Installation Guide

/ ForgeRock Directory Services 5

Latest update: 5.0.0

Mark Craig

ForgeRock AS
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Guide to installing ForgeRock® Directory Services software.

Table of Contents

Preface

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

The ForgeRock Common REST API works across the platform to provide common ways to access web resources and collections of resources.

This guide shows you how to install, upgrade, and remove ForgeRock Directory Services software components listed in "ForgeRock Directory Services Software Components".

Read the Release Notes before you get started.

ForgeRock Directory Services Software Components

Component	Description
Directory Server and Tools	Pure Java, high-performance server that can be configured as: • An LDAPv3 directory server with the additional capability to serve directory data to REST applications over HTTP. • An LDAPv3 directory proxy server providing a single point of access to underlying directory servers. • A replication server handling replication traffic with directory servers and with other replication servers, receiving, sending, and storing changes to directory data. Server distributions include command-line tools for installing, configuring, and managing servers. The tools make it possible to script all operations.
DSML Gateway	DSML support is available through the gateway, which is a Java web application that you install in a web container.
REST to LDAP Gateway	In addition to the native server support for REST over HTTP, the REST to LDAP gateway is a Java web application that lets you configure REST access to any LDAPv3 directory server.
Client Toolkit	LDAP command-line client toolkit for scripting LDAP operations. The client toolkit includes LDAP command-line tools, and command-line tools for load testing.
Java APIs	Java server-side APIs for applications that embed and server plugins that extend directory services.

Component	Description
	Java LDAP client-side APIs used internally, and for building client applications. All Java APIs are fully supported, and have Interface Stability: Evolving. In other words, when you write applications or plugins using the APIs, be prepared to adapt to incompatible changes in both major and minor releases.

For a list of available downloads, see "*Before You Install*".

1. Who Should Read this Guide

This guide is written for anyone installing ForgeRock Directory Services software who plans to maintain directory services for client applications. Basic installation can be simple and straightforward, particularly if you are already acquainted with directory services. Upgrading a running directory service without a single point of failure that can cause downtime requires significant planning.

This guide covers the install, upgrade, and removal (uninstall) procedures that you need perform only once per version. This guide aims to provide you with an understanding of what happens when you perform the steps.

2. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

  While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

- ForgeRock product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

3. Using the ForgeRock.org Site

The ForgeRock.org site has links to source code for ForgeRock open source software, as well as links to the ForgeRock forums and technical blogs.

If you are a *ForgeRock customer*, raise a support ticket instead of using the forums. ForgeRock support professionals will get in touch to help you.

Chapter 1

Before You Install

This chapter covers requirements for running ForgeRock Directory Services software in production. It covers the following topics:

- Downloading ForgeRock Directory Services software
- Choosing hardware
- Choosing an operating system
- Preparing the Java environment
- Choosing an application server when using the DSML or REST to LDAP gateway
- Assigning FQDNs when using replication
- Using appropriately signed digital certificates

1.1. Downloading ForgeRock Directory Services Software

The ForgeRock BackStage site provides access to ForgeRock releases. ForgeRock releases are thoroughly validated for ForgeRock customers who run the software in production deployments, and for those who want to try or test a given release.

"ForgeRock Directory Services Software" describes the available software.

ForgeRock Directory Services Software

File	Description
`DS-5.0.0.zip`	Cross-platform distribution of the server software. Pure Java, high-performance server that can be configured as: • An LDAPv3 directory server with the additional capability to serve directory data to REST applications over HTTP. • An LDAPv3 directory proxy server providing a single point of access to underlying directory servers. • A replication server handling replication traffic with directory servers and with other replication servers, receiving and sending changes to directory data.

File	Description
	Server distributions include command-line tools for installing, configuring, and managing servers. The tools make it possible to script all operations. By default, this file unpacks into an `opendj/` directory.
`DS-5.0.0.msi`	Microsoft Windows native installer for the server software. By default, this installs files into a `C:\Program Files (x86)\OpenDJ\` directory.
`DS-5.0.0-1_all.deb`	Server software native packages for Debian and related Linux distributions. By default, this installs files into an `/opt/opendj/` directory.
`DS-5.0.0-1.noarch.rpm`	Server software native packages for Red Hat and related Linux distributions. By default, this installs files into an `/opt/opendj/` directory.
`DS-dsml-servlet-5.0.0.war`	Cross-platform DSML gateway web archive.
`DS-rest2ldap-servlet-5.0.0.war`	Cross-platform REST to LDAP gateway web archive.
`DS-ldap-toolkit-5.0.0.zip`	Cross-platform command-line LDAP Client Toolkit. By default, this file unpacks into an `opendj-ldap-toolkit/` directory.

The platform version number that appears in the download file names may differ from the internal version number. The internal version number for this release is 4.0.0.

1.2. Choosing Hardware

Thanks to the underlying Java platform, ForgeRock Directory Services software runs well on a variety of processor architectures. Many directory service deployments meet their service-level agreements without the very latest or very fastest hardware.

1.2.1. Fulfilling Memory Requirements

When installing an directory server for evaluation, you need 256 MB memory (32-bit) or 1 GB memory (64-bit) available, with 150 MB free disk space for the software and a small set of sample data.

For installation in production, read the rest of this section. You need at least 2 GB memory for a directory server and four times the disk space needed for initial production data in LDIF format. A replicated directory server stores data, indexes for the data, operational attribute data, and historical information for replication. The server configuration trades disk space for performance and resilience, compacting and purging data for good performance and for protection against temporary outages. In addition, leave space for growth in database size as client applications modify and add entries over time.

For a more accurate estimate of the disk space needed, import a known fraction of the initial LDIF with the server configured for production. Run tests to estimate change and growth in directory data, and extrapolate from the actual space occupied in testing to estimate the disk space required in production.

Directory servers almost always benefit from caching all directory database files in system memory. Reading from and writing to memory is much faster than reading from and writing to disk storage.

For large directories with millions of user directory entries, there might not be room to install enough memory to cache everything. To improve performance in such cases, use quality solid state drives either for all directory data, or as an intermediate cache between memory and disk storage.

1.2.2. Choosing a Processor Architecture

Processor architectures that provide fast single thread execution tend to help ForgeRock Directory Services software deliver the lowest response times. For top-end performance in terms of sub-millisecond response times and of throughput ranging from tens of thousands to hundreds of thousands of operations per second, the latest x86/x64 architecture chips tend to perform better than others.

Chip multi-threading (CMT) processors can work well for directory servers providing pure search throughput, though response times are higher. However, CMT processors are slow to absorb hundreds or thousands of write operations per second. Their slower threads get blocked waiting on resources, and thus are not optimal for deployments with high write throughput requirements.

1.2.3. Fulfilling Network Requirements

On systems with fast processors and enough memory to cache directory data completely, the network can become a bottleneck. Even if a single 1 Gbit Ethernet interface offers plenty of bandwidth to handle your average traffic load, it can be too small for peak traffic loads. Consider using separate interfaces for administrative traffic and for application traffic.

To estimate the network hardware required, calculate the size of the data returned to applications during peak load. For example, if you expect to have a peak load of 100,000 searches per second, each returning a full 8 KB entry, you require a network that can handle 800 MB/sec (3.2 Gbit/sec) throughput, not counting other operations, such as replication traffic.

1.2.4. Fulfilling Storage Requirements

Note

The directory server does not currently support network file systems such as NFS for database storage. Provide sufficient disk space on local storage such as internal disk or an attached disk array.

For a directory server, storage hardware must house both directory data, including historical data for replication, and server logs. On a heavily used server, you might improve performance by putting access logs on dedicated storage.

Storage must keep pace with throughput for write operations. Write throughput can arise from modify, modify DN, add, and delete operations, and from bind operations when a login timestamp is recorded, and when account lockout is configured, for example.

In a replicated topology, a directory server writes entries to disk when they are changed, and a replication server writes changelog entries. The server also records historical information to resolve potential replication conflicts.

As for network throughput, base storage throughput required on peak loads rather than average loads.

1.3. Choosing an Operating System

ForgeRock Directory Services 5 software is supported on the following operating systems:

- Linux 2.6 and later
- Microsoft Windows Server 2008, 2008 R2, 2012, and 2012 R2
- Oracle Solaris 10, 11

In order to avoid directory database file corruption after crashes or power failures on Linux systems, enable file system write barriers and make sure that the file system journaling mode is ordered. For details on how to enable write barriers and how to set the journaling mode for data, see the options for your file system in the **mount** command manual page.

1.3.1. Setting Maximum Open Files

An OpenDJ server needs to be able to open many file descriptors, especially when handling thousands of client connections. Linux systems in particular often set a limit of 1024 per user, which is too low to handle many client connections to an OpenDJ server.

When setting up an OpenDJ server for production use, make sure the server can use at least 64K (65536) file descriptors. For example, when running the server as user `opendj` on a Linux system that uses `/etc/security/limits.conf` to set user level limits, you can set soft and hard limits by adding these lines to the file:

```
opendj soft nofile 65536
opendj hard nofile 131072
```

The example above assumes the system has enough file descriptors available overall. You can check the Linux system overall maximum as follows:

```
$ cat /proc/sys/fs/file-max
204252
```

1.3.2. Preventing Interference With Antivirus Software

Prevent antivirus and intrusion detection systems from interfering with OpenDJ software.

Before using OpenDJ software with antivirus or intrusion detection software, consider the following potential problems:

Interference with normal file access

Antivirus and intrusion detection systems that perform virus scanning, sweep scanning, or deep file inspection are not compatible with OpenDJ file access, particularly database file access.

Antivirus and intrusion detection software can interfere with the normal process of opening and closing database working files. They may incorrectly mark such files as suspect to infection due to normal database processing, which involves opening and closing files in line with the database's internal logic.

Prevent antivirus and intrusion detection systems from scanning database and changelog database files.

At minimum, configure antivirus software to whitelist the OpenDJ server database files. By default, exclude the following file system directories from virus scanning:

- `/path/to/opendj/changelogDb/` (if replication is enabled)

  Prevent the antivirus software from scanning these changelog database files.

- `/path/to/opendj/db/`

  Prevent the antivirus software from scanning database files, especially `*.jdb` files.

Port blocking

Antivirus and intrusion detection software can block ports that OpenDJ uses to provide directory services.

Make sure that your software does not block the ports that OpenDJ software uses. For details, see "Limiting System and Administrative Access" in the *Security Guide*.

Negative performance impact

Antivirus software consumes system resources, reducing resources available to other services including OpenDJ servers.

Running antivirus software can therefore have a significant negative impact on OpenDJ server performance. Make sure that you test and account for the performance impact of running antivirus software before deploying OpenDJ software on the same systems.

1.4. Preparing the Java Environment

ForgeRock Directory Services software consists of pure Java applications. ForgeRock Directory Services servers and clients run on any system with full Java support. ForgeRock Directory Services is tested on a variety of operating systems, and supported on those listed in "Choosing an Operating System".

ForgeRock Directory Services software requires Java 7 or 8, specifically at least the Java Standard Edition runtime environment, or the corresponding Java Development Kit to compile Java plugins and applications.

Note

ForgeRock validates ForgeRock Directory Services software with OpenJDK and Oracle JDK, and does occasionally run sanity tests with other JDKs such as the IBM JDK and Azul's Zulu. Support for very specific Java and hardware combinations is best-effort. This means that if you encounter an issue when using a particular JVM/hardware combination, you must also demonstrate the problem on a system that is widespread and easily tested by any member of the community.

ForgeRock recommends that you keep your Java installation up-to-date with the latest security fixes.

Make sure you have a required Java environment installed on the system. If your default Java environment is not appropriate, set `OPENDJ_JAVA_HOME` to the path to the correct Java environment, or set `OPENDJ_JAVA_BIN` to the absolute path of the **java** command. The `OPENDJ_JAVA_BIN` environment variable is useful if you have both 32-bit and 64-bit versions of the Java environment installed, and want to make sure you use the 64-bit version.

1.5. Choosing an Application Server

OpenDJ servers run as standalone Java services, and do not depend on an application server.

The REST to LDAP and DSML gateway applications run on Apache Tomcat and Jetty.

ForgeRock supports only stable application container releases. See the Tomcat and Jetty documentation for details about the right container to use with your Java environment.

1.6. Assigning FQDNs For Replication

ForgeRock Directory Services replication requires use of fully qualified domain names, such as `opendj.example.com`.

Host names like `my-laptop.local` are acceptable for evaluation. In production, and when using replication across systems, you must either ensure DNS is set up correctly to provide fully qualified domain names, or update the hosts file (`/etc/hosts` or `C:\Windows\System32\drivers\etc\hosts`) to supply unique, fully qualified domain names.

1.7. Getting Digital Certificates Signed

If you plan to configure SSL or TLS to secure network communications between the server and client applications, install a properly signed digital certificate that your client applications recognize, such as one that works with your organization's PKI or one signed by a recognized certificate authority.

To use the certificate during installation, the certificate must be located in a file-based keystore supported by the JVM (JKS, JCEKS, PKCS#12), or on a PKCS#11 token. To import a signed certificate into a keystore, use the Java **keytool** command.

For details, see "Preparing For Secure Communications" in the *Administration Guide*.

1.8. Special Requests

If you have a special request regarding support for a combination not listed here, contact ForgeRock at info@forgerock.com.

Chapter 2
Installing Server Software Files

Follow the appropriate procedure for your operating system:

- "To Unpack Server Software From the Cross-Platform Zip"
- "To Install Server Software From the Debian Package"
- "To Install Server Software From the RPM Package"
- "To Install Server Software With the Windows Installer Package"

To Unpack Server Software From the Cross-Platform Zip

You can use the .zip delivery on any supported operating system.

Installation is a multi-stage process. You unpack the server software with the **unzip** command. You set up the server with the **setup** command:

1. Prepare for installation as described in "*Before You Install*".

2. Unpack the cross-platform .zip file in the file system directory where you want to install the server.

   The **setup** command, described in setup(1), uses the directory where you unzipped the files as the installation directory, and does not ask you where to install the server. If you want to install elsewhere on the file system, unzip the files in that location.

   Unzipping the .zip file creates a top-level `opendj` directory in the directory where you unzipped the file. On Windows systems if you unzip the file with Right-Click > Extract All, remove the trailing `opendj-4.0.0` directory from the folder you specify.

3. Run the **setup** command to set up the server.

To Install Server Software From the Debian Package

On Debian and related Linux distributions such as Ubuntu, you can install server software from the Debian package.

Installation is a multi-stage process. You install the software using the system package manager. You set up the server with the **setup** command:

1. Prepare for installation as described in "*Before You Install*".

   In particular, install a Java runtime environment if none is installed yet:

   ```
   $ sudo apt-get install default-jre
   ```

2. Install the server package:

   ```
   $ sudo dpkg -i DS*.deb
   ```

   The Debian package installs server files in the `/opt/opendj` directory, generates service management scripts, adds documentation files under `/usr/share/doc/opendj`, and adds man pages under `/opt/opendj/share/man`.

   The files are owned by root by default, making it easier to have the server listen on ports such as 389 and 636.

3. Set up the server with the **setup** command, **sudo /opt/opendj/setup**.

To Install Server Software From the RPM Package

On Red Hat and related Linux distributions such as Fedora and CentOS, you can install server software from the RPM package.

Installation is a multi-stage process. You install the software using the system package manager. You set up the server with the **setup** command:

1. Prepare for installation as described in "*Before You Install*".

   In particular, install a Java runtime environment if none is installed yet:

   You might need to download an RPM to install the Java runtime environment, and then install the RPM by using the **rpm** command:

   ```
   $ su
   Password:
   # rpm -ivh jre-*.rpm
   ```

2. Install the server package:

   ```
   # rpm -i DS*.rpm
   ```

   The RPM package installs server files in the `/opt/opendj` directory, generates service management scripts, and adds man pages under `/opt/opendj/share/man`.

The files are owned by root by default, making it easier to have the server listen on ports such as 389 and 636.

3. Set up the server with the **setup** command, **/opt/opendj/setup**.

   By default, the server starts in run levels 2, 3, 4, and 5.

To Install Server Software With the Windows Installer Package

You can install server software on Windows using the .msi installer package.

Installation is a multi-stage process. You install the software with the Windows installer package wizard. You set up the server with the **setup** command:

1. Prepare for installation as described in "*Before You Install*".

   Prevent antivirus and intrusion detection systems from interfering with OpenDJ software.

   Before using OpenDJ software with antivirus or intrusion detection software, consider the following potential problems:

   Interference with normal file access

   Antivirus and intrusion detection systems that perform virus scanning, sweep scanning, or deep file inspection are not compatible with OpenDJ file access, particularly database file access.

   Antivirus and intrusion detection software can interfere with the normal process of opening and closing database working files. They may incorrectly mark such files as suspect to infection due to normal database processing, which involves opening and closing files in line with the database's internal logic.

   Prevent antivirus and intrusion detection systems from scanning database and changelog database files.

   At minimum, configure antivirus software to whitelist the OpenDJ server database files. By default, exclude the following file system directories from virus scanning:

   - `/path/to/opendj/changelogDb/` (if replication is enabled)

     Prevent the antivirus software from scanning these changelog database files.

   - `/path/to/opendj/db/`

     Prevent the antivirus software from scanning database files, especially `*.jdb` files.

Port blocking

Antivirus and intrusion detection software can block ports that OpenDJ uses to provide directory services.

Make sure that your software does not block the ports that OpenDJ software uses. For details, see "Limiting System and Administrative Access" in the *Security Guide*.

Negative performance impact

Antivirus software consumes system resources, reducing resources available to other services including OpenDJ servers.

Running antivirus software can therefore have a significant negative impact on OpenDJ server performance. Make sure that you test and account for the performance impact of running antivirus software before deploying OpenDJ software on the same systems.

2. Install the server files in one of the following ways:

   - Install using the MSI package:

     a. Double-click the Windows installer package, `DS-5.0.0.msi`, to start the install wizard.

     b. In the Destination Folder screen, set the folder where the wizard installs the server files.

        The default location is under Program Files on the system drive. For example, if the system drive is C:, the default location is `C:\Program Files (x86)\opendj\`, as the native executable is a 32-bit application, though you can run the server in a 64-bit Java environment.

   - Use the Microsoft **msiexec.exe** command to install the files.

     The following example installs the server files under `C:\opendj-4.0.0`, writing an installation log file, `install.log`, in the current folder:

     ```
     C:\>msiexec /i DS-5.0.0.msi /l* install.log /q OPENDJ=C:\opendj-4.0.0
     ```

3. Start the installation.

   When installation is finished, the server files are found in the location you specified as Destination Folder. You must still run the **setup** command before you can use the server.

4. Run the **setup** command to set up the server.

Chapter 3

Installing a Directory Server

This chapter covers installation of *directory servers*. Directory servers store local copies of user data, and can be replicas of other directory servers.

Directory servers can be protected from directory client access by *directory proxy servers*. Directory proxy servers hide the implementation details of a directory server deployment from client applications. For details on installing a standalone directory proxy server, see "*Installing a Directory Proxy Server*".

Directory server replicas send updates to and receive updates from *replication servers*, which are servers that do not store user data, but instead are dedicated to transmitting replication messages. A directory server can run a local replication server in the same process. Alternatively, it can connect to a replication server running in another process, either on the same system or on a remote system. For details on installing a standalone replication server, see "*Installing a Replication Server*".

3.1. Setting Up a Directory Server

Use the setup(1) command-line tool. When used without subcommands or options, the command is interactive. For hints about setup choices, see "Directory Server Setup Parameters".

When performing a non-interactive, silent installation, specify at least all mandatory options as part of the command.

The following options are mandatory.

If you use only these options, the command sets up a server listening only on an administration port. The administration port is protected by a key pair generated at setup time with a self-signed certificate:

- `--adminConnectorPort {port}` (conventional port number: 4444)
- `--hostname {hostname}`
- `--rootUserDN {rootUserDN}` (default: `cn=Directory Manager`)
- `--rootUserPassword {rootUserPassword}`

To Set Up a Directory Server

After installing the server files as described in "*Installing Server Software Files*", follow these steps:

1. Run the **setup directory-server** command.

   The command is located where you installed the files, `/path/to/opendj/setup`.

   When setting up a directory server, you can optionally omit the **directory-server** subcommand.

   The following example shows non-interactive setup for evaluation. The example imports sample data from Example.ldif. The server listens for requests on the ports used in examples throughout the documentation. It uses a generated key pair and self-signed certificate when negotiating secure connections:

```
# Set up a directory server for evaluation.
$ /path/to/opendj/setup \
 directory-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPassword password \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpPort 8080 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --baseDN dc=example,dc=com \
 --ldifFile Example.ldif \
 --acceptLicense
```

   The following example shows non-interactive setup for production. This example creates a base DN but does not import LDIF at setup time. It activates only secure traffic for HTTPS. It uses an existing key pair, rather than a generated key pair with a self-signed certificate:

```
# Set up a directory server for production.
$ /path/to/opendj/setup \
 directory-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPasswordFile /tmp/pwd.txt \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --certNickname server-cert \
 --usePkcs12keyStore /path/to/keystore.p12 \
 --keyStorePasswordFile /tmp/keystore.pin \
 --enableStartTLS \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --baseDN dc=example,dc=com \
 --addBaseEntry \
 --productionMode \
 --acceptLicense
```

2. (Optional) Run the **status** command to review the configuration:

```
$ /path/to/opendj/bin/status
```

Directory Server Setup Parameters

Parameter	Description
Type of server	A directory server holds user data. A proxy server forwards requests to remote directory servers.
Instance path	Server setup uses tools and templates installed with the software to generate the instance files required to run an instance of a server. By default, all the files are co-located. This parameter lets you separate the files. Set the instance path to place generated files in a different location from the tools, templates, and libraries you installed. Interactive setup suggests co-locating the software with the instance files. You cannot use a single software installation for multiple servers. Tools for starting and stopping the server process, for example, work with a single configured server. They do not have a mechanism to specify an alternate server location. If you want to set up another server, install another copy of the software, and run that copy's **setup** command.
Root user DN	The root user DN identifies a user who can perform all operations allowed for the server, called root user due to the similarity to the UNIX root user. The name used in the documentation is the default name: `cn=Directory Manager`. For additional security in production environments, use a different name.
Root user password	The root user authenticates with simple, password-based authentication. Use a strong password here unless this server is only for evaluation.
Harden for production use	By opting to harden the server configuration for production, you increase security. The primary cost of increased security is that evaluating the software and demonstrating features can require additional configuration. For that reason, examples in the documentation assume you do not use this option. Setting up a server in hardened production mode leads to the following settings: • The default backend database for directory servers, `userRoot`, uses data confidentiality to encrypt potentially sensitive data on disk. As described in "Encrypting Directory Data" in the *Administration Guide*, if you configure data confidentiality before replication, the destination server's keys disappear when you configure replication. The destination server can no longer decrypt any of its data. (When you initialize replication, a source server

Parameter	Description
	initializes a destination server. The **dsreplication initialize** command references the destination server with its `--hostDestination` and `--portDestination` options.) When you install multiple directory servers that you will configure as replicas, either import data after installation, or import data for only one replica that you then use to initialize other replicas. In the latter case, leave the database empty for all other replicas. • Global access control allows only the following access: • Anonymous users can request the StartTLS extended operation, and the Get Symmetric Key extended operation. The Get Symmetric Key extended operation is an OpenDJ-specific operation for internal use. An OpenDJ server requires Get Symmetric Key extended operation access to create and share secret keys for encryption. • Anonymous users can read the root DSE operational attributes that describe server capabilities, including among other information, what security protocols and cipher suites the server supports. • Authenticated users can read the LDAP directory schema. • Authenticated users can request the LDAP Password Modify extended operation, the Who am I? extended operation, and the Cancel extended operation. • Authenticated users can request the Pre-Read and Post-Read controls, the Subtree Delete control, and the Permissive Modify control. These controls are used by the REST to LDAP gateway. Authenticated users can also request the ForgeRock Transaction ID control. This is a ForgeRock-specific control for internal use that permits transmission of transaction IDs through platform components for use as a key to correlation of Common Audit events. For a longer explanation of these settings, see "Reconsider Default Global Access Control" in the *Security Guide*. • The protocol version and cipher suites for securing connections are restricted to those using strong encryption. The protocol version is restricted to `TLSv1.2`. The cipher suites used when negotiating a secure connection call for a server certificate using an elliptic curve (EC) key algorithm or an RSA key algorithm. If you provide your own keystore when setting up the server in production mode, make sure that the certificate key algorithm is EC or RSA. Otherwise the server will not be able to negotiate secure connections. For details and examples, see "To Restrict Protocols and Cipher Suites" in the *Security Guide*. • The Crypto Manager requires encrypted communication between servers.

Parameter	Description
	The Crypto Manager is described in "Cryptographic Key Management" in the *Security Guide*. - The anonymous HTTP authorization mechanism for REST access is disabled. As a result, REST access does not permit anonymous requests. - OpenDJ native file-based access loggers and the replication error logger have UNIX/Linux file permissions set to `600` (only the server account has read-write access to log files). This setting does not affect Common Audit loggers, such as the JSON file-based audit loggers. Adjust system settings to ensure appropriate access to files. For additional information and recommendations on setting the UNIX/Linux `umask` appropriately and on setting ACLs on Windows systems, see "Setting Appropriate File Permissions" in the *Security Guide*. - The random password generator generates 10-character alphanumeric passwords. - The default password policy for normal users requires passwords at least 8 characters in length, and prevents use of common passwords. The default password policy for root DN users requires passwords at least 8 characters in length, prevents use of common passwords, and requires that authentication be secure to avoid exposing credentials over the network. - The CRAM-MD5 and DIGEST-MD5 SASL mechanisms are disabled.
Fully qualified directory server host name	The server uses the fully qualified host name in self-signed certificates and for identification between replicated servers. Interactive setup suggests the hostname of the local host. If this server is only for evaluation, then you can use an FQDN such as `laptop.local`. Otherwise, use an FQDN that other hosts can resolve to reach your server, and that matches the FQDN in the server certificate.
Administration port	This is the service port used to configure the server and to run tasks. The port used in the documentation is 4444, which is the initial port suggested during interactive setup. If the suggested port is not free, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found.
Start the server	If you do not start the server during setup, use the **/path/to/opendj/bin/start-ds** command later.
Keystore for securing connections	Setup requires a keystore with the keys for securing connections to the administration port, and to any other secure ports you configure during setup.

Parameter	Description
	You can choose to use an existing keystore supported by the JVM, which can be either a file-based keystore or a PKCS#11 token. The existing keystore must protect the keystore and all private keys with the same PIN or password. If you choose a PKCS#11 token, you must first configure access through the JVM, as the only input to the **setup** command is the PIN. If you do not have an existing keystore, the **setup** command can generate a key pair in a new PKCS#12 keystore, and self-sign the public key certificate. This is the default choice during interactive setup. Other applications will not recognize self-signed certificates unless they have explicitly trusted the certificate. For example, you import the certificate into the application's truststore, or supply a copy at runtime as a CA certificate parameter. Public key security is often misunderstood. Before making security choices for production systems, read "*Managing Certificates and Private Keys*" in the *Security Guide*.
LDAP and LDAPS port	The reserved port for LDAP is 389. The reserved port for LDAPS is 636. Examples in the documentation use 1389 and 1636, which are accessible to non-privileged users. If you install the server with access to privileged ports (< 1024), and the reserved port is not yet in use, then interactive setup suggests the reserved port number. If the port is not free or cannot be used due to lack of privileges, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found. The LDAP StartTLS extended operation is a standard operation to negotiate a secure connection starting on the cleartext LDAP port.
HTTP and HTTPS ports	The reserved port for HTTP is 80. The reserved port for HTTPS is 443. The interactive setup initially suggests 8080 and 8443 instead. If the initially suggested port is not free or cannot be used due to lack of privileges, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found. Examples in the documentation use 8080 and 8443. When you enable HTTP or HTTPS at setup time, only the administrative endpoints are enabled, `/admin/config` and `/admin/monitor`. For access to user data, see "To Set Up REST Access to User Data" in the *Administration Guide*.
Prepare data storage	One of the choices when setting up a directory server is whether to prepare for and optionally add data during setup, or to handle data storage as a separate, post-setup step. You have several options for adding directory data: • Leave the database empty if you want create backend databases and import directory data separately, after completing the setup process. For details, see "*Managing Directory Data*" in the *Administration Guide*.

Parameter	Description
	• Create only the base DN entry if you want to prepare a backend database, but to load directory data separately, after completing the setup process. A base DN, such as `dc=example,dc=com`, is the DN suffix shared by all DNs in your directory data. If the concept of base DN is new to you, briefly read "About Data In LDAP Directories" in the *Administration Guide*. Before adding directory data, you must create at least one base DN. If the directory data belongs in more than one suffix, use non-interactive mode to create multiple base DNs, or load some of the data after completing the setup process. When you choose to create a base DN entry, and therefore to create a data storage backend, interactive mode can present a choice of data storage types. If you are not sure which type to choose, briefly read "About Database Backends" in the *Administration Guide*. • Import data from an LDIF file if you already have data in LDIF and you want to load directory data as part of the setup process. LDAP data interchange format (LDIF) is the standard text format for expressing LDAP data. The documentation mainly uses sample data from . If you have LDIF already, but the data uses attributes or object classes not defined in the default schema, choose to leave the database empty, or to create a base DN entry during setup. After setup, add schema definitions as described in "*Managing Schema*" in the *Administration Guide*, and then import the data from LDIF. When you choose to import LDIF, and therefore to create a data storage backend, interactive mode can present a choice of data storage types. If you are not sure which type to choose, briefly read "About Database Backends" in the *Administration Guide*. • Load automatically-generated sample data for testing or evaluation. This option lets you have the **setup** command generate an arbitrarily large number of similar user entries. Each user entry has a `uid` RDN like `user.`*`number`*. Each user entry's password is `password`. When you choose to load generated entries, and therefore to create a data storage backend, interactive mode can present a choice of data storage types. If you are not sure which type to choose, briefly read "About Database Backends" in the *Administration Guide*.

Chapter 4

Installing a Directory Proxy Server

This chapter covers installation of standalone *directory proxy servers*. A standalone directory proxy server forwards LDAP requests for user data to remote directory servers. Directory proxy servers make it possible to provide a single point of access to a directory service, and to hide implementation details from client applications.

Unlike standalone directory proxy servers, *directory servers* store local copies of user data, and can replicate that data with other directory servers. For details on installing a directory server, see "*Installing a Directory Server*".

4.1. Setting Up a Directory Proxy Server

Use the setup(1) command-line tool. When used without subcommands or options, the command is interactive. For hints about setup choices, see "Directory Proxy Server Setup Parameters".

When performing a non-interactive, silent installation, specify at least all mandatory options as part of the command.

The following options are mandatory.

If you use only these options, the command sets up a server listening only on an administration port. The administration port is protected by a key pair generated at setup time with a self-signed certificate:

- `--adminConnectorPort {port}` (conventional port number: 4444)
- `--hostname {hostname}`
- `--rootUserDN {rootUserDN}` (default: `cn=Directory Manager`)
- `--rootUserPassword {rootUserPassword}`

To Set Up a Directory Proxy Server

After installing the server files as described in "*Installing Server Software Files*", follow these steps:

1. If you have not already done so, create an account for the proxy in the remote directory service.

   This account is used to connect to the remote directory service. The directory proxy server binds with this account, and forwards LDAP requests on behalf of other users. The account must have the same bind DN and bind password on all remote directory servers.

The account must have the right to perform proxied authorization on the remote directory service. When using OpenDJ directory services, see "Configuring Proxied Authorization" in the *Developer's Guide* for details. Otherwise, read about how to set up proxied authorization in your directory server documentation.

The examples in this procedure use the account with bind DN `cn=Proxy,ou=Apps,dc=example,dc=com` and bind password `password`.

2. Run the **setup proxy-server** command.

   The command is located where you installed the files, `/path/to/opendj/setup`.

   The following example sets up a directory proxy server that discovers remote servers by connecting to a replication server. It forwards all requests to public naming contexts of remote servers. (Generally this means requests targeting user data, as opposed to the proxy's configuration, schema, or monitoring statistics.) It uses the least requests load balancing algorithm:

   ```
   $ /path/to/opendj/setup \
    proxy-server \
    --rootUserDN "cn=Directory Manager" \
    --rootUserPassword password \
    --hostname opendj.example.com \
    --ldapPort 1389 \
    --ldapsPort 1636 \
    --adminConnectorPort 4444 \
    --replicationServer rs.example.com:4444 \
    --replicationBindDN "uid=admin,cn=Administrators,cn=admin data" \
    --replicationBindPassword password \
    --proxyUserBindDN cn=Proxy,ou=Apps,dc=example,dc=com \
    --proxyUserBindPassword password \
    --proxyUsingStartTLS \
    --useJvmTrustStore \
    --acceptLicense
   ```

   If you are just trying out the software, you can use the `--trustAll` option. Do not use this option in production environments, however.

   The following example sets up a directory proxy server that has a static list of remote servers to connect to. It forwards only requests targeting `dc=example,dc=com`. It uses the default affinity load balancing algorithm:

```
$ /path/to/opendj/setup \
 proxy-server \
 --rootUserDN "cn=Directory Manager" \
 --rootUserPassword password \
 --hostname opendj.example.com \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --adminConnectorPort 4444 \
 --staticPrimaryServer local-data-center-ldap1.example.com:636 \
 --staticPrimaryServer local-data-center-ldap2.example.com:636 \
 --staticSecondaryServer remote-data-center-ldap1.example.com:636 \
 --staticSecondaryServer remote-data-center-ldap2.example.com:636 \
 --baseDN dc=example,dc=com \
 --proxyUserBindDN cn=Proxy,ou=Apps,dc=example,dc=com \
 --proxyUserBindPassword password \
 --proxyUsingSSL \
 --useJvmTrustStore \
 --acceptLicense
```

When you set up a directory proxy server, access control is implemented using global access control policy entries, rather than global ACIs. For more information about global access control policies, see "About Global Access Control Policies" in the *Administration Guide*.

3. (Optional) Run the **status** command to review the configuration:

```
$ /path/to/opendj/bin/status
```

4. If the LDAP schema differ on the directory servers and the proxy server, align the LDAP schema of the proxy server with the LDAP schema of the remote directory servers.

   For more information, see "*Managing Schema*" in the *Administration Guide*.

For more information, see "*Configuring LDAP Proxy Services*" in the *Administration Guide*.

Directory Proxy Server Setup Parameters

Parameter	Description
Type of server	A directory server holds user data. A proxy server forwards requests to remote directory servers.
Instance path	Server setup uses tools and templates installed with the software to generate the instance files required to run an instance of a server. By default, all the files are co-located. This parameter lets you separate the files. Set the instance path to place generated files in a different location from the tools, templates, and libraries you installed. Interactive setup suggests co-locating the software with the instance files.

Parameter	Description
	You cannot use a single software installation for multiple servers. Tools for starting and stopping the server process, for example, work with a single configured server. They do not have a mechanism to specify an alternate server location. If you want to set up another server, install another copy of the software, and run that copy's **setup** command.
Root user DN	The root user DN identifies a user who can perform all operations allowed for the server, called root user due to the similarity to the UNIX root user. The name used in the documentation is the default name: `cn=Directory Manager`. For additional security in production environments, use a different name.
Root user password	The root user authenticates with simple, password-based authentication. Use a strong password here unless this server is only for evaluation.
Harden for production use	By opting to harden the server configuration for production, you increase security. The primary cost of increased security is that evaluating the software and demonstrating features can require additional configuration. For that reason, examples in the documentation assume you do not use this option. Setting up a server in hardened production mode leads to the following settings: • The default backend database for directory servers, `userRoot`, uses data confidentiality to encrypt potentially sensitive data on disk. As described in "Encrypting Directory Data" in the *Administration Guide*, if you configure data confidentiality before replication, the destination server's keys disappear when you configure replication. The destination server can no longer decrypt any of its data. (When you initialize replication, a source server initializes a destination server. The **dsreplication initialize** command references the destination server with its `--hostDestination` and `--portDestination` options.) When you install multiple directory servers that you will configure as replicas, either import data after installation, or import data for only one replica that you then use to initialize other replicas. In the latter case, leave the database empty for all other replicas. • Global access control allows only the following access: • Anonymous users can request the StartTLS extended operation, and the Get Symmetric Key extended operation. The Get Symmetric Key extended operation is an OpenDJ-specific operation for internal use. An OpenDJ server requires Get Symmetric Key extended operation access to create and share secret keys for encryption. • Anonymous users can read the root DSE operational attributes that describe server capabilities, including among other information, what security protocols and cipher suites the server supports. • Authenticated users can read the LDAP directory schema.

Parameter	Description
	• Authenticated users can request the LDAP Password Modify extended operation, the Who am I? extended operation, and the Cancel extended operation. • Authenticated users can request the Pre-Read and Post-Read controls, the Subtree Delete control, and the Permissive Modify control. These controls are used by the REST to LDAP gateway. Authenticated users can also request the ForgeRock Transaction ID control. This is a ForgeRock-specific control for internal use that permits transmission of transaction IDs through platform components for use as a key to correlation of Common Audit events. For a longer explanation of these settings, see "Reconsider Default Global Access Control" in the *Security Guide*. • The protocol version and cipher suites for securing connections are restricted to those using strong encryption. The protocol version is restricted to `TLSv1.2`. The cipher suites used when negotiating a secure connection call for a server certificate using an elliptic curve (EC) key algorithm or an RSA key algorithm. If you provide your own keystore when setting up the server in production mode, make sure that the certificate key algorithm is EC or RSA. Otherwise the server will not be able to negotiate secure connections. For details and examples, see "To Restrict Protocols and Cipher Suites" in the *Security Guide*. • The Crypto Manager requires encrypted communication between servers. The Crypto Manager is described in "Cryptographic Key Management" in the *Security Guide*. • The anonymous HTTP authorization mechanism for REST access is disabled. As a result, REST access does not permit anonymous requests. • OpenDJ native file-based access loggers and the replication error logger have UNIX/Linux file permissions set to `600` (only the server account has read-write access to log files). This setting does not affect Common Audit loggers, such as the JSON file-based audit loggers. Adjust system settings to ensure appropriate access to files. For additional information and recommendations on setting the UNIX/Linux `umask` appropriately and on setting ACLs on Windows systems, see "Setting Appropriate File Permissions" in the *Security Guide*. • The random password generator generates 10-character alphanumeric passwords. • The default password policy for normal users requires passwords at least 8 characters in length, and prevents use of common passwords.

Parameter	Description
	The default password policy for root DN users requires passwords at least 8 characters in length, prevents use of common passwords, and requires that authentication be secure to avoid exposing credentials over the network. - The CRAM-MD5 and DIGEST-MD5 SASL mechanisms are disabled.
Fully qualified directory server host name	The server uses the fully qualified host name in self-signed certificates and for identification between replicated servers. Interactive setup suggests the hostname of the local host. If this server is only for evaluation, then you can use an FQDN such as `laptop.local`. Otherwise, use an FQDN that other hosts can resolve to reach your server, and that matches the FQDN in the server certificate.
Administration port	This is the service port used to configure the server and to run tasks. The port used in the documentation is 4444, which is the initial port suggested during interactive setup. If the suggested port is not free, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found.
Start the server	If you do not start the server during setup, use the **/path/to/opendj/bin/start-ds** command later.
Keystore for securing connections	Setup requires a keystore with the keys for securing connections to the administration port, and to any other secure ports you configure during setup. You can choose to use an existing keystore supported by the JVM, which can be either a file-based keystore or a PKCS#11 token. The existing keystore must protect the keystore and all private keys with the same PIN or password. If you choose a PKCS#11 token, you must first configure access through the JVM, as the only input to the **setup** command is the PIN. If you do not have an existing keystore, the **setup** command can generate a key pair in a new PKCS#12 keystore, and self-sign the public key certificate. This is the default choice during interactive setup. Other applications will not recognize self-signed certificates unless they have explicitly trusted the certificate. For example, you import the certificate into the application's truststore, or supply a copy at runtime as a CA certificate parameter. Public key security is often misunderstood. Before making security choices for production systems, read "*Managing Certificates and Private Keys*" in the *Security Guide*.
LDAP and LDAPS port	The reserved port for LDAP is 389. The reserved port for LDAPS is 636. Examples in the documentation use 1389 and 1636, which are accessible to non-privileged users.

Parameter	Description
	If you install the server with access to privileged ports (< 1024), and the reserved port is not yet in use, then interactive setup suggests the reserved port number. If the port is not free or cannot be used due to lack of privileges, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found. The LDAP StartTLS extended operation is a standard operation to negotiate a secure connection starting on the cleartext LDAP port.
HTTP and HTTPS ports	The reserved port for HTTP is 80. The reserved port for HTTPS is 443. The interactive setup initially suggests 8080 and 8443 instead. If the initially suggested port is not free or cannot be used due to lack of privileges, interactive setup adds 1000 to the port number and tries again, repeatedly adding 1000 until a free port is found. Examples in the documentation use 8080 and 8443. When you enable HTTP or HTTPS at setup time, only the administrative endpoints are enabled, `/admin/config` and `/admin/monitor`. For access to user data, see "To Set Up REST Access to User Data" in the *Administration Guide*.
Directory server discovery	A directory proxy server uses a *service discovery mechanism* to discover and connect to remote LDAP directory servers. A service discovery mechanism identifies a set of directory servers that the proxy can forward requests to. When preparing to configure a service discovery mechanism, choose one of these alternatives: **Replication service discovery mechanism** This mechanism contacts OpenDJ replication servers to discover directory servers to forward LDAP requests to. Each replication server maintains information about the replication topology that allows the proxy server to discover directory server replicas. This mechanism only works with replicated OpenDJ servers. A replication service discovery mechanism configuration includes a bind DN and password to connect to replication servers. It uses this account to read configuration data under `cn=admin data` and `cn=config`. The account must have access and privileges to read that configuration data, and it must exist with the same credentials on all replication servers. **Static service discovery mechanism** This mechanism maintains a static list of directory server *`host:port`* combinations. You must enumerate the servers to forward LDAP requests to. This mechanism is designed to work with all LDAPv3 directory servers that support proxied authorization.

Parameter	Description
	All remote directory servers are expected to be equivalent replicas of each other. In distributed deployments, nearby remote directory servers may be set as primary and others as secondary. The proxy attempts first to forward requests to primary servers. If no primary servers are available, then the proxy forwards requests to secondary servers until the primary servers become available again. This is useful, for example, to prevent a proxy from load balancing some requests over WAN links even though directory servers on the LAN are ready to receive requests. For a replication service discovery mechanism, you identify the primary server group by its replication group ID, as described in "Replication Groups" in the *Administration Guide*. For a static service discovery mechanism, you enumerate primary and secondary servers. The connection-level security (SSL, StartTLS) options for the service discover mechanism determine how the proxy secures connections to the remote directory services. Use secure connections in production deployments to avoid sending simple bind (bind DN/password) credentials in cleartext.
Proxy user credentials	A directory proxy server uses a proxy DN and password to connect to remote directory servers, and proxied authorization for forwarded LDAP requests. This proxy account must exist with the same credentials on all remote directory servers, and must be able to use the standard proxied authorization control. For details on proxied authorization and how to configure OpenDJ directory servers to allow it, see "Configuring Proxied Authorization" in the *Developer's Guide*.
LDAP request forwarding	A proxy can forward LDAP requests for all public naming contexts supported by remote servers, or forward only requests targeting specified base DNs. A public naming context is a subtree of user entries held by the directory server such as `dc=example,dc=com`. Public naming contexts generally include user data and exclude operational suffixes such as `cn=config` and `cn=schema`. Public naming contexts are published on a directory server's root DSE. The service discovery mechanism can therefore determine them dynamically.
Load balancing algorithm	When configuring a proxy backend, choose one of these load balancing alternatives: **`affinity`** This load balancing algorithm routes requests with the same target DN to the same server. Affinity load balancing helps when applications update and then reread the same entry in quick succession. With an add or modify request on an entry that is quickly followed by a read of the entry, the request to replicate the update can take longer than the read request, depending on network latency. Affinity load balancing forwards the read request to the same server that processed the update, ensuring that the client obtains the expected result. In terms of the CAP theorem, affinity load balancing provides consistency and availability, but not partition tolerance. As this algorithm lacks partition

Parameter	Description
	tolerance, only use it to load balance requests in environments where partitions are unlikely, such as a single data center with all remote directory servers on the same network. `least-requests` This load balancing algorithm routes requests to the LDAP server with the fewest active requests from the current directory proxy server. Least requests load balancing helps to spread requests equitably across a pool of replicated servers. In terms of the CAP theorem, least requests load balancing provides availability and partition tolerance, but not consistency. A write request followed by a read request of the same entry might be forwarded to different remote directory servers.

Chapter 5

Installing a Replication Server

This chapter covers installation of standalone *replication servers*. Replication servers do not serve user data, but instead are dedicated to transmitting replication messages.

Directory server replicas send updates to and receive updates from replication servers. As an alternative to having standalone replication servers, a directory server can run a local replication server in the same process. For details on installing a directory server, see "*Installing a Directory Server*".

5.1. Setting Up a Replication Server

Follow the appropriate procedures to set up and initially configure the replication server.

To Create Standalone Servers

As described in "*Managing Data Replication*" in the *Administration Guide*, some deployments require standalone directory servers and standalone replication servers. A standalone directory server is one that does not relay replication messages, but only stores data. A standalone replication server is one that only relays replication messages, and does not store user data.

After installing the server files for two standalone servers, as described in "*Installing Server Software Files*", follow these steps:

1. Set up a server to be a standalone directory server.

   The following example sets up a standalone directory server:

   ```
   $ /path/to/opendj/setup \
    directory-server \
    --rootUserDN "cn=Directory Manager" \
    --rootUserPassword password \
    --hostname ds-only.example.com \
    --ldapPort 1389 \
    --ldapsPort 1636 \
    --httpPort 8080 \
    --httpsPort 8443 \
    --adminConnectorPort 4444 \
    --baseDN dc=example,dc=com \
    --ldifFile Example.ldif \
    --acceptLicense
   ```

2. Set up another server to be a standalone replication server.

   Notice that none of the user data-related options—`--backendType`, `--baseDN`, `--ldifFile`—and no user access ports are supplied in the following example:

   ```
   $ /setup \
    --rootUserDN "cn=Directory Manager" \
    --rootUserPassword password \
    --hostname rs-only.example.com \
    --adminConnectorPort 4444 \
    --acceptLicense
   ```

3. Configure replication between the standalone directory server and the standalone replication server.

   Notice in the following example that the standalone directory server is not a replication server (`--noReplicationServer1`), and has no replication port. Also notice that the standalone replication server is only a replication server (`--onlyReplicationServer2`):

   ```
   $ /path/to/opendj/bin/dsreplication \
    configure \
    --adminUID admin \
    --adminPassword password \
    --baseDN dc=example,dc=com \
    --host1 ds-only.example.com \
    --port1 4444 \
    --bindDN1 "cn=Directory Manager" \
    --bindPassword1 password \
    --noReplicationServer1 \
    --host2 rs-only.example.com \
    --port2 4444 \
    --bindDN2 "cn=Directory Manager" \
    --bindPassword2 password \
    --replicationPort2 8989 \
    --onlyReplicationServer2 \
    --trustAll \
    --no-prompt
   ```

   In order to initialize replication, you must first add at least one more directory server to the topology. Otherwise, there is nowhere to replicate the data, because the standalone replication server has no user data backend.

   For details on configuring data replication, see "*Managing Data Replication*" in the *Administration Guide*.

Chapter 6
Installing the REST to LDAP Gateway

This chapter explains how to install the REST to LDAP gateway web application.

To Install the REST to LDAP Gateway

The REST to LDAP gateway functions as a web application in a web application container. The REST to LDAP gateway runs independently of the LDAPv3 directory service. As an alternative to the gateway, you can configure HTTP access to a directory server as described in "To Set Up REST Access to User Data" in the *Administration Guide*.

You configure the gateway to access your directory service by editing configuration files in the deployed web application:

`WEB-INF/classes/config.json`

> This file defines how the gateway connects to LDAP directory servers, and how user identities extracted from HTTP requests map to LDAP user identities.
>
> For details, see "Gateway Configuration File" in the *Reference*.

`WEB-INF/classes/logging.properties`

> This file defines logging properties, and can be used when the gateway runs in Apache Tomcat.

`WEB-INF/classes/rest2ldap/rest2ldap.json`

> This file defines which LDAP features the gateway uses.
>
> For details, see "Gateway REST2LDAP Configuration File" in the *Reference*.

`WEB-INF/classes/rest2ldap/endpoints/api/example-v1.json`

> This file defines JSON resource to LDAP entry mappings.
>
> You can edit this file, and define additional files for alternative APIs and versions of APIs. For details, see "Mapping Configuration File" in the *Reference*.

Follow these steps to install the REST to LDAP gateway:

1. Prepare for installation as described in "*Before You Install*".

2. Deploy the .war file according to the instructions for your application server.

3. Edit the configuration files in the deployed gateway web application.

   At minimum adjust the following configuration settings in `WEB-INF/classes/config.json`:

   - `primaryLDAPServers`: Set the correct directory server host names and port numbers.
   - `authentication`: Set the correct simple bind credentials.

     The LDAP account used to authenticate needs to perform proxied authorization as described in "Configuring Proxied Authorization" in the *Developer's Guide*.

   The default sample configuration works with generated example data or with the sample content in Example.ldif. If your data is different, then you must also change the JSON resource to LDAP entry mapping settings, described in "Mapping Configuration File" in the *Reference*.

   For details regarding the configuration, see "*REST to LDAP Configuration*" in the *Reference*.

   When connecting to a directory service over LDAPS or LDAP and StartTLS, you can configure the trust manager to use a file-based truststore for server certificates that the gateway should trust. This allows the gateway to validate server certificates signed, for example, by a certificate authority that is not recognized by the Java environment when setting up LDAPS or StartTLS connections. See "Preparing For Secure Communications" in the *Administration Guide* for an example of how to use the Java **keytool** command to import a server certificate into a truststore file.

4. (Optional) If necessary, adjust the log level.

   Log levels are defined in java.util.logging.Level.

   By default, the log level is set to `INFO`, and the gateway logs HTTP request-related messages. To have the gateway log LDAP request-related messages, set the log level to `FINEST` in one of the following ways:

   - If the REST to LDAP gateway runs in Apache Tomcat, edit `WEB-INF/classes/logging.properties` to set `org.forgerock.opendj.rest2ldap.level = FINEST`. For details on Tomcat's implementation of the logging API, see *Logging in Tomcat*.

     Messages are written to `CATALINA_BASE/logs/rest2ldap.yyyy-MM-dd.log`.

   - If the REST to LDAP gateway runs in Jetty, make sure you set the log level system property when starting Jetty: `-Dorg.forgerock.opendj.rest2ldap.level=FINEST`.

     Messages are written to the Jetty log.

5. Restart the REST to LDAP gateway or the application server to make sure the configuration changes are taken into account.

6. Make sure that the directory service is up, and then check that the gateway is connecting correctly.

The following command reads Babs Jensen's entry through the gateway to a directory server holding data from Example.ldif. In this example, the gateway is deployed under `/rest2ldap`:

```
$ curl \
 --user bjensen:hifalutin
 http://opendj.example.com:8080/rest2ldap/api/users/bjensen?_prettyPrint=true
{
  "_id" : "bjensen",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta" : { },
  "userName" : "bjensen@example.com",
  "displayName" : [ "Barbara Jensen", "Babs Jensen" ],
  "name" : {
    "givenName" : "Barbara",
    "familyName" : "Jensen"
  },
  "description" : "Original description",
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1862",
    "emailAddress" : "bjensen@example.com"
  },
  "uidNumber" : "1076",
  "gidNumber" : "1000",
  "homeDirectory" : "/home/bjensen",
  "manager" : {
    "_id" : "trigden",
    "displayName" : "Torrey Rigden"
  }
}
```

If you generated example data, Babs Jensen's entry is not included. Instead, try a generated user such as `http://user.0:password@opendj.example.com:8080/rest2ldap/api/users/user.0`.

Chapter 7
Installing the DSML Gateway

This chapter explains how to install the DSML gateway web application.

To Install the DSML gateway

The DSML gateway functions as a web application in a web application container. The DSML gateway runs independently of the directory service. You configure the gateway to access a directory service by editing parameters in the gateway configuration file, `WEB-INF/web.xml`:

1. Prepare for installation as described in "*Before You Install*".

2. Deploy the .war file according to the instructions for your application server.

3. Edit `WEB-INF/web.xml` to ensure the parameters are correct.

   At minimum, make sure the correct values are set for `ldap.host` and `ldap.port`.

4. Restart the web application according to the instructions for your application server.

Chapter 8

Before You Upgrade

This chapter lists requirements to fulfill before upgrading ForgeRock Directory Services server software, especially before upgrading the software in a production environment, in addition to requirements listed in "*Before You Install*". It covers the following topics:

- Supported upgrade paths
- Required credentials
- Java upgrades
- Backup files
- Servers that run as Windows services

8.1. Following a Supported Upgrade Path

"Server Upgrade Paths" indicates what you can upgrade.

Server Upgrade Paths

From...	To...	Important Notes
Official ForgeRock release, version 2.4 or 2.5	Official ForgeRock release, directory server or replication server	Not supported. Workaround: First upgrade all servers in the deployment to use at least 2.6.0 before upgrading further. For details on upgrading to that version, see *Upgrading to OpenDJ 2.6.0*.
Official ForgeRock release, version 2.6.0 or later	Official ForgeRock release, same edition of directory server or replication server	Supported.
Official ForgeRock release, OEM edition, version 3.0.0 or later	Official ForgeRock release, directory server or replication server	Supported. The OEM edition did not include Berkeley DB Java Edition, and did not support JE backends. Instead, it uses PDB backends for local data.

From...	To...	Important Notes
		The upgrade process adds support for JE backends. The PDB backend type is deprecated and will be removed in a future release. Change your PDB backends to JE backends as described in "To Move a PDB Backend to a JE Backend".
Trial edition release, version 3.0.0 or later	Official ForgeRock release, directory server or replication server	Supported.
Unofficial build, version 2.6.0 or later	Official ForgeRock release, directory server	Not supported. Workaround: Install the new directory server as a replica of other servers. Use replication to bring the new server up to date before retiring older servers.

8.2. Obtaining the Required Credentials

Perform the upgrade procedure as the user who owns the server files.

Make sure you have the credentials to run commands as this user.

8.3. Upgrading Java

The new server requires a supported Java environment listed in "Preparing the Java Environment".

If the server uses an older version, install a newer Java version before upgrading. To enable the server to use the newer Java version, edit the `default.java-home` setting in the `opendj/config/java.properties` file.

8.4. Backing Up Server Files

Before upgrading, perform a full file system backup of the current server in order to revert if the upgrade fails.

Due to changes to the backup archive format, make sure you stop the directory server and *back up the file system directory where the current server is installed* rather than creating a backup archive with the **backup** command.

8.5. Disabling the Server as a Windows Service

If you are upgrading the server on Windows, and it is registered as a Windows service, disable the server as a Windows service before upgrade, as in the following example:

```
C:\path\to\opendj\bat> windows-service.bat --disableService
```

After upgrade, you can enable the server as a Windows service again.

Chapter 9

Upgrading a Directory Server

This chapter shows how to upgrade a directory server. It includes the following procedures:

- "To Upgrade a Directory Server"
- "To Upgrade Replicated Servers"
- "To Add a New Replica to an Existing Topology"
- "To Move a PDB Backend to a JE Backend"

> **Important**
>
> Failure to follow the upgrade instructions can result in the loss of all user data.

To Upgrade a Directory Server

Follow these steps to upgrade a directory server:

1. Prepare for upgrade as described in "*Before You Upgrade*".
2. Stop the server.
3. Proceed to upgrade the server:
   - When upgrading a server installed from the cross-platform .zip:
     a. Unpack the new files over the old files as described in "*Installing Server Software Files*".
     b. Run the **upgrade** command, described in upgrade(1), to bring the server configuration and, if possible, user data up to date with the new software delivery.

        By default, the **upgrade** command runs interactively, requesting confirmation before making important configuration changes. For some potentially long-duration tasks, such as rebuilding indexes, the default choice is to defer the tasks until after upgrade.

        You can use the `--no-prompt` option to run the command non-interactively. In this case, the `--acceptLicense` option lets you accept the license terms non-interactively.

        When using the `--no-prompt` option, if the **upgrade** command cannot complete because it requires confirmation for a potentially long or critical task, then it exits with an error and a message about how to finish making the changes. You can add the `--force` option to force a non-interactive upgrade to continue in this case, also performing long running and critical tasks.

- When upgrading a server installed from native packages, use the system package management tools.

4. Start the upgraded server.

   At this point the upgrade process is complete. See the resulting `upgrade.log` file for a full list of operations performed.

   Replication updates the upgraded server with changes that occurred during the upgrade process.

   When you upgrade from version 3.0 or earlier, the upgrade process leaves the HTTP connection handler disabled.

   The newer configuration is not compatible with the previous configuration. You must rewrite your configuration according to "*REST to LDAP Configuration*" in the *Reference*, and then configure the server to use the new configuration. For details, see "RESTful Client Access Over HTTP" in the *Administration Guide*.

5. (Optional) If you disabled the server as a Windows service in order to upgrade, enable the server as a Windows service again as in the following example:

   ```
   C:\path\to\opendj\bat> windows-service.bat --enableService
   ```

To Upgrade Replicated Servers

> **Important**
>
> The directory server upgrade process is designed to support a rolling (sequential) upgrade of replicated servers.
>
> Do not upgrade all replicated servers at once in parallel, as this removes all replication changelog data simultaneously, breaking replication.

For each server in the replication topology, follow these steps:

1. Direct client application traffic away from the server to upgrade.

2. Upgrade the server.

3. Direct client application traffic back to the upgraded server.

To Add a New Replica to an Existing Topology

Newer directory servers have updates to LDAP schemas that enable support for new features. The newer schemas are not all compatible with older servers.

When adding a new server to a replication topology with older servers and following the instructions in "Configuring Replication Settings" in the *Administration Guide*, also follow these recommendations:

1. Configure replication using the **dsreplication** command delivered with the new server.
2. Use the `--noSchemaReplication` or the `--useSecondServerAsSchemaSource` option to avoid copying the newer schema to the older server.

   It is acceptable to copy the older schema to the newer server, though it prevents use of new features that depend on newer schema.
3. If applications depend on Internet-Draft change numbers, see "To Align Draft Change Numbers" in the *Administration Guide*.

To Move a PDB Backend to a JE Backend

Although the **dsconfig** command does not provide a way to change a database backend type, it is possible to make the change. The high-level steps are as follows:

1. Stop the directory server.
2. Export the data in the PDB backend to LDIF.
3. Update the configuration to change the backend type.

   See the LDIF modifications shown in "Example Script for Changing a PDB Backend to a JE Backend".
4. Import the data from LDIF into the new JE backend.
5. Start the directory server.

Example Script for Changing a PDB Backend to a JE Backend

The following Bash script demonstrates how to change a PDB backend to a JE Backend:

```
#!/usr/bin/env bash
#
# The contents of this file are subject to the terms of the Common Development and
# Distribution License (the License). You may not use this file except in compliance with the
# License.
#
# You can obtain a copy of the License at legal-notices/CDDLv1.0.txt. See the License for the
# specific language governing permission and limitations under the License.
#
# When distributing Covered Software, include this CDDL Header Notice in each file and include
# the License file at legal-notices/CDDLv1.0.txt. If applicable, add the following below the CDDL
# Header, with the fields enclosed by brackets [] replaced by your own identifying
# information: "Portions Copyright [year] [name of copyright owner]".
#
# Copyright 2017 ForgeRock AS.
#
```

```
if test $# -ne 1
then
  echo "Usage: $0 backendID"
  echo "Migrate a PDB backend to a JE backend with all the data."
  echo "Run this script from the server base directory, such as /path/to/opendj."
  exit 1
fi

# Check that the server is stopped.
echo "Verifying that the server is stopped..."
./bin/status -n -s
if test $? -ne 0
then
  echo "The Directory Server must be stopped to migrate a backend."
  echo "Please stop the server and relaunch the script."
  exit 1
fi
echo ""

# Check for instance.loc.
LOC=.
if [ -f ./instance.loc ]
then
  LOC=`cat ./instance.loc`
elif [ -f /etc/opendj/instance.loc ]
then
  LOC=`cat /etc/opendj/instance.loc`
fi

# Check the backendID.
echo "Verifying the backend $1"
DN=`./bin/ldifsearch "$LOC"/config/config.ldif "(&(objectclass=ds-cfg-pdb-backend)(ds-cfg-backend-id=$1))"
 dn | grep "^dn:"`
if [ -z "$DN" ]
then
  echo "Could not find a PDB backend with this name. Exiting."
  exit 2
fi

echo "Exporting data to /tmp/data_$$"
# Export data from the PDB backend.
./bin/export-ldif --offline -n "$1" -l /tmp/data_$$
if test $? -ne 0
then
  echo
  exit 3
fi

echo "Updating configuration"
# Change the PDB backend configuration to a JE backend configuration.
./bin/ldifmodify -o "$LOC"/config/config.ldif.$$ "$LOC"/config/config.ldif << EOF
$DN
changetype: modify
delete: objectClass
objectClass: ds-cfg-pdb-backend
-
add: objectClass
objectClass: ds-cfg-je-backend
-
```

```
replace: ds-cfg-java-class
ds-cfg-java-class: org.opends.server.backends.jeb.JEBackend
EOF

if test $? -ne 0
then
  echo "Modifications failed. Restoring the original configuration"
  exit 4
fi

cp "$LOC"/config/config.ldif.$$ "$LOC"/config/config.ldif
echo "Configuration updates done."
echo "Importing data..."
# Import the data into the JE backend.
./bin/import-ldif --offline -n $1 -l /tmp/data_$$
if test $? -ne 0
then
  echo "Importing data failed."
  echo "The exported data file is /tmp/data_$$"
  exit 5
fi
echo "Backend $1 converted successfully from PDB to JE."
rm /tmp/data_$$
rm "$LOC"/config/config.ldif.$$
```

Chapter 10

Upgrading a Replication Server

This chapter shows how to upgrade a standalone replication server, meaning a replication server that has no user data backends.

To Upgrade a Standalone Replication Server

If the server holds user data, consider it a directory server and see "*Upgrading a Directory Server*" instead.

1. Prepare for upgrade as described in "*Before You Upgrade*".

2. Stop the server.

3. Unpack the new files over the old files as described in "*Installing Server Software Files*".

4. Run the **upgrade** command, described in upgrade(1), to bring the server configuration data up to date with the new software delivery.

   By default, the **upgrade** command runs interactively, requesting confirmation before making important configuration changes.

   You can use the `--no-prompt` option to run the command non-interactively. In this case, the `--acceptLicense` option lets you accept the license terms non-interactively.

   When using the `--no-prompt` option, if the **upgrade** command cannot complete because it requires confirmation for a potentially long or critical task, then it exits with an error and a message about how to finish making the changes. You can add the `--force` option to force a non-interactive upgrade to continue in this case, also performing long running and critical tasks.

5. Start the upgraded server.

   At this point the upgrade process is complete. See the resulting `upgrade.log` file for a full list of operations performed.

6. (Optional) If you disabled the server as a Windows service in order to upgrade, enable the server as a Windows service again as in the following example:

   ```
   C:\path\to\opendj\bat> windows-service.bat --enableService
   ```

Chapter 11

Upgrading the REST to LDAP Gateway

Replace the REST to LDAP gateway with the newer version, as for a fresh installation, and rewrite the configuration to work with the new version.

For details, see "*Installing the REST to LDAP Gateway*".

Chapter 12
Upgrading the DSML Gateway

Replace the DSML gateway with the newer version, as for a fresh installation.

For details, see "*Installing the DSML Gateway*".

Chapter 13
Removing Server Software

This chapter covers uninstallation and includes the following procedures:

- "To Uninstall Cross-Platform Server Software"
- "To Uninstall the Debian Package"
- "To Uninstall the RPM Package"
- "To Uninstall the Windows Installer Package"

To Uninstall Cross-Platform Server Software

Follow these steps to remove software installed from the cross-platform .zip:

1. Log in as the user who installed and runs the server.
2. Stop replication as described in "To Stop Replication Permanently For a Replica" in the *Administration Guide*.
3. Stop the server.

```
$ /path/to/opendj/bin/stop-ds --quiet
```

4. Delete the files manually:

```
$ rm -rf /path/to/opendj
```

To Uninstall the Debian Package

When you uninstall the Debian package from the command-line, the server is stopped if it is running:

1. Stop replication as described in "To Stop Replication Permanently For a Replica" in the *Administration Guide*.
2. Purge the package from your system:

```
$ sudo dpkg --purge opendj
```

3. (Optional) Remove any remaining server configuration files and directory data:

```
$ sudo rm -rf /opt/opendj
```

To Uninstall the RPM Package

When you uninstall the RPM package from the command-line, the server is stopped if it is running:

1. Stop replication as described in "To Stop Replication Permanently For a Replica" in the *Administration Guide*.

2. Remove the package from your system:

```
# rpm -e opendj
```

3. (Optional) Remove the server configuration files and any directory data:

```
$ sudo rm -rf /opt/opendj
```

To Uninstall the Windows Installer Package

When you uninstall the files installed from the Windows installer package, only the installed files are removed:

1. Stop replication as described in "To Stop Replication Permanently For a Replica" in the *Administration Guide*.

2. Remove installed files in one of the following ways:

   - Use Windows Control Panel.

     a. Open Windows Control Panel and browse to the page to uninstall a program.

     b. Find `ForgeRock OpenDJ` in the list and uninstall it.

   - Use the **msiexec** command.

     The following command quietly removes installed files:

     ```
     C:\>msiexec /x DS-5.0.0.msi /q
     ```

3. (Optional) Manually remove the server configuration files and any directory data.

Installation Reference

Table of Contents

Name

setup

Synopsis

`setup` {subcommand} {options}

Description

This utility can be used to install an OpenDJ instance either as a directory server, a replication server or a proxy server.

Options

The **setup** command takes the following options:

Command options:

`--acceptLicense`

Automatically accepts the product license (if present).

Default: false

`--adminConnectorPort {port}`

Port on which the Administration Connector should listen for communication.

`-D | --rootUserDn {rootUserDN}`

DN for the initial root user for the Directory Server.

Default: cn=Directory Manager

`--instancePath {path}`

Path were the instance should be set up.

Default: /root/workspace/OpenDJ_-_Release/target/checkout/.

`-j | --rootUserPasswordFile {rootUserPasswordFile}`

Path to a file containing the password for the initial root user for the Directory Server.

`-N | --certNickname {nickname}`

Nickname of a keystore entry containing a certificate that the server should use when negotiating secure connections using StartTLS or SSL. Multiple keystore entries may be provided by using this option multiple times.

`-O | --doNotStart`

Do not start the server when the configuration is completed.

Default: false

`--productionMode`

Harden default configuration for production use.

Default: false

`-Q | --quiet`

Use quiet mode.

Default: false

`-S | --skipPortCheck`

Skip the check to determine whether the specified ports are usable.

Default: false

`-u | --keyStorePasswordFile {keyStorePasswordFile}`

Path to a file containing the keystore password. The keystore password is required when you specify an existing file-based keystore (JKS, JCEKS, PKCS#12).

`--useJavaKeyStore {keyStorePath}`

Path of a JKS keystore containing the certificate(s) that the server should use when negotiating secure connections using StartTLS or SSL.

`--useJceks {keyStorePath}`

Path of a JCEKS keystore containing the certificate(s) that the server should use when negotiating secure connections using StartTLS or SSL.

`--usePkcs11KeyStore`

Use certificate(s) in a PKCS#11 token that the server should use when accepting SSL-based connections or performing StartTLS negotiation.

Default: false

`--usePkcs12KeyStore {keyStorePath}`

Path of a PKCS#12 keystore containing the certificate(s) that the server should use when negotiating secure connections using StartTLS or SSL.

`-w | --rootUserPassword {rootUserPassword}`

Password for the initial root user for the Directory Server.

`-W | --keyStorePassword {keyStorePassword}`

Keystore cleartext password. The keystore password is required when you specify an existing file-based keystore (JKS, JCEKS, PKCS#12).

General options:

`-V | --version`

Display Directory Server version information.

Default: false

`-H | --help`

Display this usage information.

Default: false

Subcommands

The **setup** command supports the following subcommands:

setup directory-server

Install an OpenDJ directory server instance. See "setup directory-server --help" for specific options.

Options

The **setup directory-server** command takes the following options:

`-q | --enableStartTls`

Enable StartTLS to allow secure communication with the server using the LDAP port.

Default: false

`-p | --ldapPort {port}`

Port on which the Directory Server should listen for LDAP communication.

`-Z | --ldapsPort {port}`

Port on which the Directory Server should listen for LDAPS communication. The LDAPS port will be configured and SSL will be enabled only if this argument is explicitly specified.

`-a | --addBaseEntry`

Indicates whether to create the base entry in the Directory Server database.

Default: false

-t | --backendType {backendType}

The type of the userRoot backend. Available backend type(s): je.

Default: je

-b | --baseDn {baseDN}

Base DN for user information in the Directory Server. Multiple base DNs may be provided by using this option multiple times.

-l | --ldifFile {ldifFile}

Path to an LDIF file containing data that should be added to the Directory Server database. Multiple LDIF files may be provided by using this option multiple times.

-R | --rejectFile {rejectFile}

Write rejected entries to the specified file.

-d | --sampleData {numEntries}

Specifies that the database should be populated with the specified number of sample entries.

--skipFile {skipFile}

Write skipped entries to the specified file.

-h | --hostname {host}

The fully-qualified directory server host name that will be used when generating self-signed certificates for LDAP SSL/StartTLS, the administration connector, and replication.

Default: localhost.localdomain

--httpPort {port}

Port on which the server should listen for HTTP communication.

--httpsPort {port}

Port on which the server should listen for HTTPS communication.

setup proxy-server

Install an OpenDJ proxy server instance. There are two ways to specify the servers to be contacted by the proxy. They can either be listed exhaustively or retrieved from an existing replication topology. See "setup proxy-server --help" for specific options.

Options

The **setup proxy-server** command takes the following options:

`-q | --enableStartTls`

Enable StartTLS to allow secure communication with the server using the LDAP port.

Default: false

`-p | --ldapPort {port}`

Port on which the Directory Server should listen for LDAP communication.

`-Z | --ldapsPort {port}`

Port on which the Directory Server should listen for LDAPS communication. The LDAPS port will be configured and SSL will be enabled only if this argument is explicitly specified.

`--useJceksTrustStore {trustStorePath}`

Use existing JCEKS truststore file to use to trust the remote server certificates.

`--useJavaTrustStore {trustStorePath}`

Use existing JKS truststore file to use to trust the remote server certificates.

`--loadBalancingAlgorithm {algorithm}`

Algorithm to use to load balance between servers. Available algorithms are 'affinity, least-requests'.

Default: affinity

`--usePkcs12TrustStore {trustStorePath}`

Use existing PKCS12 truststore file to use to trust the remote server certificates.

`--staticPrimaryServer {host:port}`

Static server to contact when available before contacting secondary servers. Multiple servers may be provided by using this option multiple times.

`--proxyUserBindDn {proxyBindDN}`

The bind DN for forwarding LDAP requests to remote servers. This bind DN must be present on all the remote servers.

Default: cn=proxy

--proxyUserBindPassword {proxyBindPassword}

Password associated with the proxy bind DN. The bind password must be the same on all the remote servers.

--proxyUserBindPasswordFile {proxyBindPasswordFile}

Path to a file containing the password associated with the proxy bind DN. The bind password must be the same on all the remote servers.

--replicationBindDn {bindDN}

The bind DN for periodically reading replication server configurations. The bind DN must be present on all replication servers and directory servers, it must be able to read the server configuration.

--replicationBindPassword {bindPassword}

The bind password for periodically reading replication server configurations. The bind password must be the same on all replication and directory servers.

--replicationBindPasswordFile {bindPasswordFile}

Path to a file containing the bind password for periodically reading replication server configurations. The bind password must be the same on all replication and directory servers.

--replicationPreferredGroupId {domainGroupIDNumber}

Replication domain group ID number of directory server replicas to contact when available before contacting other replicas. If this option is not specified then all replicas will be treated the same.

--replicationServer {host:port}

Replication server to contact periodically in order to discover backend servers. Multiple replication servers may be provided by using this option multiple times.

--baseDn {baseDN}

Base DN for user information in the Proxy Server. Multiple base DNs may be provided by using this option multiple times. If no base DNs are defined then the proxy will forward requests to all public naming contexts of the remote servers.

--staticSecondaryServer {host:port}

Static server to contact when all primary servers are unavailable. Multiple servers may be provided by using this option multiple times.

--proxyUsingSsl

Use SSL to secure communications with remote servers.

Default: false

`--proxyUsingStartTls`

Use Start TLS to secure communication with remote servers.

Default: false

`-T | --trustStorePassword {trustStorePassword}`

Truststore cleartext password.

`-U | --trustStorePasswordFile {path}`

Path to a file containing the truststore password.

`--useJvmTrustStore`

Use the JVM truststore for validating remote server certificates.

Default: false

`-X | --trustAll`

Trust all server SSL certificates.

Default: false

`-h | --hostname {host}`

The fully-qualified directory server host name that will be used when generating self-signed certificates for LDAP SSL/StartTLS, the administration connector, and replication.

Default: localhost.localdomain

`--httpPort {port}`

Port on which the server should listen for HTTP communication.

`--httpsPort {port}`

Port on which the server should listen for HTTPS communication.

Exit Codes

0

The command completed successfully.

> 0

An error occurred.

Examples

The following command installs OpenDJ directory server, enabling StartTLS and importing 100 example entries without interaction.

```
$ /path/to/opendj/setup directory-server --adminConnectorPort 4444 -t pdb -b dc=example,dc=com -d 100 \
 -D "cn=Directory Manager" -w password -h opendj.example.com -p 1389 \
 --enableStartTLS


Validating parameters..... Done
Configuring certificates..... Done
Configuring server..... Done
Importing automatically-generated data (100 entries)......... Done
Starting directory server............... Done

To see basic server status and configuration, you can
 launch
/path/to/opendj/bin/status
```

Name

upgrade — upgrade OpenDJ configuration and application data

Synopsis

`upgrade` {options}

Description

Upgrades OpenDJ configuration and application data so that it is compatible with the installed binaries.

This tool should be run immediately after upgrading the OpenDJ binaries and before restarting the server.

NOTE: this tool does not provide backup or restore capabilities. Therefore, it is the responsibility of the OpenDJ administrator to take necessary precautions before performing the upgrade.

This utility thus performs only part of the upgrade process, which includes the following phases for a single server.

1. Get and unpack a newer version of OpenDJ directory server software.
2. Stop the current OpenDJ directory server.
3. Overwrite existing binary and script files with those of the newer version, and then run this utility before restarting OpenDJ.
4. Start the upgraded OpenDJ directory server.

> **Important**
>
> This utility *does not back up OpenDJ before you upgrade, nor does it restore OpenDJ if the utility fails*. In order to revert a failed upgrade, make sure you back up OpenDJ directory server before you overwrite existing binary and script files.

By default this utility requests confirmation before making important configuration changes. You can use the `--no-prompt` option to run the command non-interactively.

When using the `--no-prompt` option, if this utility cannot complete because it requires confirmation for a potentially very long or critical task, then it exits with an error and a message about how to finish making the changes. You can add the `--force` option to force a non-interactive upgrade to continue in this case, also performing long running and critical tasks.

After upgrading, see the resulting `upgrade.log` file for a full list of operations performed.

Options

The **upgrade** command takes the following options:

Command options:

`--acceptLicense`

Automatically accepts the product license (if present).

Default: false

`--force`

Forces a non-interactive upgrade to continue even if it requires user interaction. In particular, long running or critical upgrade tasks, such as re-indexing, which require user confirmation will be skipped. This option may only be used with the 'no-prompt' option.

Default: false

`--ignoreErrors`

Ignores any errors which occur during the upgrade. This option should be used with caution and may be useful in automated deployments where potential errors are known in advance and resolved after the upgrade has completed.

Default: false

Utility input/output options:

`-n | --no-prompt`

Use non-interactive mode. If data in the command is missing, the user is not prompted and the tool will fail.

Default: false

`-Q | --quiet`

Use quiet mode.

Default: false

`-v | --verbose`

Use verbose mode.

Default: false

General options:

`-V | --version`

Display Directory Server version information.

Default: false

`-H | --help`

Display this usage information.

Default: false

Exit Codes

0

The command completed successfully.

2

The command was run in non-interactive mode, but could not complete because confirmation was required to run a long or critical task.

See the error message or the log for details.

other

An error occurred.

See the *OpenDJ Installation Guide* for an example upgrade process for OpenDJ directory server installed from the cross-platform (.zip) delivery.

Native packages (.deb, .rpm) perform more of the upgrade process, stopping OpenDJ if it is running, overwriting older files with newer files, running this utility, and starting OpenDJ if it was running when you upgraded the package(s).