or shutdown. The Directory Server provides provides a number of alert handler implementations, including:

- **Error Log Alert Handler**. Sends administrative alerts to the configured server error logger(s).
- **Exec Alert Handler.** Executes a specified command on the local system if an administrative alert matching the criteria for this alert handler is generated by the directory server. Information about the administrative alert will be made available to the executed application as arguments provided by the command.
- **Groovy Scripted Alert Handler.** Provides alert handler implementations defined in a dynamically-loaded Groovy script that implements the ScriptedAlertHandler class defined in the Server SDK.
- **JMX Alert Handler**. Sends administrative alerts to clients using the Java Management Extensions (JMX) protocol. UnboundID uses JMX for monitoring entries and requires that the JMX connection handler be enabled.
- **SMTP Alert Handler**. Sends administrative alerts to clients via email using the Simple Mail Transfer Protocol (SMTP). The server requires that one or more SMTP servers be defined in the global configuration.
- **SNMP Alert Handler**. Sends administrative alerts to clients using the Simple Network Monitoring Protocol (SNMP). The server must have an SNMP agent capable of communicating via SNMP 2c.

- **SNMP Subagent Alert Handler.** Sends SNMP traps to a master agent in response to administrative alerts generated within the server.
- **Third Party Alert Handler.** Provides alert handler implementations created in third-party code using the Server SDK.

If enabled, the Directory Server can generate administrative alerts when the events occur (summarized in the table below).

TABLE 14-2. **Administrative Alert Types**

Alert Type	Severity	Description
access-control-change	Info	Indicates that access control configuration has been changed.
access-control-disabled	Warning	Indicates that access control evaluation has been disabled.
access-control-enabled	Info	Indicates that access control evaluation has been enabled.
access-control-parse-failure	Error	Indicates that an error occurred while attempting to parse an access control rule.
access-log-criteria-matched	Info	Indicates that an access log message matched the criteria for the admin alert access log publisher.
backend-end-initialization-failed	Error	Indicates that an attempt to initialize the backend failed.
cannot-acquire-shared-backend-lock	Error	Indicates that an error occurred while attempting to acquire a shared backend lock.
cannot-copy-schema-files	Error	Indicates that an error occurred while attempting to copy schema files during a schema update.
cannot-decode-entry	Error	Indicates that an error occurred while attempting to decode an entry stored in a backend.
cannot-find-recurring-task	Error	Indicates that the definition for a recurring task could not be found.
cannot-register-backend	Error	Indicates that an error occurred while trying to register a backend.
cannot-register-shared-backend-lock	Error	Indicates that an error occurred while trying to release a shared backend lock.
cannot-rename-current-task-file	Error	Indicates that an error occurred while trying to rename the current task backing file.
cannot-rename-new-task-file	Error	Indicates that an error occurred while trying to rename the new task backing file.
cannot-restore-backup	Error	Indicates that an error occurred while trying to restore a backup.
cannot-schedule-recurring-task-iteration	Error	Indicates that an error occurred while trying to schedule a recurring task iteration.
cannot-write-configuration	Error	Indicates that an error occurred while trying to write the updated server configuration.
cannot-write-new-schema-files	Error	Indicates that an error occurred while trying to update schema files.
cannot-write-server-state-file	Error	Indicates that an error occurred while attempting to write the server status file.

TABLE 14-2. **Administrative Alert Types**

Alert Type	Severity	Description
cannot-write-task-backing-file	Error	Indicates that an error occurred while trying to write the task backing file.
config-change	Info	Indicates the a configuration change has been made in the server.
deadlock-detected	Error	Indicates that a deadlock has been detected in the JVM™ in which the Directory Server is running.
duplicate-alerts-suppressed	Error	Indicates that duplicate alert notifications have been suppressed.
entering-lockdown-mode	Warning	Indicates that the server is entering lockdown mode, in which it will only allow operations from root users.
external-config-file-edit-handled	Warning	Indicates that the server has detected an external edit to the configuration file with the server online, but that it was able to copy the modifications into a separate file without applying them.
external-config-file-edit-lost	Error	Indicates that the server has detected an external edit to the configuration file with the server online and was unable to copy the modifications into a separate file.
external-server-initialization-failed	Error	Indicates that an attempt to initialize an external server failed.
force-gc-complete	Info	Indicates that the server has completed a forced garbage collection.
force-gc-starting	Info	Indicates that the server is about to force a synchronous garbage collection.
index-degraded	Warning	Indicates that a backend is operating with a degraded index that needs to be rebuilt before that index may be used.
index-rebuild-completed	Info	Indicates that a backend is in the progress of rebuilding one or more indexes.
index-rebuild-in-progress	Info	Indicates that a backend is in the progress of rebuilding one or more indexes.
invalid-privilege	Warning	Indicates that a user has been configured with an invalid privilege.
je-recovery-required	Fatal	Indicates that a backend using the Oracle Berkeley DB Java Edition (JE) has encountered a server error and requires recovery.
large-attribute-update-error	Error	Indicates that an error occurred while updating large attribute information in the Directory Server, and the large attribute information may be out of sync with the entry contents.
ldap-connection-handler-cannot-listen	Fatal	Indicates that an error occurred when the LDAP connection handler tried to start listening for client connections and therefore the connection handler will be disabled.
ldap-connection-handler-consecutive-failures	Fatal	Indicates that the LDAP connection handler has experienced consecutive failures and will be disabled.
ldap-connection-handler-uncaught-error	Fatal	Indicates that the LDAP connection handler has encountered an uncaught error and will be disabled.
ldif-backend-cannot-write	Error	Indicates that an error occurred while trying to write the backing file for the LDIF backend.

TABLE 14-2. **Administrative Alert Types**

Alert Type	Severity	Description
ldif-connection-handler-io-error	Error	Indicates that the LDIF connection handler encountered an I/O error that prevented it from processing.
ldif-connection-handler-parse-error	Error	Indicates that the LDIF connection handler encountered an I/O error that has prevented it from processing.
leaving-lockdown-mode	Info	Indicates that the server is leaving lockdown mode and resuming normal operation.
logging-error	Error	Indicates that an error occurred while attempting to log a message.
low-disk-space-error	Error	Indicates that the amount of usable disk space has dropped below the low space error threshold.
low-disk-space-warning	Warning	Indicates that the amount of usable disk space has dropped below the configured low space warning threshold.
out-of-disk-space-error	Fatal	Indicates that the amount of usable disk space has dropped below the configured out of space error threshold.
replication-backlogged	Warning	Indicates that the replication backlog has exceeded the replication backlog count alert threshold for longer than the replication backlog duration alert threshold.
replication-backlog	Warning	Indicates that a replication server has known changes that have not been applied yet.
replication-monitor-data-unresolved	Warning	Indicates that replication monitor data is unavailable from `cn=monitor`.
replication-unresolved-conflict	Error	Indicates that the multi-master replication cannot automatically resolve a conflict.
replication-plugin-message-serialization-failure	Warning	Indicates that the replication plug-in failed to serialize or de-serialize a replication message.
replication-replay-failed	Error	Indicates that the server has failed to replay a replication operation.
replication-server-changelog-failure	Error	Indicates that the replication server encountered an error while accessing the replication changelog database.
replication-server-listen-failure	Error	Indicates that the replication server encountered an error while trying to listen on the configured replication port. There may be another application listening on the same port or the replication server host name may not be resolvable. Check the replication server configuration.
replication-unresolved-conflict	Error	Indicates that the server has detected a replication conflict that could not be resolved.
server-jvm-paused	Warning	Indicates that the server's JVM paused for some reason possibly due to misconfiguration.
server-shutting-down	Info	Indicates that the server has begun the shutdown process.
server-started	Info	Indicates that the server has completed that startup process.
server-starting	Info	Indicates that the server is starting.
system-nanotime-stopped	Error	Indicates that `System.nanoTime()` has stopped advancing.
thread-exit-holding-lock	Error	Indicates that a thread has exited while still holding one or more locks.

TABLE 14-2. **Administrative Alert Types**

Alert Type	Severity	Description
uncaught-exception	Error	Indicates that the server has detected an uncaught exception that may have caused a thread to terminate.
unique-attribute-sync-conflict	Error	Indicates that the server has detected a unique attribute conflict that was introduced from synchronization.
unique-attribute-sync-error	Error	Indicates that the server has encountered an error while attempting to detect unique attribute conflicts via synchronization.
unrecognized-alert-type	Error	Indicates that an unrecognized alert type was encountered. This should never be used for any alert that is generated, but only for cases in which the server needs to create an alert type from a string but the string does not match any recognized type.
user-defined-error	Error	Indicates that an error alert notification has been generated by third-party code.
user-defined-fatal	Fatal	Indicates that a fatal error alert notification was generated by third-party code.
user-defined-info	Info	Indicates that an informational alert notification was generated by third-party code.
user-defined-warning	Warning	Indicates that a warning alert notification was generated by third-party code.
worker-thread-caught-error	Error	Indicates that a worker thread caught an unexpected error.
work-queue-backlogged	Error	Indicates that the work queue has become significantly backlogged and operations have been required to wait a significant length of time to be processed.
work-queue-full	Error	Indicates that the work queue is full and has rejected a client request.
work-queue-no-threads-remaining	Fatal	Indicates that all worker threads have been terminated due to errors and the server must shut down.
worker-thread-caught-error	Error	Indicates that a worker thread has caught an unexpected error that has caused it to be terminated.

Configuring the JMX Connection Handler and Alert Handler

You can configure the JMX connection handler and alert handler respectively using the **`dsconfig`** tool. Any user allowed to receive JMX notifications must have the **`jmx-read`** and **`jmx-notify`** privileges. By default, these privileges are not granted to any users (including root users or global administrators). For security reasons, we recommend that you create a separate user account that does not have any other privileges but these. Although not shown in this section, you can configure the JMX connection handler and alert handler using **`dsconfig`** in interactive command-line mode, which is visible on the "Standard" object menu.

To Configure the JMX Connection Handler

1. Use **dsconfig** to enable the JMX Connection Handler.

   ```
   $ bin/dsconfig set-connection-handler-prop \
     --handler-name "JMX Connection Handler" \
     --set enabled:true \
     --set listen-port:1689
   ```

2. Add a new non-root user account with the **jmx-read**, and **jmx-notify** privileges. This account can be added using the **ldapmodify** tool using an LDIF representation like:

   ```
   dn: uid=jmx-user,dc=example,dc=com
   changetype: add
   objectClass: top
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   uid: jmx-user
   givenName: JMX
   sn: User
   cn: JMX User
   userPassword: password
   ds-privilege-name: jmx-read
   ds-privilege-name: jmx-notify
   ```

To Configure the JMX Alert Handler

Use **dsconfig** to configure the JMX Alert Handler.

```
$ bin/dsconfig set-alert-handler-prop --handler-name "JMX Alert Handler" \
  --set enabled:true
```

Configuring the SMTP Alert Handler

By default, there is no configuration entry for an SMTP alert handler. To create a new instance of an SMTP alert handler, use the **dsconfig** tool.

To Configure the SMTP Alert Handler

Use the **dsconfig** tool to configure the SMTP alert handler. The following command enables the handler and sets the sender-address, recipient-address, message-subject and message-body for the alert.

```
$ bin/dsconfig create-alert-handler \
  --handler-name "SMTP Alert Handler" \
  --type smtp \
  --set enabled:true \
  --set "sender-address:alerts@example.com" \
  --set "recipient-address:administrators@example.com" \
  --set "message-subject:Directory Admin Alert \%\%alert-type\%\%" \
  --set "message-body:Administrative alert:\\n\%\%alert-message\%\%"
```

Configuring the SNMP Subagent Alert Handler

You can configure the SNMP Subagent alert handler using the **dsconfig** tool, which is visible at the "Standard" object menu. Before you begin, you need an SNMP Subagent capable of communicating via SNMP 2c. For more information on SNMP, see “Monitoring Using SNMP” on page 472.

To Configure the SNMP Subagent Alert Handler

- Use **dsconfig** to configure the SNMP subagent alert handler. The **server-host-name** is the address of the system running the SNMP subagent. The **server-port** is the port number on which the subagent is running. The **community-name** is the name of the SNMP community that is used for the traps.

```
$ bin/dsconfig set-alert-handler-prop \
  --handler-name "SNMP Subagent Alert Handler" \
  --set enabled:true \
  --set server-host-name:host2 \
  --set server-port:162 \
  --set community-name:public
```

Note	The Directory Server also supports a *SNMP Alert Handler,* which is used in deployments that do not enable an SNMP subagent.

Working with the Alerts Backend

The Directory Server stores recently generated admin alerts in an Alerts Backend under the **cn=alerts** branch. The backend makes it possible to obtain admin alert information over LDAP for use with remote monitoring. The backend's primary job is to process search operations for alerts. It does not support add, modify, or modify DN operations of entries in the **cn=alerts** backend.

The alerts persist on disk in the **config/alerts.ldif** file so that they can survive server restarts. By default, the alerts remain on disk for seven days before being removed. However, administrators can configure the number of days for alert retention using the **dsconfig** tool. The administrative alerts of Warning level or worse that have occurred in the last 48 hours are viewable from the output of the status command-line tool and in the Directory Management Console.

To View Information in the Alerts Backend

Use **ldapsearch** to view the admin alerts.

```
$ bin/ldapsearch --port 1389 --bindDN "uid=admin,dc=example,dc=com" \
  --bindPassword secret --baseDN cn=alerts "(objectclass=*)"
```

```
dn: cn=alerts
objectClass: top
objectClass: ds-alert-root
cn: alerts

dn: ds-alert-id=3d1857a2-e8cf-4e80-ac0e-ba933be59eca,cn=alerts
objectClass: top
objectClass: ds-admin-alert
ds-alert-id: 3d1857a2-e8cf-4e80-ac0e-ba933be59eca
ds-alert-type: server-started
ds-alert-severity: info
ds-alert-type-oid: 1.3.6.1.4.1.32473.2.11.33
ds-alert-time: 20110126041442.622Z
ds-alert-generator: com.unboundid.directory.server.core.directory.server
ds-alert-message: The Directory Server has started successfully
```

To Modify the Alert Retention Time

1. Use **dsconfig** to change the maximum time information about generated admin alerts is retained in the Alerts backend. After this time, the information gets purged from the directory server. The minimum retention time is 0 milliseconds, which immediately purges the alert information.

```
$ bin/dsconfig set-backend-prop --backend-name "alerts" \
  --set "alert-retention-time: 2 weeks"
```

2. View the property using **dsconfig**.

```
$ bin/dsconfig get-backend-prop --backend-name "alerts" \
  --property alert-retention-time

Property             : Value(s)
---------------------:---------
alert-retention-time : 2 w
```

Note	The Directory Server also supports a Admin Alert Access Log Publisher that generates administrative alerts when certain criteria appear in the File-Based Access Log. See "Managing Admin Alert Access Logs" on page 457.

To Configure Duplicate Alert Suppression

Use **dsconfig** to configure the maximum number of times an alert is generated within a particular timeframe for the same condition. The **duplicate-alert-time-limit** property specifies the length of time that must pass before duplicate messages are sent over the administrative alert framework. The **duplicate-alert-limit** property specifies the maximum number of duplicate alert messages should be sent over the administrative alert framework in the time limit specified in the **duplicate-alert-time-limit** property.

```
$ bin/dsconfig set-global-configuration-prop \
--set duplicate-alert-limit:2 \
--set "duplicate-alert-time-limit:3 minutes"
```

Testing Alerts

The Directory Server provides a mechanism to verify that it will properly handle various kinds of alerts as expected. Administrators can create a test entry to generate an arbitrary type of alert to verify that the server takes the appropriate action, such as sends out an email message, SNMP trap, JMX notification or however else notifications are configured to be handled. Typically, administrators can use this mechanism in a QA server for verification.

To Test Alerts

- Create a test entry, and add it to the server using **`ldapmodify`**.

```
dn:  ds-task-id=Test Leaving Lockdown Mode Alert,cn=Scheduled Tasks,cn=Tasks
objectClass: top
objectClass: ds-task
objectClass: ds-task-alert
ds-task-id: Test Leaving Lockdown Mode Alert
ds-task-class-name: com.unboundid.directory.server.tasks.AlertTask
ds-task-alert-type: leaving-lockdown-mode
ds-task-alert-message: Testing
ds-task-alert-remove-unavailable-type: entering-lockdown-mode

$ bin/ldapmodify --port 1389 --bindDN "cn=Directory Manager" --bindPassword \
  --defaultAdd --fileName test-alert-entry.ldif
```

This entry causes the server to remove "entering-lockdown-mode" to the set of unavailable alert types listed in the general monitor entry, and will also cause it to generate an administrative alert with an alert type of "leaving-lockdown-mode". This is basically the behavior that the server exhibits when you take it out of lockdown mode, but can be tested without actually putting the server in lockdown mode. Note however that this entry might have an impact on external systems like the Directory Proxy Server if they are consuming alerts or looking at the set of degraded or unavailable alert types.

15 Troubleshooting

Overview

The UnboundID Directory Server provides a highly-reliable directory service. However, if problems do arise (whether from issues in the Directory Server itself or a supporting component, like the JVM, operating system, or hardware), then it is essential to be able to diagnose the problem quickly to determine the underlying cause and the best course of action to take towards resolving it.

This chapter provides information about how to perform this analysis to help ensure that the problem is resolved as quickly as possible. It targets cases in which the Directory Server is running on Sun Solaris or Linux systems, but much of the information can be useful on other platforms as well.

This chapter presents the following information:

Using the Collect Support Data Tool

The Directory Server provides a significant amount of information about its current state including any problems that it has encountered during processing. If a problem occurs, the first step is to run the **`collect-support-data`** tool in the **`bin`** directory. The tool aggregates all relevant support files into a zip file that administrators can send to your authorized support provider for analysis. The tool also runs data collector utilities, such as **`jps`**, **`jstack`**, and **`jstat`** plus other diagnostic tools for Solaris and Linux machines, and bundles the results in the zip file.

Table 15-1 presents a summary of the data collectors that the `collect-support-data` tool archives in zip format. If an error occurs during processing, you can re-run the specific data collector command and send the results to your authorized support provider.

TABLE 15-1. Directory Server Commands Used in the Collect-Support-Data Tool

Data Collector	Description
status	Runs `status -F` to show the full version information of the Directory Server (Unix, Windows).
server-state	Runs `server-state` to show the current state of the Directory Server process (Unix, Windows).
dsreplication status	Runs `dsreplication status` to show the current state of the replication topology (Unix, Windows). If the `--noReplicationStatus` option is used, the replication status information is not collected.

TABLE 15-2. JDK Commands Used in the Collect-Support-Data Tool

Data Collector	Description
jps	Java Virtual Machine Process status tool. Reports information on the JVM (Solaris, Linux, Windows, Mac OS).
jstack	Java Virtual Machine Stack Trace. Prints the stack traces of threads for the Java process (Solaris, Linux, Windows, Mac OS).
jstat	Java Virtual Machine Statistics Monitoring Tool. Displays performance statistics for the JVM (Solaris, Linux, Windows, Mac OS).
jinfo	Displays the Java configuration information for the Java process (Solaris, Linux, Windows, Mac OS).

TABLE 15-3. Linux Commands Used in the collect-support-data Tool

Data Collector	Description
tail	Displays the last few lines of a file. Tails the /`var/logs/messages` directory.
uname	Prints system, machine, and operating system information.
ps	Prints a snapshot of the current active processes.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
cat	Concatenates the following files and prints to standard output: • `/proc/cpuinfo` • `/proc/meminfo` • `/etc/hosts` • `/etc/nsswitch.conf` • `/etc/resolv.conf`
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
uptime	Prints the time the server has been up and active.
dmesg	Prints the message buffer of the kernel.

TABLE 15-3. **Linux Commands Used in the collect-support-data Tool**

Data Collector	Description
vmstat	Prints information about virtual memory statistics.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.
pstack	Prints an execution stack trace on an active processed specified by the pid.
top	Prints a list of active processes and how much CPU and memory each process is using.

TABLE 15-4. **Solaris Commands Used in the collect-support-data Tool**

Data Collector	Description
uname	Prints system, machine, and operating system information.
ps	Prints a snapshot of the current active processes.
zonename	Prints the name of the current zone.
zoneadm	Prints the name of the current configured in verbose mode.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
zfs	Prints basic ZFS information: dataset pool names, and their used, available, referenced, and mountpoint properties.
zpool	Print a zpool's status.
fmdump	Prints the log files managed by the Solaris Fault Manager.
prtconf	Prints the system configuration information.
prtdiag	Prints the system diagnostic information.
cat	Concatenates the following files and prints to standard output: • **`/etc/system`** • **`/etc/release`** • **`/etc/hosts`** • **`/etc/nsswitch.conf`** • **`/etc/resolv.conf`**
tail	Displays the last few lines of a file. Tails the **`/var/adm/messages`** directory.
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
uptime	Prints the time the server has been up and active.
dmesg	Prints the message buffer of the kernel.
patchadd	Prints the patches added to the system if any (Solaris, not OpenSolaris).
vmstat	Prints information about virtual memory statistics.
iostat	Prints disk I/O and CPU utilization information.
mpstat	Prints performance statistics for all logical processors.
pstack	Prints an execution stack trace on an active processed specified by the pid.
prstat	Prints resource usage.

TABLE 15-5. MacOS Commands Used in the collect-support-data Tool

Data Collector	Description
uname	Prints system, machine, and operating system information.
uptime	Prints the time the server has been up and active.
ps	Prints a snapshot of the current active processes.
system_profiler	Prints system hardware and software configuration.
vm_stat	Prints machine virtual memory statistics.
tail	Displays the last few lines of a file. Tails the **`/var/log/system.log`** directory.
netstat	Prints the state of network interfaces, protocols, and the kernal routing table.
ifconfig	Prints information on all interfaces.
df	Prints the amount of available disk space for filesystems in 1024-byte units.
sample	Profiles a process during an interval.

Available Tool Options

The **`collect-support-data`** tool has some important options that you should be aware of:

- **--noLdap**. Specifies that no effort should be made to collect any information over LDAP. This option should only be used if the server is completely unresponsive or will not start and only as a last resort.

- **--noReplicationStatus**. Specifies that no effort should be made to collect **`dsreplication status`** information.

- **--pid {pid}**. Specifies the ID of an additional process from which information is to be collected. This option is useful for troubleshooting external server tools and can be specified multiple times for each external server, respectively.

- **--sequential**. Use this option to diagnose “Out of Memory” errors. The tool collects data in parallel to minimize the collection time necessary for some analysis utilities. This option specifies that data collection should be run sequentially as opposed to in parallel. This action has the effect of reducing the initial memory footprint of this tool at a cost of taking longer to complete.

- **--reportCount {count}**. Specifies the number of reports generated for commands that supports sampling (for example, vmstat, iostat, or mpstat). A value of 0 (zero) indicates that no reports will be generated for these commands. If this option is not specified, it defaults to 10.

- **--reportInterval {interval}**. Specifies the number of seconds between reports for commands that support sampling (for example, mpstat). This option must have a value greater than 0 (zero). If this option is not specified, it default to 1.

- **--maxJstacks {number}**. Specifies the number of `jstack` samples to collect. If not specified, the default number of samples collected is 10.
- **--collectExpensiveData**. Specifies that data on expensive or long running processes be collected. These processes are not collected by default, because they may impact the performance of a running server.
- **--comment {comment}**. Provides the ability to submit any additional information about the collected data set. This comment will be added to the generated archive as a README file.
- **--includeBinaryFiles**. Specifies that binary files be included in the archive collection. By default, all binary files are automatically excluded in data collection.
- **--adminPassword {adminPassword}**. Specifies the global administrator password used to obtain `dsreplication status` information.
- **--adminPasswordFile {adminPasswordFile}**. Specifies the file containing the password of the global administrator used to obtain `dsreplication status` information.

To Run the Collect Support Data Tool

1. Go to the installation directory.

```
$ cd /ds/UnboundID-DS
```

2. Use the `collect-support-data` tool. Make sure to include the host, port number, bind DN, and bind password.

```
$ bin/collect-support-data --hostname 127.0.0.1 --port 389 \
  --bindDN "uid=admin,dc=example,dc=com" --bindPassword secret \
  --serverRoot /opt/UnboundID-DS --pid 1234
```

3. Email the zip file to your Authorized Support Provider.

Note	You might need to change the .zip file extension to something like .zap to get through your mail servers.

Directory Server Troubleshooting Information

The Directory Server has a comprehensive default set of log files and monitor entries that are useful when troubleshooting a particular server problem.

Error Log

By default, this log file is available at **`logs/errors`** below the server install root and it provides information about warnings, errors, and other significant events that occur within the server. A number of messages are written to this file on startup and shutdown, but while the server is running there is normally little information written to it. In the event that a problem does occur, however, the server writes information about that problem to this file.

The following is an example of a message that might be written to the error log:

```
[11/Apr/2011:10:31:53.783 -0500] category=CORE severity=NOTICE
msgID=458887 msg="The Directory Server has started successfully"
```

The **`category`** field provides information about the area of the server from which the message was generated. Available categories include:

ACCESS_CONTROL	PROTOCOL	SCHEMA
ADMIN	JEB	SYNC
ADMIN_TOOL	LOG	TASK
BACKEND	PLUGIN	THIRD_PARTY
CONFIG	PROXY	TOOLS
CORE	QUICKSETUP	USER_DEFINED
DSCONFIG	REPLICATION	UTIL
EXTENSIONS	RUNTIME_INFORMATION	VERSION

The **`severity`** field provides information about how severe the server considers the problem to be. Available severities include:

- **`DEBUG`** – Used for messages that provide verbose debugging information and do not indicate any kind of problem. Note that this severity level is rarely used for error logging, as the Directory Server provides a separate debug logging facility as described below.

- **`INFORMATION`** – Used for informational messages that can be useful from time to time but are not normally something that administrators need to see.

- **`MILD_WARNING`** – Used for problems that the server detects, which can indicate something unusual occurred, but the warning does not prevent the server from completing the task it was working on. These warnings are not normally something that should be of concern to administrators.

- **`MILD_ERROR`** – Used for problems detected by the server that prevented it from completing some processing normally but that are not considered to be a significant problem requiring administrative action.

- **`NOTICE`** – Used for information messages about significant events that occur within the server and are considered important enough to warrant making available to administrators under normal conditions.

- **`SEVERE_WARNING`** – Used for problems that the server detects that might lead to bigger problem in the future and should be addressed by administrators.

- `SEVERE_ERROR` – Used for significant problems that have prevented the server from successfully completing processing and are considered important.
- `FATAL_ERROR` – Used for critical problems that arise which might leave the server unable to continue processing operations normally.

The messages written to the error log may be filtered based on their severities in two ways. First, the error log publisher has a `default-severity` property, which may be used to specify the severity of messages logged regardless of their category. By default, this includes the `NOTICE`, `SEVERE_WARNING`, `SEVERE_ERROR`, and `FATAL_ERROR` severities.

You can override these severities on a per-category basis using the `override-severity` property. If this property is used, then each value should consist of a category name followed by an equal sign and a comma-delimited set of severities that should be logged for messages in that category. For example, the following override severity would enable logging at all severity levels in the `PROTOCOL` category:

```
protocol=debug,information,mild-warning,mild-error,notice,severe-warning,severe-
error,fatal-error
```

Note that for the purposes of this configuration property, any underscores in category or severity names should be replaced with dashes. Also, severities are not inherently hierarchical, so enabling the `DEBUG` severity for a category will not automatically enable logging at the `INFORMATION`, `MILD_WARNING`, or `MILD_ERROR` severities.

The error log configuration may be altered on the fly using tools like `dsconfig`, the web administration console, or the LDIF connection handler, and changes will take effect immediately. You can configure multiple error logs that are active in the server at the same time, writing to different log files with different configurations. For example, a new error logger may be activated with a different set of default severities to debug a short-term problem, and then that logger may be removed once the problem is resolved, so that the normal error log does not contain any of the more verbose information.

server.out Log

The `server.out` file holds any information written to standard output or standard error while the server is running. Normally, it includes a number of messages written at startup and shutdown, as well as information about any administrative alerts generated while the server is running. In most cases, this information is also written to the error log. However, the `server.out` file can also contain output generated by the JVM. For example, if garbage collection debugging is enabled, or if a stack trace is requested via "kill -QUIT" as described in a later section, then output is written to this file.

Debug Log

The debug log provides a means of obtaining information that can be used for troubleshooting problems but is not necessary or desirable to have available while the server is functioning normally. As a result, the debug log is disabled by default, but it can be enabled and configured at any time.

Some of the most notable configuration properties for the debug log publisher include:

- `enabled` – Indicates whether debug logging is enabled. By default, it is disabled.
- `log-file` – Specifies the path to the file to be written. By default, debug messages are written to the `logs/debug` file.
- `default-debug-level` – Specifies the minimum log level for debug messages that should be written. The default value is "`error`," which only provides information about errors that occur during processing (for example, exception stack traces). Other supported debug levels include `warning`, `info`, and `verbose`. Note that unlike error log severities, the debug log levels are hierarchical. Configuring a specified debug level enables any debugging at any higher levels. For example, configuring the `info` debug level automatically enables the `warning` and `error` levels.
- `default-debug-category` – Specifies the categories for debug messages that should be written. Some of the most useful categories include `caught` (provides information and stack traces for any exceptions caught during processing), `database-access` (provides information about operations performed in the underlying database), `protocol` (provides information about ASN.1 and LDAP communication performed by the server), and `data` (provides information about raw data read from or written to clients).

As with the error and access logs, multiple debug loggers can be active in the server at any time with different configurations and log files in order to help isolate information that might be relevant to a particular problem.

Note	Enabling one or more debug loggers can have a significant impact on server performance. We recommend that debug loggers be enabled only when necessary, and they be scoped so that only pertinent debug information is recorded.

Debug targets can be used to futher pare down the set of messages generated. For example, you can specify that debug logs be generated only within a specific class or package. If you need to enable the debug logger, you should work with your authorized support provider to best configure the debug target and intpret the output.

Replication Repair Log

The replication repair log is written to `logs/replication` by default and records information about processing performed by the replication repair service. This log is used to resolve replication conflicts that can arise. For example, if the same entry is modified at the same time on two different systems, or if an attempt is made to create entries with the same DN at the same time on two different systems, the Directory Server records these events.

Config Audit Log and the Configuration Archive

The configuration audit log provides a record of any changes made to the server configuration while the server is online. This information is written to the **`logs/config-audit.log`** file and provides information about the configuration change in the form that may be used to perform the operation in a non-interactive manner with the **`dsconfig`** command. Other information written for each change includes:

- Time that the configuration change was made.
- Connection ID and operation ID for the corresponding change, which can be used to correlate it with information in the access log.
- DN of the user requesting the configuration change and the method by which that user authenticated to the server.
- Source and destination addresses of the client connection.
- Command that can be used to undo the change and revert to the previous configuration for the associated configuration object.

In addition to information about the individual changes that are made to the configuration, the Directory Server maintains complete copies of all previous configurations. These configurations are provided in the **`config/archived-configs`** directory and are gzip-compressed copies of the **`config/config.ldif`** file in use before the configuration change was made. The filenames contain time stamps that indicate when that configuration was first used.

Access and Audit Log

The access log provides information about operations processed within the server. The default access log file is written to **`logs/access`**, but multiple access loggers can be active at the same time, each writing to different log files and using different configurations.

By default, a single access log message is generated, which combines the elements of request, forward, and result messages. If an error is encountered while attempting to process the request, then one or more forward-failed messages may also be generated.

```
[01/Jun/2011:11:10:19.692 -0500] CONNECT conn=49 from="127.0.0.1" to="127.0.0.1" pro-
tocol="LDAP+TLS" clientConnectionPolicy="default"

[01/Jun/2011:11:10:19.764 -0500] BIND RESULT conn=49 op=0 msgID=1 version="3"
dn="cn=Directory Manager" authType="SIMPLE" resultCode=0 etime=0.401 authDN="cn=Direc-
tory Manager,cn=Root DNs,cn=config" clientConnectionPolicy="default"

[01/Jun/2011:11:10:19.769 -0500] SEARCH RESULT conn=49 op=1 msgID=2 base="ou=Peo-
ple,dc=example,dc=com" scope=2 filter="(uid=1)" attrs="ALL" resultCode=0 etime=0.549
entriesReturned=1

[01/Jun/2011:11:10:19.788 -0500] DISCONNECT conn=49 reason="Client Unbind"
```

Each log message includes a timestamp indicating when it was written, followed by the operation type, the connection ID (which is used for all operations processed on the same client connection), the operation ID (which can be used to correlate the request and response log messages for the operation), and the message ID used in LDAP messages for this operation.

The remaining content for access log messages varies based on the type of operation being processed, and whether it is a request or a result message. Request messages generally include the most pertinent information from the request, but generally omit information that is sensitive or not useful.

Result messages include a "**`resultCode`**" element that indicates whether the operation was successful or if failed and an "**`etime`**" element that indicates the length of time in milliseconds that the server spent processing the operation. Other elements that might be present include the following:

- **`origin=replication`** – Operation that was processed as a result of data synchronization (for example, replication) rather than a request received directly from a client.
- **`message`** – Text that was included in the **`diagnosticMessage`** field of the response sent to the client.
- **`additionalInfo`** – Additional information about the operation that was not included in the response sent back to the client.
- **`authDN`** – DN of the user that authenticated to the server (typically only included in bind result messages).
- **`authzDN`** – DN of an alternate authorization identify used when processing the operation (for example, if the proxied authorization control was included in the request).
- **`authFailureID`** – Unique identifier associated with the authentication failure reason (only included in non-successful bind result messages).
- **`authFailureReason`** – Information about the reason that a bind operation failed that might be useful to administrators but was not included in the response to the client for security reasons.
- **`responseOID`** – OID included in an extended response returned to the client.
- **`entriesReturned`** – Number of matching entries returned to the client for a search operation.
- **`unindexed=true`** – Indicates that the associated search operation could not be sufficiently processed using server indexes and a significant traversal through the database was required.

Note that this is not an exhaustive list, and elements that are not listed here may also be present in access log messages. The Commercial Edition of the LDAP SDK provides an API for parsing access log messages and provides access to all elements that they may contain.

The Directory Server provides a second access log implementation called the *audit log*, which is used to provide detailed information about write operations (add, delete, modify, and modify DN) processed within the server. If the audit log is enabled, the entire content of the change is written to the audit log file (which defaults to **`logs/audit`**) in LDIF form.

The UnboundID Directory Server also provides a very rich classification system that can be used to filter the content for access log files. This can be helpful when debugging problems with client applications, because it can restrict log information to operations processed only by a particular application (for example, based on IP address and/or authentication DN), only failed operations, or only operations taking a long time to complete, etc.

Setup Log File

The **`setup`** tool writes a log file providing information about the processing that it performs. By default, this log file is written to **`logs/setup.log`** although a different name may be used if a file with that name already exists, because the **`setup`** tool has already been run. The full path to the setup log file is provided when the setup tool has completed.

Tool Log Files

Many of the administrative tools provided with the Directory Server (for example, **`import-ldif`**, **`export-ldif`**, **`backup`**, **`restore`**, etc.) can take a significant length of time to complete write information to standard output or standard error or both while the tool is running. They also write additional output to files in the **`logs/tools`** directory (for example, **`logs/tools/import-ldif.log`**). The information written to these log files can be useful for diagnosing problems encountered while they were running. When running via the server tasks interface, log messages generated while the task is running may alternately be written to the server error log file.

je.info and je.config Files

The primary data store used by the Directory Server is the Oracle Berkeley DB Java Edition (JE). The Directory Server provides two primary sources of information about processing within the database.

The first is logging performed by the JE code itself, and is written into the **`je.info.0`** file in the directory containing the database files (for example, **`db/userRoot/je.info.0`**). In the event of a problem within JE itself, useful information about the nature of the problem may be written to this log. The level of information written to this log file is controlled by the `db-`**`logging-level`** property in the backend configuration object. It uses the standard Java logging framework for logging messages, so the standard **`SEVERE`**, **`WARNING`**, **`INFO`**, **`CONFIG`**, **`FINE`**, **`FINER`**, and **`FINEST`** levels are available.

The second is configuration information used when opening the database environment. When the backend database environment is opened, then the Directory Server will also write a file named **`je.config`** in the directory containing the database files (for example, **`db/userRoot/je.config`**) with information about the configuration used.

LDAP SDK Debug Log

This log can be used to help examine the communication between the Directory Server and the Directory Proxy Server. It contains information about exceptions that occur during processing, problems establishing and terminating network connections, and problems that occur during the reading and writing of LDAP messages and LDIF entries. You can configure the types of debugging that should be enabled, the debug level that should be used, and whether debug messages should include stack traces. As for other file-based loggers, you can also specify the rotation and retention policies.

Monitor Entries

While the Directory Server is running, it generates a significant amount of information available through monitor entries. Monitor entries are available over LDAP in the "`cn=monitor`" subtree. The types of monitor entries that are available include:

- General Monitor Entry (`cn=monitor`) – Provides a basic set of general information about the server.
- Active Operations Monitor Entry (`cn=Active Operations,cn=monitor`) – Provides information about all operations currently in progress in the server.
- Backend Monitor Entries (`cn={id} Backend,cn=monitor`) – Provides information about the backend, including the number of entries, the base DN(s), and whether it is private.
- Client Connections Monitor Entry (`cn=Client Connections,cn=monitor`) – Provides information about all connections currently established to the server.
- Connection Handler Monitor Entry (`cn={name},cn=monitor`) – Provides information about the configuration of each connection handler and the client connections established to it.
- Database Environment Monitor Entries (`cn={id} Database Environment,cn=monitor`) – Provides statistics and other data from the Oracle Berkeley DB Java Edition database environment used by the associated backend.
- Disk Space Usage Monitor Entry (`cn=Disk Space Usage,cn=monitor`) – Provides information about the amount of usable disk space available to server components.
- JVM Memory Usage Monitor Entry (`cn=JVM Memory Usage,cn=monitor`) – Provides information about garbage collection activity, the amount of memory available to the server, and the amount of memory consumed by various server components.
- JVM Stack Trace Monitor Entry (`cn=JVM Stack Trace,cn=monitor`) – Provides a stack trace of all threads in the JVM.
- LDAP Statistics Monitor Entries (`cn={name} Statistics,cn=monitor`) – Provides information about the number of each type of operation requested and bytes transferred over the connection handler.
- Processing Time Histogram Monitor Entry (`cn=Processing Time Histogram,cn=monitor`) – Provides information about the number of percent of operations that completed in various response time categories.

- System Information Monitor Entry (`cn=System Information,cn=monitor`) – Provides information about the underlying JVM and system.
- Version Monitor Entry (`cn=Version,cn=monitor`) – Provides information about the Directory Server version.
- Work Queue Monitor Entry (`cn=Work Queue,cn=monitor`) – Provides information about the state of the Directory Server work queue, including the number of operations waiting on worker threads and the number of operations that have been rejected because the queue became full.

Directory Server Troubleshooting Tools

The UnboundID Directory Server provides a set of tools that can also be used to obtain information for diagnosing and solving problems.

Server Version Information

If it becomes necessary to contact your authorized support provider, then it will be important to provide precise information about the version of the Directory Server software that is in use. If the server is running, then this information can be obtained from the "`cn=Version,cn=monitor`" entry. It can also be obtained using the command:

```
$ bin/status --fullVersion
```

This command outputs a number of important pieces of information, including:

- Major, minor, point and patch version numbers for the server.
- Source revision number from which the server was built.
- Build information including build ID with time stamp, OS, user, Java and JVM version for the build.
- Auxiliary software versions: Oracle Berkeley DB JE, JZlib, SNMP4J (SNMP4J, Agent, Agentx), and LDAP SDK for Java version.

LDIF Connection Handler

The Directory Server provides an LDIF connection handler that provides a way to request operations that do not require any network communication with the server. This can be particularly helpful if a configuration problem or bug in the server has left a connection handler unusable, or if all worker threads are busy processing operations.

The LDIF connection handler is enabled by default and looks for LDIF files to be placed in the `config/auto-process-ldif` directory. This directory does not exist by default, but if it is created and an LDIF file is placed in it containing one or more changes to be processed, then those changes will be applied.

Any changes that can be made over LDAP can be applied through the LDIF connection handler. It is primarily intended for administrative operations like updating the server configuration or scheduling tasks, although other types of changes (including changes to data contained in the server) can be processed. As the LDIF file is processed, a new file is written with comments for each change providing information about the result of processing that change.

dbtest Tool

The **`dbtest`** tool provides a utility that can be used to obtain general information about the data in a backend database. The tool dumps information about entries or keys, and raw data from the database. It can also find keys that have exceeded the entry limit.

For example, the following command can be used to dump a list of all keys in the **`objectClass.equality`** that have exceeded the entry threshold:

```
$ bin/dbtest dump-database-container \
  --backendID userRoot \
  --baseDN "dc=example,dc=com" \
  --databaseName objectClass.equality \
  --onlyExceedingLimit
```

Note that on a large database, many **`dbtest`** operations might take a very long time to complete as it is necessary to examine every record in the associated database. You can use the database name option to list a specific database. The following command displays information about the **`uid.equality`** database in the **`dc=example,dc=com`** entry container in the **`userRoot`** backend.

```
$ bin/dbtest list-database-containers -n userRoot -b "dc=example,dc=com" \
  -d uid.equality
```

Embedded Profiler

If the Directory Server appears to be running slowly, then it is helpful to know what operations are being processed in the server. The JVM Stack Trace monitor entry can be used to obtain a point-in-time snapshot of what the server is doing, but in many cases, it might be useful to have information collected over a period of time.

The embedded profiler is configured so that it is always available but is not active by default so that it has no impact on the performance of the running server. Even when it is running, it has a relatively small impact on performance, but it is recommended that it remain inactive when it is not needed. It can be controlled using the **`dsconfig`** tool or the web administration console by managing the "Profiler" configuration object in the "Plugin" object type, available at the standard object level. The **`profile-action`** property for this configuration object can have one of the following values:

- start – Indicates that the embedded profiler should start capturing data in the background.

- stop – Indicates that the embedded profiler should stop capturing data and write the information that it has collected to a “**`logs/profile{timestamp}`**” file.

- cancel – Indicates that the embedded profiler should stop capturing data and discard any information that it has collected.

Any profiling data that has been captured can be examined using the `profiler-viewer` tool. This tool can operate in either a text-based mode, in which case it dump a formatted text representation of the profile data to standard output, or it can be used in a graphical mode that allows the information to be more easily understood.

To Invoke the Profile Viewer in Text-based Mode

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z
```

To Invoke the Profile View in GUI Mode

```
$ bin/profile-viewer --fileName logs/profile.20110101000000Z --useGUI
```

Oracle Berkeley DB Java Edition Utilities

The Oracle Berkeley DB Java Edition (JE) itself provides a number of utilities that can be used for performing various types of low-level debugging in the database environment. These utilities should generally not be used unless you are advised to do so by your authorized support provider, but they provide access to information about the underlying database environment that is not available through any other means.

Troubleshooting Resources for Java Applications

Because the UnboundID Directory Server is written entirely in Java, it is possible to use standard Java debugging and instrumentation tools when troubleshooting problems with the Directory Server. In many cases, obtaining the full benefit of these tools requires access to the Directory Server source code. These Java tools should be used under the advisement of your authorized support provider.

Java Troubleshooting Documentation

There are a number of documents providing general information about troubleshooting Java-based applications. Some of these documents include:

- http://www.oracle.com/technetwork/java/javase/index-138283.html – Troubleshooting Java SE

- http://www.oracle.com/technetwork/java/javase/index-137495.html – Troubleshooting Guide for Java SE 6 with HotSpot VM

- http://www.sun.com/bigadmin/hubs/java/troubleshoot/ – BigAdmin Page on Java SE Troubleshooting
- http://www.oracle.com/technetwork/java/javase/tools6-unix-139447.html – Tools for troubleshooting Java on Solaris and Linux

Java Troubleshooting Tools

The Java Development Kit provides a number of very useful tools to obtain information about Java applications and diagnosing problems. These tools are not included with the Java Runtime Environment™ (JRE), so the full Java Development Environment (JDK) should always be installed and used to run the UnboundID Directory Server.

jps

The jps tool is a Java-specific version of the UNIX **`ps`** tool. It can be used to obtain a list of all Java processes currently running and their respective process identifiers. When invoked by a non-root user, it will list only Java processes running as that user. When invoked by a root user, then it lists all Java processes on the system.

This tool can be used to see if the Directory Server is running and if a process ID has been assigned to it. This process ID can be used in conjunction with other tools to perform further analysis.

This tool can be run without any arguments, but some of the more useful arguments that include:

- -v – Includes the arguments passed to the JVM for the processes that are listed.
- -m – Includes the arguments passed to the main method for the processes that are listed.
- -l – (lowercase L). Include the fully qualified name for the main class rather than only the base class name.

Additional documentation for the **`jps`** tool is available at:

http://java.sun.com/javase/6/docs/techs/tools/share/jps.html

jstack

The **`jstack`** tool is used to obtain a stack trace of a running Java process, or optionally from a core file generated if the JVM happens to crash. A stack trace can be extremely valuable when trying to debug a problem, because it provides information about all threads running and exactly what each is doing at the point in time that the stack trace was obtained.

Stack traces are helpful when diagnosing problems in which the server appears to be hung or behaving slowly. Java stack traces are generally more helpful than native stack traces, because Java threads can have user-friendly names (as do the threads used by the UnboundID Directory Server), and the frame of the stack trace may include the line number of the source file to which it corresponds. This is useful when diagnosing problems and often allows them to be identified and resolved quickly.

In order to obtain a stack trace from a running JVM, use the command:

```
jstack {processID}
```

where *{processID}* is the process ID of the target JVM as returned by the `jps` command. To obtain a stack trace from a core file from a Java process, use the command:

```
jstack {pathToJava} {pathToCore}
```

where *{pathToJava}* is the path to the java command from which the core file was created, and *{pathToCore}* is the path to the core file to examine. In either case, the stack trace is written to standard output and includes the names and call stacks for each of the threads that were active in the JVM.

In many cases, no additional options are necessary. The "`-l`" option can be added to obtain a long listing, which includes additional information about locks owned by the threads. The "`-m`" option can be used to include native frames in the stack trace.

Additional documentation for the jstack tool is available at http://java.sun.com/javase/6/docs/techs/tools/share/jstack.html.

jmap

The `jmap` tool is used to obtain information about the memory consumed by the JVM. It is very similar to the native `pmap` tool provided by many operating systems. As with the `jstack` tool, `jmap` can be invoked against a running Java process by providing the process ID, or against a core file, like:

```
jmap {processID}
jmap {pathToJava} {pathToCore}
```

Some of the additional arguments include:

- **-dump:live,format=b,file=filename** – Dump the live heap data to a file that can be examined by the jhat tool

- **-heap** – Provides a summary of the memory used in the Java heap, along with information about the garbage collection algorithm in use.

- **-histo:live** – Provides a count of the number of objects of each type contained in the heap. If the ":live" portion is included, then only live objects are included; otherwise, the count include objects that are no longer in use and are garbage collected.

Additional information about the `jmap` tool can be found at http://java.sun.com/javase/6/docs/techs/tools/share/jmap.html.

jhat

The `jhat` (Java Heap Analysis Tool) utility provides the ability to analyze the contents of the Java heap. It can be used to analyze a heap dump file, which is generated if the Directory

Server encounters an out of memory error (as a result of the "**`-XX:+HeapDumpOnOutOfMemory-Error`**" JVM option) or from the use of the **`jmap`** command with the "**`-dump`**" option.

The **`jhat`** tool acts as a web server that can be accessed by a browser in order to query the contents of the heap. Several predefined queries are available to help determine the types of objects consuming significant amounts of heap space, and it also provides a custom query language (OQL, the Object Query Language) for performing more advanced types of analysis.

The **`jhat`** tool can be launched with the path to the heap dump file, like:

```
jhat /path/to/heap.dump
```

This command causes the **`jhat`** web server to begin listening on port 7000. It can be accessed in a browser at **`http://localhost:7000`** (or **`http://address:7000`** from a remote system). An alternate port number can be specified using the "**`-port`**" option, like:

```
jhat -port 1234 /path/to/heap.dump
```

In order to issue custom OQL searches, access the web interface using the URL **`http://localhost:7000/oql/`** (the trailing slash must be provided). Additional information about the OQL syntax may be obtained in the web interface at **`http://localhost:7000/oqlhelp/`**. Additional information for the **`jhat`** tool may be found at http://java.sun.com/javase/6/docs/techs/tools/share/jhat.html.

jstat

The **`jstat`** tool is used to obtain a variety of statistical information from the JVM, much like the **`vmstat`** utility that can be used to obtain CPU utilization information from the operating system. The general manner to invoke it is as follows:

```
jstat {type} {processID} {interval}
```

The *{interval}* option specifies the length of time in milliseconds between lines of output. The *{processID}* option specifies the process ID of the JVM used to run the Directory Server, which can be obtained by running `jps` as mentioned previously. The *{type}* option specifies the type of output that should be provided. Some of the most useful types include:

- -class – Provides information about class loading and unloading.
- -compile – Provides information about the activity of the JIT complex.
- -printcompilation – Provides information about JIT method compilation.
- -gc – Provides information about the activity of the garbage collector.
- -gccapacity – Provides information about memory region capacities.

Java Diagnostic Information

In addition to the tools listed in the previous section, the JVM can provide additional diagnostic information in response to certain events.

Garbage Collection Diagnostic Information

If the JVM is not properly configured, then garbage collection activity can cause significant pauses in the execution of the Java code running in it. While the `jstat` utility with the `-gc` option can be used to obtain some information about garbage collection activity, there are additional arguments that can be added to the JVM to use when running the server to provide additional detail.

Those arguments include:

```
-XX:+PrintGCDetails
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationConcurrentTime
-XX:+PrintGCApplicationStoppedTime
-XX:+PrintGCDateStamps
```

In order to run the Directory Server with these options, edit the `config/java.properties` file and add them to the end of the line that begins with "`start-ds.java-args`". After the file has been saved, invoke the following command to make those new arguments take effect the next time the server is started:

```
$ bin/dsjavaproperties
```

JVM Crash Diagnostic Information

If the JVM itself should happen to crash for some reason, then it generates a fatal error log with information about the state of the JVM at the time of the crash. By default, this file is named `hs_err_pid{processID}.log` and is written into the directory that is the base of the Directory Server installation. This file includes information the underlying cause of the JVM crash, information about the threads running and Java heap at the time of the crash, the options provided to the JVM, environment variables that were set, and information about the underlying system. More information about the content that may be written to this log file may be found at http://java.sun.com/javase/6/webs/trouble/TSG-VM/html/felog.html.

Troubleshooting Resources in the Operating System

The underlying operating system also provides a significant amount of information that can help diagnose issues that impact the performance and the stability of the Directory Server. In some cases, problems with the underlying system can be directly responsible for the issues seen with the directory server, and in others system, tools can help narrow down the cause of the problem.

Identifying Problems with the Underlying System

If the underlying system itself is experiencing problems, it can adversely impact the function of applications running on it. Places to look for problems in the underlying system include:

- The system log file (/var/adm/messages on Solaris and /var/log/messages on Linux). Information about faulted or degraded devices or other unusual system conditions are written there.
- On Solaris systems, if the fault management system has detected a problem with a system component, information about that problem is obtain by running the **`fmdump`** command.
- If the ZFS filesystem is in use, then the zpool status command provides information about read errors, write errors, or data checksum errors.

Examining CPU Utilization

Observing CPU utilization for the Directory Server process and the system as a whole provides clues as to the nature of the problem.

System-Wide CPU Utilization

To investigate CPU consumption of the system as a whole, use the **`vmstat`** command with a time interval in seconds, like:

```
vmstat 5
```

The specific output of this command varies between different operating systems, but it includes the percentage of the time the CPU was spent executing user-space code (user time), the percentage of time spent executing kernel-space code (system time), and the percentage of time not executing any code (idle time).

If the CPUs are spending most of their time executing user-space code, the available processors are being well-utilized. If performance is poor or the server is unresponsive, it can indicate that the Directory Server is not optimally tuned. If there is a high system time, it can indicate that the system is performing excessive disk and/or network I/O, or in some cases, there can be some other system-wide problem like an interrupt storm. If the system is mostly idle but the Directory Server is performing poorly or is unresponsive, there can be a resource constraint elsewhere (for example, waiting on disk or memory access, or excessive lock contention), or the JVM can be performing other tasks like stop-the-world garbage collection that cannot be run heavily in parallel.

Per-CPU Utilization

To investigate CPU consumption on a per-CPU basis, use the **`mpstat`** command with a time interval in seconds, like:

```
mpstat 5
```

On Linux systems, it might be necessary to add "**-P ALL**" to the command, like:

```
mpstat -P ALL 5
```

Among other things, this shows the percentage of time each CPU has spent in user time, system time, and idle time. If the overall CPU utilization is relatively low but **mpstat** reports that one CPU has a much higher utilization than the others, there might be a significant bottleneck within the server or the JVM might be performing certain types of garbage collection which cannot be run in parallel. On the other hand, if CPU utilization is relatively even across all CPUs, there is likely no such bottleneck and the issue might be elsewhere.

Per-Process Utilization

To investigate CPU consumption on a per-process basis, use the **prstat** tool on Solaris or the **top** utility on Linux. If a process other than the Java process used to run the Directory Server is consuming a significant amount of available CPU, it might be interfering with the ability of the Directory Server to run effectively.

If the **mpstat** command showed that one CPU was much more heavily utilized than the others, it might be useful to identify the thread with the highest CPU utilization as it is likely the one that is a bottleneck preventing other threads from processing. On Solaris, this can be achieved by using the **prstat** command with the "-L" option, like:

```
prstat -L -p {processID}
```

This command will cause each thread to be displayed on a separate line, with the LWPID (lightweight process identifier) displayed as the last item on each line, separated from the process name by a slash. The thread that is currently consuming the largest amount of CPU will be displayed at the top of the list, and the **pstack** command can be used to identify which thread is responsible.

Examining Disk Utilization

If the underlying system has a very high disk utilization, it can adversely impact Directory Server performance. It could delay the ability to read or write database files or write log files. It could also raise concerns for server stability if excessive disk I/O inhibits the ability of the cleaner threads to keep the database size under control.

The **iostat** tool may be used to obtain information about the disk activity on the system. On Solaris systems, this should be invoked using the "-x" and "-n" arguments, like:

```
iostat -x -n 5
```

On Linux systems, **iostat** should be invoked with the "-x" argument, like:

```
iostat -x 5
```

A number of different types of information will be displayed, but in order to obtain an initial feel for how busy the underlying disks are, look at the "%b" column on Solaris and the "%util" column on Linux. Both of these fields show the percentage of the time that the underlying disks are actively servicing I/O requests. A system with a high disk utilization likely exhibits poor Directory Server performance.

If the high disk utilization is on one or more disks that are used to provide swap space for the system, the system might not have enough free memory to process requests. As a result, it might have started swapping blocks of memory that have not been used recently to disk. This can cause very poor server performance. It is important to ensure that the server is configured appropriately to avoid this condition. If this problem occurs on a regular basis, then the server is likely configured to use too much memory. If swapping is not normally a problem but it does arise, then check to see if there are any other processes running, which are consuming a significant amount of memory, and check for other potential causes of significant memory consumption (for example, large files in a **`tmpfs`** filesystem).

On Solaris systems using ZFS, you can use the **`zpool iostat {interval}`** command to obtain information about I/O activity on a per-pool basis. While this command provides a useful display of the number of read and write operations and the amount of data being read from and written to the disks, it does not actually show how busy the underlying disks. As a result, the **`zpool iostat`** command is generally not as useful as the traditional **`iostat`** command for identifying potential I/O bottlenecks.

Examining Process Details

There are a number of tools provided by the operating system that can help examine a process in detail.

ps

The standard **`ps`** tool can be used to provide a range of information about a particular process. For example, the command can be used to display the state of the process, the name of the user running the process, its process ID and parent process ID, the priority and nice value, resident and virtual memory sizes, the start time, the execution time, and the process name with arguments:

```
ps -fly -p {processID}
```

Note that for a process with a large number of arguments, the standard `ps` command displays only a limited set of the arguments based on available space in the terminal window. In that case, the BSD version of the `ps` command (available on Solaris as **`/usr/ucb/ps`**) can be used to obtain the full command with all arguments, like:

```
/usr/ucb/ps auxwww {processID}
```

pstack

The **`pstack`** command can be used to obtain a native stack trace of all threads in a process. While a native stack trace might not be as user-friendly as a Java stack trace obtained using **`jstack`**, it includes threads that are not available in a Java stack trace. For example, the command displays those threads used to perform garbage collection and other housekeeping tasks. The general usage for the pstack command is:

```
pstack {processID}
```

dbx / gdb

A process debugger provides the ability to examine a process in detail. Like **`pstack`**, a debugger can obtain a stack trace for all threads in the process, but it also provides the ability to examine a process (or core file) in much greater detail, including observing the contents of memory at a specified address and the values of CPU registers in different frames of execution. The GNU debugger **`gdb`** is widely-used on Linux systems and is available on Solaris, but the Sun Studio debugger **`dbx`** is generally preferred over **`gdb`** on Solaris.

Note that using a debugger against a live process interrupts that process and suspends its execution until it detaches from the process. In addition, when running against a live process, a debugger has the ability to actually alter the contents of the memory associated with that process, which can have adverse effects. As a result, it is recommended that the use of a process debugger be restricted to core files and only used to examine live processes under the direction of your authorized support provider.

pfiles / lsof

In order to examine the set of files that a process is using (including special types of files, like sockets) on Solaris, you can use the **`pfiles`** command, like:

```
pfiles {processID}
```

On Linux systems, the `lsof` tool can be used, like:

```
lsof -p {processID}
```

Tracing Process Execution

If a process is unresponsive but is consuming a nontrivial amount of CPU time, or if a process is consuming significantly more CPU time than is expected, it might be useful to examine the activity of that process in more detail than can be obtained using a point-in-time snapshot like you can get with **`pstack`** or a debugger. For example, if a process is performing a significant amount of disk reads and/or writes, it can be useful to see which files are being accessed. Similarly, if a process is consistently exiting abnormally, then beginning tracing for that process just before it exits can help provide additional information that cannot be captured in a core file (and if the process is exiting rather than being terminated for an illegal operation, then no core file may be available).

On Solaris systems, the **`dtrace`** tool provides an unmatched mechanism for tracing the execution of a process in extremely powerful and flexible ways, but it is also relatively complex and describing its use is beyond the scope of this document. In many cases, however, observing the system calls made by a process can reveal a great deal about what it is doing. This can be accomplished using the **`truss`** utility on Solaris or the **`strace`** tool on Linux.

The `truss` utility is very powerful and has a lot of options, but two of the most useful forms in which it may be invoked are:

- `truss -f -p {processID}` – Provides a basic overview of all system calls being made by the specified process (and any subprocesses that it creates) and their associated return values.
- `truss -fear all -p {processID}` – Provides an extremely verbose trace of all system call activity, including details about data being read from or written to files and sockets.

In both cases, the output may be written to a file instead of the terminal window by adding the `-o {path}` option. Further, rather than observing an already-running process, it is possible to have `truss` launch the process and trace execution over its entire life span by replacing `-p {processID}` with name and arguments for the command to invoke.

On Linux systems, the basic equivalent of the first truss variant above is:

```
strace -f -p {processID}
```

Consult the `strace` manual page for additional information about using it to trace process execution on Linux.

Examining Network Communication

Because the UnboundID Directory Server is a network-based application, it can be valuable to observe the network communication that it has with clients. The Directory Server itself can provide details about its interaction with clients by enabling debugging for the `protocol` or `data` debug categories, but there may be a number of cases in which it is useful to view information at a much lower level. A network sniffer, like the `snoop` tool on Solaris or the `tcpdump` tool on Linux, can be used to accomplish this.

There are many options that can be used with these tools, and their corresponding manual pages will provide a more thorough explanation of their use. However, in order to perform basic tracing to show the full details of the packets received for communication on port `389` with remote host `1.2.3.4`, the following commands can be used on Solaris and Linux, respectively:

```
snoop -d {interface} -r -x 0 host 1.2.3.4 port 389

tcpdump -i {interface} -n -XX -s 0 host 1.2.3.4 and port 389
```

On Solaris systems, the `snoop` command provides enhanced support for parsing LDAP communication (but only when the Directory Server is listening on the default port of `389`). By adding the "`-v`" argument to the snoop command line, a verbose breakdown of each packet will be displayed, including protocol-level information. It does not appear that the `tcpdump` tool provides support for LDAP parsing. However, in either case it is possible to write capture data to a file rather than displaying information on the terminal (using "`-o {path}`" with `snoop`, or "`-w {path}`" with `tcpdump`), so that information can be later analyzed with a graphical tool like Wireshark, which provides the ability to interpret LDAP communication on any port.

Note that enabling network tracing generally requires privileges that are not available to normal users and therefore may require root access. On Solaris systems, granting the

net_rawaccess privilege to a user should be sufficient to allow that user to run the **snoop** utility.

Common Problems and Potential Solutions

This section describes a number of different types of problems that can occur and common potential causes for them.

General Methodology to Troubleshoot a Problem

When a problem is detected, UnboundID recommends using the following general methodology to isolate the problem:

1. Run the **bin/status** tool or look at the server status in the web console. The **status** tool provides a summary of the server's current state with key metrics and a list of recent alerts.

2. Look in the server logs. In particular, view the following logs:

   - logs/error
   - logs/failed-ops
   - logs/expensive-ops

3. Use system commands, such as **vmstat** and **iostat** to determine if the server is bottlenecked on a system resource like CPU or disk throughput.

4. For performance problem (especially intermittent ones like spikes in response time), enabling the **periodic-stats-logger** can help to isolate problems, because it stores important server performance information on a per-second basis. The **periodic-stats-logger** can save the information in a csv-formatted file that can be loaded into a spreadsheet. The information this logger makes available is very configurable. You can create multiple loggers for different types of information or a different frequency of logging (for example, hourly data in addition to per-second data). For more information, see "Profiling Server Performance Using the Periodic Stats Logger" on page 484.

5. For replication problem, run **dsreplication status** and look at the **logs/replication** file.

6. For more advanced users, run the **collect-support-data** tool on the system, unzip the archive somewhere, and look through the collected information. This is often useful when administrators most familiar with the directory service do not have direct access to the systems where the production servers are running. They can examine the **collect-support-**

data archive on a different server. For more information, see “Using the Collect Support Data Tool” on page 507.

Important	Run the **collect-support-data** tool whenever there is a problem whose cause is not easily identified, so that this information can be passed back to Support before corrective action can be taken.

The Server Will Not Run Setup

If the **setup** tool does not run properly, some of the most common reasons include:

A Suitable Java Environment Is Not Available

The UnboundID Directory Server requires that Java 1.6.0_25 to be installed on the system and made available to the server, and it must be installed prior to running **setup**. If the **setup** tool does not detect that a suitable Java environment is available, it will refuse to run.

In order to ensure that this does not happen, the `setup` tool should be invoked with an explicitly-defined value for the `JAVA_HOME` environment variable that specifies the path to the Java installation that should be used. For example:

```
env JAVA_HOME=/ds/java ./setup
```

If this still does not work for some reason, then it can be that the value specified in the provided `JAVA_HOME` environment variable can be overridden by another environment variable. If that occurs, try the following command, which should override any other environment variables that can be set:

```
env UNBOUNDID_JAVA_HOME="/ds/java" \
    UNBOUNDID_JAVA_BIN="" ./setup
```

Oracle Berkeley DB Java Edition Is Not Available

If the version of the Directory Server that you are using was not provided with the Oracle Berkeley DB Java Edition library, then it must be manually downloaded and the appropriate JAR file placed in the **lib** directory before running **setup**. See the **lib/downloading-je.txt** file for instructions on obtaining the appropriate library.

Unexpected Arguments Provided to the JVM

If the **setup** script attempts to launch the **java** command with an invalid set of Java arguments, it might prevent the JVM from starting. By default, no special options are provided to the JVM when running **setup**, but this might not be the case if either the **JAVA_ARGS** or **UNBOUNDID_JAVA_ARGS** environment variable is set. If the **setup** tool displays an error message that indicates that the Java environment could not be started with the provided set of arguments, then invoke the following command before trying to re-run **setup**:

```
unset JAVA_ARGS UNBOUNDID_JAVA_ARGS
```

The Server Has Already Been Configured or Used

The `setup` tool is only intended to provide the *initial* configuration for the Directory Server. It refuses to run if it detects that the `setup` tool has already been run, or if an attempt has been made to start the Directory Server prior to running the `setup` tool. This protects an existing Directory Server installation from being inadvertently updated in a manner that could harm an existing configuration or data set.

If the Directory Server has been previously used and if you want to perform a fresh installation, it is recommended that you first remove the existing installation, create a new one and run `setup` in that new installation. However, if you are confident that there is nothing of value in the existing installation (for example, if a previous attempt to run `setup` failed to complete successfully for some reason but it will refuse to run again), the following steps can be used to allow the `setup` program to run:

- Remove the `config/config.ldif` file and replace it with the `config/update/config.ldif`.*{revision}* file containing the initial configuration.
- If there are any files or subdirectories below the `db` directory, then remove them.
- If a `config/java.properties` file exists, then remove it.
- If a `lib/setup-java-home` script (or `lib\set-java-home.bat` file on Microsoft Windows) exists, then remove it.

The Server Will Not Start

If the Directory Server does not start, then there are a number of potential causes.

The Server or Other Administrative Tool Is Already Running

Only a single instance of the Directory Server can run at any time from the same installation root. If an instance is already running, then subsequent attempts to start the server will fail. Similarly, some other administrative operations (for example, restoring a backup or importing data from an LDIF file) can also prevent the server from being started. In such cases, the attempt to start the server should fail with a message like:

```
The Directory Server could not acquire an exclusive lock on file /ds/UnboundID-DS/
locks/server.lock:  The exclusive lock requested for file /ds/UnboundID-DS/locks/
server.lock was not granted, which indicates that another process already holds a
shared or exclusive lock on that file.  This generally means that another instance
of this server is already running
```

If the Directory Server is not running (and is not in the process of starting up or shutting down) and there are no other tools like `restore` or `import-ldif` running that could prevent the server from being started, and the server still believes that it is running, then it is possible that a previously-held lock was not properly released. In that case, you can try removing all of the files in the locks directory before attempting to start the server.

If you wish to have multiple instances running at the same time on the same system, then you should create a completely separate installation in another location on the filesystem.

There Is Not Enough Memory Available

When the Directory Server is started, the JVM attempts to allocate all memory that it has been configured to use. If there is not enough free memory available on the system, then the Directory Server generates an error message that indicates that the server could not be started with the specified set of arguments. Note that it is possible that an invalid option was provided to the JVM (as described below), but if that same set of JVM arguments has already been used successfully to run the server, then it is more likely that the system does not have enough memory available.

There are a number of potential causes for this:

- If the amount of memory in the underlying system has changed (for example, system memory has been removed, or if the Directory Server is running in a zone or other type of virtualized container and a change has been made to the amount of memory that container will be allowed to use), then the Directory Server might need to be re-configured to use a smaller amount of memory than had been previously configured.

- Another process running on the system is consuming a significant amount of memory so that there is not enough free memory available to start the server. If this is the case, then either terminate the other process to make more memory available for the Directory Server, or reconfigure the Directory Server to reduce the amount of memory that it attempts to use.

- The Directory Server was just shut down and an attempt was made to immediately restart it. In some cases, if the server is configured to use a significant amount of memory, then it can take a few seconds for all of the memory that had been in use by the server, when it was previously running, to be released back to the operating system. In that case, run the **`vmstat`** command and wait until the amount of free memory stops growing before attempting to restart the server.

- For Solaris-based systems only, if the system has one or more ZFS filesystems (even if the Directory Server itself is not installed on a ZFS filesystem), but it has not been configured to limit the amount of memory that ZFS can use for caching, then it is possible that ZFS caching is holding onto a significant amount of memory and cannot release it quickly enough when it is needed by the Directory Server. In that case, the system should be re-configured to limit the amount of memory that ZFS is allowed to use as described in the Restricting ZFS Memory Consumption section on page 19.

- If the system is configured with one or more memory-backed filesystems for example, **`tmpfs`** used for /**`tmp`** for Solaris), then look to see if there are any large files that can be consuming a significant amount of memory in any of those locations. If so, then remove them or relocate them to a disk-based filesystem.

- For Linux systems only, if there is a mismatch between the large pages setting for the JVM and the large pages reserved in the operating system. For more information, see “Configure Large Page Support (Linux)” on page 13.

If nothing else works and there is still not enough free memory to allow the JVM to start, then as a last resort, try rebooting the system.

An Invalid Java Environment or JVM Option Was Used

If an attempt to start the Directory Server fails with an error message indicating that no valid Java environment could be found, or indicates that the Java environment could not be started with the configured set of options, then you should first ensure that enough memory is available on the system as described above. If there is a sufficient amount of memory available, then other causes for this error can include the following:

- The Java installation that was previously used to run the server no longer exists (for example, an updated Java environment was installed and the old installation was removed). In that case, update the **`config/java.properties`** file to reference to path to the new Java installation and run the **`bin/dsjavaproperties`** command to apply that change.

- The Java installation used to run the server has been updated and the server is trying to use the correct Java installation but one or more of the options that had worked with the previous Java version no longer work with the new version. In that case, it is recommended that the server be re-configured to use the previous Java version, so that it can be run while investigating which options should be used with the new installation.

- If an **`UNBOUNDID_JAVA_HOME`** or **`UNBOUNDID_JAVA_BIN`** environment variable is set, then its value may override the path to the Java installation used to run the server as defined in the **`config/java.properties`** file. Similarly, if an **`UNBOUNDID_JAVA_ARGS`** environment variable is set, then its value might override the arguments provided to the JVM. If this is the case, then explicitly unset the **`UNBOUNDID_JAVA_HOME`**, **`UNBOUNDID_JAVA_BIN`**, and **`UNBOUNDID_JAVA_ARGS`** environment variables before trying to start the server.

Note that any time the **`config/java.properties`** file is updated, the **`bin/dsjavaproperties`** tool must be run to apply the new configuration. If a problem with the previous Java configuration prevents the **`bin/dsjavaproperties`** tool from running properly, then it can be necessary to remove the **`lib/set-java-home`** script (or **`lib\set-java-home.bat`** file on Microsoft Windows) and invoke the **`bin/dsjavaproperties`** tool with an explicitly-defined path to the Java environment, like:

```
env UNBOUNDID_JAVA_HOME=/ds/java bin/dsjavaproperties
```

An Invalid Command-Line Option Was Provided

There are a small number of arguments that are provided when running the **`bin/start-ds`** command, but in most cases, none are required. If one or more command-line arguments were provided for the **`bin/start-ds`** command and any of them is not recognized, then the server provides an error message indicating that an argument was not recognized and displays version information. In that case, correct or remove the invalid argument and try to start the server again.

The Server Has an Invalid Configuration

If a change is made to the Directory Server configuration using an officially-supported tool like **dsconfig** or the directory management console, the server should validate that configuration change before applying it. However, it is possible that a configuration change can appear to be valid at the time that it is applied, but does not work as expected when the server is restarted. Alternately, a change in the underlying system can cause a previously-valid configuration to become invalid.

In most cases involving an invalid configuration, the Directory Server displays (and writes to the error log) a message that explains the problem, and this can be sufficient to identify the problem and understand what action needs to be taken to correct it. If for some reason the startup failure does not provide enough information to identify the problem with the configuration, then look in the **logs/config-audit.log** file to see what recent configuration changes have been made with the server online, or in the **config/archived-configs** directory to see if there might have been a recent configuration change resulting from a direct change to the configuration file itself that was not made through a supported configuration interface.

If the server does not start as a result of a recent invalid configuration change, then it can be possible to start the server using the configuration that was in place the last time that the server started successfully (for example, the "last known good" configuration). This can be achieved using the **--useLastKnownGoodConfig** option:

```
$ bin/start-ds --useLastKnownGoodConfig
```

Note that if it has been a long time since the last time the server was started and a number of configuration changes have been made since that time, then the last known good configuration can be significantly out of date. In such cases, it can be preferable to manually repair the configuration.

If there is no last known good configuration, if the server no longer starts with the last known good configuration, or if the last known good configuration is significantly out of date, then manually update the configuration by editing the **config/config.ldif** file. In that case, you should make sure that the server is offline and that you have made a copy of the existing configuration before beginning. You might wish to discuss the change with your authorized support representative before applying it to ensure that you understand the correct change that needs to be made.

Note In addition to manually-editing the config file, you can look at previous achived configurations to see if the most recent one works. You can also use the **ldif-diff** tool to compare the configurations in the archive to the current configuration to see what is different.

You Do Not Have Sufficient Permissions

The Directory Server should only be started by the user or role used to initially install the server. In most cases, if an attempt is made to start the server as a user or role other than the one used to create the initial configuration, then the server will fail to start, because the user will not have sufficient permissions to access files owned by the other user, such as database and log files. However, if the server was initially installed as a non-root user and then the server is started by the root account, then it can no longer be possible to start the server as a non-root user because new files that are created would be owned by root and could not be written by other users.

If the server was inadvertently started by root when it is intended to be run by a non-root user, or if you wish to change the user account that should be used to run the server, then it should be sufficient to simply change ownership on all files in the Directory Server installation, so that they are owned by the user or role under which the server should run. For example, if the Directory Server should be run as the "ds" user in the "other" group, then the following command can be used to accomplish this (invoked by the root user):

```
chown -R ds:other /ds/UnboundID-DS
```

The Server Has Crashed or Shut Itself Down

You can first check the current server state by using the **`bin/server-state`** command. If the Directory Server was previously running but is no longer active, then the potential reasons include the following:

- The Directory Server was shut down by an administrator. Unless the server was forcefully terminated (for example, using "**`kill -9`**"), then messages are written to the **`error`** and **`server.out`** logs explaining the reason for the shutdown.
- The Directory Server was shut down when the underlying system crashed or was rebooted. If this is the case, then running the **`uptime`** command on the underlying system shows that it was recently booted.
- The Directory Server process was terminated by the underlying operating system for some reason (for example, the out of memory killer on Linux). If this happens, then a message will be written to the system error log.
- The Directory Server decided to shut itself down in response to a serious problem that had arisen. At present, this should only occur if the server has detected that the amount of usable disk space has become critically low, or if significant errors have been encountered during processing that left the server without any remaining worker threads to process operations. If this happens, then messages are written to the **`error`** and **`server.out`** logs (if disk space is available) to provide the reason for the shutdown.
- The JVM in which the Directory Server was running crashed. If this happens, then the JVM should dump a fatal error log (a **`hs_err_pid{processID}.log`** file) and potentially a core file.

In the event that the operating system itself crashed or terminated the process, then you should work with your operating system vendor to diagnose the underlying problem. If the JVM crashed or the server shut itself down for a reason that is not clear, then contact your authorized support provider for further assistance.

The Server Will Not Accept Client Connections

You can first check the current server state by using the **`bin/server-state`** command. If the Directory Server does not appear to be accepting connections from clients, then potential reasons include the following:

- The Directory Server is not running.
- The Directory Server is in the process of starting up but has not yet begun accepting client connections. If the Directory Server is configured to automatically load database contents into memory on startup, then this can take a significant length of time (depending on the size of the database) although it writes messages to the error log as priming progresses. If the Directory Server was not gracefully shut down and the database needs to perform some form of recovery, then information about recovery progress should be written to the newest **`je.info.*`** file in the database directory.
- The underlying system on which the Directory Server is installed is not running.
- The Directory Server is running but is not reachable as a result of a network or firewall configuration problem. If that is the case, then connection attempts should time out rather than be rejected.
- If the Directory Server is configured to allow secure communication via SSL or StartTLS, then a problem with the key manager and/or trust manager configuration can cause connections to be rejected. If that is the case, then messages should be written to the server access log for each failed connection attempt.
- If the Directory Server has been configured with a maximum allowed number of connections, then it can be that the maximum number of allowed client connections are already established. If that is the case, then messages should be written to the server access log for each rejected connection attempt.
- If the Directory Server is configured to restrict access based on the address of the client, then messages should be written to the server access log for each rejected connection attempt.
- If a connection handler encounters a significant error, then it can stop listening for new requests. If this occurs, then a message should be written to the server error log with information about the problem. Another solution is to restart the server. A third option is to restart the connection handler using the LDIF connection handler to make it available again. To do this, create an LDIF file that disables and then re-enables the connection handler, create the **`config/auto-process-ldif`** directory if it does not already exist, and then copy the LDIF file into it.

The Server is Unresponsive

You can first check the current server state by using the **`bin/server-state`** command. If the Directory Server process is running and appears to be accepting connections but does not respond to requests received on those connections, then potential reasons for this behavior include:

- If all worker threads are busy processing other client requests, then new requests that arrive will be forced to wait in the work queue until a worker thread becomes available. If this is the case, then a stack trace obtained using the **`jstack`** command shows that all of the worker threads are busy and none of them are waiting for new requests to process.

Note If all of the worker threads are tied up processing the same operation for a long time, the server will also issue an alert that it might be deadlocked, which may not actually be the case. All threads might be tied up processing unindexed searches.

- If a request handler is stuck performing some expensive processing for a client connection, then other requests sent to the server on connections associated with that request handler is forced to wait until the request handler is able to read data on those connections. If this is the case, then only some of the connections can experience this behavior (unless there is only a single request handler, in which it will impact all connections), and stack traces obtained using the **`jstack`** command shows that a request handler thread is continuously blocked rather than waiting for new requests to arrive. Note that this scenario is a theoretical problem and one that has not appeared in production.

- If the JVM in which the Directory Server is running is not properly configured, then it can be forced to spend a significant length of time performing garbage collection, and in severe cases, could cause significant interruptions in the execution of Java code. In such cases, a stack trace obtained from a **`pstack`** of the native process should show that most threads are idle but at least one thread performing garbage collection is active. It is also likely that one or a small number of CPUs is 100% busy while all other CPUs are mostly idle. The server will also issue an alert after detecting a long JVM pause (due to garbage collection). The alert will include details of the pause.

- If the JVM in which the Directory Server is running has hung for some reason, then the **`pstack`** utility should show that one or more threads are blocked and unable to make progress. In such cases, the system CPUs should be mostly idle.

- If a network or firewall configuration problem arises, then attempts to communicate with the server cannot be received by the server. In that case, a network sniffer like **`snoop`** or **`tcpdump`** should show that packets sent to the system on which the Directory Server is running are not receiving TCP acknowledgement.

- If the system on which the Directory Server is running has become hung or lost power with a graceful shutdown, then the behavior is often similar to that of a network or firewall configuration problem.

If it appears that the problem is with the Directory Server software or the JVM in which it is running, then you need to work with your authorized support provider to fully diagnose the problem and determine the best course of action to correct it.

The Server is Slow to Respond to Client Requests

If the Directory Server is running and does respond to clients, but clients take a long time to receive responses, then the problem can be attributable to a number of potential problems. In these cases, use the Periodic Stats Logger, which is a valuable tool to get per-second monitoring information on the Directory Server. The Periodic Stats Logger can save the information in csv format for easy viewing in a spreadsheet. For more information, see “Profiling Server Performance Using the Periodic Stats Logger” on page 484. The potential problems that cause slow responses to client requests are as follows:

- The server is not optimally configured for the type of requests being processed, or clients are requesting inefficient operations. If this is the case, then the access log should show that operations are taking a long time to complete and they will likely be unindexed. In that case, updating the server configuration to better suit the requests, or altering the requests to make them more efficient, could help alleviate the problem. In this case, view the expensive operations access log in **`logs/expensive-ops`**, which by default logs operations that take longer than 1 second. You can also run the **`bin/status`** command or view the status in the web console to see the Directory Server’s Work Queue information (also see the next bullet point).

- The server is overwhelmed with client requests and has amassed a large backlog of requests in the work queue. This can be the result of a configuration problem (for example, too few worker thread configured), or it can be necessary to provision more systems on which to run the Directory Server software. Symptoms of this problem appear similar to those experienced when the server is asked to process inefficient requests, but looking at the details of the requests in the access log show that they are not necessarily inefficient requests. Run the **`bin/status`** command to view the Work Queue information. If everything is performing well, you should not see a large queue size or a server that is near 100% busy. The %Busy statistic is calculated as the percentage of worker threads that are busy processing operations.

  ```
            --- Work Queue ---
             : Recent : Average : Maximum
  -----------:--------:---------:--------
  Queue Size : 10     : 1       : 10
  % Busy     : 17     : 14      : 100
  ```

  You can also view the expensive operations access log in **`logs/expensive-ops`**, which by default logs operations that take longer than 1 second.

- The server is not configured to fully cache all of the data in the server, or the cache is not yet primed. In this case, **`iostat`** reports a very high disk utilization. This can be resolved by configuring the server to fully cache all data, and to load database contents into memory on startup. If the underlying system does not have enough memory to fully cache the entire data set, then it might not be possible to achieve optimal performance for operations that need data which is not contained in the cache. For more information, see “Tuning Disk-Bound Deployments” on page 84.

- If the JVM is not properly configured, then it will need to perform frequent garbage collection and periodically pause execution of the Java code that it is running. In that case, the server error log should report that the server has detected a number of pauses and can include tuning recommendations to help alleviate the problem.

- If the Directory Server is configured to use a large percentage of the memory in the system, then it is possible that the system has gotten low on available memory and has begun swapping. In this case, **`iostat`** should report very high utilization for disks used to hold swap space, and commands like **`swap -l`** on Solaris or **`cat /proc/meminfo`** on Linux can report a large amount of swap memory in use. Another cause of swapping is if **`swappiness`** is not set to 0 on Linux. For more information, see “Disable File System Swapping (Linux)” on page 12.

- If another process on the system is consuming a significant amount of CPU time, then it can adversely impact the ability of the Directory Server to process requests efficiently. Isolating the processes (for example, using processor sets) or separating them onto different systems can help eliminate this problem.

The Server Returns Error Responses to Client Requests

If a large number of client requests are receiving error responses, then view the **`logs/failed-ops`** log, which is an access log for only failed operations. The potential reasons for the error responses include the following:

- If clients are requesting operations that legitimately should fail (for example, they are targeting entries that do not exist, are attempting to update entries in a way that would violate the server schema, or are performing some other type of inappropriate operation), then the problem is likely with the client and not the server.

- If a portion of the Directory Server data is unavailable (for example, because an online LDIF import or restore is in progress), then operations targeting that data will fail. Those problems will be resolved when the backend containing that data is brought back online. During the outage, it might be desirable to update proxy servers or load balancers or both to route requests away from the affected server. As of Directory Server version 3.1 or later, the Directory Server will indicate that it is in a degraded status and the proxy server will route around it.

- If the Directory Server work queue is configured with a maximum capacity and that capacity has been reached, then the server begins rejecting all new requests until space is available in the work queue. In this case, it might be necessary to alter the server configuration or the client requests or both, so that they can be processed more efficiently, or it might be necessary to add additional server instances to handle some of the workload.

- If an internal error occurs within the server while processing a client request, then the server terminates the connection to the client and logs a message about the problem that occurred. This should not happen under normal circumstances, so you will need to work with your authorized support provider to diagnose and correct the problem.

- If a problem is encountered while interacting with the underlying database (for example, an attempt to read from or write to disk failed because of a disk problem or lack of available disk space), then it can begin returning errors for all attempts to interact with the database until the backend is closed and re-opened and the database has been given a change to recover itself. In these cases, the **`je.info.*`** file in the database directory should provide information about the nature of the problem.

The Server is experiencing problems with replication

If replication does not appear to be functioning properly, then first check the **`dsreplication status`** command, which shows all of the servers that are replicating and whether they are back-logged or not. Next, you can check the server error log, replication repair log, and replication monitor entries may provide information about the nature of the underlying problem. Potential reasons that replication may not be functioning as expected include the following:

- Replication has not yet been configured between systems or has been disabled.
- If a server has been offline for a period of time or has fallen far enough behind such that it is missing changes, which are no longer present in any of the replication databases, then that server must be re-initialized with an up-to-date copy of the data from another server.
- If the environment is comprised of a heterogeneous set of systems, then it is possible that some of the systems might not be able to keep up with the maximum throughput achieved by other servers in the topology. In such cases, the slower servers might not be fast enough to remain in sync with the other servers in the environment.
- If the environment contains systems in multiple data centers and the network links between the data centers are insufficient for the volume of changes that must be processed, then servers might not be able to remain in sync under a high volume of changes.
- A network or firewall configuration problem has arisen, which prevents or interferes with communication between servers.
- An internal problem within the server has caused replication to stop functioning properly. The Directory Server logs the event in the error log in this case. Run the **`collect-support-data`** tool, so that the details of the problems can be passed to Support. Then, try restarting the Directory Server.

The Server behaves differently from Oracle DSEE

After migrating from a Oracle Directory Server Enterprise Edition (DSEE) configuration to an UnboundID Directory Server, follow the tuning procedures in “Oracle DSEE Compatibility” on page 85 if the Directory Server behaves differently from the Oracle DSEE server.

Problems with the Directory Management Console

If a problem arises when trying to use the directory management console, then potential reasons for the problem may include the following:

- The web application container used to host the console is not running. If an error occurs while trying to start it, then consult the logs for the web application container.
- If a problem occurs while trying to authenticate to the web application container, then make sure that the target Directory Server is online. If it is online, then the access log may provide information about the reasons for the authentication failure.
- If a problem occurs while attempting to interact with a directory server instance using the Directory Management Console, then the access and error logs for that Directory Server instance might provide additional information about the underlying problem.
- **Console runs out of memory (PermGen)**. If you are running a Management Console for a Directory Server while running an UnboundID Directory Proxy Server Management Console and UnboundID Synchronization Server Management Console, you may see a Java PermGen error as follows:

  ```
  Exception in thread "http-bio-8080-exec-7" java.lang.OutOfMemoryError: PermGen
  Space
  ```

  For a servlet container, such as Tomcat, you can specify additional arguments to pass to the JVM by creating a `bin/setenv.sh` file (or `setenv.bat` for Windows) that sets the CATALINA_OPTS variable. The `startup.sh` script will automatically pick this up. For example:

  ```
  #!/bin/bash

  # The following may be modified to change JVM memory arguments.
  MAX_HEAP_SIZE=512m
  MIN_HEAP_SIZE=$MAX_HEAP_SIZE
  MAX_PERM_SIZE=256m

  CATALINA_OPTS="-Xmx${MAX_HEAP_SIZE} -Xms${MIN_HEAP_SIZE} -XX:MaxPerm-
  Size=${MAX_PERM_SIZE}"
  ```

Problems with the HTTP Connection Handler

When problems with the HTTP Connection Handler occur, first look at the HTTP connection handler log to diagnose the issue. The following section shows HTTP log examples when various errors occur.

- **Failed Request Due to a Non-Existent Resource**. The server receives a status code 404, which indicates the server could not match the URI.

  ```
  [15/Mar/2012:17:39:39 -0500] RESULT requestID=0 from="10.2.1.113:52958"
  method="GET" url="https://10.2.1.113:443/Aleph/Users/uid=user.1,ou=people,dc=exam-
  ple,dc=com" requestHeader="Host: x2270-11.example.lab" requestHeader="Accept: */*"
  requestHeader="User-Agent: curl/7.21.6 (i386-pc-solaris2.10) libcurl/7.21.6
  OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7" authorizationType="Basic" sta-
  ```

```
tusCode=404 etime=81.484 responseContentLength=103 responseHeader="Access-Control-
Allow-Credentials: true" responseContentType="application/json"
```

- **Failed Request due to a Malformed Request Body**. The server receives a status code 400, which indicates that the request had a malformed syntax in its body.

```
[15/Mar/2012:17:47:23 -0500] RESULT requestID=10 from="10.2.1.113:55284"
method="POST" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host: x2270-
11.example.lab" requestHeader="Expect: 100-continue" requestHeader="Accept: */*"
requestHeader="Content-Type: application/json" requestHeader="User-Agent: curl/
7.21.6 (i386-pc-solaris2.10) libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22
libssh2/1.2.7" authorizationType="Basic" requestContentType="application/json"
requestContentLength=5564 statusCode=400 etime=15.272 responseContentLength=133
responseContentType="application/json"
```

- **Failed Request due to an Unsupported HTTP Method**. The server receives a status code 405, which indicates that the specified method (e.g., "PATCH") in the request line is not allowed for the resource identified in the URI.

```
[15/Mar/2012:17:48:59 -0500] RESULT requestID=11 from="10.2.1.113:55763"
method="PATCH" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host: x2270-
11.example.lab" requestHeader="Accept: */*" requestHeader="Content-Type: applica-
tion/json" requestHeader="User-Agent: curl/7.21.6 (i386-pc-solaris2.10) libcurl/
7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7" authorization-
Type="Basic" requestContentType="application/json" statusCode=405 etime=6.807
responseContentLength=0 responseHeader="Allow: POST, GET, OPTIONS, HEAD"
```

- **Failed Request due to an Unsupported Media Type**. The server receives a status code 415, which indicates that the request entity is in a format that is not supported by the requested resource.

```
[15/Mar/2012:17:44:45 -0500] RESULT requestID=4 from="10.2.1.113:54493"
method="POST" url="https://10.2.1.113:443/Aleph/Users" requestHeader="Host: x2270-
11.example.lab" requestHeader="Accept: */*" requestHeader="Content-Type: applica-
tion/atom+xml" requestHeader="User-Agent: curl/7.21.6 (i386-pc-solaris2.10) lib-
curl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/1.2.7"
authorizationType="Basic" requestContentType="application/atom+xml" requestContent-
Length=3 statusCode=415 etime=6.222 responseContentLength=1402 responseH-
eader="Cache-Control: must-revalidate,no-cache,no-store" responseContentType="text/
html;charset=ISO-8859-1"
```

- **Failed Request due to an Authentication Error**. The server receives a status code 401, which indicates that the request requires user authentication.

```
[15/Mar/2012:17:46:06 -0500] RESULT requestID=8 from="10.2.1.113:54899"
method="GET" url="https://10.2.1.113:443/Aleph/Schemas" requestHeader="Host: x2270-
11.example.lab" requestHeader="Accept: */*" requestHeader="User-Agent: curl/7.21.6
(i386-pc-solaris2.10) libcurl/7.21.6 OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22 libssh2/
1.2.7" authorizationType="Basic" statusCode=401 etime=2.751 responseContent-
Length=63 responseHeader="WWW-Authenticate: Basic realm=SCIM" responseH-
eader="Access-Control-Allow-Credentials: true" responseContentType="application/
json"
```

Providing Information for Support Cases

If a problem arises that you are unable to fully diagnose and correct on your own, then contact your authorized support provider for assistance. In order to ensure that the problem can be addressed as quickly as possible, you should be sure to provide all of the information that the support personnel may need to fully understand the underlying cause by running the **`collect-support-data`** tool, and then sending the generated zip file to your authorized support provider. It is good practice to run this tool and send the ZIP file to your authorized support provider before any corrective action has taken place.

16 Command-Line Tools

Overview

The UnboundID Directory Server provides a full suite of command-line tools necessary to administer the server. The command-line tools are available in the **`bin`** directory for UNIX or Linux systems and **`bat`** directory for Microsoft Windows systems.

This chapter presents the following topics:

Using the Help Option

The following table provides a brief description of each command-line tool. You can view detailed argument options and examples by typing **`--help`** with the command.

```
bin/dsconfig --help
```

For those utilities that support additional subcommands (**`dsconfig`**, **`dsframework`**, **`dsreplication`**, **`manage-account`**), you can get a list of the subcommands by typing **`--help-subcommands`**.

```
bin/dsconfig --help-subcommands
```

You can also get more detailed subcommand information by typing **`--help`** with the specific subcommand.

```
bin/dsconfig list-log-publishers --help
```

Note	For detailed information and examples of the command-line tools, see the *UnboundID Directory Server Command-Line Tool Reference.*

Available Command-Line Utilities

The Directory Server provides the following command-line utilities, which can be run directly in interactive or non-interactive modes or can be included in scripts.

TABLE 16-1. Command-Line Utilities

Command-Line Tool	Description
authrate	Perform repeated authentications against an LDAP directory server, where each authentication consists of a search to find a user followed by a bind to verify the credentials for that user.
backup	Run full or incremental backups on one or more directory server backends. This utility also supports the use of a properties file to pass predefined command-line arguments. See “Managing the Tools Properties File” on page 550 for more information.
base64	Encode raw data using the base64 algorithm or decode base64-encoded data back to its raw representation.
collect-support-data	Collect and package system information useful in troubleshooting problems. The information is packaged as a ZIP archive that can be sent to a technical support representative.
create-rc-script	Create an Run Control (RC) script that may be used to start, stop, and restart the Directory Server on UNIX-based systems.
dbtest	Inspect the contents of Directory Server backends that store their information in Oracle® Berkeley DB Java Edition databases.
dsconfig	View and edit the Directory Server configuration.
dsframework	Manage administrative server groups or the global administrative user accounts that are used to configure servers within server groups.
dsjavaproperties	Configure the JVM™ arguments used to run the Directory Server and associated tools. Before launching the command, edit the properties file located in `config/java.properties` to specify the desired JVM options and `JAVA_HOME`.
dsreplication	Manage data replication between two or more Directory Server instances.
dump-dns	Obtain a listing of all of the DNs for all entries below a specified base DN in the Directory Server.
encode-password	Encode user passwords with a specified storage scheme or determine whether a given clear-text value matches a provided encoded password.
encryption-settings	Manage the server encryption settings database.
enter-lockdown-mode	Request that the Directory Server enter lockdown mode, during which it only processes operations requested by users holding the `lockdown-mode` privilege.
export-ldif	Export data from a Directory Server backend in LDIF form.
import-ldif	Import LDIF data into a Directory Server backend.
ldap-diff	Compare the contents of two LDAP servers.
ldap-result-code	Display and query LDAP result codes.
ldapcompare	Perform LDAP compare operations in the Directory Server.
ldapdelete	Perform LDAP delete operations in the Directory Server.

TABLE 16-1. **Command-Line Utilities**

Command-Line Tool	Description
ldapmodify	Perform LDAP modify, add, delete, and modify DN operations in the Directory Server.
ldappasswordmodify	Perform LDAP password modify operations in the Directory Server.
ldapsearch	Perform LDAP search operations in the Directory Server.
ldif-diff	Compare the contents of two LDIF files, the output being an LDIF file needed to bring the source file in sync with the target.
ldifmodify	Apply a set of modify, add, and delete operations against data in an LDIF file.
ldifsearch	Perform search operations against data in an LDIF file.
leave-lockdown-mode	Request that the Directory Server leave lockdown mode and resume normal operation.
list-backends	List the backends and base DNs configured in the Directory Server.
make-ldif	Generate LDIF data based on a definition in a template file.
manage-account	Access and alter password policy state properties for user entries.
manage-tasks	Access information about pending, running, and completed tasks scheduled in the Directory Server.
migrate-ldap-schema	Migrate schema information from an existing LDAP server into an UnboundID Directory Server instance.
migrate-sun-ds-config	Update an instance of the UnboundID Directory Server to match the configuration of an existing Sun Java System 5.x, 6.x, or 7.x directory instance.
modrate	Perform repeated modifications against an LDAP directory server.
move-entry	Move one or more LDAP entries from one server to another.
parallel-update	Perform add, delete, modify, and modify DN operations concurrently using multiple threads.
profile-viewer	View information in data files captured by the Directory Server profiler.
rebuild-index	Rebuild index data within a backend based on the Berkeley DB Java Edition. Note that this tool uses different approaches to rebuilding indexes based on whether it is running in online mode (as a task) rather than in offline mode. Running in offline mode will often provide significantly better performance. Also note that running in online mode will prevent the server from using that index while the rebuild is in progress, so some searches may behave differently while a rebuild is active than when it is not.
remove-backup	Safely remove a backup and optionally all of its dependent backups from the specified directory server backend.
restore	Restore a backup of a Directory Server backend.
revert-update	Returns a server to the version before the last update was performed. Unlike the `update` tool, this tool operates on the local instance from which it is invoked. This tool relies on files from the history directory that are created during an update to restore the server to a prior state. It should be noted that this tool does not revert database files to prior states. Therefore, any changes made to the directory data between the time of the update and time of reversion will be lost.
review-license	Review and/or indicate your acceptance of the product license.

TABLE 16-1. **Command-Line Utilities**

Command-Line Tool	Description
scramble-ldif	Obscure the contents of a specified set of attributes in an LDIF file.
search-and-mod-rate	Perform repeated searches against an LDAP directory server and modify each entry returned.
searchrate	Perform repeated searches against an LDAP directory server.
server-state	View information about the current state of the Directory Server process.
setup	Perform the initial setup for a directory server instance.
start-ds	Start the Directory server.
status	Display basic server information.
stop-ds	Stop or restart the Directory Server.
sum-file-sizes	Calculate the sum of the sizes for a set of files.
summarize-access-log	Generate a summary of one or more access logs to display a number of metrics about operations processed within the server.
summarize-config	Generate a configuration summary of either a remote or local Directory Server instance. By default, only basic components and properties will be included. To include advanced components, use the `--advanced` option.
uninstall	Uninstall the Directory Server.
update	Update the Directory Server to a newer version by downloading and unzipping the new server install package on the same host as the server you wish to update. Then, use the `update` tool from the new server package to update the older version of the server. Before upgrading a server, you should ensure that it is capable of starting without severe or fatal errors. During the update process, the server is stopped if running, then the update performed, and a check is made to determine if the newly updated server starts without major errors. If it cannot start cleanly, the update will be backed out and the server returned to its prior state. See the `revert-update` tool for information on reverting an update.
validate-acis	Validates a set of access control definitions contained in an LDAP server (including Oracle DSEE instances) or an LDIF file to determine whether they are acceptable for use in the UnboundID Directory Server. Note that the output generated by this tool will be LDIF, but each entry in the output will have exactly one ACI, so entries which have more than one ACI will appear multiple times in the output with different ACI values.
validate-ldif	Validate the contents of an LDIF file against the server schema.
verify-index	Verify that indexes in a backend using the Oracle Berkeley DB Java Edition are consistent with the entry data contained in the database.

Managing the Tools Properties File

The UnboundID Directory Server supports the use of a tools properties file that simplifies command-line invocations by reading in a set of arguments for each tool from a text file. Each property is in the form of name/value pairs that define predetermined values for a tool's arguments. Properties files are convenient when quickly testing the Directory Server in multiple environments.

The Directory Server supports two types of properties file: default properties files that can be applied to all command-line utilities or tool-specific properties file that can be specified using the `--propertiesFilePath` option with the following tools:

backup	ldapcompare	restore
dsconfig	ldapdelete	setup
dsframework	ldapmodify	status
dsreplication	ldappasswordmodify	stop-ds
export-ldif	ldapsearch	uninstall
import-ldif	manage-tasks	

Creating a Properties File

You can create a properties file with a text editor by specifying each argument, or option, using standard Java properties file format (name=value). For example, you can create a simple properties file that define a set of LDAP connection parameters as follows:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
baseDN=dc=example,dc=com
```

Next, you can specify the location of the file using the `--propertiesFilePath /path/to/File` option with the command-line tool. For example, if you save the previous properties file as `bin/mytool.properties`, you can specify the path to the properties file with `ldapsearch` as follows:

```
$ bin/ldapsearch --propertiesFilePath bin/mytools.properties "(objectclass=*)"
```

Properties files do not allow quotation marks of any kind around values. Any spaces or special characters should be escaped. For example,

```
bindDN=cn=QA\ Managers,ou=groups,dc=example,dc=com
```

The following is not allowed as it contains quotation marks:

```
bindDN=cn="QA Managers,ou=groups,dc=example,dc=com"
```

Tool-Specific Properties

The Directory Server also supports properties for specific tool options using the format: `tool.option=value`. Tool-specific options have precedence over general options. For example, the following properties file uses `ldapsearch.port=2389` for `ldapsearch` requests by the client. All other tools that use the properties file uses `port=1389`.

```
hostname=server1.example.com
port=1389
ldapsearch.port=2389
bindDN=cn=Directory\ Manager
```

Specifying Default Properties Files

The Directory Server provides a default properties files that apply to all command-line utilities used in client requests. A default properties file, **`tools.properties`**, is located in the **`<server-root>config`** directory.

If you place a custom properties file that has a different filename as **`tools.properties`** in this default location, you need to specify the path using the **`--propertiesFilePath`** option. If you make changes to the tools.properties file, you do not need the **`--propertiesFilePath`** option. See the examples in the next section.

Evaluation Order Summary

The Directory Server uses the following evaluation ordering to determine options for a given command-line utility:

- All options used with a utility on the command line takes precedence over any options in any properties file.
- If the **`--propertiesFilePath`** option is used with no other options, the Directory Server takes its options from the specified properties file.
- If no options are used on the command line including the **`--propertiesFilePath`** option (and **`--noPropertiesFile`**), the Directory Server searches for the **`tools.properties`** file at **`<directory-instance>/config/`**.
- If no default properties file is found, the Directory Server generates an error.
- Tool-specific properties (for example, **`ldapsearch.port=3389`**) have precedence over general properties (for example, **`port=1389`**).

Evaluation Order Example

Given the following properties file that is saved as **`<server-root>/bin/tools.properties`**:

```
hostname=server1.example.com
port=1389
bindDN=cn=Directory\ Manager
bindPassword=secret
baseDN=dc=example,dc=com
```

The Directory Server locates a command-line options in a specific priority order.

1. All options presented with the tool on the command line take precedence over any options in any properties file. In the following example, the client request is run with the options specified on the command line (port and baseDN). The command uses the **`bindDN`** and **`bindPassword`** arguments specified in the properties file.

```
$ bin/ldapsearch --port 2389 --baseDN ou=People,dc=example,dc=com \
  --propertiesFilePath bin/tools.properties "(objectclass=*)"
```

2. Next, if you specify the properties file using the **`--propertiesFilePath`** option and no other command-line options, the Directory Server uses the specified properties file as follows:

   ```
   $ bin/ldapsearch --propertiesFilePath bin/tools.properties \
     "(objectclass=*)"
   ```

3. If no options are presented with the tool on the command line and the **`--noPropertiesFile`** option is not present, the Directory Server attempts to locate any default **`tools.properties`** file in the following location:

   ```
   <directory-instance>/config/tools.properties
   ```

   Assume that you move your **`tools.properties`** file from **`<directory-instance>/bin`** to the **`<directory-instance>/config`** directory. You can then run your tools as follows:

   ```
   $ bin/ldapsearch "(objectclass=*)"
   ```

   The Directory Server can be configured so that it does not search for any properties file by using the **`--noPropertiesFile`** option. This options tells the Directory Server to use only those options specified on the command line. The **`--propertiesFilePath`** and **`--noPropertiesFile`** options are mutually exclusive and cannot be used together.

4. If no default **`tools.properties`** file is found and no options are specified with the command-line tool, then the tool generates an error for any missing arguments.

Index