

UnboundID[®] Directory Server

Product Description

Version 3.2

September 21, 2011



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

512-600-7700
www.UnboundID.com

Copyright

Copyright ©2011 UnboundID Corp.
All rights reserved.

This document constitutes an unpublished, copyrighted work and contains valuable trade secrets and other confidential information belonging to UnboundID® Corp. None of the foregoing material may be copied, duplicated or disclosed to third parties without the express written permission of UnboundID Corp.

“UnboundID” is a registered trademark of UnboundID Corp. UNIX® is a registered trademark in the United States and other countries, licensed exclusively through The Open Group. All other registered and unregistered trademarks in this document are the sole property of their respective owners.

The contents of this publication are presented for information purposes only and are provided “as is.” While every effort has been made to ensure the accuracy of the contents, the contents are not to be construed as warranties or guarantees, expressed or implied, regarding the products or services described herein or their use or applicability. We reserve the right to modify or improve the design or specifications of such products at any time without notice.

UnboundID® Directory Server

Product Description

Introduction

Today, data centers are under increased pressure to manage rising technology costs while maximizing processing efficiencies and meeting aggressive performance objectives. For example, the explosive growth of mobile smartphones and web applications coupled with the rising bandwidth requirements necessary to process large numbers of transactions for billions of entries are stressing data centers. Many corporations are leveraging virtualization technologies, adopting energy-saving green strategies, and automating security, risk, and compliance processes as user activity and enterprise mobility systems increase. Your Identity Management system must also satisfy these requirements.

To meet these challenges, UnboundID® Corp has developed a next-generation, feature-rich, and blazingly-fast directory server that provides excellent performance (high throughput and low latency) while providing a robust, highly scalable and extensible directory services solution.

This document presents the features and benefits of the UnboundID Directory Server, which significantly lowers your Total Cost of Ownership (TCO) while providing vastly increased performance for lower equipment and storage costs.

Corporations are consolidating their data center operations through reductions or mergers yet must plan for user account growth and rising web application usage. Your directory services must be able to meet these demands.

The staff at UnboundID Corp. has a long track record of successful LDAP technology deployments.

The UnboundID Product Family

The UnboundID product family integrates disparate systems to provide a robust and high performance directory services solution. The **UnboundID® Directory Server**, together with its performance and scalability, offers all of the standard traditional LDAPv3 server functions, plus powerful Relational Database Management System (RDBMS) features, such as transactions, joins, and event notifications. The **UnboundID® Directory Proxy Server** is a fast, scalable load-balancing server that seamlessly distributes client requests to the backend servers. The **UnboundID® Synchronization Server** provides a point-to-point, directory-centric solution for fast, low-latency data synchronization between different directory systems and platforms. The **UnboundID® LDAP SDK for Java** is a feature-rich, fast, and user-friendly API for client communications with LDAPv3 directory servers.

For more information about the UnboundID offerings, go to the UnboundID web site at:

www.unboundid.com

The UnboundID Directory Server is the centerpiece of the UnboundID family of Directory Services products that include the UnboundID Directory Proxy Server, the UnboundID Synchronization Server, and the UnboundID LDAP SDK for Java.

The UnboundID products are currently being deployed in production systems around the world.

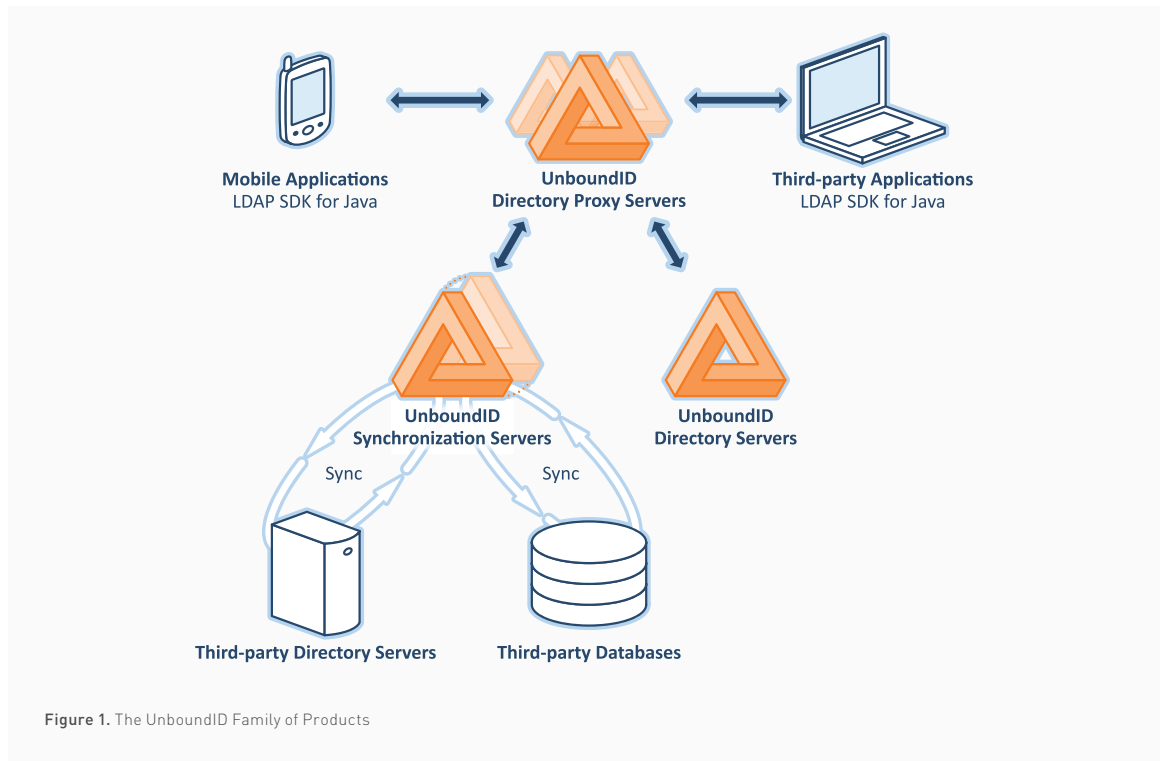


Figure 1. The UnboundID Family of Products

What is the UnboundID Directory Server?

The UnboundID Directory Server (DS) is a state-of-the-art, high performance, LDAP directory server for high-scale production environments. This next-generation directory server centrally stores and maintains user, network device, security, and application data.

The UnboundID Directory Server is a fully LDAPv3-compliant directory server designed for high performance concurrent processing. The Directory Server features fast response times, fast read and write under heavy loads, efficient memory management, highly available N-way multi-master replication, wide security and data protection features, high scalability, and broad extensibility. The Directory Server provides a web management console and a rich set of command-line tools to manage and monitor the server. The logging system provides a large variety of standard and customized features, such as, custom access logs, filtered log publishers, and admin alert logs that automatically trigger notifications when a certain access message appears. The Directory Server also supports unique transactional capabilities that were only available on Relational Database Management System (RDBMS) systems, such as the ability to process multiple reads and writes as a single transaction.

The UnboundID Directory Server is a 100% Java™-based LDAPv3-compliant server for high performance, portability, extensibility, and scalability.

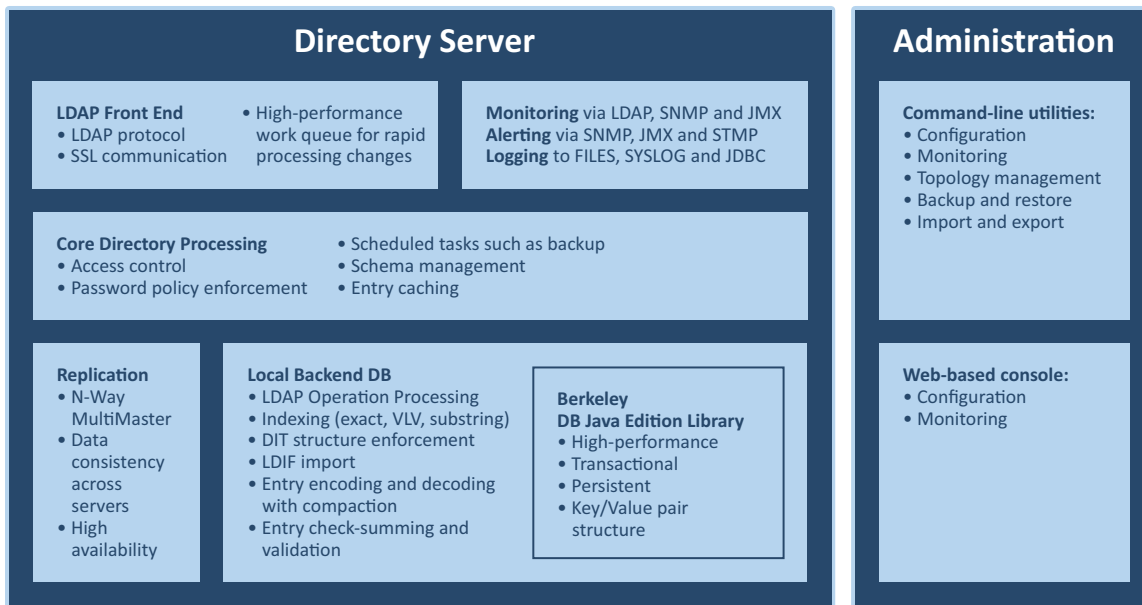


Figure 2. The UnboundID Directory Server

UnboundID Directory Server Capabilities At-A-Glance

TOTAL COST OF OWNERSHIP	
Platform Support <ul style="list-style-type: none"> • Runs on any Java™ 6 compliant Virtual Machine (JVM™) • VMware Certified • Designed for platform independence 	Benefits <ul style="list-style-type: none"> • Enables maximum utilization of existing hardware. • Supports deployment across virtualized environments
HIGH PERFORMANCE THROUGHPUT	
Performance <ul style="list-style-type: none"> • More read and write transactions per second than competing solutions • Sub-millisecond response times • Fast cache loading of data at startup for superior out-of-the-blocks performance 	Benefits <ul style="list-style-type: none"> • Better response time allows the use of real-time information, e.g., location and network data • Meets the demanding requirements of SLAs to support customer satisfaction and retention • Improves availability with faster startup, backup and recovery
Architecture <ul style="list-style-type: none"> • Provides the ability to customize the implementation by means of customized plug-ins and extended operations. • Based on a Concurrent Processing Model and an entry-level locking system for efficient updates • Uses Oracle® Berkeley DB Java Edition 	Benefits <ul style="list-style-type: none"> • Supports small to large networks at a fraction of the cost of traditional solutions, saves equipment costs as requirements increase • Limits impact on other updates in progress compared to page-level locking systems • Provides a log structured filesystem, which is more efficient than monolithic database files
Small Memory Footprint <ul style="list-style-type: none"> • Innovative memory storage capabilities: both in-memory and on-disk-requires 40-80% less space than competing solutions 	Benefits <ul style="list-style-type: none"> • Reduces data center cost and environmental impact by requiring fewer machines
REPLICATION, DATA INTEGRITY & SECURITY	
Replication <ul style="list-style-type: none"> • Provides robust N-way multi-master replication • Replication servers can be grouped by location • Immediate notifications are generated when data conflicts or errors occur 	Benefits <ul style="list-style-type: none"> • Ensures high availability in the event a server goes down to failure or maintenance • Easy management and monitoring for local or multi-site environments • Easy monitoring and management
Data Integrity <ul style="list-style-type: none"> • Uses entry digests to detect corruption of data • Provides data validation for backups before restoration 	Benefits <ul style="list-style-type: none"> • Has minimal impact on service performance during detection and validation • Enables real-time data integrity auditing • Improves accountability
Security <ul style="list-style-type: none"> • Uses robust and extensible security model fully compliant with LDAP security standards • Improved SSL, startTLS, message digests, privileges, access control, password policy, SASL mechanism support • Provides the ability to encrypt data-at-rest down to the entry level • Allows the assignment of specific levels of security to sensitive attributes 	Benefits <ul style="list-style-type: none"> • Ensures customer, employee, company information is secure • Supports various authentication and authorization means to access the directory server • Ensures that all data-at-rest is secured and protected against data leakage • Assign specific sensitive attribute definitions to information accessed or replicated between servers ensuring that data is protected from unauthorized individuals eavesdropping on network communication
FLEXIBILITY	
Flexibility <ul style="list-style-type: none"> • Extends the capabilities of the core server to support complex environments 	Benefits <ul style="list-style-type: none"> • Ensures that the solution is capable of adapting to changing business requirements

UnboundID Directory Server Capabilities At-A-Glance (cont.)

PRODUCTIVITY & OPERATIONS	
Transactions <ul style="list-style-type: none"> • Can perform multiple read or write operations in a single transaction • Provides batched or interactive transactions 	Benefits <ul style="list-style-type: none"> • Ensures no interruptions from other connections • Enables SQL-like BEGIN/COMMIT/ROLLBACK functionality
Backup & Restore <ul style="list-style-type: none"> • Supports full and incremental backups • Supports compressed, encrypted, and hashed backups • Supports verifying the integrity of backups without restore • Can schedule backups or restores 	Benefits <ul style="list-style-type: none"> • Incremental backups ensure efficient storage capabilities with lower total storage requirements • Encryption and hashing secures backups. Compression lowers storage requirements for the backups
Import & Export <ul style="list-style-type: none"> • Dramatically improved LDIF import performance • Supports compressed, encrypted, and hashed LDIF files • Supports filtered entries or attributes for imports or exports 	Benefits <ul style="list-style-type: none"> • Faster means to import your data • Secures imports and exports. Compression lowers LDIF file storage requirements • Provides an efficient means to import or export only those entries or attributes needed to different systems
INSTRUMENTATION & MANAGEMENT	
Change Notification <ul style="list-style-type: none"> • Enables informational and critical notification to an event using open APIs 	Benefits <ul style="list-style-type: none"> • Ensures complete visibility and control over server operations and instrumentation
Server Configuration <ul style="list-style-type: none"> • Supports command-line tools and a web console • Records every server configuration change, user who made it and how to reverse it 	Benefits <ul style="list-style-type: none"> • Ensures simple way to access the configuration. • Ensures security and accountability with the ability to playback or undo configurations
Logging <ul style="list-style-type: none"> • Employs an extensive set of logging capabilities for access, error, and debug: admin alerts, filtered logging, customized logging, JDBC™ support, syslog support 	Benefits <ul style="list-style-type: none"> • Ensures fast and effective troubleshooting to meet the needs of any data center
Administration Alerts <ul style="list-style-type: none"> • Supports an alert framework to notify administrators about significant events via JMX™, SNMP or SMTP • Provides an API to define custom alert handlers 	Benefits <ul style="list-style-type: none"> • Ensures fast and effective management in the event of a problem or event • Custom alerts can be tailored to specific production environments for more efficient management
Real-Time Monitoring <ul style="list-style-type: none"> • All monitor entries are consolidated in a single location • Exposes all monitor information via JMX, LDAP, or the console • Provides a monitoring API for customization 	Benefits <ul style="list-style-type: none"> • Easier to locate monitoring information • Provides a more comprehensive monitoring capability • Allows custom components to expose their own monitoring information

Total Cost of Ownership

The UnboundID Directory Server's broad portability, high performance (high throughput and low latency), low memory footprint, and rich feature set provides an efficient and effective directory services solution for mission-critical environments that lowers the total cost of ownership (TCO) over other directory vendors' solutions. Increased performance for a decreased TCO makes the UnboundID Directory Server an ideal solution for your Identity Management objectives.

The UnboundID Directory Server lowers the TCO, delivering up to 500% more performance with only 20% of the equipment and storage.

Platform Support

The UnboundID Directory Server is a pure Java application, fully portable to any platform that has a Java runtime, including the following:

- Solaris™ (x86, x64, and SPARC systems)
- Linux® (x86, x64, Itanium, Power, and S/390 systems)
- Microsoft® Windows® (x86, x64, and Itanium systems)
- AIX® (Power systems)
- HP-UX® (PA-RISC and Itanium systems)
- Mac OS X® (x64 systems)
- Azul® Compute Appliance
- VMware-Certified

Because the UnboundID Directory Server is compiled to Java bytecode rather than machine or OS-specific code, only a single version of the product is necessary to work on all platforms. For products compiled to native binaries, vendors are required to limit which platforms they can support and provide a separate set of binaries for each platform, and even separate binaries for 32-bit and 64-bit architectures on the same platform. The UnboundID Directory Server allows administrators to easily manage a single build in a broad platform environment. In most cases, installing a new Directory Server instance in an existing environment can be as easy as copying the contents of an instance that has already been configured as desired, even if the new installation is on a system with a different OS or CPU architecture.

The UnboundID Directory Server is distributed in single zip file for quick and easy installation.

Our design also supports portability between different operating systems. Other directory server vendors use OS-specific database formats that lock the user into a specific platform. UnboundID's approach ensures that customers can move a directory server instance between machines without having to re-import data. This approach allows customers to migrate to faster, less expensive systems without the worry of compatibility or platform-support issues, and thus, reducing the total cost of ownership.

The UnboundID Directory Server is fully portable across different operating systems and CPU architectures and does not require platform-specific releases.

High Performance Throughput

UnboundID's main objective has been to design and to produce a robust directory server with industry-leading performance and low operation response times (latency). We are constantly investigating ways to increase performance and lower response times.

The Directory Server has superior sub-millisecond response times for read and write operations.

Architecture

The UnboundID Directory Server's architecture ensures a high degree of performance throughput using concurrent processing, entry encoding compaction, and optimized JVM configuration.

- **Concurrent Processing Model:** The UnboundID Directory Server is based on a concurrent processing model that optimizes the performance of powerful multi-core, multiprocessor systems. The Directory Server avoids expensive locking whenever possible. Many directory servers from other vendors implement locking over a wide scope, such as locking the entire backend to ensure that only one write may be in progress anywhere in that backend at a time. When locking is required, the Directory Server uses a fine-grained locking system to limit the impact on other concurrent operations that may be in progress. During database updates, the Directory Server uses entry-level locking to limit the impact on other updates in progress.
- **Optimized JVM Configuration:** While other vendors' products require extensive manual tuning after installation, the UnboundID Directory Server automatically tunes its JVM configuration settings and database cache for optimal performance on the machine based on system characteristics, such as memory size and the number of CPUs.

The UnboundID Directory Server was built from the ground up after carefully analyzing the requirements in today's market versus current directory server offerings from other vendors.

The out-of-the-box configuration settings are highly optimized for performance and throughput, and require little manual tuning.

The Directory Server also dynamically determines the number of worker threads, the number of LDAP request handlers, the number of database lock tables at startup based on the number of CPUs on the system. Using these system characteristics, the Directory Server provides a high level of performance for its out-of-box configuration.

- **Entry Encoding Compaction:** The UnboundID Directory Server optimizes performance and scalability by encoding entries for database storage. The Directory Server compacts each entry encoding to allow more information for a given amount of memory. UnboundID has designed an efficient entry decoding process to eliminate the need for an entry cache. The efficiency of the entry decoding process allows more dedicated memory to database caching than for other operations. Entry encoding compaction and efficient entry decoding help increase Directory Server throughput and lower latency. Other vendors' products typically require both entry caching and database caching for optimal performance, which wastes memory resources and are less efficient.
- **Directory Data Priming:** The UnboundID Directory Server can automatically load (prime) directory data into memory at startup before accepting any client requests. The Directory Server's priming mechanism is fast and greatly improves performance by avoiding I/O accesses for even the very first client request.

The Directory Server stores entries using a compact binary representation, which allows tokenized object class sets, attribute descriptions, and entry DNs to further reduce the memory size for entry encoding. Other directory vendors store entries as LDIF representations, which require a relatively large on-disk memory footprint and is difficult to parse.

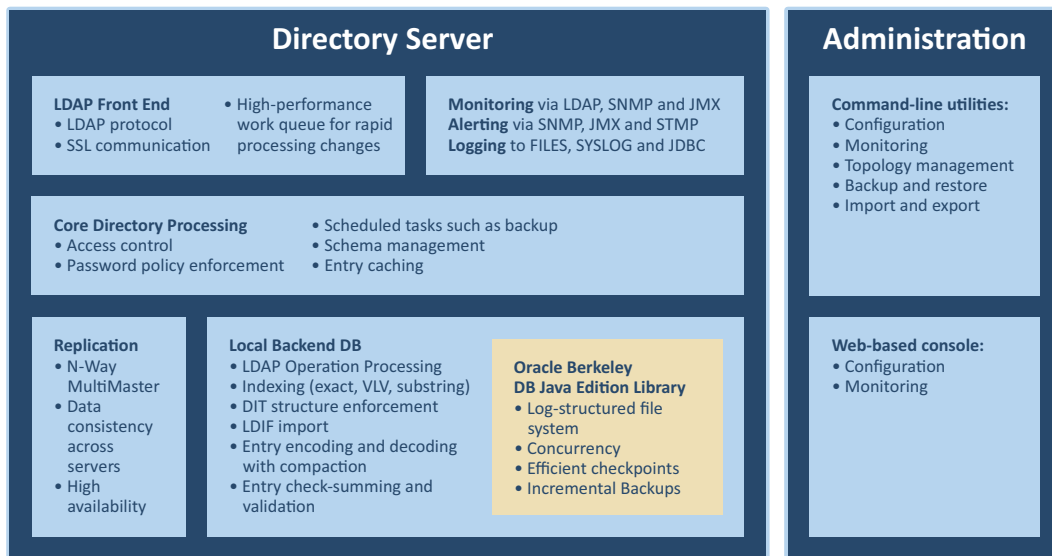


Figure 3. The UnboundID Directory Server Architecture

Oracle® Berkeley DB Java Edition

The UnboundID Directory Server uses the Oracle Berkeley DB Java Edition (JE) as its primary data store repository. The Oracle Berkeley DB Java Edition is a high performance, highly scalable, embedded transactional database with full support for Atomicity, Consistency, Isolation, and Durability (ACID) semantics, and characterized by key-value pairs for storing user data objects in the Directory Server. Oracle Berkeley DB Java Edition offers a number of advantages over other types of databases (for example, native Berkeley DB), including:

- **Log-Structured Filesystem:** JE uses a log-structured filesystem, which writes its data (Java objects) sequentially to log files and does not use monolithic database files. A log-structured filesystem eliminates the need to tune page sizes, which is normally required in RDBMS fixed-page database systems and provides better support for maintaining compact databases. JE only performs sequential writes and runs an automatic cleaning mechanism to ensure the database remains compact. JE can run in 32-bit mode or 64-bit mode without any differences.
- **Concurrency:** JE uses entry-level locking, which provides a high level of concurrency in terms of database accessibility. Other directory vendor's solutions use page-level locking, which locks multiple entries even if only one entry is being updated. Page-level locking can delay or prevent access to other entries on the same page.
- **Efficient Checkpoints:** Checkpoints in the Oracle Berkeley DB Java Edition have less work to do after a failover than in other types of databases and are more efficient. Other directory server vendors that use a page-sized database must use monolithic database files with separate transaction logs. In these systems, checkpoints are expensive due to the fact that data must be moved from transaction logs into the main database files.

Directory vendors that use page-based memory storage databases waste space (both on-disk and in-memory cache) as there will be space at the end of a page that isn't big enough to hold everything. The UnboundID Directory Server uses just enough space as is required.

The UnboundID Directory Server uses entry-level locking, which allows multiple concurrent writes. Other vendors that use monolithic databases use backend-wide write locks, which allow only one write operation at a time.

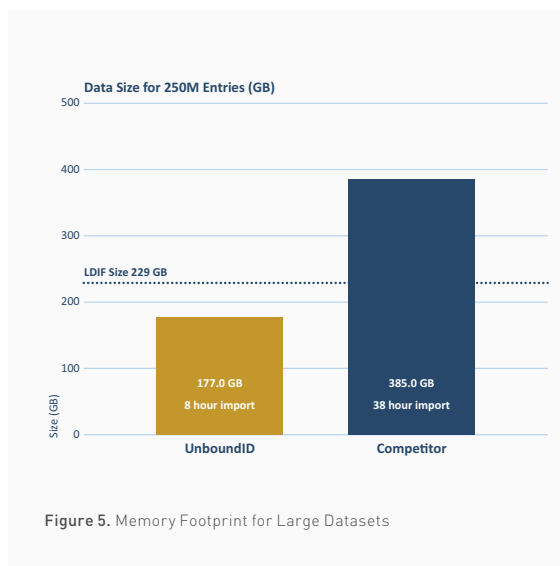
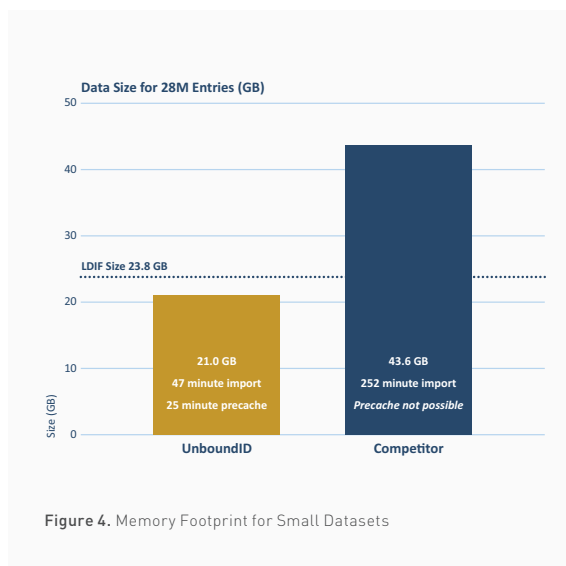
- Incremental Backups:** Because JE uses a log-structured filesystem, it allows the UnboundID Directory Server to use a separate database environment for each backend. This feature, in turn, supports an efficient incremental backup mechanism in addition to full backup capabilities for each backend. Administrators can backup and restore each backend individually and configure the amount of memory for caching on each backend. Other directory vendors that use the native Berkeley DB use monolithic database files and a separate transaction log file that prevents the use of incremental backups.

Small Memory Footprint

To achieve the maximum performance and throughput, UnboundID recommends that the Directory Server have as small an on-disk footprint as possible by taking full advantage of the database cache. UnboundID recommends that administrators allocate enough JVM memory to fully load the Directory Server’s dataset and runtime environment into database cache. Because the Directory Server first accesses the cache, fully caching the dataset increases throughput and decreases response time by bypassing disk I/O. Administrators can configure the percentage of JVM heap memory allocated to the database cache as well as preload the database cache at startup to improve the server’s processing throughput. The Directory Server also provides an entry cache that can be used together with the database cache or in place of it.

Using the cache to keep the on-disk footprint small is important, since it is faster and easier to work with smaller backups and because fast highly-available storage systems can be expensive. Because the on-disk and in-memory footprints are both directly related to how the data is stored by the server, maintaining a compact memory footprint ensures optimal performance and throughput.

The UnboundID Directory Server has designed a memory efficient system that has a significantly reduced on-disk footprint to that of our competitors. A smaller database footprint translates directory to a smaller cache footprint with improved performance; thus, reducing the memory costs associated with the system.



Compact Representation of Directory Data

The UnboundID Directory Server employs various mechanisms for a compact representation of directory data, such as compact byte encoding, DN tokens, special attribute type compaction, gzip compression, and separate large attribute backends.

- Compact Byte Encoding:** Entry IDs represent 64-bit values that require up to eight bytes to store. However, for values that can be represented using up to 31 bits (which is the case for the first two billion entries), the UnboundID Directory Server uses a more compact four-byte encoding, which saves four bytes per entry ID, both in the entry database as well as in the index ID lists. Other vendors store entries in the backend database using LDIF representation, which requires a significant amount of space and is expensive to parse.

Other vendor solutions require the use of an Entry cache to maximize performance. While the UnboundID Directory Server provides an Entry Cache, the Directory Server’s compact representation of the entry data improves throughput without the need for an entry cache.

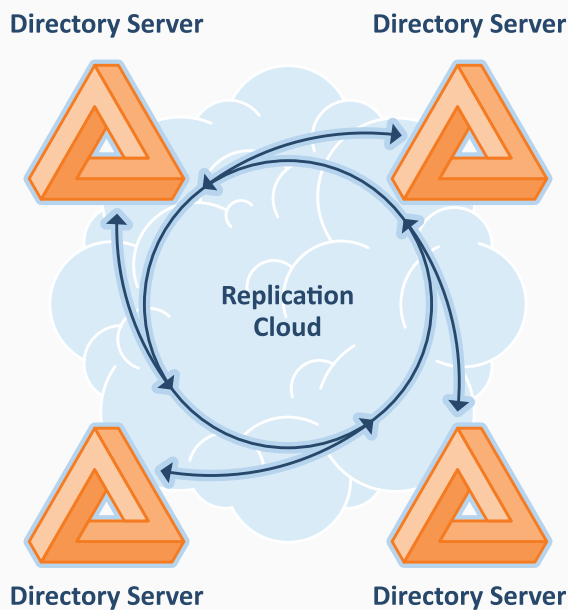
- **DN Tokens:** The UnboundID Directory Server provides a mechanism for encoding entry DNs in a compact manner. It maintains a set of tokens for parent DNs and includes only the string portion of the DN, which is relative to the defined token that matches the largest possible portion of the entry DN. For example, if a token is defined for "ou=People,dc=example,dc=com," then the DN "uid=test.user,ou=People,dc=example,dc=com" would be represented as simply as "uid=test.user" followed by the token for "ou=People,dc=example,dc=com." The Directory Server automatically defines tokens for any base DNs configured in the user backend, but administrators can define additional parent DNs for which to create tokens. The Directory Server also provides the ability to tokenize each attribute description used in entries to further improve memory efficiency.
- **Special Attribute Type Compaction:** Special compaction is applied to the values of a limited set of attribute types. In particular, the values of the entryUUID, createTimeStamp, and modifyTimeStamp attributes are stored in a compact manner, which requires less space than their string representations.
- **Gzip Compression:** The UnboundID Directory Server provides the ability to use gzip compression to further compress entry contents after all other compaction has been performed. When enabled, the backend automatically compresses entries before storing them in the database. Similarly, when the compressed entry is accessed, the Directory Server automatically decompresses the entry.
- **Large Attributes:** Similar to storing Large Objects (LOBs) in a separate database table, the UnboundID Directory Server stores a configurable set of large attributes in a specific backend separately from the encoded entry. This method improves performance for search operations that do not normally request the large attribute, because a much smaller amount of data is decoded. It also increases the cache efficiency, because the large attributes are cached separately from the remaining attributes in the entry.

By tokenizing common DNs, UnboundID found, for example, that a 40 byte DN can be compacted down to 14 bytes. Over the full range of your DIT, the available memory savings can be significant and can be used for other processes.

Large attributes are those attributes that store graphic, voice, or audio files that are from 4k to 40k or much larger in size.

Replication, Data Integrity, and Security

The UnboundID Directory Server provides a robust, centralized replication model for efficient data redundancy and fast failover. The data itself is protected through the Directory Server's integrity mechanisms and ensures that tampering or unauthorized modification of the directory data is not possible. The Directory Server also provides extensive support for secure data communication using SSL or StartTLS, access control, fine-grained privileges system, SASL authentication mechanisms and data encryption.



The UnboundID Directory Server allows administrators to define groups of replication servers by location, which are co-located in the same physical network or city to minimize communication over WAN links.

Figure 6. Centralized Replication Model

Replication

The UnboundID Directory Server has a robust replication mechanism that ensures data remains consistent across all replica servers in the environment. Unlike other vendor systems that use a point-to-point model, the UnboundID Directory Server uses a centralized model to provide a robust and highly available replication system with a dramatically simpler configuration.

Many vendors require that each replicated server be explicitly configured with information about the other servers in the environment. With the UnboundID Directory Server, administrators can provide the replica servers with connection information about one of the replication servers. Then, the Directory Server automatically discovers the rest of the topology in order to provide the necessary redundancy. This configuration is dramatically more efficient than a point-to-point model during initial setup, and even more so when adding a new server into an existing replication topology.

Another benefit of the UnboundID replication model over more traditional point-to-point models is that the Directory Server can be separated from the replication server itself, similar to the changelog provided by other directory servers. The replication server can be run in the same JVM instance as the Directory Server, in a dedicated JVM instance on the same machine, or in a dedicated instance on a different machine. The UnboundID replication model supports writable Directory Server instances without a local changelog. The replication servers can also run as dedicated instance, so that all Directory Servers instances can pool systems resources for caching directory data and processing client requests; thus, ensuring the highest possible replication performance.

Data Integrity

The UnboundID Directory Server uses data integrity mechanisms to ensure that the data is secure and has not been tampered with. Several features to help protect the integrity of the data as follows:

- **Cryptographic Digests:** The UnboundID Directory Server provides a configuration option that can include a cryptographic digest of the entry contents as part of the entry data. When retrieving an entry from the database, the server provides the ability to re-compute the entry digest and verify that the stored digest matches the computed value. The digest protects against data corruption that may have been introduced at the storage level or in the path between the storage and the Directory Server process.
- **Intelligent Log Rotation:** Whenever the UnboundID Directory Server needs to overwrite an existing file (e.g., log file, database file, etc.), it always writes the new version to a temporary file in the same filesystem as the existing file. The Directory Server then renames the current file for archiving and renames the new file into place. This process ensures that there is always a valid copy of the file on disk in the event that an error occurs while writing the new file; thus, no logging information will be lost. For some particularly important files (like the configuration file), the Directory Server maintains historical versions of the file, so that you can see previous contents that it may have held.
- **LDAP Assertion Control:** The UnboundID Directory Server provides support for the LDAP assertion control, defined in RFC 4528, which requires that the server only process the requested operation if a provided search filter matches the target entry. The control provides a mechanism for client-side data integrity to help prevent two clients from inadvertently modifying the same entry at the same time.
- **Entry Checksums:** The UnboundID Directory Server provides a **ds-entry-checksum** virtual attribute, whose value is a checksum of that entry. The Directory Server also supports bulk verification.
- **Backup Protection:** The UnboundID Directory Server provides the ability to protect the integrity of backup contents using a number of mechanisms. 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 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.
- **LDIF Export Protection:** The UnboundID Directory Server provides the ability to protect the integrity of LDIF exports. As with backups, the Directory Server can generate an optional digitally-signed cryptographic digest of the LDIF content, which can be verified when the file is imported into the server.

The UnboundID Directory Server provides an N-way multi-master system that is based on a centralized replication model for high-availability and fast failover.

The UnboundID Directory Server can perform bulk verification of the entries as they are accessed.

The LDAP Assertion Control and entry checksum virtual attribute provides an ideal method to ensure that an entry has not been modified since it was last read.

The UnboundID Directory Server generates and sends an admin alert for any entry found to not match its checksum.

Security

Security is another important area of focus for the UnboundID Directory Server. The Directory Server provides a number of security-related features, including the following:

- **SSL/StartTLS Support:** The UnboundID Directory Server supports the use of SSL and StartTLS to encrypt communication with the client. Administrators can configure different certificates for each connection handler as well as map certificates to user entries based on their fingerprint.
- **Message Digests/Encryption Algorithms:** The UnboundID Directory Server supports the use of a number of one-way message digests (e.g., CRYPT, MD5, SHA-1, and SHA-2 256-bit, 384-bit, and 512-bit, with or without salt), as well as a number of reversible encryption algorithms (3DES, AES, Blowfish, and RC4) for storing passwords.
- **Password Policy Support:** The UnboundID Directory Server provides extensive password policy support including features, such as, customizable password attributes, 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, such as, one-way digests (MD5, SMD5, SSHA256, SSHA384, SSHA512) and reversible digests (BASE64, 3DES, AES, RC4, BLOWFISH). Administrators can also support a number of password validators, such as maximum password length, similarity to current password, and other features.
- **SASL Mechanism Support:** The UnboundID Directory Server supports a number of SASL mechanisms, including ANONYMOUS, CRAM-MD5, DIGEST-MD5, EXTERNAL, GSSAPI, and PLAIN. 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 UnboundID Directory Server allows reversible encryption to store the passwords, so that they will not need to be stored in clear text. This mechanism is available for 3DES, AES, RC4, Blowfish, CRAM-MD5 and DIGEST-MD5.
- **Full-Featured Access Control System:** The UnboundID Directory Server includes 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 the use of specific types of controls and extended operations and provides stronger validation for access control rules before accepting them compared to other vendors' directory servers.
- **Fine-Grained Privilege Subsystem:** The UnboundID Directory Server includes a fine-grained privilege subsystem to grant capabilities to normal users that normally would be only available to root users. Thus, multiple root users can be defined with their own separate accounts, rather than requiring a single shared root account. The privileges subsystem can also be used to grant or revoke administrative capabilities for different users. SASL authentication is supported for root users, and root users can be made subject to password policy constraints.
- **Multiple Client Connection Policies:** The UnboundID Directory Server provides the ability to define multiple client connection policies to categorize clients based on properties, such as the address, protocol, authentication identity, and communication security. Client connection policies control which operations are allowed and which portions of the DIT are accessible. Restrictions enforced by client connection policies can supersede permissions granted by access control or privileges. Administrators can use this mechanism to restrict what clients can do based on how they are communicating with the server. For example, even root users can be prevented from accessing the server configuration unless they are on the local LAN or use certificate-based authentication.
- **Encrypted/Digitally-Signed Backups or Exports:** The UnboundID Directory Server provides the ability to encrypt and/or digitally sign backups or LDIF exports to ensure that the data is protected.
- **Applying Entry Encryption:** The UnboundID Directory Server provides the ability to encrypt stored data from unauthorized access. When data encryption is enabled in the server, then encryption will be applied to an entire entry. Applying the encryption to the entire entry rather than only to a specified set of attributes dramatically simplifies the server configuration, allows for a more compact representation of the encrypted data, and ensures that other potential references to sensitive data are also protected.
- **Configuring Sensitive Attributes:** The UnboundID Directory Server offers specific mechanisms to protect information accessed or replicated between servers. This communication security may be enabled independently of data encryption. 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. With sensitive attributes, you may prevent particular attributes from being returned to insecure clients. In other cases, it may be useful to have additional

The UnboundID Directory Server supports various authorization and authentication methods.

The UnboundID Directory Server supports the use of the authentication password syntax, described in RFC3112.

Root users can authenticate via SASL mechanisms on the UnboundID Directory Server.

The UnboundID Directory Server automatically sends an alert for any operation that results in a change in the access control rules. It also enters in lockdown mode if it detects an invalid or malformed access control rule.

levels of protection in place to ensure that some attributes are even more carefully protected. For instance, you may want to prevent a particular attribute from ever being returned in a search request, or from being modified. These kinds of protection may be achieved using sensitive attribute definitions.

Productivity and Operations

The UnboundID Directory Server provides atomic transactional functionality in the form of batched and interactive transactions, which are typically available for RDBMS databases. The Directory Server also provides robust data maintenance tools for import, export, backup and restore operations.

Directory Server Transactional Functionality

Historically, LDAP directory servers have been ACID compliant but provided very little transactional functionality to clients. Transactions require that any single operation be atomic, so that if a modification alters multiple attributes in an entry, the operation either completely succeeds or completely fails. However, for most LDAP servers, there was no such guarantee or protection that any single operation is atomic across multiple operations. For example, adding an entry and then adding that entry's DN to the administrator group cannot be guaranteed to be atomic.

Atomic transactions provide the ability to submit multiple operations as a single operation.

To provide a transactional model that provides robust atomicity, the UnboundID Directory Server provides full support for the following atomic operations:

- **Modify-Increment Extension (RFC 4525):** This RFC defines an LDAP modification type to increment the integer value of an attribute type by a specified amount (or a negative amount to decrement the value). This function is used to atomically increase or decrease an integer value by a given amount without needing to know the previous or resulting value. This RFC solves a classic race condition in which a client reads the value and writes a new value based on what was read, during which time another client altered that value between the read and write operations.
- **Read Entry Controls (RFC 4527):** This RFC defines LDAP pre-read and post-read controls to retrieve a copy of an entry immediately before it is deleted, modified, or renamed, or to retrieve a copy of an entry immediately after it has been added, modified, or renamed. These functions are particularly useful when used in conjunction with the Modify-Increment modification control. Another capability of the Read Entry Controls is that they can increment the value by a specified amount and then retrieve the updated value. The controls can also be used to obtain information about changes that involved operational attributes generated by the server. For example, administrators can use the controls to obtain the entryUUID value that was assigned when an entry was created.
- **Assertion Control (RFC 4528):** The RFC defines the LDAP assertion control to request that an operation only be processed if the target entry matches a filter included in the control value. The Assertion Control ensures that a value is only updated if the current value is what the client expects.

Batched Transactions

In the UnboundID Directory Server, our implementation of draft-zeitenga-ldap-txn is called batched transactions, a mechanism for grouping multiple write operations together as a single atomic unit. Using batched transactions, administrators can create, modify, or delete multiple entries in a batch file and have them run in an atomic manner, meaning that either all the operations succeed, or none of them do (for example, if a failure occurs with one of the batched processes). In general, the server simply batches the operations, which are part of the transaction, and does not process them until the client requests that the server commit that transaction. Note that this implementation does not provide any capability for interjecting read operations into the process. There is no way to retrieve data from the server within the transaction and use that information in the course of updating other entries.

The UnboundID Directory Server uses its own UnboundID-specific OIDs to support the Batched Transaction draft even though it has not reached a level of maturity. If any other directory vendors provides support for transactions, as described in this draft, their implementations would likely use different OIDs for the control and extended operations needed to implement this function with the UnboundID implementation.

Interactive Transactions

The UnboundID Directory Server provides interactive transactions, which enable read and write operations to be combined within the same transaction. This function is not based on any public specification but is based on UnboundID's design. The interactive transaction process is similar to that of batched transactions except that interactive transactions can include search and compare operations within the transaction. Also, interactive transaction operations can be processed immediately as they are requested, rather than batched up and processed only when the commit request is received.

UnboundID developed interactive transactions based on our own design. There is no proposed standard specification for this capability. Interactive transactions require specific coding for its implementation in client applications.

The basic process used for invoking an interactive transaction is as follows:

1. Send a start interactive transaction extended request to the Directory Server.
2. Read the start interactive transaction extended response from the Directory Server. If the server accepts the request, then the response contains a transaction ID that is used for the remainder of the transaction.
3. Issue one or more requests that should be processed as part of the interactive transaction. In each request, include an interactive transaction specification request control that contains the transaction ID obtained from the start interactive transaction response from step 2.
4. After all operations have been processed, send an end interactive transaction extended request to the Directory Server indicating whether the transaction should be committed or aborted.

The UnboundID Directory Server also ensures a level of isolation while a transaction is in progress, so that other clients do not see altered data until it has been committed. The Directory Server does not allow any such clients to read or update any entry that have been altered during the course of the transaction until the transaction itself has been committed or aborted.

Backup and Restore

The UnboundID Directory Server writes its backup to a single file, which is easier to manage than archiving each file individually. The Directory Server provides the following capabilities for backup and restore operations:

- **Incremental Backups:** The Directory Server provides support for incremental backups, which archive only those portions of the database that have changed since the last full or incremental backup. Incremental backups require less time and take up less space than full backups. When restoring, the full backup and any intermediate backups are automatically restored. Any files that are no longer needed are skipped.
- **Backup and Restore Individually:** The Directory Server provides the ability to back up and restore each backend individually rather than requiring the entire server to be backed up and restored as a single unit.
- **Backup Compression:** The Directory Server provides backup compression, so that they take up less space on disk. Backup compression may take more time to generate compared to a non-compressed backup, but it is generally faster to restore from a compressed backup than a non-compressed one.
- **Encryption and Digitally-Signed Backup:** The Directory Server provides encryption and/or digitally-signed backups to secure the data.
- **Scheduled Tasks:** The Directory Server provides a Tasks subsystem that can schedule backups or restores at pre-specified times. Administrators can monitor all active Tasks from the command line and can start, stop, or modify a scheduled backup or restore on the system.

The UnboundID Directory Server provides comprehensive backup and restore tools that can be run immediately or scheduled as tasks.

Import and Export

The UnboundID Directory Server provides the following capabilities for LDIF import and export:

- **Inclusion and Exclusion Filtering:** When importing or exporting data, entries can be included or excluded based on their location in the Directory Information Tree (DIT). The UnboundID Directory Server can also include or exclude entries or individual attributes based whether they match a provided filter.
- **Export Compression:** The UnboundID Directory Server provides data compression for LDIF files during export as well as automatic import decompression capabilities from a compressed LDIF file.
- **Encryption and Digitally-Signed Exports:** The UnboundID Directory Server provides data encryption and/or digitally-signed LDIF files during export. Likewise, the Directory Server supports import capabilities from an encrypted and/or digitally-signed import file.
- **Scheduled Tasks:** Like backups and restores, the UnboundID Directory Server supports scheduled data imports or exports that are run as tasks at pre-specified times. Administrators can monitor all active Tasks from the command line and can start, stop, or modify a scheduled import or export on the system.

Instrumentation and Management

The UnboundID Directory Server provides a robust change notification system to monitor changes or significant events within the server. Administrators can manage the Directory Server configuration using a set of command-line tools and a Directory Server Management web console.

Change Notification

The UnboundID Directory Server provides a number of mechanisms to obtain information about changes or other significant events within the server:

- **Persistent Search Control:** The persistent search control (as described in draft-ietf-ldapext-psearch) is used by clients to request notification of changes that match a given set of criteria. This control is a very useful feature but requires a long-running search on the client. Note that if the search is interrupted (e.g., because of a lost connection between the client and server), it can be difficult to ensure that all changes are caught.
- **LDAP Change Log:** The Directory Server provides an LDAP-accessible changelog (as described in draft-good-ldap-changelog) that can be accessed by clients to obtain historical information about changes within the server. The LDAP change is useful for cases in which the client does not require real-time notification of changes or cannot maintain long-running connections to the server.
- **Subscription Framework:** The Directory Server provides a change subscription framework, in which criteria can be defined in the server configuration. The server will invoke custom code whenever a change is processed in the server that matches that configuration. Although the subscription framework requires custom code to run whenever a matching operation is processed, it ensures that no changes will be missed, because the server invokes that code itself without a client-side component. This feature could also be implemented using the plugin API although the change subscription framework is specifically designed for this capability.
- **Configuration APIs:** The Directory Server provides APIs to intercept changes to the server configuration. These APIs can be used by server components to detect and validate configuration changes as requested by clients. They provide the ability to reject the change or immediately react to it in order to apply the new configuration. This API is generally intended for server components that have some configuration associated with them but are used by custom components, such as plugins.
- **Logging:** The Directory Server provides extensive logging capabilities for access, error, and debug messages (described in a later section). The access logging framework obtains information about operations processed within the server. An error logging framework provides information on warnings or errors about potential problems generated during normal processing. The debug logging framework provides troubleshooting information necessary to track any errors or conditions.

Administrators can use the persistent search control to be notified of changes. This control can be used instead of or in addition to other alert mechanisms, such as, email or SNMP traps.

The UnboundID Synchronization Server uses the LDAP change log to sync changes to its topology endpoints.

Server Configuration

The UnboundID Directory Server has a full-featured set of administration tools to install and manage the server. It provides a number of administrative interfaces, including a web-based administration console and a dsconfig command-line tool. The UnboundID® Directory Server Management Console is a graphical web application that provides access to the server's configuration. The console provides the same functionality for managing the configuration as the dsconfig command-line tool, and also provides easy access to monitor data and server documentation. Both tools allow you to safely manage the configuration of a whole topology of servers as easily as a single server.

The Directory Server configuration can be accessed using the dsconfig command-line tool and the web-based Directory Server Management Console.

The Directory Server Management Console

The Directory Server Management Console is a web application that allows administrators to use a web browser to easily view and manage their topology.

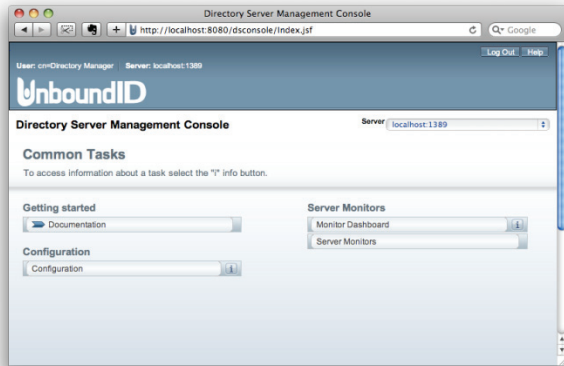


Figure 7. UnboundID Directory Server Management Console

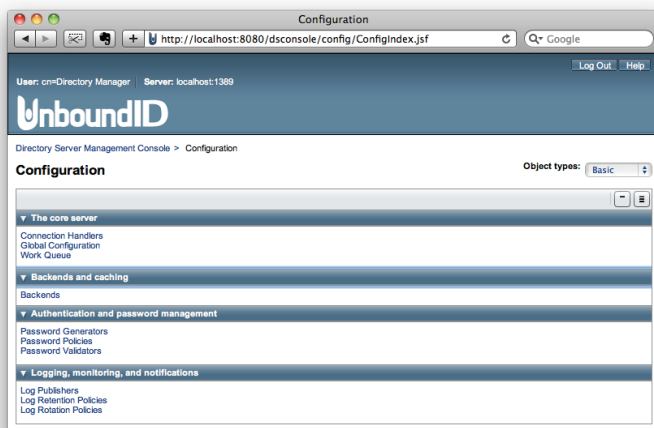


Figure 8. UnboundID Directory Server Management Console

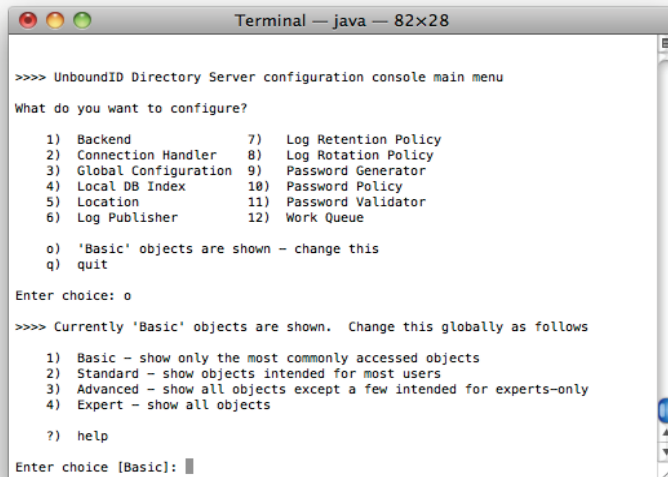
The Management Console provides an Object Type menu, which allows you to display the properties relevant to the administrator's experience level.

The dsconfig Command-Line Tool

The **dsconfig** tool is a command-line utility with several modes of operation: a menu-driven interactive mode, a non-interactive mode that can be invoked using command-line arguments, and a batch mode in which configuration changes can be read from a file to efficiently make multiple changes in succession. Every configuration change made to a server is recorded in a configuration audit log that can be easily replayed using the dsconfig batch mode. This feature allows you to automatically configure other servers in your topology using an existing configuration.

- **Object Menu Levels:** Like the Directory Server Management Console, the **dsconfig** tool provides four menu options that display only those server properties for each level. For example, the Advanced and the Expert menus show many properties that require LDAP directory server expertise on the part of the administrator. Any arbitrary change to some of these properties could lead to server malfunction.

The dsconfig tool can be run interactively from the command line or within a script. All changes are recorded within a configuration audit log that also records the commands you can use to back out of the change.



```
Terminal — java — 82x28

>>>> UnboundID Directory Server configuration console main menu

What do you want to configure?

 1) Backend                7) Log Retention Policy
 2) Connection Handler     8) Log Rotation Policy
 3) Global Configuration   9) Password Generator
 4) Local DB Index        10) Password Policy
 5) Location               11) Password Validator
 6) Log Publisher         12) Work Queue

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

Enter choice: o

>>>> Currently 'Basic' objects are shown. Change this globally as follows

 1) Basic - show only the most commonly accessed objects
 2) Standard - show objects intended for most users
 3) Advanced - show all objects except a few intended for experts-only
 4) Expert - show all objects

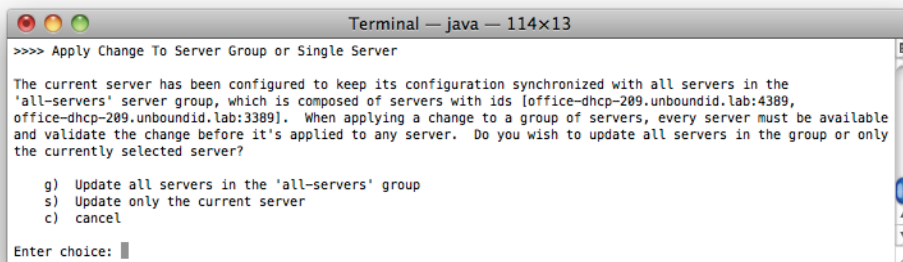
 ?) help

Enter choice [Basic]:
```

Figure 9. The dsconfig Command-Line Utility with Four Different Menu Levels

- **Server Groups:** Both the web-based administration console and the dsconfig utility have many powerful functions for managing a directory environment. For example, the tools provide the ability to make configuration changes to multiple servers in a server group, rather than requiring administrators to manage each server individually, which helps avoid operation inconsistency. If a configuration change is to be applied to multiple servers, the configuration tool will first verify that all servers will accept that change before attempting to apply it anywhere.

Administrators can use server groups to safely apply a configuration change to multiple servers.



```
Terminal — java — 114x13

>>>> Apply Change To Server Group or Single Server

The current server has been configured to keep its configuration synchronized with all servers in the
'all-servers' server group, which is composed of servers with ids [office-dhcp-209.unboundid.lab:4389,
office-dhcp-209.unboundid.lab:3389]. When applying a change to a group of servers, every server must be available
and validate the change before it's applied to any server. Do you wish to update all servers in the group or only
the currently selected server?

 g) Update all servers in the 'all-servers' group
 s) Update only the current server
 c) cancel

Enter choice:
```

Figure 10. Making changes to multiple servers in a server group

Configuration Property Components

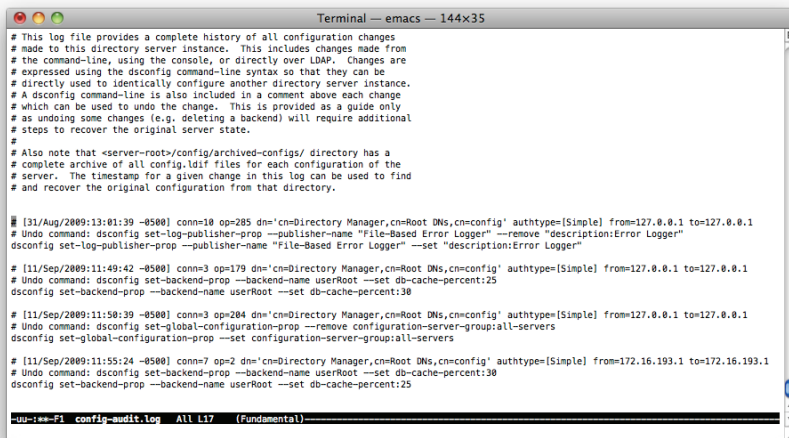
The Directory Server defines all configurable elements in a common set of configuration definitions, which specify the set of configuration objects and the types of properties associated with those objects. The configuration framework validates all configuration changes prior to being accepted.

The Directory Server uses the configuration definitions in a number of ways. The Directory Server reads its configuration using the configuration definitions, as well as detects, validates, and reacts to changes made online to the server configuration. The administrative interfaces, the dsconfig tool and the Directory Server Management Console, use the configuration definitions to discover what properties are available and to perform client-side validation.

Configuration Audit Log

The Directory Server also provides mechanisms for keeping track of configuration changes made over time, regardless of the tool used to make those changes. The configuration audit log provides a record of all configuration changes in a form that is compatible for use with the dsconfig batch mode for easy playback, and also includes comments indicating when the change was made, the administrator that requested the change, and a command that may be used to revert the change if desired. In addition, the Directory Server archives complete copies of each configuration that it has used, so you can see exactly what configuration was in use at any given time in the past.

The configuration audit log records comments that detail when and by whom each change was made, along with an analogous Undo command that can be used to revert the change.



```
Terminal — emacs — 144x35
# This log file provides a complete history of all configuration changes
# made to this directory server instance. This includes changes made from
# the command-line, using the console, or directly over LDAP. Changes are
# expressed using the dsconfig command-line syntax so that they can be
# directly used to identically configure another directory server instance.
# A dsconfig command-line is also included in a comment above each change
# which can be used to undo the change. This is provided as a guide only
# as undoing some changes (e.g. deleting a backend) will require additional
# steps to recover the original server state.
#
# Also note that <server-root>/config/archived-configs/ directory has a
# complete archive of all config.ldif files for each configuration of the
# server. The timestamp for a given change in this log can be used to find
# and recover the original configuration from that directory.

# [31/Aug/2009:13:01:39 -0500] conn=10 op=285 dn='cn=Directory Manager,cn=Root DNs,cn=config' authType=[Simple] from=127.0.0.1 to=127.0.0.1
# Undo command: dsconfig set-log-publisher-prop --publisher-name "File-Based Error Logger" --remove "description:Error Logger"
dsconfig set-log-publisher-prop --publisher-name "File-Based Error Logger" --set "description:Error Logger"

# [11/Sep/2009:11:49:42 -0500] conn=3 op=179 dn='cn=Directory Manager,cn=Root DNs,cn=config' authType=[Simple] from=127.0.0.1 to=127.0.0.1
# Undo command: dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:25
dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:30

# [11/Sep/2009:11:50:39 -0500] conn=3 op=204 dn='cn=Directory Manager,cn=Root DNs,cn=config' authType=[Simple] from=127.0.0.1 to=127.0.0.1
# Undo command: dsconfig set-global-configuration-prop --remove configuration-server-group:all-servers
dsconfig set-global-configuration-prop --set configuration-server-group:all-servers

# [11/Sep/2009:11:55:24 -0500] conn=7 op=2 dn='cn=Directory Manager,cn=Root DNs,cn=config' authType=[Simple] from=172.16.193.1 to=172.16.193.1
# Undo command: dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:30
dsconfig set-backend-prop --backend-name userRoot --set db-cache-percent:25

--w--F1 config-audit.log All L17 (Fundamental)
```

Figure 11. The Configuration Audit Log

Logging

The UnboundID Directory Server has a rich logging subsystem whose features help it stand out from the competition. Some of the most significant logging features include the following:

- **Multiple Loggers of Any Type:** The UnboundID Directory Server supports any number of active loggers of any type (access, error, or debug) in the server. Each logger has its own configuration and generally has a minimal impact on overall server performance.
- **Filtered Access Logging:** The UnboundID Directory Server provides the ability to control which types of access log messages are written on a very fine-grained level. Administrators can configure each access logger so that it will only include messages that meet certain criteria, including information about the client (such as its address, protocol, communication security level, and authentication state), the request (such as the type of operation, the location and content of the target entry, and any included request controls), and the result (such as the operation result code, the length of time required to process the request, and any included response controls). For example, an administrator can easily create an access log that only records information about operations that did not complete successfully, operations that took longer than some length of time to complete, operations targeting some portion of the DIT, or operations requested by a particular client application.

The Directory Server provides a full featured Logging Subsystem that is capable of numerous configuration options for any requirement. For example, administrators can set up separate logs for different applications or set up filtered logs that filter only a pre-determined set of criteria that can automatically send admin alert notifications when an event occurs. The logging possibilities are endless.

- **Multiple Message Consolidation:** The UnboundID Directory Server provides the ability to consolidate multiple pieces of information into a single access log message. Traditionally, one access log message is generated when the client establishes a connection, one message for each request received by the server, and one message with information about the result of processing each operation, etc. The Directory Server provides an option to consolidate all information about an operation in a single message for easy viewing. This message can be configured to include details about the request and results as well as the address and authentication identity of the client that requested it. Often, applications keep connections open for a long time, so details about these connections may be buried in a log file that has been rotated or even deleted. On busy servers, log messages for requests and responses may be separated far apart in the same log file or even in different log files. Plus, providing all information about an operation on a single line is much easier to examine using text processing tools, like the UNIX `grep` command.
- **Access Logs Filtering:** The UnboundID Directory Server can filter access log messages based on any type of client information available to the Directory Server (e.g., the address of the client, information about the authenticated users, type of authentication used, whether the communication is secure, etc.), the content of the request, and the result of operation processing. For example, administrators can create access logs that contain only information about operations that did not complete successfully, or operations that took more than some length of time to complete, or all operations.
- **Centralized Logging Capabilities:** The UnboundID Directory Server provides capabilities for centralizing access and error log messages across multiple server instances. In addition to logging to files on the server filesystem, the Directory Server can log to a UNIX syslog server and/or to a relational database. Log messages may include an instance name field, which can identify the Directory Server instance that generated a log message.
- **Intermediate Client Control:** The UnboundID Directory Server supports the Intermediate Client Control, a custom control, to help track operations between multiple systems in the environment. For example, if the backend server is an UnboundID Directory Server instance, then the backend server log information for that request will include information about the address and port of the Directory Proxy Server instance and the connection ID and operation ID used to identify that operation in the Directory Proxy Server. Other applications can be developed to make use of this control (available in the Commercial Edition of the UnboundID LDAP SDK for Java) to include information about themselves to help better track requests from those applications through the directory environment. The control makes correlation possible, so that inefficient or abusive activity can be tracked to its origin.
- **Access/Error Log Parsing APIs:** The Commercial Edition of the UnboundID LDAP SDK for Java (which is included with the Directory Server) provides APIs for reading and parsing access and error log messages. A summarize-access-log tool is also provided, which uses this API to examine access log messages generated by the server and reports a number of metrics from the data those log files contain, including information about the number and percentage of each type of operation, the average length of time required to process each type of operation, a breakdown of the processing times into a number of predefined buckets, a breakdown of the result codes returned, and a breakdown of the most common search scopes, filter types, and entry counts.

The UnboundID Directory Server provides a convenience tool that generates a summary of the server's access logs. The generated file breaks down the important metrics that would be otherwise hard to determine by examining the access logs on their own.

Administrative Alerts

The UnboundID Directory Server provides an administrative alert framework that can be used to notify administrators of any significant warnings, errors, or other noteworthy events that occur in the server. Administrators can be notified using existing alert notification handlers via log messages, email, SNMP traps, or JMX notifications. You can also configure the Directory Server to execute a specified command whenever an alert is generated, with information about that alert available as command-line arguments. All administrative alerts are also exposed via LDAP as entries below a base DN of "cn=alerts", and you can use the persistent search operation to be automatically notified over LDAP of any new alerts generated by the server.

The administrative alert framework allows the Directory Server to generate and handle alerts delivered over different means.

The administrator can select the events that apply for each type of notification based on the severity level or the specific type of alert. For example, it may be desirable to log information about all types of alerts, but only generate e-mail messages or SNMP traps for warnings or errors. Some sample events include the following:

- **Startup/Shutdown:** Sends an alert when the directory server completes the startup process or begins the shutdown process.
- **Applied Configuration Changes:** Sends an alert when a configuration change is applied to the server.

- **Changes in Backend Server Availability:** Sends an alert whenever there is a change in the health check state of any of the backend servers.
- **Scheduled Task Errors:** Sends an alert if there is an error processing a scheduled task, such as doing a backup.
- **Disk Space Usage:** Sends an alert if the available disk space drops below a configured threshold.
- **Unapplied Replication Changes:** Sends an alert if the backlog of unapplied replication changes exceeds the configured threshold.

Real-Time Monitoring

The UnboundID Directory Server exposes real-time monitoring information in a single branch (cn=monitor) and can be accessed using the Directory Server Management Console, Java Management Extensions (JMX), or directly over LDAP. The UnboundID Directory Server's monitoring information includes the following:

The UnboundID Directory Server's alerting system can be used with the monitoring framework to immediately notify administrators of any problems.

Active Connections	Provides information about all active connections including when the connection was established, from which source address, which user established the connection, the number of operations completed, and the number in progress.
Active Operations	Provides information about all active operations including when the operation started, the connection that initiated the operation, the type of the operation, and details of the operation itself, e.g., the search filter and base DN for a search operation.
LDAP Statistics	Provides information about the number of operations of each type that have been processed by the directory server.
Processing Time Histogram	Provides a breakdown per operation type of how long operations have taken in the Directory Server. A cumulative count is stored for operations that took less than 1ms, 2ms, 3ms, 5ms, 10ms, etc.
Work Queue Busyness	Provides the percentage of time that worker threads have been busy actively processing operations, and exposes the recent percent busy; and the average and maximum percent busy since startup.
Backend Database Statistics	Provides information about the backend database including the number of entries in each backend, the percentage full of the database cache, and the on-disk size of the database.
Disk Space	Provides information about the disk utilization of all filesystems used by the Directory Server represented as an absolute value and a percentage of the overall filesystem size.
Stack Traces	Provides a current stack trace of all threads in the Directory Server.
System Information	Provides information about the system on which the Directory Server is running, including details about the hardware, operating system, and JVM being used.

The Commercial Edition of the UnboundID LDAP SDK for Java provides an API for retrieving and parsing various types of monitor entries from the UnboundID Directory Server.

LDAP SDK for Java

The UnboundID LDAP SDK for Java is a fast, user-friendly, and powerful Java API for client communications with LDAP directory servers. UnboundID developed the LDAP SDK for Java because of the stagnation, limited feature set, usability problems, and performance issues with other Java-based LDAP APIs. The LDAP SDK for Java works with any LDAP version 3-compliant directory server.

The LDAP SDK for Java comes in two versions: a Standard Edition, which is freely available for use with any type of LDAPv3 directory server, and a Commercial Edition, which is included with the UnboundID Directory Server and UnboundID Directory Proxy Server. The Commercial Edition provides all of the capabilities of the Standard Edition, as well as additional features that are specific to the UnboundID Servers, including support for additional controls and extended operations and APIs for retrieving and interacting with administrative alerts, monitor entries, scheduled tasks, and log messages. For more information, see the LDAP SDK for Java web site at the following URL:

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 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.

The UnboundID LDAP SDK offers better performance, better ease of use, and more features than other Java-based LDAP APIs.

The UnboundID Server SDK offers extensibility to help developers customize the core server functionality.

About UnboundID Corp.

UnboundID is a leading provider of real-time identity management software for cloud, mobile and social applications. UnboundID is a privately-held company based in Austin, Texas and is funded by Silverton Partners.

For more information, visit www.unboundid.com.

The UnboundID staff members are experts in the Identity Management field and have a proven track record of many successful deployments.



UnboundID and the UnboundID logo are trademarks of UnboundID Corp.
All other product or service names are trademarks of their respective companies.