Getting Started

/ Directory Services 6.5

Latest update: 6.5.6

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 getting started quickly with ForgeRock® Directory Services software.

Table of Contents

Overview

Use this guide to get a quick, hands-on looks at what Directory Services software can do. You will download, install, and use DS on your local computer. Expect to spend 30-120 minutes working through this guide.

What You Will Learn

Topic	Covered In...
Where to get DS software	"To Download DS Software"
How to set up DS software	"To Install DS Software"
Where to find DS tools	"To Install DS Software"
How to check DS server status	"To Install DS Software"
Basic LDAP concepts	"*First Steps With LDAP*"
How to create, read, update, and delete LDAP entries	"*First Steps With LDAP*"
LDAP read, search, lookup concepts	"Search an LDAP Directory"
LDAP authentication concepts	"Modify an LDAP Entry"
How to create, read, update, and delete over HTTP	"*First Steps With REST*"
How to modify the DS server configuration	"Create a Resource"
Data replication concepts	"*First Steps With Replication*"
How to measure LDAP performance	"*First Steps With Performance Tools*"
Where to look for more advanced information	"*Where To Go From Here*"

Chapter 1
Installing DS

This chapter shows you how to download and install a single directory server and bundled command-line tools on your local computer.

Follow these procedures in order to get started:

- "To Prepare For Installation"
- "To Download DS Software"
- "To Install DS Software"

To Prepare For Installation

Before downloading and installing DS software, follow these steps:

1. To evaluate DS software, make sure you have 10 GB free disk space for the software and for sample data.
2. Verify that you have a supported Java version installed on your local computer.

   Directory Services 6.5 software requires Java 8 or 11, specifically at least the Java Standard Edition runtime environment, or the corresponding Java Development Kit to compile Java plugins and applications.
3. Make sure the **curl** command is available.

   The HTTP examples in "*First Steps With REST*" use the **curl** command, which is described at https://curl.haxx.se.

To Download DS Software

Directory Services software is free to download, evaluate, and use for developing your applications and solutions. Follow these steps:

1. If you do not have an account on ForgeRock BackStage, https://backstage.forgerock.com, sign up for one.
2. Sign in to ForgeRock BackStage.
3. Find and download the latest Directory Services ZIP distribution.

To Install DS Software

After downloading Directory Services software, follow these steps to unpack and install a directory server:

1. Unzip the `.zip` file into the file system directory where you want to install the server.

   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-6.5.6` directory from the folder you specify.

   The documentation shows the installation file system directory as `/path/to/opendj`.

   For example:

   ```
   $ unzip ~/Downloads/DS-6.5.6.zip -d /path/to
   ```

2. Use the **setup** command to set up a running server with sample data.

   The following example runs the command non-interactively. Use the same settings shown here to be able to copy and paste the commands shown in this guide:

   ```
   $ /path/to/opendj/setup \
    directory-server \
    --rootUserDn "cn=Directory Manager" \
    --rootUserPassword password \
    --monitorUserPassword password \
    --hostname localhost \
    --ldapPort 1389 \
    --httpPort 8080 \
    --adminConnectorPort 4444 \
    --baseDn dc=example,dc=com \
    --sampleData 10000 \
    --acceptLicense
   Validating parameters... Done
   Configuring certificates... Done
   Configuring server... Done
   Importing automatically-generated data (10000 entries)... Done
   Starting directory server... Done

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

   The **setup directory-server** command shown here has the following options:

   `--rootUserDn "cn=Directory Manager"`
   `--rootUserPassword password`

   > These options set the credentials for the directory superuser. This user has privileges to perform any and all administrative operations, and is not subject to access control. It is called the root user due to the similarity to the UNIX root user.

The root user *distinguished name* (DN) identifies the directory superuser. In LDAP, a DN is the fully qualified name for a directory entry. The name used here is the default name: `cn=Directory Manager`.

`--monitorUserPassword password`

The monitor user has the privilege to read monitoring data. No `--monitorUserDn` option is set, so the DN defaults to `uid=Monitor`.

`--hostname localhost`

The server uses the fully qualified host name in self-signed certificates and for identification between replicated servers.

Using `localhost` is a shortcut suitable only for evaluation on your local computer. In all other cases, set this to the fully qualified domain name, such as `ds.example.com`.

`--ldapPort 1389`

The reserved port for LDAP is 389. Connections to this port can be secured with StartTLS, but are not secure by default.

Examples in the documentation use 1389, which is accessible to non-privileged users.

`--httpPort 8080`

The reserved port for HTTP is 80.

HTTP client applications access directory data and monitoring information on this port.

Examples in the documentation use 8080, which is accessible to non-privileged users.

`--adminConnectorPort 4444`

This is the service port used to configure the server and to run tasks. Connections to this port are secured with TLS.

The port used in the documentation is 4444, which is the initial port suggested during interactive setup.

`--baseDn dc=example,dc=com`

A base DN is the suffix shared by all DNs in a set of directory data.

LDAP entries are arranged hierarchically in the directory. The hierarchical organization resembles a file system on a PC or a web server, often visualized as an upside down tree structure, or a pyramid. In the same way a full path uniquely identifies each file or folder in a file system, a DN uniquely identifies each LDAP entry.

Each DN consists of components separated by commas, such as `uid=user.6,ou=People,dc=example,dc=com`. The base DN matches the final components of each DN in that branch of the directory. A DN's components reflect the hierarchy of directory entries. The user entry with DN `uid=user.6,ou=People,dc=example,dc=com` is said to be under the organization unit entry `ou=People,dc=example,dc=com`, which in turn is under `dc=example,dc=com`.

Basic components have the form *`attribute-name=attribute-value`*, such as `dc=com`. In the example `dc=com`, the attribute `dc` (DNS domain component) has the value `com`. The DN `dc=example,dc=com` reflects the DNs domain name `example.com`.

`--sampleData 10000`

This option generates 10 000 sample LDAP user entries under `dc=example,dc=com`. All users have the same password, literally `password`.

`--acceptLicense`

Remove this option to read the license and then accept it interactively.

Alternatively, you can run the **setup** command interactively by starting it without options.

3. Add the DS tools to your PATH.

   - In **sh** or **bash**:

     ```
     $ export PATH=${PATH}:/path/to/opendj/bin
     ```

   - In Windows **cmd.exe**:

     ```
     C:\>set PATH=%PATH%;C:\path\to\opendj\bat
     ```

   The tools are shell and bat scripts. Notice that shell scripts are found in a `bin` directory, and that bat scripts are found in a `bat` directory.

4. Run the **status** command:

   ```
   $ status \
    --bindDn "cn=Directory Manager" \
    --bindPassword password \
    --hostname localhost \
    --port 4444 \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePasswordFile /path/to/opendj/config/keystore.pin
   ```

   The **status** command uses a secure connection to the administration port. To trust the server's self-signed certificate, the command uses the server's own keystore as the truststore.

   Read the output that the **status** command displays.

Chapter 2
First Steps With LDAP

This chapter shows you how to use bundled DS command-line tools to send LDAP requests to the server you set up in "*Installing DS*". If you have not added the bundled LDAP command-line tools location to your PATH, see Step 3.

Tip

LDAP is short for Lightweight Directory Access Protocol, a standard Internet protocol. In LDAP, the word "Directory" refers to a type of distributed database originally modeled on a telephone directory.

A telephone directory, such as an old phone book, indexes entries for people and organizations in alphabetical order by name. You can use someone's name to quickly look up their telephone number and address. An LDAP server generalizes this design to use multiple indexes, and to let you define what you want to store in a directory entry.

Try these examples in order to start learning about LDAP operations:

- "Search an LDAP Directory"
- "Modify an LDAP Entry"
- "Add an LDAP Entry"
- "Delete an LDAP Entry"

Search an LDAP Directory

Searching the directory is like searching for a phone number in a phone book. You can look up a subscriber's phone number because you know the subscriber's last name. In other words, you use the value of an attribute to find entries that have attributes of interest.

When looking up a subscriber's entry in a phone book, you need to have some idea where they live in order to pick the right phone book. For example, a Los Angeles subscriber cannot be found in the New York phone book. In an LDAP directory, you need to know at least the base DN to search under.

For this example, assume you know a user's full name, `Abagail Abadines`, and that Abagail's entry is under the base DN `dc=example,dc=com`. You want to look up Abagail's email and office location. The following command sends an appropriate LDAP search request to the server you installed:

```
$ ldapsearch \
 --hostname localhost \
 --port 1389 \
 --baseDn dc=example,dc=com \
 "(cn=Abagail Abadines)" \
 cn mail street l
dn: uid=user.6,ou=People,dc=example,dc=com
cn: Abagail Abadines
mail: user.6@example.com
street: 61394 Adams Street
l: Lynchburg
```

Notice the following characteristics of the search:

- The base DN `dc=example,dc=com` tells the server where to look for Abagail's entry. Servers can hold data for multiple base DNs, so this is important information.

  It is possible to restrict the scope of the search, but the default is to search the entire subtree under the base DN.

- The command uses a *search filter*, `"(cn=Abagail Abadines)"`, which tells the server, "Find entries whose `cn` attribute exactly matches the string `Abagail Abadines` without regard to case sensitivity."

  The `cn` (`commonName`) attribute is a standard attribute for full names.

  Internally, the directory server has an equality index for the `cn` attribute. The directory uses the index to quickly find matches for `abagail abadines`. The default behavior in LDAP is to ignore case, so `"(cn=Abagail Abadines)"`, `"(cn=abagail abadines)"`, and `"(CN=ABAGAIL ABADINES)"` are equivalent.

  If more than one entry matches the filter, the server returns multiple entries.

- The filter is followed by a list of LDAP attributes, `cn mail street l`. This tells the server to return only the specified attributes in the search result entries. By default, if you do not specify the attributes to return, the server returns all the user attributes that you have the right to read.

- The result shows attributes from a single entry. Notice that an LDAP entry, represented here in the standard LDIF format, has a flat structure with no nesting.

  The DN that uniquely identifies the entry is `uid=user.6,ou=People,dc=example,dc=com`. Multiple entries can have the same attribute values, but each must have a unique DN. This is the same as saying that the leading *relative distinguished name* (RDN) value must be unique at this level in the hierarchy. Only one entry directly under `ou=People,dc=example,dc=com` has the RDN `uid=user.6`.

  The `mail`, `street`, `l` (location), and `uid` attributes are all standard LDAP attributes like `cn`.

For additional examples, see "Searching the Directory" in the *Developer's Guide*.

Modify an LDAP Entry

When searching for LDAP entries as shown in "Search an LDAP Directory", you might get results without authenticating to the directory.

When modifying directory data, however, you must authenticate first. LDAP servers must know who you are to determine what you have access to.

Consider the following example, where Abagail chooses to modify the description on her own entry:

```
$ ldapmodify \
 --hostname localhost \
 --port 1389 \
 --bindDn uid=user.6,ou=People,dc=example,dc=com \
 --bindPassword password <<EOF
dn: uid=user.6,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: New description
EOF
# MODIFY operation successful for DN uid=user.6,ou=People,dc=example,dc=com
```

- The modification is expressed in standard LDAP Data Interchange Format (LDIF).

  The LDIF specifies the DN of the target entry to modify. It then indicates that the change to perform is an LDAP modify, and that the value `New description` is to replace existing values of the `description` attribute.

- Abagail's authentication credentials are provided with the `--bindDn` and `--bindPassword` options. Notice that the user identifier is Abagail's DN.

  Authentication operations bind an LDAP identity to a connection. In LDAP, a client application connects to the server, then binds an identity to the connection. An LDAP client application keeps its connection open until it finishes performing its operations. The server uses the identity bound to the connection to make authorization decisions for subsequent operations, such as search and modify requests.

  If no credentials are provided, then the identity for the connection is that of an anonymous user. As a directory administrator, you can configure access controls for anonymous users just as you configure access controls for other users.

  A simple bind involving a DN and a password is just one of several supported authentication mechanisms. The documentation frequently shows simple binds in examples because this kind of authentication is so familiar. Alternatives include authenticating with a digital certificate, or using Kerberos.

- Notice that the result is a comment indicating success. The command's return code—0, but not shown in the example—also indicates success.

  The scripts and applications that you write should use and trust LDAP return codes.

For additional examples, see "Modifying Entry Attributes" in the *Developer's Guide*. Also see "Changing Passwords" in the *Developer's Guide*.

Add an LDAP Entry

Authorized users can modify attributes, and can also add and delete directory entries.

The following example adds a new user entry to the directory:

```
$ ldapmodify \
 --hostname localhost \
 --port 1389 \
 --bindDn "cn=Directory Manager" \
 --bindPassword password <<EOF
dn: uid=newuser,ou=People,dc=example,dc=com
uid: newuser
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
cn: New User
sn: User
ou: People
mail: newuser@example.com
userPassword: changeme
EOF
# ADD operation successful for DN uid=newuser,ou=People,dc=example,dc=com
```

- The entry to add is expressed in standard LDIF.
- The bind DN for the user requesting the add is `cn=Directory Manager`. It is also possible to authorize regular users to add entries.

For additional examples, see "Adding Entries" in the *Developer's Guide*.

Delete an LDAP Entry

The following example deletes the user added in "Add an LDAP Entry":

```
$ ldapdelete \
 --hostname localhost \
 --port 1389 \
 --bindDn "cn=Directory Manager" \
 --bindPassword password \
 uid=newuser,ou=People,dc=example,dc=com
# DELETE operation successful for DN uid=newuser,ou=People,dc=example,dc=com
```

Notice that the **ldapdelete** command specifies the entry to delete by its DN.

For additional examples, see "Deleting Entries" in the *Developer's Guide*.

Chapter 3

First Steps With REST

This chapter shows you how to send RESTful HTTP requests to the server you set up in "*Installing DS*".

Tip

ForgeRock Directory Services let you access directory data over HTTP as well as LDAP. The feature is known as REST to LDAP because it transforms REST operations into LDAP operations.

The server maps JSON resources to LDAP entries in order to convert HTTP operations to LDAP internally. The directory server you installed is bundled with a default mapping file for sample data. You can configure your own mapping depending on your directory data and the JSON objects you want.

Try these examples in order to start learning about HTTP access to directory data:

- "Create a Resource"
- "Read a Resource"
- "Update a Resource"
- "Delete a Resource"

Create a Resource

When you configured an HTTP port at setup time, you enabled the DS server to respond to HTTP requests. However, you must make a further change to the server configuration to enable HTTP access to your directory data.

Because the default REST to LDAP mapping file is an example designed to work with sample data, it is not guaranteed to work with *your* data. As a result, the API to access your directory data is not enabled by default.

The following command enables the API to access directory data using the default example mapping file:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --endpoint-name /api \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePasswordFile /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Like the **status** command you ran in "To Install DS Software", the **dsconfig** command authenticates using the directory superuser credentials. It uses a secure connection to the administration port. To trust the server's self-signed certificate, the command uses the server's own keystore as the truststore. It sends a configuration request to enable the `/api` endpoint, where the directory server exposes the HTTP API to access directory data.

In order to make updates to the directory, authorize a user to do so. The following example authorizes Abagail (`user.6`) to make changes to directory data:

```
$ ldapmodify \
 --hostname localhost \
 --port 1389 \
 --bindDn "cn=Directory Manager" \
 --bindPassword password <<EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=example,dc=com") (targetattr = "*")
 (version 3.0;acl "Abagail can update entries"; allow(all)
  userdn = "ldap:///uid=user.6,ou=People,dc=example,dc=com";)
EOF
```

With the API enabled, and Abagail authorized to make changes to the directory, use the API to create a user over HTTP as in the following example:

```
$ curl \
 --request POST \
 --user user.6:password \
 --header "Content-Type: application/json" \
 --data '{
  "_id": "newuser",
  "_schema":"frapi:opendj:rest2ldap:user:1.0",
  "contactInformation": {
    "telephoneNumber": "+1 408 555 1212",
    "emailAddress": "newuser@example.com"
  },
  "name": {
    "givenName": "New",
    "familyName": "User"
  },
  "displayName": ["New User"],
  "manager": {
    "_id": "user.6",
    "displayName": "Abagail Abadines"
```

```
  }
 }' \
 http://opendj.example.com:8080/api/users?_prettyPrint=true
{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<timestamp>"
  },
  "userName" : "newuser@example.com",
  "displayName" : [ "New User" ],
  "name" : {
    "givenName": "New",
    "familyName": "User"
  },
  "manager" : {
    "_id" : "user.6",
    "displayName" : "Abagail Abadines"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  }
}
```

- The user performing the HTTP POST is Abagail.

  The default authorization mechanism for HTTP access is HTTP Basic authentication. Abagail's HTTP user ID, `user.6`, is mapped to her LDAP DN, `uid=user.6,ou=people,dc=example,dc=com`. REST to LDAP uses Abagail's DN and password to perform a simple LDAP bind for authentication. The directory can then use its LDAP-based access control mechanisms to authorize the operation that she requests.

- The form of the JSON data respects the API defined by the mapping file. The example mapping file is a JSON format configuration file with comments. See `/path/to/opendj/config/rest2ldap/endpoints/api/example-v1.json`.

- The successful response is the JSON resource that the command created.

  Fields whose names start with `_` are reserved. For details, see "About ForgeRock Common REST" in the *Developer's Guide*.

For additional details, see "Create" in the *Developer's Guide* and "Creating Resources" in the *Developer's Guide*.

Read a Resource

The following example reads the user resource created in "Create a Resource":

```
$ curl \
 --request GET \
 --user user.6:password \
 --header "Content-Type: application/json" \
 http://opendj.example.com:8080/api/users/newuser?_prettyPrint=true
{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<timestamp>"
  },
  "userName" : "newuser@example.com",
  "displayName" : [ "New User" ],
  "name" : {
    "givenName": "New",
    "familyName": "User"
  },
  "manager" : {
    "_id" : "user.6",
    "displayName" : "Abagail Abadines"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  }
}
```

Notice that Abagail authenticates for this HTTP GET request. If no credentials are specified, the response is the HTTP 401 Unauthorized:

```
{"code":401,"reason":"Unauthorized","message":"Invalid Credentials"}
```

In other words, the HTTP Basic authorization mechanism requires authentication even for read operations.

For additional details, see "Read" in the *Developer's Guide* and "Reading a Resource" in the *Developer's Guide*. You can also query collections of resources as described in "Query" in the *Developer's Guide* and "Querying Resource Collections" in the *Developer's Guide*.

Update a Resource

The following HTTP PUT request replaces the user resource created in "Create a Resource":

```
$ curl \
 --request PUT \
 --user user.6:password \
 --header "Content-Type: application/json" \
 --header "If-Match: *" \
 --data '{
  "_id": "newuser",
  "_schema":"frapi:opendj:rest2ldap:user:1.0",
  "contactInformation": {
    "telephoneNumber": "+1 234 567 8910",
    "emailAddress": "updated.user@example.com"
```

```
  },
  "name": {
    "givenName": "Updated",
    "familyName": "User"
  },
  "displayName": ["Updated User"],
  "manager": {
    "_id" : "user.6",
    "displayName" : "Abagail Abadines"
  }
 }' \
 http://opendj.example.com:8080/api/users/newuser?_prettyPrint=true
{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<timestamp>"
  },
  "userName" : "newuser@example.com",
  "displayName" : [ "Updated User" ],
  "name" : {
    "givenName" : "Updated",
    "familyName" : "User"
  },
  "manager" : {
    "_id" : "user.6",
    "displayName" : "Abagail Abadines"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 234 567 8910",
    "emailAddress" : "updated.user@example.com"
  }
}
```

Resources are versioned using revision numbers. A revision is specified in the resource's `_rev` field. The `--header "If-Match: *"` ensures the resource is replaced, regardless of its revision. Alternatively, you can set `--header "If-Match: revision"` to replace only the expected revision of the resource.

For additional details, see "Update" in the *Developer's Guide* and "Updating Resources" in the *Developer's Guide*. You can also patch resources instead of replacing them entirely. See "Patch" in the *Developer's Guide* and "Patching Resources" in the *Developer's Guide*.

Delete a Resource

The following HTTP DELETE request deletes the user resource updated in "Update a Resource":

```
$ curl \
 --request DELETE \
 --user user.6:password \
 --header "Content-Type: application/json" \
 http://opendj.example.com:8080/api/users/newuser?_prettyPrint=true
{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<timestamp>"
  },
  "userName" : "newuser@example.com",
  "displayName" : [ "Updated User" ],
  "name" : {
    "givenName" : "Updated",
    "familyName" : "User"
  },
  "manager" : {
    "_id" : "user.6",
    "displayName" : "Abagail Abadines"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 234 567 8910",
    "emailAddress" : "updated.user@example.com"
  }
}
```

For additional details, see "Delete" in the *Developer's Guide* and "Deleting Resources" in the *Developer's Guide*.

Chapter 4

First Steps With Replication

This chapter demonstrates the power of directory data replication. Replication provides automatic data synchronization between directory servers. It ensures that all directory servers eventually share a consistent set of directory data.

Tip

Replication requires two or more directory servers and additional configuration. This chapter takes you though the setup process quickly, providing UNIX shell commands that you can reuse. It does not explain each command in detail.

Once you have worked through this chapter, you can read "*Managing Data Replication*" in the *Administration Guide* for additional details.

Setting Up Replication

Add a second directory server, and set up replication between the second server and the server you set up in "*Installing DS*". The following shell script demonstrates the process:

```
# Unpack files for a second, replica server:
cd ~/Downloads && unzip ~/Downloads/DS-6.5.6.zip && mv opendj /path/to/replica

# Set up a second, replica server configuring only the necessary ports:
/path/to/replica/setup \
 directory-server \
 --rootUserDn "cn=Directory Manager" \
 --rootUserPassword password \
 --hostname localhost \
 --ldapPort 11389 \
 --adminConnectorPort 14444 \
 --baseDN dc=example,dc=com \
 --acceptLicense

# Configure replication between the servers (make them aware of each other, ready to replicate):
dsreplication \
 configure \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --host1 localhost \
 --port1 4444 \
 --bindDN1 "cn=Directory Manager" \
 --bindPassword1 password \
 --replicationPort1 8989 \
```

```
 --host2 localhost \
 --port2 14444 \
 --bindDN2 "cn=Directory Manager" \
 --bindPassword2 password \
 --replicationPort2 18989 \
 --trustAll \
 --no-prompt

# Initialize replication between the servers (copy data from the source to the destination server):
dsreplication \
 initialize \
 --adminUID admin \
 --adminPassword password \
 --baseDN dc=example,dc=com \
 --hostSource localhost \
 --portSource 4444 \
 --hostDestination localhost \
 --portDestination 14444 \
 --trustAll \
 --no-prompt
```

Trying Replication

After working through "Setting Up Replication", demonstrate to yourself that replication is operational when the service is running smoothly. Use the commands in the following example:

```
# Update a description on the first server:
ldapmodify \
 --hostname localhost \
 --port 1389 \
 --bindDn uid=user.6,ou=People,dc=example,dc=com \
 --bindPassword password <<EOF
dn: uid=user.6,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Replicate this
EOF

# On the first server, read the description "Replicate this" to see the effects of your change:
ldapsearch \
 --hostname localhost \
 --port 1389 \
 --baseDn dc=example,dc=com \
 "(cn=Abagail Abadines)" \
 description

# On the second server, read the description "Replicate this" to check that the change has been
 replicated:
ldapsearch \
 --hostname localhost \
 --port 11389 \
 --baseDn dc=example,dc=com \
 "(cn=Abagail Abadines)" \
 description
```

Also demonstrate that replication works around crashes and network interruptions. Use the commands in the following example:

```
# Stop the second server to simulate a network outage or server crash:
/path/to/replica/bin/stop-ds

# On the first server, update the description again:
ldapmodify \
 --hostname localhost \
 --port 1389 \
 --bindDn uid=user.6,ou=People,dc=example,dc=com \
 --bindPassword password <<EOF
dn: uid=user.6,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Second server is stopped
EOF

# On the first server, read the description "Second server is stopped" to see the change:
ldapsearch \
 --hostname localhost \
 --port 1389 \
 --baseDn dc=example,dc=com \
 "(cn=Abagail Abadines)" \
 description

# Start the second server again to simulate recovery:
/path/to/replica/bin/start-ds

# On the second server, read the description "Second server is stopped" to check that replication has
 resumed:
ldapsearch \
 --hostname localhost \
 --port 11389 \
 --baseDn dc=example,dc=com \
 "(cn=Abagail Abadines)" \
 description
```

Tip

Unlike some databases, DS replication does not operate in active-passive mode. Instead, you read and write on any running server. Replication replays your changes as soon as possible.

As an exercise, demonstrate this by stopping the first server, modifying an entry on the second server, and then restarting the first server. Search for the modified entry on the first server to see that replication replays the change in that direction as well.

Reading the Changelog

Some applications require notification when directory data updates occur. For example, IDM can sync directory data with another database. Other applications start additional processing when certain updates occur.

Once configured as described in "Setting Up Replication", replicated DS directory servers publish an external change log over LDAP. This changelog allows authorized client applications to read changes to directory data.

The following example command reads the changelog:

```
$ ldapsearch \
 --hostname localhost \
 --port 1389 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --baseDN cn=changelog \
 --control "1.3.6.1.4.1.26027.1.5.4:false" \
 "(&)" \
 changes changeLogCookie targetDN
```

When looking at the output of the command, notice that the `changes` values are base64-encoded in LDIF because they include line breaks. You can use the DS **base64** command to decode them. For details, see "Change Notification For Your Applications" in the *Administration Guide*.

Chapter 5

First Steps With Performance Tools

This chapter demonstrates directory server performance. DS directory servers offer high throughput and low response times for most operations.

Tip

Before trying the examples in this chapter, work through the previous chapters. You should have two directory server replicas running on your local computer as described in "*First Steps With Replication*".

DS software includes the following command-line tools for measuring performance of common LDAP operations:

- **addrate** measures LDAP adds and deletes
- **authrate** measures LDAP binds
- **modrate** measures LDAP modifications
- **searchrate** measures LDAP searches

Measuring LDAP Modification Performance

Use the following script to measure LDAP modifications:

```
# Run modrate for 10 seconds against the first server, and observe the performance numbers:
modrate \
 --maxDuration 10 \
 --hostname localhost \
 --port 1389 \
 --bindDn uid=user.6,ou=People,dc=example,dc=com \
 --bindPassword password \
 --noRebind \
 --numConnections 4 \
 --numConcurrentRequests 4 \
 --targetDn "uid=user.{1},ou=people,dc=example,dc=com" \
 --argument "rand(0,10000)" \
 --argument "randstr(16)" \
 "description:{2}"

# Read monitoring information concerning the number of modify requests on the LDAP port:
ldapsearch \
 --hostname localhost \
 --port 1389 \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN "cn=LDAP,cn=connection handlers,cn=monitor" \
 "(&)" \
 ds-mon-requests-modify
```

When reading the **modrate** command output, notice that it shows statistics for throughput (operations/second), response times (milliseconds), and errors/second. If you expect all operations to succeed and yet `err/sec` is not 0.0, the command options are no doubt incorrectly set.

Notice that the monitoring attributes hold similar, alternative statistics.

Measuring LDAP Search Performance

Use the following script to measure LDAP searches:

```
# Run searchrate for 10 seconds against the first server, and observe the performance numbers:
searchrate \
 --maxDuration 10 \
 --hostname localhost \
 --port 1389 \
 --bindDn uid=user.6,ou=People,dc=example,dc=com \
 --bindPassword password \
 --noRebind \
 --numConnections 4 \
 --numConcurrentRequests 4 \
 --baseDn "dc=example,dc=com" \
 --argument "rand(0,10000)" \
 "(uid=user.{})"

# Read monitoring information concerning the number of subtree search requests on the LDAP port:
ldapsearch \
 --hostname localhost \
 --port 1389 \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN "cn=LDAP,cn=connection handlers,cn=monitor" \
 "(&)" \
 ds-mon-requests-search-sub
```

Notice that **searchrate** command output resembles that of the **modrate** command. The **searchrate** output also indicates how many entries each search returned.

Checking Modifications Were Replicated

After running the performance tools, check that both replicas have the same data. The following example uses the **ldifdiff** tool to make the comparison:

```
$ ldifdiff \
 <(ldapsearch --hostname localhost --port 1389 --bindDn "cn=Directory Manager" --bindPassword password --
baseDn dc=example,dc=com "(&)") \
 <(ldapsearch --hostname localhost --port 11389 --bindDn "cn=Directory Manager" --bindPassword password --
baseDn dc=example,dc=com "(&)")
# No differences were detected between the source and target LDIF files
```

To retrieve all entries, the searches are performed as directory superuser. The superuser is not subject to resource limits, which prevent other users from using an unfair share of system resources, and which would prevent the server from returning all entries. The command should return 0, with no differences as shown above.

Chapter 6

Where To Go From Here

Once you have worked through the other chapters in this guide, try the following suggestions:

Learn More About DS Replication

Data replication is sometimes called the "killer feature" of LDAP directories. Its strengths are in enabling very high availability for directory services even during network outages, and automatically resolving conflicts that can occur when the network is down, for example. LDAP directories have been improving and hardening replication features for decades.

Its weaknesses are that replication protocols have not been standardized for interoperability, and that unwary developers can misunderstand its property of eventual consistency if they are too used to the strong, immediate consistency of monolithic, transactional databases.

Replication necessarily involves multiple servers and additional configuration. You can learn more about it by reading "*Managing Data Replication*" in the *Administration Guide*, and trying the examples in that chapter.

Browse Other DS Documentation

DS Documentation

Document	Topics Covered
Release Notes	Compatibility changes Documentation updates Fixed bugs Known issues Limitations Prerequisites Release levels and interface stability What's new
Installation Guide	Installing servers and gateways Upgrading servers and gateways Uninstalling servers and gateways
Deployment Guide	Deployment checklists Deployment patterns Planning the deployment

Document	Topics Covered
	Sizing and provisioning systems What each DS component does
Administration Guide	Access control Account lockout Alerts Attribute value uniqueness Backup, restore Changing certificates Data confidentiality Export, import Indexing LDAP schema Logging Monitoring Moving servers Notifications Pass-through authentication Password Policy Privileges Proxy services Replication Resource limits Samba password sync Starting, stopping, restarting servers Tools and constraints Troubleshooting Tuning Understanding directory services
Security Guide	Administrative roles and security Authentication PKI and the Crypto Manager Protecting files Securing connections Securing Java Securing the operating system Securing web containers for gateway applications Securing DS servers Security features Threats
Developer's Guide	Best practices for application developers Collective attributes Embedding DS servers Groups How to perform LDAP operations How to use REST APIs Server plugins Understanding LDAP Virtual attributes

Document	Topics Covered
Reference	File layout LDAP result codes Monitoring metrics Ports used REST to LDAP configuration Supported LDAP controls Supported LDAP extended operations Supported locales and languages Supported RFCs, standards Tools reference
Configuration Reference	Exhaustive reference for the **dsconfig** server configuration command
LDAP Schema Reference	Exhaustive reference for default LDAP schema, including monitoring attributes and object classes
DS Javadoc	Evolving LDAP SDK and server APIs, including ForgeRock common APIs
Log Message Reference	DS server error log messages by category and ID

Try Third-Party Tools With DS Software

LDAP is a standard protocol, and so you can use LDAP-compliant third-party tools to manage DS directory data.

The following is a non-exhaustive list of popular, third-party LDAP tools:

- Admin4
- Apache Directory Studio
- JXplorer and JXWorkBench
- phpLDAPadmin
- Softerra LDAP Administrator
- web2ldap

Many software solutions include support for LDAP authentication and LDAP-based address books.

Be aware that ForgeRock does not offer technical support for third-party tools, and does not necessarily endorse particular tools.

Use DS With AM

For details, see the following documentation:

- *Planning With Providers* and *Example Deployment Topology* in the AM *Deployment Planning Guide*
- *Preparing External Data Stores* and *Implementing the Core Token Service* in the AM *Installation Guide*

  You can install DS directory servers for use as external AM stores. For details, "*Using Directory Server Setup Profiles*" in the *Installation Guide*.
- *Directory Services Configuration Properties* in the AM *Setup and Maintenance Guide*

Use DS With IDM

For details, see the following documentation:

- *About the Repository* and *Selecting a Repository* in the IDM *Installation Guide*

  Also see "To Use DS as an External IDM Repository" in the *Installation Guide*.
- *One Way Synchronization From LDAP to IDM*, *Two Way Synchronization From LDAP to IDM*, and other LDAP-related chapters in the IDM *Samples Guide*
- *Understanding the DS Repository Configuration* and *Generic and Explicit Mappings With a DS Repository* in the IDM *Integrator's Guide*
- *Synchronizing Passwords With ForgeRock Directory Services (DS)* in the IDM *Password Synchronization Plugin Guide*

Remove DS Software

For details, see "To Uninstall Cross-Platform Server Software" in the *Installation Guide*.

Appendix A. Getting Support

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, see https://www.forgerock.com.

ForgeRock has staff members around the globe who support our international customers and partners. For details on ForgeRock's support offering, including support plans and service level agreements (SLAs), visit https://www.forgerock.com/support.

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.