PingDS

June 5, 2025

DIRECTORY SERVICES

Version: 7.2

Table of Contents

Start here

Use this guide to get a quick, hands-on look 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.

Install DS

Install DS software.

Learn LDAP

Use DS LDAP tools.

Learn REST/HTTP

Access DS over HTTP.

Learn Replication

Replicate DS data.

Measure Performance

Measure LDAP operations.

Learn Access Control

Learn DS ACIs.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

The common REST API provides ForgeRock Identity Platform software common ways to access web resources and collections of resources.

Install DS

> **Tip**
>
> DS software has no GUI. Instead, DS software is bundled with command-line tools.
> Because LDAP is standard, you can use third-party GUI tools to view and edit directory data. For a short list, see Try third-party tools.

Prepare for installation

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.

   For details, check the supported Java versions.
3. If you plan to run the Bash-based REST API examples, make sure the `curl` command is available.

   For details, see the curl site.

Download DS software

1. If you do not have an account on Ping Identity Backstage, sign up for one.
2. Sign in to Backstage.
3. Find and download the latest Directory Services ZIP distribution.

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-7.2.5` directory from the folder you specify.

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

   For example:

   Bash

   ```
   $ unzip ~/Downloads/DS-7.2.5.zip -d /path/to
   ```

PowerShell

```
PS C:\path\to> Expand-Archive DS-7.2.5.zip C:\path\to
```

Zsh

```
% unzip ~/Downloads/DS-7.2.5.zip -d /path/to
```

2. Generate and save a deployment ID using the deployment ID password of your choice.

   You will use this ID and its password when setting up DS servers in your deployment. The DS server uses the two together when generating other keys to protect shared secret keys and secure connections to other DS servers:

Bash

```
$ /path/to/opendj/bin/dskeymgr create-deployment-id --deploymentIdPassword password
<deployment-id>
$ export DEPLOYMENT_ID=<deployment-id>
```

PowerShell

```
PS C:\path\to> C:\path\to\opendj\bat\dskeymgr.bat create-deployment-id --deploymentIdPassword password
<deployment-id>
```

Zsh

```
% /path/to/opendj/bin/dskeymgr create-deployment-id --deploymentIdPassword password
<deployment-id>
$ export DEPLOYMENT_ID=<deployment-id>
```

3. Use the `setup` command to set up a server with the `ds-evaluation` profile. The evaluation profile includes Example.com sample data, more lenient access control, and some other features.

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:

Bash

```
$ /path/to/opendj/setup \
 --serverId first-ds \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDn uid=admin \
 --rootUserPassword password \
 --monitorUserPassword password \
 --hostname localhost \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --replicationPort 8989 \
 --profile ds-evaluation \
 --start \
 --acceptLicense
Validating parameters..... Done
Configuring certificates..... Done
Configuring server... Done
Configuring profile DS evaluation..... Done
Starting directory server..... Done

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

PowerShell

```
PS C:\path\to> C:\path\to\opendj\setup.bat `
 --serverId first-ds `
 --deploymentId <deployment-id> `
 --deploymentIdPassword password `
 --rootUserDn uid=admin `
 --rootUserPassword password `
 --monitorUserPassword password `
 --hostname localhost `
 --ldapPort 1389 `
 --ldapsPort 1636 `
 --httpsPort 8443 `
 --adminConnectorPort 4444 `
 --replicationPort 8989 `
 --profile ds-evaluation `
 --start `
 --acceptLicense
Validating parameters..... Done
Configuring certificates..... Done
Configuring server..... Done
Configuring profile DS evaluation..... Done
Starting directory server..... Done

To see basic server status and configuration, you can launch
C:\path\to\opendj\bat\status
```

Zsh

```
% /path/to/opendj/setup \
 --serverId first-ds \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDn uid=admin \
 --rootUserPassword password \
 --monitorUserPassword password \
 --hostname localhost \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --replicationPort 8989 \
 --profile ds-evaluation \
 --start \
 --acceptLicense
Validating parameters..... Done
Configuring certificates..... Done
Configuring server... Done
Configuring profile DS evaluation..... Done
Starting directory server.............. Done

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

The `setup` command shown here has the following options:

`--serverId first-ds`

A server identifier string that is unique across servers in your deployment.

`--deploymentId <deployment-id>`

The *deployment ID* is a random string generated using the `dskeymgr` command. It is paired with a *deployment ID password*, which is a random string that you choose, and that you must keep secret.

Together, the deployment ID and password serve to generate the shared master key that DS servers in the deployment require for protecting shared encryption secrets. By default, they also serve to generate a private CA and keys for TLS to protect communication between DS servers.

When you deploy multiple servers together, reuse the same deployment ID and password for each server installation.

`--deploymentIdPassword password`

This is a random string that you choose, and that you must keep secret. It is paired with the deployment ID.

`--rootUserDn uid=admin`

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: `uid=admin`.

`--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 domain name for identification between replicated servers.

Using `localhost` is a shortcut suitable only for evaluation on your local computer. In production, 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.

`--ldapsPort 1636`

The reserved port for LDAPS is `636`. Connections to this port are secured with TLS.

Examples in the documentation use `1636`, which is accessible to non-privileged users.

`--httpsPort 8443`

The reserved port for HTTPS is `443`.

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

Examples in the documentation use `8443`, 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.

`--replicationPort 8989`

This is the service port used for replication messages.

The port used in the documentation is `8989`, which is the initial port suggested during interactive setup.

`--profile ds-evaluation`

The setup profile adds hard-coded entries for users like Babs Jensen, and groups like Directory Administrators. It also generates 100,000 sample LDAP user entries. All generated users have the same password, literally `password`. The generated user accounts are helpful for performance testing.

All entries are added under the base DN `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=bjensen,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=bjensen,ou=People,dc=example,dc=com` is under the organizational 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`.

`--start`

By default, the `setup` command does not start the server. This lets you complete any necessary configuration steps before starting the server for the first time, which may initiate the replication process.

In this case, you have no further configuration to do. This option causes the server to start immediately.

`--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.

4. Add the DS tools to your PATH to avoid having to specify the full path for each command:

Bash

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

PowerShell

```
PS C:\path\to> $env:PATH += ";C:\path\to\opendj\bat"
```

Zsh

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

5. Run the `status` command:

Bash

```
$ status \
 --bindDn uid=admin \
 --bindPassword password \
 --hostname localhost \
 --port 4444 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

PowerShell

```
PS C:\path\to> status.bat `
 --bindDn uid=admin `
 --bindPassword password `
 --hostname localhost `
 --port 4444 `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin
```

Zsh

```
% status \
 --bindDn uid=admin \
 --bindPassword password \
 --hostname localhost \
 --port 4444 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

The `status` command uses a secure connection to the administration port. To trust the server's certificate, the command uses the server's own truststore.

Read the output that the `status` command displays.

Learn LDAP

LDAP is short for Lightweight Directory Access Protocol, a standard Internet protocol. The examples that follow show you how to use bundled DS command-line tools to send LDAP requests.

Before you try the examples, set up a server, as described in Install DS. Make sure you added the command-line tools to your PATH:

Bash

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

PowerShell

```
PS C:\path\to> $env:PATH += ";C:\path\to\opendj\bat"
```

Zsh

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

Search

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, `Babs Jensen`, and that Babs Jensen's entry is under the base DN `dc=example,dc=com`. You want to look up Babs Jensen's email and office location. The following command sends an appropriate LDAP search request to the server you installed:

Bash

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(cn=Babs Jensen)" \
 cn mail street l

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
l: San Francisco
mail: bjensen@example.com
street: 201 Mission Street Suite 2900
```

PowerShell

```
PS C:\path\to> ldapsearch.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=bjensen,ou=People,dc=example,dc=com `
 --bindPassword hifalutin `
 --baseDn dc=example,dc=com `
 "(cn=Babs Jensen)" `
 cn mail street l

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
l: San Francisco
mail: bjensen@example.com
street: 201 Mission Street Suite 2900
```

Zsh

```
% ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(cn=Babs Jensen)" \
 cn mail street l

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
l: San Francisco
mail: bjensen@example.com
street: 201 Mission Street Suite 2900
```

Notice the following characteristics of the search:

- The command makes a secure connection to the server using LDAPS.

  The command relies on the server's truststore to trust the CA certificate used to sign the server certificate.

- The base DN option, `--baseDn dc=example,dc=com`, tells the server where to look for Babs Jensen'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=Babs Jensen)"`, which tells the server, "Find entries whose `cn` attribute exactly matches the string `Babs Jensen` without regard to case."

  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 `babs jensen`. The default behavior in LDAP is to ignore case, so `"(cn=Babs Jensen)"`, `"(cn=babs jensen)"`, and `"(CN=BABS JENSEN)"` 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=bjensen,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=bjensen`.

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

For additional examples, see LDAP search.

Modify

You installed the server with the `ds-evaluation` profile. That profile grants access to search Example.com data 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.

In the following example Babs Jensen modifies the description on her own entry:

Bash

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin <<EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: New description
EOF

# MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com
```

PowerShell

```
PS C:\path\to> New-Item -Path . -Name "description.ldif" -ItemType "file" -Value @"
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: New description
"@
PS C:\path\to> ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=bjensen,ou=People,dc=example,dc=com `
 --bindPassword hifalutin `
 description.ldif

# MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com
```

Zsh

```
% ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin <<EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: New description
EOF

# MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com
```

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

- 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.

- 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 LDAP updates, and Passwords and accounts.

Add

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

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

Bash

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --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: chngthspwd
EOF

# ADD operation successful for DN uid=newuser,ou=People,dc=example,dc=com
```

PowerShell

```
PS C:\path\to> New-Item -Path . -Name "user.ldif" -ItemType "file" -Value @"
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: chngthspwd
"@
PS C:\path\to> ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=admin `
 --bindPassword password `
 user.ldif

# ADD operation successful for DN uid=newuser,ou=People,dc=example,dc=com
```

Zsh

```
% ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --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: chngthspwd
EOF

# ADD operation successful for DN uid=newuser,ou=People,dc=example,dc=com
```

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

For additional examples, see LDAP updates.

Delete

The following example deletes the user added in Add:

Bash

```
$ ldapdelete \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --bindPassword password \
 uid=newuser,ou=People,dc=example,dc=com

# DELETE operation successful for DN uid=newuser,ou=People,dc=example,dc=com
```

PowerShell

```
PS C:\path\to> ldapdelete.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=admin `
 --bindPassword password `
 uid=newuser,ou=People,dc=example,dc=com

# DELETE operation successful for DN uid=newuser,ou=People,dc=example,dc=com
```

Zsh

```
% ldapdelete \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --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 LDAP updates.

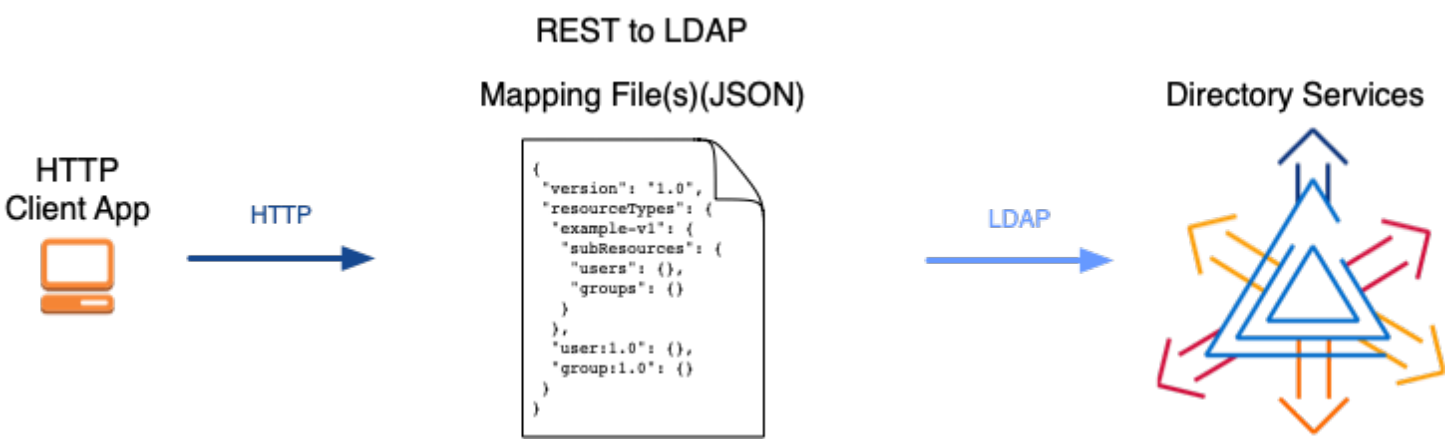
Learn REST/HTTP

The examples that follow show you how to send RESTful HTTP requests to the directory server.

Before you try the examples, set up a server, as described in Install DS.

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.



Prepare

Get the deployment CA certificate used to trust the server in the RESTful examples:

When using the Bash or Zsh examples with `curl` commands, get the deployment CA certificate in PEM format, which the `curl` command can read:

Bash

```
$ dskeymgr \
 export-ca-cert \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --outputFile ca-cert.pem
```

Zsh

```
% dskeymgr \
 export-ca-cert \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --outputFile ca-cert.pem
```

For PowerShell examples, first configure Windows to trust the deployment CA certificate. Import the deployment CA from the server truststore using Microsoft Management Console (MMC):

1. Run Microsoft Management Console (`mmc.exe`).

2. Add the certificates snap-in so you can import the deployment CA certificate:
    - In the console, select File > Add/Remove Snap-in, then Add.
    - Select Certificates from the list of snap-ins and click Add.
    - Finish the wizard.
3. Import the deployment CA certificate using the snap-in:
    - Select Console Root > Trusted Root Certification Authorities > Certificates.
    - In the Action menu, select Import to open the wizard.
    - Use the wizard to import the deployment CA certificate from the server truststore file, `C:\path\to\opendj\config\keystore`.

      The truststore password is the text in the file `C:\path\to\opendj\config\keystore.pin`.

Create

Use the REST API to create a user resource over HTTP:

Bash

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user admin: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": "bjensen",
    "displayName": "Babs Jensen"
  }
 }' \
 https://localhost:8443/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" : "bjensen",
    "_rev" : "<revision>"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  }
}
```

PowerShell

```
PS C:\path\to> $Credentials =
[System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes("admin:password"))
$Headers = @{
    Authorization = "Basic $Credentials"
}
Invoke-RestMethod `
 -Uri https://localhost:8443/api/users `
 -Method Post `
 -Headers $Headers `
 -ContentType application/json `
 -Body @"
 {
  "_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": "bjensen",
    "displayName": "Babs Jensen"
  }
 }
"@ | ConvertTo-JSON

{
    "_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":  "bjensen",
                    "_rev":  "<revision>"
                },
    "contactInformation":  {
                               "telephoneNumber":  "+1 408 555 1212",
                               "emailAddress":  "newuser@example.com"
                           }
}
```

Zsh

```
% curl \
 --request POST \
 --cacert ca-cert.pem \
 --user admin: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": "bjensen",
    "displayName": "Babs Jensen"
  }
 }' \
 "https://localhost:8443/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" : "bjensen",
    "_rev" : "<revision>"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  }
}
```

- The command makes a secure connection to the server using HTTPS.
- The user performing the HTTP POST is the directory superuser.

  The default authorization mechanism for HTTP access is HTTP Basic authentication. The superuser's HTTP user ID, `admin`, is mapped to the LDAP DN, `uid=admin`. REST to LDAP uses the 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.

- 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 DS REST APIs.

For additional details, see DS REST APIs and Create.

Read

Use the REST API to read the user resource created in Create:

Bash

```
$ curl \
 --request GET \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --header "Content-Type: application/json" \
 https://localhost:8443/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" : "bjensen",
    "_rev" : "<revision>"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  }
}
```

PowerShell

```
PS C:\path\to> $Credentials =
[System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes("bjensen:hifalutin"))
$Headers = @{
    Authorization = "Basic $Credentials"
}
Invoke-RestMethod `
 -Uri https://localhost:8443/api/users/newuser `
 -Method Get `
 -Headers $Headers `
 -ContentType application/json | ConvertTo-JSON

{
    "_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":  "bjensen",
                    "_rev":  "<revision>"
                },
    "contactInformation":  {
                               "telephoneNumber":  "+1 408 555 1212",
                               "emailAddress":  "newuser@example.com"
                           }
}
```

Zsh

```
% curl \
 --request GET \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --header "Content-Type: application/json" \
 "https://localhost:8443/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" : "bjensen",
    "_rev" : "<revision>"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  }
}
```

Notice that Babs Jensen 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 DS REST APIs and Read. You can also query collections of resources, as described in Query.

Update

Use the REST API to replace the user resource created in Create:

Bash

```
$ curl \
 --request PUT \
 --cacert ca-cert.pem \
 --user admin: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" : "bjensen",
    "displayName" : "Babs Jensen"
  }
 }' \
 https://localhost:8443/api/users/newuser?_prettyPrint=true

{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<timestamp>",
    "lastModified" : "<timestamp>"
  },
  "displayName" : [ "Updated User" ],
  "name" : {
    "givenName" : "Updated",
    "familyName" : "User"
  },
  "manager" : {
    "_id" : "bjensen",
    "_rev" : "<revision>"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 234 567 8910"
  }
}
```

PowerShell

```
PS C:\path\to> $Credentials =
[System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes("admin:password"))
$Headers = @{
    "Authorization" = "Basic $Credentials"
    "If-Match"      = "*"
}
Invoke-RestMethod `
 -Uri https://localhost:8443/api/users/newuser `
 -Method Put `
 -Headers $Headers `
 -ContentType application/json `
 -Body @"
 {
  "_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" : "bjensen",
    "displayName" : "Babs Jensen"
  }
 }
"@ | ConvertTo-JSON

{
    "_id":  "newuser",
    "_rev":  "<revision>",
    "_schema":  "frapi:opendj:rest2ldap:user:1.0",
    "_meta":  {
                  "created":  "<timestamp>",
                  "lastModified":  "<timestamp>"
              },
    "displayName":  [
                        "Updated User"
                    ],
    "name":  {
                 "givenName":  "Updated",
                 "familyName":  "User"
             },
    "manager":  {
                    "_id":  "bjensen",
                    "_rev":  "<revision>"
                },
    "contactInformation":  {
                               "telephoneNumber":  "+1 234 567 8910"
                           }
}
```

Zsh

```
% curl \
 --request PUT \
 --cacert ca-cert.pem \
 --user admin: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" : "bjensen",
    "displayName" : "Babs Jensen"
  }
 }' \
 "https://localhost:8443/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" : "bjensen",
    "_rev" : "<revision>"
  },
  "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 DS REST APIs and Update. You can also patch resources instead of replacing them entirely. See Patch.

Delete

Use the REST API to delete the user resource updated in Update:

Bash

```
$ curl \
 --request DELETE \
 --cacert ca-cert.pem \
 --user admin:password \
 --header "Content-Type: application/json" \
 https://localhost:8443/api/users/newuser?_prettyPrint=true

{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<timestamp>",
    "lastModified" : "<timestamp>"
  },
  "displayName" : [ "Updated User" ],
  "name" : {
    "givenName" : "Updated",
    "familyName" : "User"
  },
  "manager" : {
    "_id" : "bjensen",
    "_rev" : "<revision>"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 234 567 8910"
  }
}
```

PowerShell

```
PS C:\path\to> $Credentials =
[System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes("admin:password"))
$Headers = @{
    Authorization = "Basic $Credentials"
}
Invoke-RestMethod `
 -Uri https://localhost:8443/api/users/newuser `
 -Method Delete `
 -Headers $Headers `
 -ContentType application/json | ConvertTo-JSON

{
    "_id":  "newuser",
    "_rev":  "<revision>",
    "_schema":  "frapi:opendj:rest2ldap:user:1.0",
    "_meta":  {
                  "created":  "<timestamp>",
                  "lastModified":  "<timestamp>"
              },
    "displayName":  [
                        "Updated User"
                    ],
    "name":  {
                 "givenName":  "Updated",
                 "familyName":  "User"
             },
    "manager":  {
                    "_id":  "bjensen",
                    "_rev":  "<revision>"
                },
    "contactInformation":  {
                               "telephoneNumber":  "+1 234 567 8910"
                           }
}
```

Zsh

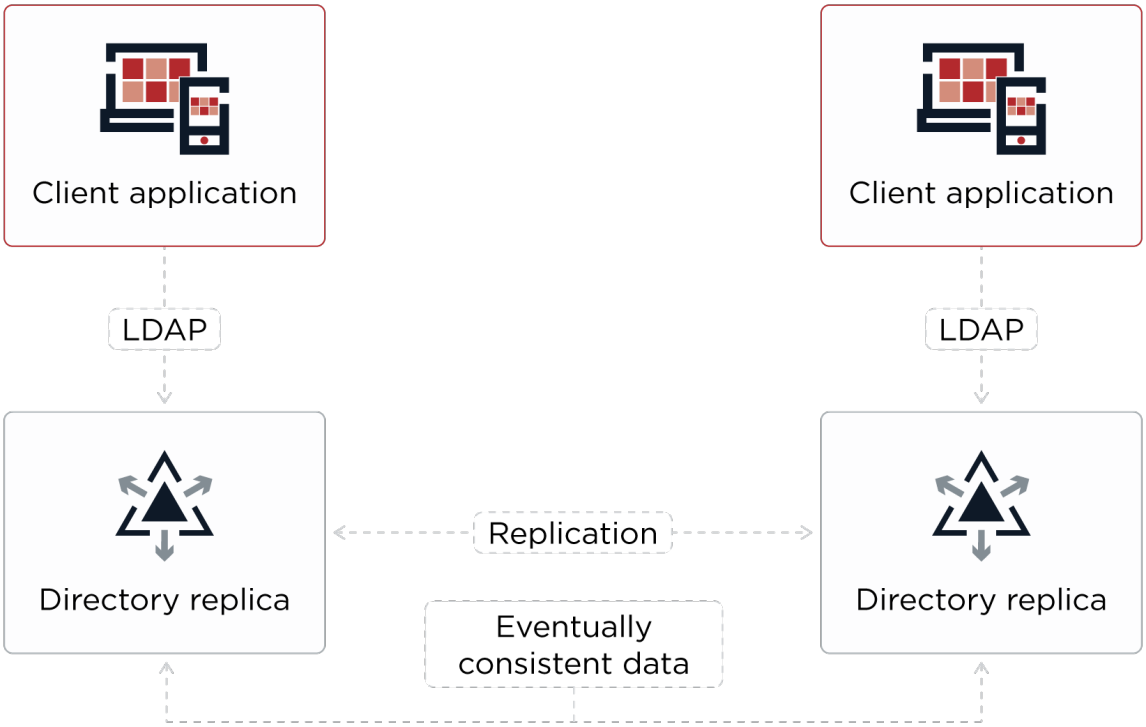
```
% curl \
 --request DELETE \
 --cacert ca-cert.pem \
 --user admin:password \
 --header "Content-Type: application/json" \
 "https://localhost:8443/api/users/newuser?_prettyPrint=true"

{
  "_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" : "bjensen",
    "_rev" : "<revision>"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 234 567 8910",
    "emailAddress" : "updated.user@example.com"
  }
}
```

For additional details, see DS REST APIs and Delete.

Learn replication

Replication provides automatic data synchronization between directory servers. It ensures that all directory servers eventually share a consistent set of directory data.

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



For a full discussion of the subject, see Replication.

Add a replica

High-level steps:

1. Unpack the files for a second directory server in a different folder.
2. Set up the new server as a replica of the first server using the generated `<deployment-id>` from Install DS.

The following example demonstrates the process:

Bash

```
# Unpack files for a second, replica server in a different folder:
cd ~/Downloads && unzip ~/Downloads/DS-7.2.5.zip && mv opendj /path/to/replica

# Set up a second, replica server:
/path/to/replica/setup \
 --serverId second-ds \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDn uid=admin \
 --rootUserPassword password \
 --hostname localhost \
 --ldapPort 11389 \
 --ldapsPort 11636 \
 --adminConnectorPort 14444 \
 --replicationPort 18989 \
 --bootstrapReplicationServer localhost:8989 \
 --profile ds-evaluation \
 --start \
 --acceptLicense
```

PowerShell

```
# Unpack files for a second, replica server in a different folder:
Expand-Archive DS-7.2.5.zip C:\Temp
Rename-Item -Path C:\Temp\opendj -NewName C:\Temp\replica
Move-Item C:\Temp\replica C:\path\to

# Set up a second, replica server:
C:\path\to\replica\setup.bat `
 --serverId second-ds `
 --deploymentId <deployment-id> `
 --deploymentIdPassword password `
 --rootUserDn uid=admin `
 --rootUserPassword password `
 --hostname localhost `
 --ldapPort 11389 `
 --ldapsPort 11636 `
 --adminConnectorPort 14444 `
 --replicationPort 18989 \
 --bootstrapReplicationServer locahost:8989 \
 --profile ds-evaluation `
 --start `
 --acceptLicense
```

Zsh

```
# Unpack files for a second, replica server in a different folder:
cd ~/Downloads && unzip ~/Downloads/DS-7.2.5.zip && mv opendj /path/to/replica

# Set up a second, replica server:
/path/to/replica/setup \
 --serverId second-ds \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDn uid=admin \
 --rootUserPassword password \
 --hostname localhost \
 --ldapPort 11389 \
 --ldapsPort 11636 \
 --adminConnectorPort 14444 \
 --replicationPort 18989 \
 --bootstrapReplicationServer localhost:8989 \
 --profile ds-evaluation \
 --start \
 --acceptLicense
```

Try replication

With the new replica set up and started, demonstrate that replication works:

Bash

```
# Update a description on the first server:
ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin <<EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Replicate this
EOF

# On the first server, read the description to see the effects of your change:
ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(cn=Babs Jensen)" \
 description

# On the second server, read the description to see the change has been replicated:
ldapsearch \
 --hostname localhost \
 --port 11636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(cn=Babs Jensen)" \
 description
```

PowerShell

```
# Update a description on the first server:
New-Item -Path . -Name "mod-desc.ldif" -ItemType "file" -Value @"
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Replicate this
"@

ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=bjensen,ou=People,dc=example,dc=com `
 --bindPassword password `
 mod-desc.ldif

# On the first server, read the description to see the effects of your change:
ldapsearch.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=bjensen,ou=People,dc=example,dc=com `
 --bindPassword hifalutin `
 --baseDn dc=example,dc=com `
 "(cn=Babs Jensen)" `
 description

# On the second server, read the description to see the change has been replicated:
ldapsearch.bat `
 --hostname localhost `
 --port 11636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=bjensen,ou=People,dc=example,dc=com `
 --bindPassword hifalutin `
 --baseDn dc=example,dc=com `
 "(cn=Babs Jensen)" `
 description
```

Zsh

```
# Update a description on the first server:
ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin <<EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Replicate this
EOF

# On the first server, read the description to see the effects of your change:
ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(cn=Babs Jensen)" \
 description

# On the second server, read the description to see the change has been replicated:
ldapsearch \
 --hostname localhost \
 --port 11636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(cn=Babs Jensen)" \
 description
```

Also demonstrate that replication works despite crashes and network interruptions:

Bash

```
# 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 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin <<EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Second server is stopped
EOF

# On the first server, read the description to see the change:
ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(cn=Babs Jensen)" \
 description

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

# On the second server, read the description to check that replication has resumed:
ldapsearch \
 --hostname localhost \
 --port 11636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(cn=Babs Jensen)" \
 description
```

PowerShell

```
# Stop the second server to simulate a network outage or server crash:
C:\path\to\replica\bat\stop-ds.bat

# On the first server, update the description again:
New-Item -Path . -Name "mod-desc2.ldif" -ItemType "file" -Value @"
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Second server is stopped
"@

ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=bjensen,ou=People,dc=example,dc=com `
 --bindPassword password `
mod-desc2.ldif

# On the first server, read the description to see the change:
ldapsearch.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=bjensen,ou=People,dc=example,dc=com `
 --bindPassword hifalutin `
 --baseDn dc=example,dc=com `
 "(cn=Babs Jensen)" `
 description

# Start the second server again to simulate recovery:
C:\path\to\replica\bat\start-ds.bat

# On the second server, read the description to check that replication has resumed:
ldapsearch.bat `
 --hostname localhost `
 --port 11636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=bjensen,ou=People,dc=example,dc=com `
 --bindPassword hifalutin `
 --baseDn dc=example,dc=com `
 "(cn=Babs Jensen)" `
 description
```

Zsh

```
# 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 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin <<EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Second server is stopped
EOF

# On the first server, read the description to see the change:
ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(cn=Babs Jensen)" \
 description

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

# On the second server, read the description to check that replication has resumed:
ldapsearch \
 --hostname localhost \
 --port 11636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(cn=Babs Jensen)" \
 description
```

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. Demonstrate this to check your understanding:

1. Stop the first server.

   Use the `stop-ds` command.

2. Modify an entry on the second server.

   For an example, see Modify.

3. Restart the first server.

   Use the `start-ds` command.

4. Search for the modified entry on the first server to check that replication replays the change.

   For an example, see Search.

Notifications

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.

Replicated DS directory servers publish an external change log over LDAP. This changelog allows authorized client applications to read changes to directory data:

Bash

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --bindPassword password \
 --baseDN cn=changelog \
 --control "ecl:false" \
 "(&)" \
 changes changeLogCookie targetDN
```

PowerShell

```
C:\> ldapsearch.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDN uid=admin `
 --bindPassword password `
 --baseDN cn=changelog `
 --control "ecl:false" `
 "(objectclass=*)" `
 changes changeLogCookie targetDN
```

Zsh

```
% ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN cn=changelog \
 --control "ecl:false" \
 "(&)" \
 changes changeLogCookie targetDN
```

When looking at the output of the command (not shown here), 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 Changelog for notifications.

Measure performance

DS directory servers offer high throughput and low response times for most operations. 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

Note

Before trying the examples that follow, work through the previous examples. You should have two directory server replicas running on your local computer, as described in Learn replication:

Modifications

Measure the LDAP modification rate:

Bash

```
# Run modrate for 10 seconds against the first server:
modrate \
 --maxDuration 10 \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --noRebind \
 --numConnections 4 \
 --numConcurrentRequests 4 \
 --targetDn "uid=user.{1},ou=people,dc=example,dc=com" \
 --argument "rand(0,100000)" \
 --argument "randstr(16)" \
 "description:{2}"

# Read number of modify requests on the LDAPS port:
ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN "cn=LDAPS,cn=connection handlers,cn=monitor" \
 "(&)" \
 ds-mon-requests-modify
```

PowerShell

```
# Run modrate for 10 seconds against the first server, and observe the performance numbers:
modrate.bat `
 --maxDuration 10 `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=bjensen,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(0,100000)" `
 --argument "randstr(16)" `
 "description:{2}"

# Read number of modify requests on the LDAPS port:
ldapsearch.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDN uid=monitor `
 --bindPassword password `
 --baseDN "cn=LDAPS,cn=connection handlers,cn=monitor" `
 "(objectclass=*)" `
 ds-mon-requests-modify
```

Zsh

```
# Run modrate for 10 seconds against the first server:
modrate \
 --maxDuration 10 \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --noRebind \
 --numConnections 4 \
 --numConcurrentRequests 4 \
 --targetDn "uid=user.{1},ou=people,dc=example,dc=com" \
 --argument "rand(0,100000)" \
 --argument "randstr(16)" \
 "description:{2}"

# Read number of modify requests on the LDAPS port:
ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN "cn=LDAPS,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. For an explanation of the command output, see modrate.

Notice that the monitoring attributes hold similar, alternative statistics.

Searches

Measure the LDAP search rate:

Bash

```
# Run searchrate for 10 seconds against the first server:
searchrate \
 --maxDuration 10 \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --noRebind \
 --numConnections 4 \
 --numConcurrentRequests 4 \
 --baseDn "dc=example,dc=com" \
 --argument "rand(0,100000)" \
 "(uid=user.{})"

# Read number of subtree search requests on the LDAPS port:
ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN "cn=LDAPS,cn=connection handlers,cn=monitor" \
 "(&)" \
 ds-mon-requests-search-sub
```

PowerShell

```
# Run searchrate for 10 seconds against the first server:
searchrate.bat `
 --maxDuration 10 `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=bjensen,ou=People,dc=example,dc=com `
 --bindPassword password `
 --noRebind `
 --numConnections 4 `
 --numConcurrentRequests 4 `
 --baseDn "dc=example,dc=com" `
 --argument "rand(0,100000)" `
 "(uid=user.{})"

# Read number of subtree search requests on the LDAPS port:
ldapsearch.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDN uid=monitor `
 --bindPassword password `
 --baseDN "cn=LDAPS,cn=connection handlers,cn=monitor" `
 "(objectclass=*)" `
 ds-mon-requests-search-sub
```

Zsh

```
# Run searchrate for 10 seconds against the first server:
searchrate \
 --maxDuration 10 \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --noRebind \
 --numConnections 4 \
 --numConcurrentRequests 4 \
 --baseDn "dc=example,dc=com" \
 --argument "rand(0,100000)" \
 "(uid=user.{})"

# Read number of subtree search requests on the LDAPS port:
ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN "cn=LDAPS,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. For an explanation of the command output, see searchrate.

Check replication

After running the performance tools, check that both replicas are up to date. The following example uses monitoring metrics to check that replication delay is zero on each replica:

Bash

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN cn=monitor \
 "(ds-mon-current-delay=*)" \
 ds-mon-current-delay

dn: ds-mon-domain-name=dc=example\,dc=com,cn=replicas,cn=replication,cn=monitor
ds-mon-current-delay: 0

dn: ds-mon-server-id=second-ds,cn=remote replicas,ds-mon-domain-
name=dc=example\,dc=com,cn=replicas,cn=replication,cn=monitor
ds-mon-current-delay: 0
```

PowerShell

```
PS C:\path\to> ldapsearch.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDN uid=monitor `
 --bindPassword password `
 --baseDN cn=monitor `
 "(ds-mon-current-delay=*)" `
 ds-mon-current-delay

dn: ds-mon-domain-name=dc=example\,dc=com,cn=replicas,cn=replication,cn=monitor
ds-mon-current-delay: 0

dn: ds-mon-server-id=second-ds,cn=remote replicas,ds-mon-domain-
name=dc=example\,dc=com,cn=replicas,cn=replication,cn=monitor
ds-mon-current-delay: 0
```

Zsh

```
% ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN cn=monitor \
 "(ds-mon-current-delay=*)" \
 ds-mon-current-delay

dn: ds-mon-domain-name=dc=example\,dc=com,cn=replicas,cn=replication,cn=monitor
ds-mon-current-delay: 0

dn: ds-mon-server-id=second-ds,cn=remote replicas,ds-mon-domain-
name=dc=example\,dc=com,cn=replicas,cn=replication,cn=monitor
ds-mon-current-delay: 0
```

Learn access control

Until now, you have used the evaluation setup profile. The evaluation profile makes it easy to access Example.com data. It helps you learn and demonstrate directory services without explicitly granting access after server setup.

In a production directory service where security is important, access is under tighter control. In most cases, access is denied by default to prevent accidental information leaks. You must explicitly grant access where required. To grant access, use access control instructions (ACIs).

> **Note**
>
> The sample ACIs described here demonstrate some but not all ACI features.
> For details, see Access control.

About ACIs

ACIs are implemented as *operational LDAP attributes*. An operational attribute is not meant to store application data, but to influence server behavior. Operational attributes are often left hidden from normal users. A server does not return operational attributes on an entry unless explicitly requested.

Each ACI influences server behavior by indicating:

- Which directory data it targets
- Which permissions it allows or denies
- Which users or groups it applies to
- Under which conditions (time, network origin, connection security, user properties) it applies

The following example ACI gives users access to change their own passwords:

```
aci: (targetattr = "authPassword || userPassword")
 (version 3.0;acl "Allow users to change their own passwords";
 allow (write)(userdn = "ldap:///self");)
```

Consider the characteristics of this ACI attribute:

Target Entries and Scope

The target entries and scope for this ACI are implicit.

The default target is the entry with this `aci` attribute.

The default scope includes the target entry and all its subordinates.

In other words, if you set this ACI on `ou=People,dc=example,dc=com`, it affects all users under that base entry. For example, Babs Jensen, `uid=bjensen,ou=People,dc=example,dc=com`, can set her own password.

Target Attributes

This ACI affects operations on either of the standard password attributes: `(targetattr = "authPassword || userPassword")`.

The ACI only has an effect when an operation targets either `authPassword` or `userPassword`, and any subtypes of those attribute types.

Permissions

This ACI affects only operations that change affected attributes: `allow (write)`.

If this is the only ACI that targets password attributes, users have access to change their own passwords, but they do not have access to *read* passwords.

Subjects

This ACI has an effect when the target entry is the same as the bind DN: `(userdn = "ldap:///self")`.

This means that the user must have authenticated to change their password.

Documentation

The wrapper around the permissions and subjects contains human-readable documentation about the ACI: `(version 3.0;acl "Allow users to change their own passwords"; … ;)`.

Version 3.0 is the only supported ACI version.

Conditions

This ACI does not define any conditions. It applies all the time, for connections from all networks, and so forth.

Server configuration settings can further constrain how clients connect. Such constraints are not specified by this ACI, however.

Use ACIs

To write ACI attributes:

- A user must have the `modify-acl` administrative privilege.

  Privileges are server configuration settings that control access to administrative operations.

- An ACI must give the user permission to change `aci` attributes.

> **Important**
>
> By default, only the directory superuser has the right to add, delete, or modify ACI attributes. In fact, the directory superuser has a privilege, `bypass-acl`, that allows the account to perform operations without regard to ACIs. Any account with permissions to change ACIs is dangerous, because the power can be misused. The user with permissions to change ACIs can give themselves full access to all directory data in their scope.

Prepare to use the examples:

Use each server's `stop-ds` command to stop any DS servers running on your computer.

This lets the new server use ports that might already be in use by another server.

1. Download the Example.ldif file, shown in the following listing:

```
#
# Copyright 2020-2022 ForgeRock AS. All Rights Reserved
#
# Use of this code requires a commercial software license with ForgeRock AS.
# or with one of its affiliates. All use shall be exclusively subject
# to such license between the licensee and ForgeRock AS.
#
dn: dc=example,dc=com
objectClass: domain
objectClass: top
dc: example

dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: Groups

dn: ou=Self Service,ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
description: Groups that authenticated users can manage on their own
ou: Self Service

dn: ou=People,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
description: Description on ou=People
ou: People

dn: uid=ACI Admin,ou=People,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
cn: ACI Admin
givenName: ACI
mail: aci-admin@example.com
ou: People
sn: Admin
uid: ACI Admin
userPassword: 5up35tr0ng

dn: uid=bjensen,ou=People,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
cn: Babs Jensen
givenName: Barbara
mail: bjensen@example.com
ou: People
sn: Jensen
uid: bjensen
userPassword: 5up35tr0ng
```

2. Save the file to your computer's temporary directory, such as `/tmp` or `C:\Temp` .

1. Unzip the DS server `.zip` file into the folder where you want to install the server.

2. Set up the directory server using the LDIF you downloaded.

   Set up the server without the evaluation setup profile, *so the access control settings are secure by default.* The default password policies require stronger passwords. The configuration grants very little access to regular users. Only `uid=admin` has access to the data:

Bash

```
$ /path/to/opendj/setup \
 --serverId learn-acis \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDn uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --hostname localhost \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --acceptLicense
$ dsconfig \
 create-backend \
 --backend-name exampleData \
 --type je \
 --set enabled:true \
 --set base-dn:dc=example,dc=com \
 --offline \
 --no-prompt
$ import-ldif \
 --backendId exampleData \
 --ldifFile /tmp/Example.ldif \
 --offline
$ start-ds --quiet
```

PowerShell

```
PS C:\path\to> C:\path\to\opendj\setup.bat `
 --serverId learn-acis `
 --deploymentId <deployment-id> `
 --deploymentIdPassword password `
 --rootUserDn uid=admin `
 --rootUserPassword str0ngAdm1nPa55word `
 --hostname localhost `
 --ldapPort 1389 `
 --ldapsPort 1636 `
 --httpsPort 8443 `
 --adminConnectorPort 4444 `
 --acceptLicense
PS C:\path\to> C:\path\to\opendj\bat\dsconfig.bat `
 create-backend `
 --backend-name exampleData `
 --type je `
 --set enabled:true `
 --set base-dn:dc=example,dc=com `
 --offline `
 --no-prompt
PS C:\path\to> C:\path\to\opendj\bat\import-ldif.bat `
 --backendId exampleData `
 --ldifFile C:\Temp\Example.ldif `
 --offline
PS C:\path\to> C:\path\to\opendj\bat\start-ds.bat --quiet
```

Zsh

```
% /path/to/opendj/setup \
 --serverId learn-acis \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDn uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --hostname localhost \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --acceptLicense
% dsconfig \
 create-backend \
 --backend-name exampleData \
 --type je \
 --set enabled:true \
 --set base-dn:dc=example,dc=com \
 --offline \
 --no-prompt
% import-ldif \
 --backendId exampleData \
 --ldifFile /tmp/Example.ldif \
 --offline
% start-ds --quiet
```

Grant the `ACI Admin` user access to modify ACIs:

Bash

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --bindPassword str0ngAdm1nPa55word << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "aci") (version 3.0;acl "ACI Admin can manage ACI attributes";
 allow (write) userdn = "ldap:///uid=ACI Admin,ou=People,dc=example,dc=com";)

dn: uid=ACI Admin,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: modify-acl
EOF
```

PowerShell

```
PS C:\path\to> New-Item -Path . -Name "aci-admin.ldif" -ItemType "file" -Value @"
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "aci") (version 3.0;acl "ACI Admin can manage ACI attributes";
 allow (write) userdn = "ldap:///uid=ACI Admin,ou=People,dc=example,dc=com";)

dn: uid=ACI Admin,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: modify-acl
"@
PS C:\path\to> ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn uid=admin `
 --bindPassword str0ngAdm1nPa55word `
 aci-admin.ldif
```

Zsh

```
% ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --bindPassword str0ngAdm1nPa55word << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "aci") (version 3.0;acl "ACI Admin can manage ACI attributes";
 allow (write) userdn = "ldap:///uid=ACI Admin,ou=People,dc=example,dc=com";)

dn: uid=ACI Admin,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: modify-acl
EOF
```

Try examples from Learn LDAP.

You find that Babs Jensen does not have the access that she had with the evaluation setup profile. For production servers, the best practice is to grant access only when required.

Examples

Prepare to use the examples before trying them. The `ACI Admin` account must have access to manage ACIs. After you add an example ACI, test users' access. For inspiration, see the examples in Learn LDAP.

ACI syntax is powerful, and sometimes difficult to get right. For details, see Access control.

ACI: access own entry

The following example grants authenticated users access to read their own entry, and modify some attributes:

Bash

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Users can read their entries";
 allow (read, search, compare) (userdn = "ldap:///self");)
-
add: aci
aci: (targetattr = "authPassword || description || displayName || homePhone ||
 jpegPhoto || preferredLanguage || userPassword")
 (version 3.0;acl "Self-service modifications for basic attributes";
 allow (write) (userdn = "ldap:///self");)
EOF
```

PowerShell

```
PS C:\path\to> New-Item -Path . -Name "self-access.ldif" -ItemType "file" -Value @"
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Users can read their entries";
 allow (read, search, compare) (userdn = "ldap:///self");)
-
add: aci
aci: (targetattr = "authPassword || description || displayName || homePhone ||
 jpegPhoto || preferredLanguage || userPassword")
 (version 3.0;acl "Self-service modifications for basic attributes";
 allow (write) (userdn = "ldap:///self");)
"@
PS C:\path\to> ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" `
 --bindPassword 5up35tr0ng `
 self-access.ldif
```

Zsh

```
% ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Users can read their entries";
 allow (read, search, compare) (userdn = "ldap:///self");)
-
add: aci
aci: (targetattr = "authPassword || description || displayName || homePhone ||
 jpegPhoto || preferredLanguage || userPassword")
 (version 3.0;acl "Self-service modifications for basic attributes";
 allow (write) (userdn = "ldap:///self");)
EOF
```

In this example, the list of attributes that users can read includes all user attributes. The list that users can modify is limited. Other attributes might be governed by other applications. For example, a user's manager might only be changed through an HR system. Perhaps the IT department is responsible for all changes to email addresses.

ACI: access subSchemaSubEntry attribute

The `subSchemaSubEntry` attribute indicates the entry holding the LDAP schema definitions that apply to the current entry. Many applications retrieve this attribute, and the associated schema, to properly display or validate attribute values.

The following example demonstrates how to grant access to read this attribute on directory entries:

Bash

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "subSchemaSubEntry")
 (version 3.0;acl "Authenticated users can read subSchemaSubEntry";
 allow (read, search, compare) (userdn = "ldap:///all");)
EOF
```

PowerShell

```
PS C:\path\to> New-Item -Path . -Name "subSchemaSubentry-access.ldif" -ItemType "file" -Value @"
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "subSchemaSubEntry")
 (version 3.0;acl "Authenticated users can read subSchemaSubEntry";
 allow (read, search, compare) (userdn = "ldap:///all");)
"@
PS C:\path\to> ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" `
 --bindPassword 5up35tr0ng `
 subSchemaSubentry-access.ldif
```

Zsh

```
% ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "subSchemaSubEntry")
 (version 3.0;acl "Authenticated users can read subSchemaSubEntry";
 allow (read, search, compare) (userdn = "ldap:///all");)
EOF
```

ACI: manage group membership

For some static groups, you might choose to let users manage their own memberships. The following example lets members of self-service groups manage their own membership:

Bash

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: ou=Self Service,ou=Groups,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "member") (version 3.0;acl "Self registration";
 allow (selfwrite) (userdn = "ldap:///uid=*,ou=People,dc=example,dc=com");)
EOF
```

PowerShell

```
PS C:\path\to> New-Item -Path . -Name "self-service-groups.ldif" -ItemType "file" -Value @"
dn: ou=Self Service,ou=Groups,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "member") (version 3.0;acl "Self registration";
 allow (selfwrite) (userdn = "ldap:///uid=*,ou=People,dc=example,dc=com");)
"@
PS C:\path\to> ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" `
 --bindPassword 5up35tr0ng `
 self-service-groups.ldif
```

Zsh

```
% ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: ou=Self Service,ou=Groups,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "member") (version 3.0;acl "Self registration";
 allow (selfwrite) (userdn = "ldap:///uid=*,ou=People,dc=example,dc=com");)
EOF
```

The `selfwrite` permission is for adding or deleting one's own DN from a group.

ACI: manage self-service groups

This example lets users create and delete self-managed groups:

Bash

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: ou=Self Service,ou=Groups,dc=example,dc=com
changetype: modify
add: aci
aci: (targattrfilters="add=objectClass:(objectClass=groupOfNames)")
 (version 3.0; acl "Users can create self-service groups";
 allow (add) (userdn = "ldap:///uid=*,ou=People,dc=example,dc=com");)
-
add: aci
aci: (version 3.0; acl "Owner can delete self-service groups";
 allow (delete) (userattr = "owner#USERDN");)
EOF
```

PowerShell

```
PS C:\path\to> New-Item -Path . -Name "self-managed-groups.ldif" -ItemType "file" -Value @"
dn: ou=Self Service,ou=Groups,dc=example,dc=com
changetype: modify
add: aci
aci: (targattrfilters="add=objectClass:(objectClass=groupOfNames)")
 (version 3.0; acl "Users can create self-service groups";
 allow (add) (userdn = "ldap:///uid=*,ou=People,dc=example,dc=com");)
-
add: aci
aci: (version 3.0; acl "Owner can delete self-service groups";
 allow (delete) (userattr = "owner#USERDN");)
"@
PS C:\path\to> ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" `
 --bindPassword 5up35tr0ng `
 self-managed-groups.ldif
```

Zsh

```
% ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: ou=Self Service,ou=Groups,dc=example,dc=com
changetype: modify
add: aci
aci: (targattrfilters="add=objectClass:(objectClass=groupOfNames)")
 (version 3.0; acl "Users can create self-service groups";
 allow (add) (userdn = "ldap:///uid=*,ou=People,dc=example,dc=com");)
-
add: aci
aci: (version 3.0; acl "Owner can delete self-service groups";
 allow (delete) (userattr = "owner#USERDN");)
EOF
```

ACI: full access

The following ACI grants Babs Jensen permission to perform all LDAP operations, allowing her full administrator access to the directory data under `dc=example,dc=com`. Babs can read and write directory data, rename and move entries, and use proxied authorization. Some operations also require administrative privileges not shown in this example:

Bash

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "* || +") (version 3.0;acl "Babs has full access";
 allow (all, export, import, proxy) (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
EOF
```

PowerShell

```
PS C:\path\to> New-Item -Path . -Name "full-access.ldif" -ItemType "file" -Value @"
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "* || +") (version 3.0;acl "Babs has full access";
 allow (all, export, import, proxy) (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
"@
PS C:\path\to> ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" `
 --bindPassword 5up35tr0ng `
 full-access.ldif
```

Zsh

```
% ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "* || +") (version 3.0;acl "Babs has full access";
 allow (all, export, import, proxy) (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
EOF
```

`(targetattr = "* || +")` permits access to all user attributes and all operational attributes. `allow (all, import, export, proxy)` permits all user operations, modify DN operations, and proxied authorization. Notice that `all` does not allow modify DN and proxied authorization.

ACI: anonymous reads and searches

In LDAP, an anonymous user is one who does not provide bind credentials. By default, most setup profiles only allow anonymous access to read information about the server's capabilities, or before using the StartTLS operation to get a secure connection before providing credentials.

Unless you set up the server with the evaluation profile, anonymous users cannot read application data by default. You can grant them access, however. First, change the global configuration to allow unauthenticated requests. Second, add an ACI to grant access to the entries.

The following command changes the global configuration property, `unauthenticated-requests-policy`, to allow unauthenticated requests:

Bash

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword str0ngAdm1nPa55word \
 --set unauthenticated-requests-policy:allow \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

PowerShell

```
PS C:\path\to> dsconfig.bat `
 set-global-configuration-prop `
 --hostname localhost `
 --port 4444 `
 --bindDN uid=admin `
 --bindPassword str0ngAdm1nPa55word `
 --set unauthenticated-requests-policy:allow `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --no-prompt
```

Zsh

```
% dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword str0ngAdm1nPa55word \
 --set unauthenticated-requests-policy:allow \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

This ACI makes all user attributes in `dc=example,dc=com` data (except passwords) world-readable:

Bash

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr != "authPassword || userPassword") (version 3.0;acl "Anonymous read-search access";
 allow (read, search, compare) (userdn = "ldap:///anyone");)
EOF
```

PowerShell

```
PS C:\path\to> New-Item -Path . -Name "anon-access.ldif" -ItemType "file" -Value @"
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr != "authPassword || userPassword") (version 3.0;acl "Anonymous read-search access";
 allow (read, search, compare) (userdn = "ldap:///anyone");)
"@
PS C:\path\to> ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" `
 --bindPassword 5up35tr0ng `
 anon-access.ldif
```

Zsh

```
% ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr != "authPassword || userPassword") (version 3.0;acl "Anonymous read-search access";
 allow (read, search, compare) (userdn = "ldap:///anyone");)
EOF
```

Notice that `ldap:///anyone` designates anonymous users and authenticated users. Do not confuse that with `ldap:///all`, which designates authenticated users only.

ACI: permit insecure access over loopback only

This ACI uses IP address and Security Strength Factor subjects to prevent insecure remote access to `dc=example,dc=com` data. In most cases, you explicitly grant permission with `allow`, making it easier to understand and to explain why the server permits a given operation. This demonstrates one use case where it makes sense to *deny* permission:

Bash

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "* || +") (version 3.0;acl "Restrict insecure LDAP to the loopback address";
 deny (all) (ip != "127.0.0.1" and ssf <= "1");)
EOF
```

PowerShell

```
PS C:\path\to> New-Item -Path . -Name "deny-cleartext.ldif" -ItemType "file" -Value @"
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "* || +") (version 3.0;acl "Restrict cleartext LDAP to the loopback address";
 deny (all) (ip != "127.0.0.1" and ssf <= "1");)
"@
PS C:\path\to> ldapmodify.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\path\to\opendj\config\keystore.pin `
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" `
 --bindPassword 5up35tr0ng `
 deny-cleartext.ldif
```

Zsh

```
% ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn "uid=ACI Admin,ou=People,dc=example,dc=com" \
 --bindPassword 5up35tr0ng << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "* || +") (version 3.0;acl "Restrict insecure LDAP to the loopback address";
 deny (all) (ip != "127.0.0.1" and ssf <= "1");)
EOF
```

- `ssf = 1` means that TLS is configured without a cipher. The server verifies integrity using packet checksums, but all content is sent in plain text.
- `ssf = 0` means that the content is sent plain text with no connection security.

About directories

A directory resembles a dictionary or a phone book. If you know a word, you can look up its entry in the dictionary to learn its definition or its pronunciation. If you know a name, you can look up its entry in the phone book to find the telephone number and street address associated with the name. If you are bored, curious, or have lots of time, you can also read through the dictionary, phone book, or directory, entry after entry.

Where a directory differs from a paper dictionary or phone book is in how entries are indexed. Dictionaries typically have one index, which is words in alphabetical order. Phone books, have one index as well, which is names in alphabetical order. Directories' entries, however, are often indexed for multiple attributes, including names, user identifiers, email addresses, and telephone numbers. This means you can look up a directory account by the user's name, their user identifier, their email address, or their telephone number, for example.

ForgeRock Directory Services are based on the Lightweight Directory Access Protocol (LDAP). Nearly all of what follows is an introduction to LDAP.

ForgeRock Directory Services also provide RESTful HTTP access to directory data. As a directory user, you will find it useful to understand the underlying LDAP model even if most users are accessing the directory over HTTP rather than LDAP.

History

Phone companies have been managing directories for many decades. The Internet itself has relied on distributed directory services like DNS since the mid 1980s.

It was not until the late 1980s, however, that experts from what is now the International Telecommunications Union published the X.500 set of international standards, including Directory Access Protocol. The X.500 standards specify Open Systems Interconnect (OSI) protocols and data definitions for general purpose directory services. The X.500 standards were designed to meet the needs of systems built according to the X.400 standards, covering electronic mail services.

Lightweight Directory Access Protocol has been around since the early 1990s. LDAP was originally developed as an alternative protocol that would allow directory access over Internet protocols rather than OSI protocols, and be lightweight enough for desktop implementations. By the mid-1990s, LDAP directory servers became generally available and widely used.

Until the late 1990s, LDAP directory servers were designed primarily with quick lookups and high availability for lookups in mind. LDAP directory servers replicate data. When an update is made, that update is applied to other peer directory servers. Thus, if one directory server goes down, lookups can continue on other servers. Furthermore, if a directory service needs to support more lookups, the administrator can simply add another directory server to replicate with its peers.

As organizations rolled out larger and larger directories serving more and more applications, they discovered the need for high availability and fast updates. Around the year 2000, directories began to support multi-master replication; that is, replication with multiple read-write servers. The organizations with the very largest directories became concerned about replicating so many changes.

The DS code base began in the mid-2000s, when engineers solving the update performance issue decided that the cost of adapting the existing C-based directory technology for high-performance updates would be higher than the cost of building new, high-performance directory using Java technology.

LDAP data

LDAP directory data is organized into entries, similar to the entries for words in the dictionary, or for subscriber names in the phone book:

```
dn: uid=bjensen,ou=People,dc=example,dc=com
uid: bjensen
cn: Babs Jensen
cn: Barbara Jensen
facsimileTelephoneNumber: +1 408 555 1992
gidNumber: 1000
givenName: Barbara
homeDirectory: /home/bjensen
l: San Francisco
mail: bjensen@example.com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: posixAccount
objectClass: top
ou: People
ou: Product Development
roomNumber: 0209
sn: Jensen
telephoneNumber: +1 408 555 1862
uidNumber: 1076
```

Barbara Jensen's entry has a number of attributes, such as `uid: bjensen`, `telephoneNumber: +1 408 555 1862`, and `objectClass: posixAccount`. (The `objectClass` attribute type indicates which types of attributes are required and allowed for the entry. As the entries object classes can be updated online, and even the definitions of object classes and attributes are expressed as entries that can be updated online, directory data is extensible on the fly.) When you look up her entry in the directory, you specify one or more attributes and values to match. The directory server then returns entries with attribute values that match what you specified.

The attributes you search for are indexed in the directory, so the directory server can retrieve them more quickly. Attribute values are not necessarily strings. Some attribute values, like certificates and photos, are binary.

Each entry also has a unique identifier, shown at the top of the entry, `dn: uid=bjensen,ou=People,dc=example,dc=com`. DN is an acronym for *Distinguished Name*. No two entries in the directory have the same distinguished name. DNs are typically composed of case-insensitive attributes.

Sometimes distinguished names include characters that you must escape. The following example shows an entry that includes escaped characters in the DN:

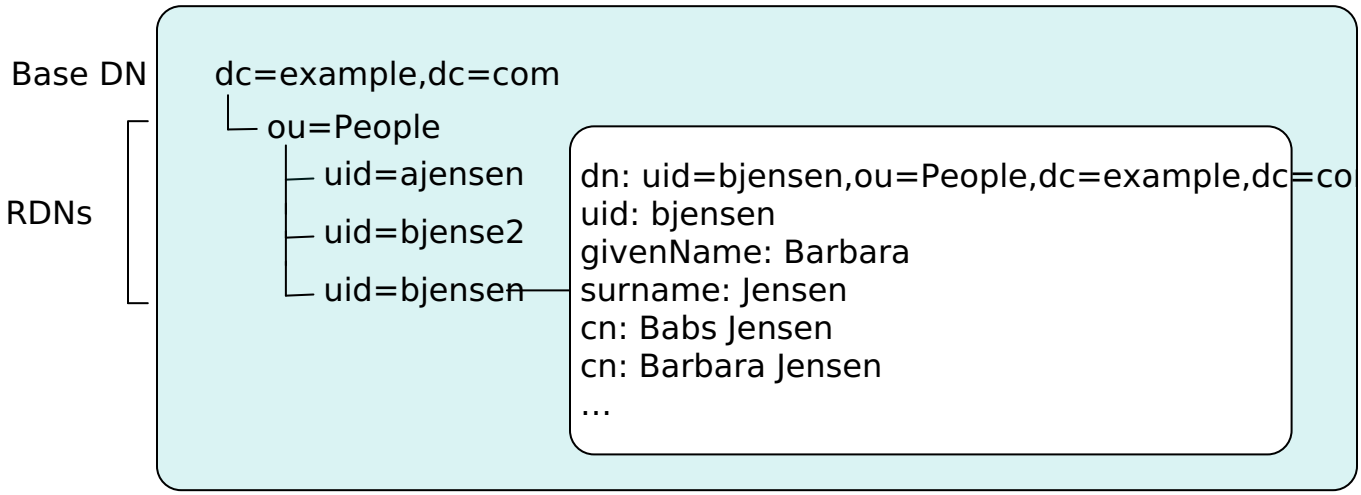
Bash

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=escape)"

dn: cn=DN Escape Characters \" # \+ \, \; \< = \> \\,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
givenName: DN Escape Characters
uid: escape
cn: DN Escape Characters " # + , ; < = > \
sn: " # + , ; < = > \
mail: escape@example.com
```

PowerShell

```
PS C:\path\to> ldapsearch.bat `
 --hostname localhost `
 --port 1636 `
 --useSsl `
 --usePkcs12TrustStore /path/to/opendj/config/keystore `
 --trustStorePassword:file /path/to/opendj/config/keystore.pin `
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com `
 --bindPassword bribery `
 --baseDN dc=example,dc=com `
 "(uid=escape)"

dn: cn=DN Escape Characters \" # \+ \, \; \< = \> \\,dc=example,dc=com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
givenName: DN Escape Characters
uid: escape
cn: DN Escape Characters " # + , ; < = > \
sn: " # + , ; < = > \
mail: escape@example.com
```

LDAP entries are arranged hierarchically in the directory. The hierarchical organization resembles a file system on a PC or a web server, often imagined as an upside down tree structure, or a pyramid. The distinguished name consists of components separated by commas, `uid=bjensen,ou=People,dc=example,dc=com`. The names are little-endian. The components reflect the hierarchy of directory entries.



Barbara Jensen's entry is located under an entry with DN `ou=People,dc=example,dc=com`, an organizational unit and parent entry for the people at Example.com. The `ou=People` entry is located under the entry with DN `dc=example,dc=com`, the base entry for Example.com. DC is an acronym for *Domain Component*. The directory has other base entries, such as `cn=config`, under which the configuration is accessible through LDAP.

A directory can serve multiple organizations, too. You might find `dc=example,dc=com`, `dc=mycompany,dc=com`, and `o=myOrganization` in the same LDAP directory. Therefore, when you look up entries, you specify the base DN to look under in the same way you need to know whether to look in the New York, Paris, or Tokyo phone book to find a telephone number.

The root entry for the directory, technically the entry with DN `""` (the empty string), is called the *root DSE*. It contains information about what the server supports, including the other base DNs it serves.

A directory server stores two kinds of attributes in a directory entry: *user attributes* and *operational attributes*. User attributes hold the information for users of the directory. All attributes shown in the entry above are user attributes. Operational attributes hold information used by the directory itself. Examples of operational attributes include `entryUUID`, `modifyTimestamp`, and `subschemaSubentry`.

When an LDAP search operation finds an entry in the directory, the directory server returns all the visible user attributes unless the search request restricts the list of attributes by specifying those attributes explicitly. The directory server does not, however, return any operational attributes unless the search request specifically asks for them.

Generally speaking, applications should change only user attributes, and leave updates of operational attributes to the server, relying on public directory server interfaces to change server behavior. An exception is access control instruction (`aci`) attributes, which are operational attributes used to control access to directory data.

Communication

In some client/server applications, like web browsing, a connection is set up and torn down for each client request.

LDAP has a different model. In LDAP, the client application connects to the server and authenticates. The client then requests any number of operations, perhaps processing results in between requests. The client finally disconnects when done, potentially days later.

The standard operations are as follows:

Bind (authenticate)

The first operation in an LDAP session usually involves the client binding to the LDAP server w ith the server authenticating the client. Authentication identifies the client's identity in LDAP terms, the identity which is later used by the server to authorize (or not) access to directory data that the client wants to lookup or change.

If the client does not bind explicitly, the server treats the client as an anonymous client. An anonymous client is allowed to do anything that can be done anonymously. What can be done anonymously depends on access control and configuration settings. The client can also bind again on the same connection.

Search (lookup)

After binding, the client can request that the server return entries based on an LDAP filter, which is an expression that the server uses to find entries that match the request, and a base DN under which to search. For example, to look up all entries for people with the email address `bjensen@example.com` in data for Example.com, you would specify a base DN such as `ou=People,dc=example,dc=com` and the filter `(mail=bjensen@example.com)`.

Compare

After binding, the client can request that the server compare an attribute value that the client specifies with the value stored on an entry in the directory.

Modify

After binding, the client can request that the server change one or more attribute values on an entry. Often administrators do not allow clients to change directory data, so allow appropriate access for client application if they have the right to update data.

Add

After binding, the client can request to add one or more new LDAP entries to the server.

Delete

After binding, the client can request that the server delete one or more entries. To delete an entry with other entries underneath, first delete the children, then the parent.

Modify DN

After binding, the client can request that the server change the distinguished name of the entry. In other words, this renames the entry or moves it to another location. For example, if Barbara changes her unique identifier from `bjensen` to something else, her DN would have to change. For another example, if you decide to consolidate `ou=Customers` and `ou=Employees` under `ou=People` instead, all the entries underneath must change distinguished names.

Renaming entire branches of entries can be a major operation for the directory, so avoid moving entire branches if you can.

Unbind

When done making requests, the client can request an unbind operation to end the LDAP session.

Abandon

When a request seems to be taking too long to complete, or when a search request returns many more matches than desired, the client can send an abandon request to the server to drop the operation in progress.

Controls and extensions

LDAP has standardized two mechanisms for extending the operations directory servers can perform beyond the basic operations listed above. One mechanism involves using LDAP controls. The other mechanism involves using LDAP extended operations.

LDAP controls are information added to an LDAP message to further specify how an LDAP operation should be processed. For example, the Server-Side Sort request control modifies a search to request that the directory server return entries to the client in sorted order. The Subtree Delete request control modifies a delete request so the server also removes child entries of the entry targeted for deletion.

One special search operation that DS servers support is Persistent Search. The client application sets up a Persistent Search to continue receiving new results whenever changes are made to data that is in the scope of the search, using the search as a form of change notification. Persistent Searches are intended to remain connected permanently, though they can be idle for long periods of time.

The directory server can also send response controls in some cases to indicate that the response contains special information. Examples include responses for entry change notification, password policy, and paged results.

For the list of supported LDAP controls, see Supported LDAP controls.

LDAP extended operations are additional LDAP operations not included in the original standard list. For example, the Cancel Extended Operation works like an abandon operation, but finishes with a response from the server after the cancel is complete. The StartTLS Extended Operation allows a client to connect to a server on an unsecure port, then starts Transport Layer Security negotiations to protect communications.

For the list of supported LDAP extended operations, see Supported LDAP extended operations.

Indexes

Directories have indexes for multiple attributes. By default, DS does not let normal users perform searches that are not indexed, because such searches mean DS servers have to scan an entire directory database when looking for matches.

As directory administrator, part of your responsibility is making sure directory data is properly indexed. DS software provides tools for building and rebuilding indexes, for verifying indexes, and for evaluating how well indexes are working.

For help better understanding and managing indexes, read Indexes.

Schema

Some databases are designed to hold huge amounts of data for a particular application. Although such databases can support multiple applications, data organization depends on the applications served.

In contrast, directories are designed for shared, centralized services. Although the first guides to deploying directory services suggested taking inventory of all the applications that would access the directory, today many directory administrators do not even know how many applications use their services. The shared, centralized nature of directory services fosters interoperability in practice. It has helped directory services be successful in the long term.

Part of what makes this possible is the shared model of directory user information, in particular the LDAP schema. LDAP schema defines what the directory can contain. This means that directory entries are not arbitrary data, but tightly codified objects whose attributes are completely predictable from publicly readable definitions. Many schema definitions are in fact standard. They are the same not just across a directory service but across different directory services.

At the same time, unlike some databases, LDAP schema and the data it defines can be extended on the fly while the service is running. LDAP schema is accessible over LDAP. One attribute of every entry is its set of `objectClass` values. This gives you as administrator great flexibility in adapting your directory service to store new data without losing or changing the structure of existing data, and without ever stopping your directory service.

For a closer look, see LDAP schema.

Access control

Directory services support fine-grained access control.

As directory administrator, you can control who has access to what data when, how, where and under what conditions by using access control instructions (ACI). You can allow some directory operations and not others. You can scope access control from the whole directory service down to individual attributes on directory entries. You can specify when, from what host or IP address, and the encryption strength required for an operation.

As ACIs are stored on entries in the directory, you can update access controls while the service is running, and even delegate that control to client applications. DS software combines the strengths of ACIs with separate administrative privileges to help you secure access to directory data.

For more information, read Access control.

Replication

DS replication consists of copying each update to the directory service to multiple directory servers. This brings both redundancy, in the case of network partitions or of crashes, and scalability for read operations. Most directory deployments involve multiple servers replicating together.

When you have replicated servers, all of which are writable, you can have replication conflicts. What if, for example, there is a network outage between two replicas, and meanwhile two different values are written to the same attribute on the same entry on the two replicas?

In nearly all cases, DS replication can resolve these situations automatically without involving you, the directory administrator. This makes your directory service resilient and safe even in the unpredictable real world.

One counterintuitive aspect of replication is that although you add directory *read* capacity by adding replicas to your deployment, you do not add directory *write* capacity by adding replicas. Each write operation must be replayed everywhere. As a result, if you have N servers, you have N write operations to replay.

Replication is also *loosely consistent*. Loosely consistent means that directory data will eventually converge to be the same everywhere, but it will not necessarily be the same everywhere at all times, or even at any time. Client applications sometimes get this wrong when they write to a pool of load balanced directory servers, immediately read back what they wrote, and are surprised that it is not the same. If your users are complaining about this, consider using a directory proxy server to mitigate their poor practices.

DSMLv2

Directory Services Markup Language (DSMLv2) v2.0 became a standard in 2001. DSMLv2 describes directory data and basic directory operations in XML format, so they can be carried in Simple Object Access Protocol (SOAP) messages. DSMLv2 further allows clients to batch multiple operations together in a single request, to be processed either in sequential order or in parallel.

DS software provides support for DSMLv2 as a DSML gateway, which is a servlet that connects to any standard LDAPv3 directory. DSMLv2 opens basic directory services to SOAP-based web services and service oriented architectures.

To set up DSMLv2 access, see Install a DSML gateway.

HTTP access

DS software can expose directory data as JSON resources over HTTP to REST clients, providing easy access to directory data for developers who are not familiar with LDAP. RESTful access depends on a configuration that describes how the JSON representation maps to LDAP entries.

Although client applications have no need to understand LDAP, the underlying implementation still uses the LDAP model for its operations. The mapping adds some overhead.

Furthermore, depending on the configuration, individual JSON resources can require multiple LDAP operations. For example, an LDAP user entry represents `manager` as a DN (of the manager's entry). The same manager might be represented in JSON as an object holding the manager's user ID and full name, in which case the software must look up the manager's entry to resolve the mapping for the manager portion of the JSON resource, in addition to looking up the user's entry. As another example, suppose a large group is represented in LDAP as a set of 100,000 DNs. If the JSON resource is configured so that a member is represented by its name, then listing that resource would involve 100,000 LDAP searches to translate DNs to names.

A primary distinction between LDAP entries and JSON resources is that LDAP entries hold sets of attributes values, whereas JSON resources are documents containing arbitrarily nested objects. As LDAP data is governed by schema, almost no LDAP objects are arbitrary collections of data. (LDAP has the object class `extensibleObject`, but its should be used sparingly.) JSON resources can hold arrays, ordered collections that can contain duplicates. LDAP attributes are sets, unordered collections without duplicates. For most directory and identity data, these distinctions do not matter. You are likely to run into them, however, if you try to turn your directory into a document store for arbitrary JSON resources.

Despite some extra cost in terms of system resources, exposing directory data over HTTP can unlock directory services for a new generation of applications. The configuration provides flexible mapping, so that you can configure views that correspond to how client applications need to see directory data.

DS software also give you a deployment choice for HTTP access. You can deploy the REST to LDAP gateway, which is a servlet that connects to any standard LDAPv3 directory, or you can activate the HTTP connection handler on a server to allow direct and more efficient HTTP and HTTPS access.

Deployment

This page serves as an introduction. When you have understood enough of the concepts to build the directory services that you want to deploy, you must still build a prototype and test it before you roll out shared, centralized services for your organization.

Start with Deployment when beginning your project.

Best practices

Follow these best practices for writing effective, maintainable, high-performance directory client applications.

Authenticate correctly

Unless your application performs only read operations, authenticate to the directory server. Some directory services require authentication to read directory data.

Once you authenticate (bind), directory servers make authorization decisions based on your identity. With servers that support proxied authorization, once authenticated, your application can request an operation on behalf of another identity, such as the identity of the end user.

Your application therefore should have an account, such as `cn=My App,ou=Apps,dc=example,dc=com`. The directory administrator can authorize appropriate access for your application's account, and monitor your application's requests to help you troubleshoot problems if they arise.

Applications can use simple, password-based authentication. When using password-based authentication, use secure connections to protect credentials over the network. For applications, prefer certificate-based authentication if possible.

Reuse connections

LDAP is a stateful protocol. You authenticate (bind), you perform operations, you unbind. The server maintains a context that lets it make authorization decisions concerning your requests. Therefore, reuse connections whenever possible.

Because LDAP supports asynchronous requests, it is normal and expected to make multiple requests over the same connection. Your application can share a pool of connections to avoid the overhead of setting them up and tearing them down.

Check connection health

In a network built for HTTP applications, your long-lived LDAP connections can get cut by network equipment configured to treat idle and old connections as stale resources to reclaim.

When you maintain a particularly long-lived connection, such as a connection for a persistent search, periodically perform a health check to maintain the connection operational.

A health check involves reading or writing an attribute on a well-known entry in your data. It can serve the purposes of maintaining the connection operational, and of verifying access to your data. A success result for a read indicates that the data is available, and the application can read it. A success result for a write indicates that the data is available, and the application can write to it. The exact check to perform depends on how your application uses the directory. Under some circumstances, your data might be temporarily read-only, for example.

When using a connection timeout, take care not to set the timeout so low that long operations, such as unindexed searches, fail to complete before the timeout.

Request exactly what you need all at once

By the time your application makes it to production, you should know what attributes you want. Request them explicitly, and request all the attributes in the same search.

For example, if you require `mail` and `cn`, then specify both attributes in your search request.

Use specific LDAP filters

The difference in results between a general filter `(mail=*@example.com)`, and a good, specific filter like `(mail=user@example.com)` can be huge numbers of entries and enormous amounts of processing time, both for the directory server that has to return search results, and for your application that has to sort through them.

Many use cases can be handled with short, specific filters. As a rule, prefer equality filters over substring filters.

DS servers reject unindexed searches by default, because unindexed searches are resource-intensive. If your application needs to use a filter that results in an unindexed search, work with the directory administrator to find a solution, such as adding the indexes required for your search filters.

Always use `&` with `!` to restrict the potential result set before returning all entries that do not match part of the filter. For example, `(&(location=Oslo)(!(mail=birthday.girl@example.com)))`.

Make modifications specific

Specific modifications help directory servers apply and replicate your changes more effectively.

When you modify attributes with multiple values, such as a static group member attribute, replace or delete specific values individually, rather than replacing the entire list of values.

Trust result codes

Trust the LDAP result code from the directory server. For example, if you request a modification, and you get a success result, consider the operation a success. Do not immediately issue a search to get the modified entry.

LDAP replication model is loosely convergent. In other words, the directory server sends you the success result *before* replicating the change to every directory server replica across the network. If you issue a read immediately after a write, a load balancer may direct the request to another replica. The result might differ from what you expect.

The loosely convergent model means that the entry could have changed since you read it. If needed, use LDAP assertions to set conditions for your LDAP operations.

Handle input securely

When taking input directly from a user or another program, use appropriate methods to sanitize the data. Failure to sanitize the input data can leave your application vulnerable to injection attacks.

For Java applications, the Directory Services `format()` methods for filters and DNs are similar to the Java `String.format()` methods. In addition to formatting the output, they escape the input objects. When building a search filter, use one of the methods of the DS APIs to escape input.

Check group membership on the account, not the group

Reading an entire large static group entry to check membership is wasteful.

If you need to determine which groups an account belongs to, request the DS virtual attribute, `isMemberOf`, when you read the account entry. Other directory servers use other names for this attribute that identifies the groups to an account belongs to.

Check support for features you use

Directory servers expose their capabilities as operational attribute values on the root DSE, which is the entry whose DN is an empty string, `""`.

This lets your application discover capabilities at run time, rather than storing configuration separately. Putting effort into checking directory capabilities makes your application easier to deploy and to maintain.

For example, rather than hard-coding `dc=example,dc=com` as a base DN in your configuration, read the root DSE `namingContexts` attribute.

Directory servers also expose their schema over LDAP. The root DSE attribute `subschemaSubentry` shows the DN of the entry for LDAP schema definitions.

Store large attribute values by reference

To serve results quickly with high availability, directory servers cache content and replicate it everywhere. If you already store large attribute values elsewhere, such as photos or audio messages, keep only a reference to external content in a user's account.

Take care with persistent search and server-side sorting

A persistent search lets your application receive updates from the server as they happen by keeping the connection open and forcing the server to check whether to return additional results any time it performs a modification in the scope of your search. Directory administrators therefore might hesitate to grant persistent search access to your application.

DS servers expose a change log to let you discover updates with less overhead. If you do have to use a persistent search instead, try to narrow the scope of your search.

DS servers support a resource-intensive, standard operation called server-side sorting. When your application requests a server-side sort, the directory server retrieves all matching entries, sorts the entries in memory, and returns the results. For result sets of any size, server-side sorting ties up server resources that could be used elsewhere. Alternatives include sorting the results after your application receives them, or working with the directory administrator to enable appropriate browsing (virtual list view) indexes for applications that must regularly page through long lists of search results.

Reuse schemas where possible

DS servers come with schema definitions for a wide range of standard object classes and attribute types. Directories use unique, IANA-registered object identifiers (OIDs) to avoid object class and attribute type name clashes. The overall goal is Internet-wide interoperability.

Therefore, reuse schema definitions that already exist whenever you reasonably can. Reuse them as is. Do not try to redefine existing schema definitions.

If you must add schema definitions for your application, extend existing object classes with AUXILIARY classes. Take care to name your schemas such that they do not clash with other names.

When you have defined schema required for your application, work with the directory administrator to add your definitions to the directory service. DS servers let directory administrators update schema definitions over LDAP. There is no need to interrupt the service to add your application. Directory administrators can, however, have other reasons why they hesitate to add your schema definitions. Coming to the discussion prepared with good schema definitions, explanations of why they should be added, and evident regard for interoperability makes it easier for the directory administrator to grant your request.

Read directory server schemas during initialization

By default, Directory Services APIs use a minimal, built-in core schema, rather than reading the schema from the server. Doing so automatically would incur a significant performance cost. Unless schemas change, your application only needs to read them once.

When you start your application, read directory server schemas as a one-off initialization step.

Once you have the directory server schema definitions, use them to validate entries.

Handle referrals

When a directory server returns a search result, the result is not necessarily an entry. If the result is a referral, then your application should follow up with an additional search based on the URIs provided in the result.

Troubleshooting: check result codes

LDAP result codes are standard, and listed in LDAP result codes.

When your application receives a result, it must rely on the result code value to determine what action to take. When the result is not what you expect, read or at least log the additional message information.

Troubleshooting: check server logs

If you can read the directory server access log, then check what the server did with your application's request. The following excerpt shows a successful search by `cn=My App,ou=Apps,dc=example,dc=com`:

```
{"eventName":"DJ-LDAP","client":{"ip":"127.0.0.1","port":59891},"server":{"ip":"127.0.0.1","port":1389},"request":
{"protocol":"LDAP","operation":"CONNECT","connId":0},"transactionId":"0","response":
{"status":"SUCCESSFUL","statusCode":"0","elapsedTime":
0,"elapsedTimeUnits":"MILLISECONDS"},"timestamp":"2016-10-20T15:48:36.933Z","_id":"11d5fdaf-79ac-4677-
a640-805db1c35af0-3"}
{"eventName":"DJ-LDAP","client":{"ip":"127.0.0.1","port":59891},"server":{"ip":"127.0.0.1","port":1389},"request":
{"protocol":"LDAP","operation":"EXTENDED","connId":0,"msgId":
1,"name":"StartTLS","oid":"1.3.6.1.4.1.1466.20037"},"transactionId":"0","response":
{"status":"SUCCESSFUL","statusCode":"0","elapsedTime":
3,"elapsedTimeUnits":"MILLISECONDS"},"timestamp":"2016-10-20T15:48:36.945Z","_id":"11d5fdaf-79ac-4677-
a640-805db1c35af0-5"}
{"eventName":"DJ-LDAP","client":{"ip":"127.0.0.1","port":59891},"server":{"ip":"127.0.0.1","port":1389},"request":
{"protocol":"LDAP","operation":"BIND","connId":0,"msgId":2,"version":"3","authType":"Simple","dn":"cn=My
App,ou=Apps,dc=example,dc=com"},"transactionId":"0","response":{"status":"SUCCESSFUL","statusCode":"0","elapsedTime":
6,"elapsedTimeUnits":"MILLISECONDS"},"userId":"cn=My
App,ou=Apps,dc=example,dc=com","timestamp":"2016-10-20T15:48:38.462Z","_id":"11d5fdaf-79ac-4677-a640-805db1c35af0-7"}
{"eventName":"DJ-LDAP","client":{"ip":"127.0.0.1","port":59891},"server":{"ip":"127.0.0.1","port":1389},"request":
{"protocol":"LDAP","operation":"SEARCH","connId":0,"msgId":
3,"dn":"dc=example,dc=com","scope":"sub","filter":"(uid=kvaughan)","attrs":
["isMemberOf"]},"transactionId":"0","response":{"status":"SUCCESSFUL","statusCode":"0","elapsedTime":
6,"elapsedTimeUnits":"MILLISECONDS","nentries":1},"timestamp":"2016-10-20T15:48:38.472Z","_id":"11d5fdaf-79ac-4677-
a640-805db1c35af0-9"}
{"eventName":"DJ-LDAP","client":{"ip":"127.0.0.1","port":59891},"server":{"ip":"127.0.0.1","port":1389},"request":
{"protocol":"LDAP","operation":"UNBIND","connId":0,"msgId":
4},"transactionId":"0","timestamp":"2016-10-20T15:48:38.480Z","_id":"11d5fdaf-79ac-4677-a640-805db1c35af0-11"}
{"eventName":"DJ-LDAP","client":{"ip":"127.0.0.1","port":59891},"server":{"ip":"127.0.0.1","port":1389},"request":
{"protocol":"LDAP","operation":"DISCONNECT","connId":0},"transactionId":"0","response":
{"status":"SUCCESSFUL","statusCode":"0","elapsedTime":0,"elapsedTimeUnits":"MILLISECONDS","reason":"Client
Unbind"},"timestamp":"2016-10-20T15:48:38.481Z","_id":"11d5fdaf-79ac-4677-a640-805db1c35af0-13"}
```


Notice that each operation type is shown in upper case. The messages track the client information, and the connection ID (`connId`) and message ID (`msgID`) numbers for filtering messages. The `elapsedTime` indicates how many milliseconds the DS server worked on the request. Result code 0 corresponds to a successful result, as described in RFC 4511.

Troubleshooting: inspect network traffic

If result codes and server logs are not enough, many network tools can interpret HTTP and LDAP packets. Install the necessary keys to decrypt encrypted packet content.

Next steps

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

Learn about 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 Replication, and trying the examples in that page.

Browse DS documentation

Category	Topics Covered
Release notes	DS features, fixes, and known issues
Deployment	Deploying Directory Services in on-premises and cloud environments
Installation	Installing DS software
Upgrade	Upgrading DS software
Configuration	Configuring DS servers after installation
Security	Ensuring a Directory Services deployment is secure
Maintenance	Day-to-day operations for maintaining DS servers
Logging	Configuring DS server logs
Monitoring	What to monitor when running DS servers, and where to look for metrics and other information
Use LDAP	How to use LDAP features and command-line tools
Use REST/HTTP	How to configure and use DS REST APIs for HTTP access
Configuration Reference	The `dsconfig` subcommands and server configuration properties
DS Javadoc	Evolving LDAP SDK and server APIs, including ForgeRock common APIs
LDAP reference	LDAP-specific features of DS software
LDAP Schema Reference	All default LDAP schema, including monitoring attributes and object classes
Log Reference	DS server error log messages by category and ID
Tools Reference	Tools bundled with DS software

Try third-party tools

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

- 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.

ForgeRock does not endorse or support third-party tools.

Use DS with AM

- Back End Directory Servers in the AM *Deployment Planning Guide*
- Preparing External Stores in the AM *Installation Guide*
- Configuring External CTS Token Stores in the AM *Core Token Service Guide*
- You can install DS directory servers for use as external AM stores.

  For details, see Setup profiles.

Use DS with IDM

- External DS Repository and Select a Repository in the IDM *Installation Guide*

  Also see Install DS as an IDM repository.
- One Way Synchronization From LDAP to IDM, Two-Way Synchronization Between LDAP and IDM, and other LDAP-related pages in the IDM *Samples Guide*
- DS Repository Configuration and Mappings With a DS Repository in the IDM *Object Modeling Guide*
- Synchronize Passwords With DS in the IDM *Password Synchronization Plugin Guide*

Remove DS software

For details, see Uninstallation.

Glossary

Abandon operation

LDAP operation to stop processing of a request in progress, after which the server drops the connection without a reply to the client application.

Access control

Control to grant or to deny access to a resource.

Access control instruction (ACI)

Instruction added as a directory entry attribute for fine-grained control over what a given user or group member is authorized to do in terms of LDAP operations and access to user data.

ACIs are implemented independently from privileges, which apply to administrative operations.

See also: Privilege

Access control list (ACL)

An access control list connects a user or group of users to one or more security entitlements. For example, users in group sales are granted the entitlement read-only to some financial data.

Access log

Server log tracing the operations the server processes including timestamps, connection information, and information about the operation itself.

Account lockout

The act of making an account temporarily or permanently inactive after successive authentication failures.

Active user

A user that has the ability to authenticate and use the services, having valid credentials.

Add operation

LDAP operation to add a new entry or entries to the directory.

Anonymous

A user that does not need to authenticate, and is unknown to the system.

Anonymous bind

A bind operation using simple authentication with an empty DN and an empty password, allowing anonymous access such as reading public information.

Approximate index

Index is used to match values that "sound like" those provided in the filter.

Attribute

Properties of a directory entry, stored as one or more key-value pairs. Typical examples include the common name (`cn`) to store the user's full name and variations of the name, user ID (`uid`) to store a unique identifier for the entry, and `mail` to store email addresses.

Attribute value assertion (AVA)

An attribute description and a matching rule assertion value for the attribute.

DS software uses AVAs in RDNs, and to determine whether an entry matches an assertion. For example, a search filter specifying the AVA `uid=bjensen` asserts that matching entries have a `uid` attribute value equal to `bjensen`.

Audit log

Type of access log that dumps changes in LDIF.

Authentication

The process of verifying who is requesting access to a resource; the act of confirming the identity of a principal.

Authorization

The process of determining whether access should be granted to an individual based on information about that individual; the act of determining whether to grant or to deny a principal access to a resource.

Backend

Repository that stores directory data. Different implementations with different capabilities exist.

Binary copy

Backup files from one replica are restored on another replica.

Bind operation

LDAP authentication operation to determine the client's identity in LDAP terms, the identity which is later used by the server to authorize (or not) access to directory data that the client wants to lookup or change.

Branch

The distinguished name (DN) of a non-leaf entry in the Directory Information Tree (DIT), and that entry and all its subordinates taken together.

Some administrative operations allow you to include or exclude branches by specifying the DN of the branch.

See also: Suffix

Collective attribute

A standard mechanism for defining attributes that appear on all the entries in a particular subtree.

Compare operation

LDAP operation to compare a specified attribute value with the value stored on an entry in the directory.

Control

Information added to an LDAP message to further specify how an LDAP operation should be processed. DS supports many LDAP controls.

Change sequence number (CSN)

An opaque string uniquely identifying a single change to directory data. A CSN indicates exactly when a change occurred on which replica. An example CSN is `010f016df804edca0000008fevaluation-only`.

DS replication uses CSNs to replay replicated operations consistently on all replicas. DS replicas record CSNs in historical data values for `ds-sync-state` and `ds-sync-hist` attributes.

When troubleshooting replication data consistency, it can be useful to interpret CSNs. Contact support for help.

Database cache

Memory space set aside to hold database content.

Debug log

Server log tracing details needed to troubleshoot a problem in the server.

Delete operation

LDAP operation to remove an existing entry or entries from the directory.

Directory

A directory is a network service which lists participants in the network such as users, computers, printers, and groups. The directory provides a convenient, centralized, and robust mechanism for publishing and consuming information about network participants.

Directory hierarchy

A directory can be organized into a hierarchy in order to make it easier to browse or manage. Directory hierarchies normally represent something in the physical world, such as organizational hierarchies or physical locations.

For example, the top level of a directory may represent a company, the next level down divisions, the next level down departments, and down the hierarchy. Alternately, the top level may represent the world, the next level down countries, next states or provinces, and next cities.

Directory Information Tree (DIT)

A set of directory entries organized hierarchically in a tree structure, where the vertices are the entries, and the arcs between vertices define relationships between entries.

Directory object

A directory object is an item in a directory. Example objects include users, user groups, computers, and more. Objects may be organized into a hierarchy and contain identifying attributes.

See also: Entry

Directory proxy server

Server that forwards LDAP requests to remote directory servers. A standalone directory proxy server does not store user data.

Directory server

Server application for centralizing information about network participants. A highly available directory service consists of multiple directory servers configured to replicate directory data.

See also: Replication

Directory Services Markup Language (DSML)

Standard language to access directory services using XML. DMSL v1 defined an XML mapping of LDAP objects, while DSMLv2 maps the LDAP Protocol and data model to XML.

Directory superuser

Directory account with privileges to do full administration of the DS server, including bypassing access control evaluation, changing access controls, and changing administrative privileges.

See also: Superuser

Distinguished name (DN)

Fully qualified name for a directory entry, such as `uid=bjensen,ou=People,dc=example,dc=com`, built by concatenating the entry RDN (`uid=bjensen`) with the DN of the parent entry (`ou=People,dc=example,dc=com`).

Domain

A replication domain consists of several directory servers sharing the same synchronized set of data.

The base DN of a replication domain specifies the base DN of the replicated data.

DSML gateway

Standalone web application that translates DSML requests from client applications to LDAP requests to a directory service, and LDAP responses from a directory service to DSML responses to client applications.

Dynamic group

Group that specifies members using LDAP URLs.

Entry

An entry is an object in the directory, defined by one of more object classes, and their related attributes.

Entry cache

Memory space set aside to hold frequently accessed, large entries, such as static groups.

Equality index

Index used to match values that correspond exactly (though generally without case sensitivity) to the value provided in the search filter.

Errors log

Server log tracing server events, error conditions, and warnings, categorized and identified by severity.

Etime

Elapsed time within the server to process a request, starting from the moment the decoded operation is available to be processed by a worker thread.

Export

Save directory data in an LDIF file.

Extended operation

Additional LDAP operation not included in the original standards. DS servers support several standard LDAP extended operations.

Extensible match index

Index for a matching rule other than approximate, equality, ordering, presence, substring or VLV, such as an index for generalized time.

External user

An individual that accesses company resources or services but is not working for the company. Typically, a customer or partner.

Filter

An LDAP search filter is an expression that the server uses to find entries that match a search request, such as `(mail=*@example.com)` to match all entries having an email address in the example.com domain.

Group

Entry identifying a set of members whose entries are also in the directory.

Generation ID

The initial state identifier for a replicated directory server base DN. It is a hash of the first 1000 entries of the base DN, computed when creating the backend, importing data from LDIF, or initializing replication.

Replication can only proceed between base DNs that have the same generation ID.

Idle time limit

Defines how long DS allows idle connections to remain open.

Import

Read in and index directory data from an LDIF file.

Inactive user

An entry in the directory that once represented a user but which is now no longer able to be authenticated.

Index

Directory server backend feature to allow quick lookup of entries based on their attribute values.

See also: Approximate index, Equality index, Extensible match index, Ordering index, Presence index, Substring index, VLV index, Index entry limit

Index entry limit

When the number of entries that an index key points to exceeds the index entry limit, DS stops maintaining the list of entries for that index key.

Internal user

An individual who works within the company either as an employee or as a contractor.

LDAP Data Interchange Format (LDIF)

Standard, portable, text-based representation of directory content.

See RFC 2849.

LDAP URL

LDAP Uniform Resource Locator, such as `ldaps://ds.example.com:636/dc=example,dc=com??sub?(uid=bjensen)`.

See RFC 2255.

LDAPS

LDAP over SSL.

Lightweight Directory Access Protocol (LDAP)

A simple and standardized network protocol used by applications to connect to a directory, search for objects and add, edit or remove objects.

See RFC 4510.

Matching rule

Defines rules for performing matching operations against assertion values. Matching rules are frequently associated with an attribute syntax, and are used to compare values according to that syntax.

For example, the `distinguishedNameEqualityMatch` matching rule can be used to determine whether two DNs are equal and can ignore unnecessary spaces around commas and equal signs, differences in capitalization in attribute names, and other discrepancies.

Modify DN operation

LDAP modification operation to request that the server change the distinguished name of an entry.

Modify operation

LDAP modification operation to request that the server change one or more attributes of an entry.

Naming context

Base DN under which client applications can look for user data.

Object class

Identifies entries that share certain characteristics. Most commonly, an entry's object classes define the attributes that must and may be present on the entry.

Object classes are stored on entries as values of the `objectClass` attribute. Object classes are defined in the directory schema, and can be *abstract* (defining characteristics for other object classes to inherit), *structural* (defining the basic structure of an entry, one structural inheritance per entry), or *auxiliary* (for decorating entries already having a structural object class with other required and optional attributes).

Object identifier (OID)

String that uniquely identifies an object, such as `0.9.2342.19200300.100.1.1` for the user ID attribute or `1.3.6.1.4.1.1466.115.121.1.15` for `DirectoryString` syntax.

Operational attribute

An attribute that has a special (operational) meaning for the server, such as `pwdPolicySubentry` or `modifyTimestamp`.

Ordering index

Index used to match values for a filter that specifies a range.

Password policy

A set of rules regarding what sequence of characters constitutes an acceptable password. Acceptable passwords are generally those that would be too difficult for another user, or an automated program to guess and thereby defeat the password mechanism.

Password policies may require a minimum length, a mixture of different types of characters (lowercase, uppercase, digits, punctuation marks, and other characters), avoiding dictionary words or passwords based on the user's name, and other attributes.

Password policies may also require that users not reuse old passwords and that users change their passwords regularly.

Password reset

Password change performed by a user other than the user who owns the entry.

Password storage scheme

Mechanism for encoding user passwords stored on directory entries. DS implements a number of password storage schemes.

Password validator

Mechanism for determining whether a proposed password is acceptable for use. DS implements a number of password validators.

Plugin

Java library with accompanying configuration that implements a feature through processing that is not essential to the core operation of DS servers.

As the name indicates, plugins can be plugged in to an installed server for immediate configuration and use without recompiling the server.

DS servers invoke plugins at specific points in the lifecycle of a client request. The DS configuration framework lets directory administrators manage plugins with the same tools used to manage the server.

Presence index

Index used to match the fact that an attribute is present on the entry, regardless of the value.

Principal

Entity that can be authenticated, such as a user, a device, or an application.

Privilege

Server configuration settings controlling access to administrative operations such as exporting and importing data, restarting the server, performing password reset, and changing the server configuration.

Privileges are implemented independently from access control instructions (ACI), which apply to LDAP operations and user data.

See also: Access control instruction

Referential integrity

Ensuring that group membership remains consistent following changes to member entries.

Referint log

Server log tracing referential integrity events, with entries similar to the errors log.

Referral

Reference to another directory location, which can be another directory server running elsewhere or another container on the same server, where the current operation can be processed.

Relative distinguished name (RDN)

Initial portion of a DN that distinguishes the entry from all other entries at the same level, such as `uid=bjensen` in `uid=bjensen,ou=People,dc=example,dc=com`.

Replica

Directory server this is configured to use replication.

Replication

Data synchronization that ensures all directory servers participating eventually share a consistent set of directory data.

Replication log

Server log tracing replication events, with entries similar to the errors log.

Replication server

Server dedicated to transmitting replication messages. A standalone replication server does not store user data.

REST to LDAP gateway

Standalone web application that translates RESTful HTTP requests from client applications to LDAP requests to directory services, and LDAP responses from directory services to HTTP responses to client applications.

Root DSE

The directory entry with distinguished name "" (empty string), where DSE is an acronym for *DSA-Specific Entry*. DSA is an acronym for *Directory Server Agent*, a single directory server.

The root DSE serves to expose information over LDAP about what the directory server supports in terms of LDAP controls, auth password schemes, SASL mechanisms, LDAP protocol versions, naming contexts, features, LDAP extended operations, and other information.

Schema

LDAP schema defines the object classes, attributes types, attribute value syntaxes, matching rules and other constrains on entries held by the directory server.

Search filter

See: Filter

Search operation

LDAP lookup operation where a client requests that the server return entries based on an LDAP filter, and a base DN under which to search.

Simple authentication

Bind operation performed with a user's entry DN and user's password.

Use simple authentication only if the network connection is secure.

Size limit

Sets the maximum number of entries returned for a search.

Static group

Group that enumerates member entries.

Subentry

An entry, such as a password policy entry, that resides with the user data but holds operational data, and is not visible in search results unless explicitly requested.

Substring index

Index used to match values specified with wildcards in the filter.

Suffix

The distinguished name (DN) of a root entry in the Directory Information Tree (DIT), and that entry and all its subordinates taken together as a single object of administrative tasks such as export, import, indexing, and replication.

Superuser

User with privileges to perform unconstrained administrative actions on DS server. This account is analogous to the UNIX `root` and Windows `Administrator` accounts.

The conventional default superuser DN is `uid=admin`. You can create additional superuser accounts, each with different administrative privileges.

Superuser privileges include the following:

- `bypass-acl`: The holder is not subject to access control.
- `privilege-change`: The holder can edit administrative privileges.
- `proxied-auth`: The holder can make requests on behalf of another user, including directory superusers.

See also: Directory superuser, Privilege

Task

Mechanism to provide remote access to server administrative functions.

DS software supports tasks to back up and restore backends, to import and export LDIF files, and to stop and restart the server.

Time limit

Defines the maximum processing time DS devotes to a search operation.

Unbind operation

LDAP operation to release resources at the end of a session.

Unindexed search

Search operation for which no matching index is available.

If no indexes are applicable, then the directory server potentially has to go through all entries to look for candidate matches. For this reason, the `unindexed-search` privilege, which allows users to request searches for which no applicable index exists, is reserved for the directory manager by default.

User

An entry that represents an individual that can be authenticated through credentials contained or referenced by its attributes. A user may represent an internal user or an external user, and may be an active user or an inactive user.

User attribute

An attribute for storing user data on a directory entry such as `mail` or `givenname`.

Virtual attribute

An attribute with dynamically generated values that appear in entries but are not persistently stored in the backend.

Virtual directory

An application that exposes a consolidated view of multiple physical directories over an LDAP interface. Consumers of the directory information connect to the virtual directory's LDAP service.

Behind the scenes, requests for information and updates to the directory are sent to one or more physical directories where the actual information resides. Virtual directories enable organizations to create a consolidated view of information that for legal or technical reasons cannot be consolidated into a single physical copy.

Virtual list view (VLV) index

Browsing index designed to help the directory server respond to client applications that need, for example, to browse through a long list of results a page at a time in a GUI.

Virtual static group

DS group that lets applications see dynamic groups as what appear to be static groups.

X.500

A family of standardized protocols for accessing, browsing and maintaining a directory. X.500 is functionally similar to LDAP, but is generally considered to be more complex, and has consequently not been widely adopted.

Deployment

This guide focuses on how to use Directory Services software to build secure, high-performance, manageable directory services. It helps directory service architects design scalable services that fit their needs.

DS Software

Use DS components.

Project Outline

Outline a successful plan.

Complete Plans

Create comprehensive plans.

Deployment Patterns

Apply best practices.

Provisioning

Prepare systems and hardware.

Checklists

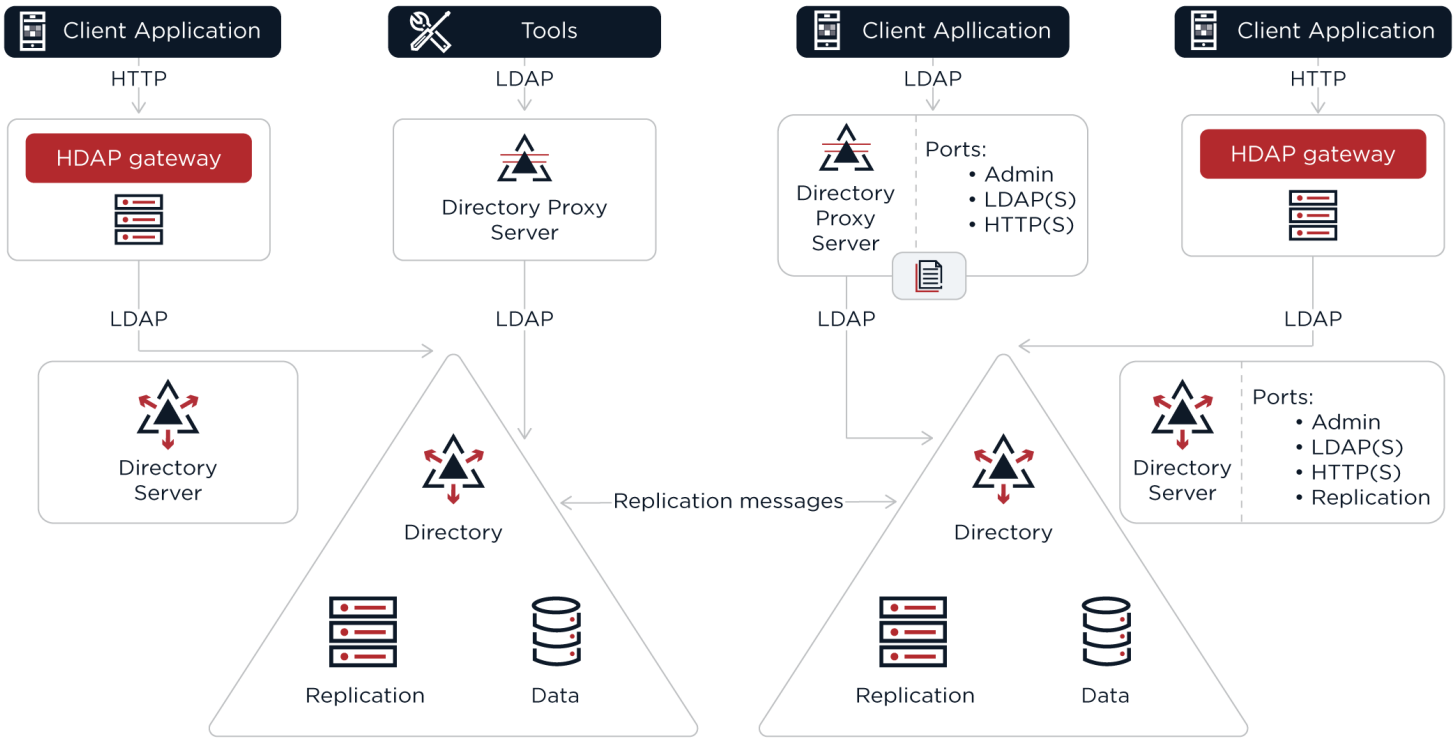
Follow checklists.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

The common REST API provides ForgeRock Identity Platform software common ways to access web resources and collections of resources.

DS software

A directory service provides LDAP and HTTP access to distributed, shared directory data. A deployed directory service consists of one or more components. Each component plays a particular role in your directory service. Before you design your deployment, you need to be familiar with the roles that each component can play:



- *Directory servers*, which maintain and serve requests for directory data.

  In most deployments, directory servers use data *replication* to ensure that their data sets eventually converge everywhere. This documentation refers to a replicated directory server as a *replica*.

- *Directory proxy servers* that forward requests for directory data to directory servers, and return directory server responses to client applications.

- *Replication servers* that transmit data replication messages among replicas.

  You can configure a directory server to act as a replication server as well. A *standalone* replication server plays only the replication server role, brokering replication change messages. It does not store directory data.

- *DSML gateways* that intermediate between DSML client applications and an LDAP directory.

- *REST to LDAP gateways* that intermediate between RESTful HTTP client applications and LDAP directories.

- LDAP client tools and server administration tools for testing and managing servers.

Directory servers

Directory servers have the following characteristics.

Roles

Directory servers provide access to their copy of the distributed directory database. A directory server usually functions as the repository of identities for users, applications, and things. They respond to requests from client applications directly or indirectly through directory proxy servers. This includes the following:

- LDAP requests for authentication, reads, and updates.

  An LDAP client application authenticates with the directory server, and then performs one or more operations before either reauthenticating to reuse the connection or ending the session and closing the connection.

- HTTP read and update requests, often including credentials for authentication.

  An HTTP request translates to one or more internal LDAP requests.

- Administrative requests, such as requests to modify the server configuration or to perform a task such as backup or LDIF export.

- JMX and SNMP requests specifically for monitoring information.

In deployments with multiple *replicas*, directory servers replay replicated operations. Expect each replica to replay every successful update to any replica.

Data

In addition to the libraries and tools delivered with the server distribution, a directory server is associated with the following persistent state information and local data:

User data

Directory servers store user data. The directory server stores the data in local storage, such as an internal disk or an attached disk array. The storage must keep pace with throughput for update operations.

The amount of user data depends entirely on the deployment, ranging from a few LDAP entries to more than a billion. The amount of user data grows or shrinks in deployment depending on the pattern of update operations.

The directory server stores user data in a backend database. For details, see Data storage.

Metadata for replication

A directory server can be a replica of other directory servers, meaning it can hold an eventually consistent copy of the data on the other replicas. To avoid an individual server becoming a single point of failure, almost all real-world deployments depend on replication.

When serving a request to update directory data, the directory server modifies its data and sends a request to a replication server. The replication server, described in Replication servers, ensures that all other replicas update their data to eventually reflect the current state of the data.

To tolerate network partitions, the directory service supports concurrent update operations on different replicas. Concurrent updates potentially cause conflicts, but directory servers can resolve most conflicts automatically. To resolve conflicts, a directory server stores historical metadata alongside user data, trading space for resilience. For details, see About Replication.

The directory server purges this historical metadata after a configurable interval. The volume of historical metadata depends on the total number of updates made to the directory service since the purge interval.

Server configuration

Each server has configuration data in its `config` directory. This includes the server configuration, mostly in LDIF format, LDAP schema definitions also in LDIF format, keys used to secure connections and perform encryption and decryption, and some additional data.

When installing a server, the `setup` command instantiates this configuration data from templates.

When upgrading a server, the `upgrade` command modifies this configuration data to apply any necessary changes.

Log files

The server writes to multiple log files by default, including error and access logs.

The server writes a message to the current access log for each operation. For high-volume directory services, log file storage must keep pace with the requests to record access to the service.

Log file retention and rotation policies prevent log file data from filling the disk. For details, see Logging. As a result of default retention policies, messages can eventually be lost unless you copy old files to another system for permanent storage.

Backup files

When you export directory data to LDIF or create a backup, the directory server writes the files to the specified directory. If you never purge or move these files, they can eventually fill the disk.

For details, see Import and Export, and Backup and restore.

System resources

When deciding how to deploy a directory server, think of it as a copy of the database. A large, high-performance, distributed database serving lots of applications requires more system resources than a small database serving one, simple application.

A directory server requires the following system resources:

- Sufficient RAM to cache frequently used data.

  For best read performance, cache the entire directory data set in memory.

- Sufficient CPU to perform any required calculations.

  Authentication operations generally use more CPU than other operations. In particular, password storage schemes like PBKDF2 are designed to consume CPU resources. Calculations for transport layer security can use CPU as well, particularly if many client requests are over short-lived HTTPS connections.

- Sufficient fast disk access to serve client applications, to replay replication operations, and to log access and errors.

  The underlying disk subsystem must serve enough input/output operations per second (IOPS) to avoid becoming a bottleneck when performing these operations. A small database that serves few client operations and changes relatively infrequently requires fewer IOPS than a large database sustaining many updates and serving many clients.

  Plan additional capacity for any backup or LDIF files stored on local partitions.

- Sufficiently fast network access to serve client applications and relay replication traffic.

When considering network requirements, keep the following points in mind:

- Each LDAP search request can return multiple response messages.
- Each request to update directory data results in corresponding replication traffic. The operation must be communicated to replication servers and replayed on each other directory server.
- Once established, and unlike most HTTP connections, LDAP connections remain open until the client closes the connection, or until the server idles the connection. This is particularly true for applications using persistent searches, which by design are intended to be permanent.

Replication servers

Standalone replication servers have the following characteristics. If you configure a directory server to play the role of a replication server as well, then the directory server also has these roles and characteristics.

Roles

Replication servers provide the following services:

- Receive and transmit change messages between replicas.

  Each replica is connected to one replication server at a time. A single standalone replication server can serve 10 or more replicas.

- Maintain information about all other replication servers and directory servers in the deployment that replicate the same data.

  Change messages travel from a connected directory server to the replication server. The replication server transmits the message to connected replicas. If there are other replication servers, the replication server transmits the message to the other replication servers, which in turn transmit the message to their connected replicas. This hub-and-spoke communication model means directory services can be composed of many individual servers.

- Respond to administrative requests.
- Respond to HTTP, JMX, LDAP, and SNMP requests for monitoring information.

In all deployments using replication, the replication service provides the foundation of directory service availability. This is as important to the directory service as a naming service is for a network. When deploying replicated directory services, start by installing the replication service.

To avoid a single point of failure, install two or more replication servers in each location.

Data

In addition to the libraries and tools delivered with the server distribution, a replication server is associated with the following persistent state information and local data:

Change data

When serving a request to update directory data, a directory server, described in Directory servers, modifies its data and sends a request to a replication server. The replication server ensures that all other replicas update their data to eventually reflect the current state of the data.

The replication protocol is proprietary. Replication servers expose a public record of changes in a change log, allowing other applications to keep up to date with changes to user data. This change data is stored in change log files. For details, see Changelog for notifications.

The replication server purges this historical metadata after a configurable interval. The volume of historical metadata depends on the updates made to the directory service since the purge interval.

Server configuration

Each server has configuration data in its `config` directory. This includes the server configuration, mostly in LDIF format, LDAP schema definitions also in LDIF format, keys used to secure connections and perform encryption and decryption, and some additional data.

When installing a server, the `setup` command instantiates this configuration data from templates.

When upgrading a server, the `upgrade` command modifies this configuration data to apply any necessary changes.

Log files

The server writes to multiple log files by default, including error and access logs.

Log file retention and rotation policies prevent log file data from filling the disk. For details, see Logging. This means, however, that messages are eventually lost unless you move old files to another system for permanent storage.

System resources

When deploying a replication server, keep its foundational role in mind. Directory servers communicate with other replicas through replication servers. Directory proxy servers rely on replication servers to find directory servers.

A replication server requires the following system resources:

- Sufficient fast disk access to log and read change messages, and to update access and error logs.

  The underlying disk subsystem must serve enough IOPS to avoid becoming a bottleneck when performing these operations.

- Sufficiently fast network access to receive and transmit change messages for multiple replicas and for each other replication server.

Directory proxy servers

Directory proxy servers have the following characteristics.

Roles

Directory proxy servers provide the following services:

- Balance load of requests to LDAP directory servers.
- Receive and transmit LDAP client requests to LDAP directory servers.
- Receive and transmit LDAP directory server responses to LDAP client applications.
- Respond to administrative requests.

- Respond to HTTP, JMX, LDAP, and SNMP requests for monitoring information.

A directory proxy server can hide the underlying directory service architecture from client applications, enabling you to build a single point of directory service access.

A directory proxy server can discover directory servers through a replication server. This capability relies, however, on the replication server configuration. If you use the proxy server with third-party directory service components, then you must manually maintain the network locations for directory servers.

A directory proxy server provides LDAP access to remote LDAP directory servers. If you want to provide HTTP access to remote LDAP directory servers, use the REST to LDAP gateway instead. For details, see REST to LDAP gateway.

Data

In addition to the libraries and tools delivered with the server distribution, a directory proxy server is associated with the following persistent state information and local data:

Server configuration

Each server has configuration data in its `config` directory. This includes the server configuration, mostly in LDIF format, LDAP schema definitions also in LDIF format, keys used to secure connections and perform encryption and decryption, and some additional data.

When installing a server, the `setup` command instantiates this configuration data from templates.

When upgrading a server, the `upgrade` command modifies this configuration data to apply any necessary changes.

Log files

The server writes to multiple log files by default, including error and access logs.

Log file retention and rotation policies prevent log file data from filling the disk. For details, see Logging. This means, however, that messages are eventually lost unless you move old files to another system for permanent storage.

System resources

In order to route requests appropriately, a directory proxy server must decode incoming requests and encode ongoing requests. It must also decode and encode incoming and outgoing responses. When deploying a directory proxy server, keep this decoding and encoding in mind, because it explains why you might need as many proxy servers as directory servers.

A directory proxy server requires the following system resources:

- Sufficient fast disk access to update access and error logs.

  The underlying disk subsystem must serve enough IOPS to avoid becoming a bottleneck when performing these operations.

- Sufficiently fast network access to receive and transmit client requests and server responses.

- Sufficient CPU to perform any required calculations.

  Request and response decoding and encoding consume CPU resources.

- Sufficient RAM to maintain active connections.

Command-line tools

When you install the files for a server component, those files include tools for setting up, upgrading, and configuring and maintaining the server and administrative tasks. The files also include LDAP command-line tools for sending LDAP requests and measuring directory service performance.

For details, see Server Commands.

DSML gateway

The standalone DSML gateway web application has the following characteristics.

You can install this component independently of directory services. For details, see Install a DSML gateway.

Roles

DSML gateways provide the following services:

- Receive HTTP DSML requests from client applications, and transmit them as LDAP requests to a directory service.
- Receive LDAP responses from a directory service, and transmit them as HTTP DSML responses to client applications.

A DSML gateway runs in a Java web application container. It is limited to one host:port combination for the LDAP directory service.

Data

A DSML gateway maintains only its own service configuration, recorded in the web application `WEB-INF/web.xml` file. It depends on the host web application container for other services, such as logging.

System resources

A DSML gateway requires the following system resources:

- Sufficiently fast network access to receive and transmit client requests and server responses.
- Sufficient CPU to perform any required calculations.

  Request and response decoding, encoding, and transformation all consume CPU resources.

  Calculations to secure network connections also consume CPU resources.
- Sufficient RAM to maintain active connections.

REST to LDAP gateway

The standalone REST to LDAP gateway web application has the following characteristics. REST refers to the representational state transfer architectural style. RESTful requests use the HTTP protocol.

You can install this component independently of directory services. For details, see Install a REST to LDAP gateway.

Roles

REST to LDAP gateways provide the following services:

- Receive HTTP requests from client applications, and transmit them as LDAP requests to a directory service.
- Receive LDAP responses from a directory service, and transmit them as HTTP responses to client applications.

A REST to LDAP gateway runs in a Java web application container. It can be configured to contact multiple LDAP directory servers.

One RESTful HTTP request can generate multiple LDAP requests. This is particularly true if the REST to LDAP mapping configuration includes references to resolve before returning response entries. For example, an LDAP user entry can have a `manager` attribute that holds the DN of the user's manager's entry. Rather than return an LDAP-specific DN in the REST response, the REST to LDAP mapping is configured to return the manager's name in the response. As a result, every time a user's manager is returned in the response, the gateway must make a request for the user's LDAP information and another request for the user's manager's name.

Data

A REST to LDAP gateway maintains only its own service configuration, recorded in files as described in REST to LDAP reference. It depends on the host web application container for other services, such as logging.

System resources

A REST to LDAP gateway requires the following system resources:

- Sufficiently fast network access to receive and transmit client requests and server responses.
- Sufficient CPU to perform any required calculations.

  Request and response decoding, encoding, and transformation all consume CPU resources.

  Calculations to secure network connections also consume CPU resources.
- Sufficient RAM to maintain active connections.

Project outline

Consider the following when preparing the high-level project plan.

Needs assessment

Needs assessment is prerequisite to developing a comprehensive deployment plan. An accurate needs assessment is critical to ensuring that your directory services implementation meets your business needs and objectives.

As part of the needs assessment, make sure you answer the following questions:

What are your business objectives?

Clarify and quantify your business goals for directory services.

Why do you want to deploy directory services?

Consider at least the following list when answering this question:

- Is this a greenfield deployment?
- Do you need to transition an existing deployment to the cloud?
- Do you need to scale existing deployment for more users, devices, or things?

If you have an existing deployment, how do you upgrade?

Consider at least the following list when answering this question:

- Do you require a graceful upgrade?
- What obsolete components need a graceful transition?

  What should their replacements be?
- What are the costs related to the change?

  How can you save cost by making the transition?

Define objectives based on your needs assessment. State your objective so that all stakeholders agree on the same goals and business objectives.

Deployment planning

Deployment planning is critical to ensuring that your directory services are properly implemented within the time frame determined by your requirements. The more thoroughly you plan your deployment, the more solid your configuration will be, and you will meet timelines and milestones while staying within budget.

A deployment plan defines the goals, scope, roles, and responsibilities of key stakeholders, architecture, implementation, and testing of your DS deployment. A good plan ensures that a smooth transition to a new product or service is configured and all possible contingencies are addressed to quickly troubleshoot and solve any issue that may occur during the deployment process.

The deployment plan also defines a training schedule for your employees, procedural maintenance plans, and a service plan to support your directory services.

Important questions

- **What key applications does your system serve?** Understand how key client applications will use your directory service and what they require. Based on this understanding, you can match service level objectives (SLOs) to operational requirements. This ensures that you focus on what is critical to your primary customers.
- **What directory data does your system serve?** Directory data can follow standard schema and be shared by many applications. Alternatively, it can be dedicated to a single application such as AM CTS or IDM repository. Key applications can impose how they access directory data, or the directory data definition can be your decision.

  In addition, know where you will obtain production data, and in what format you will obtain it. You might need to maintain synchronization between your directory service and existing data services.

- **What are your SLOs?** In light of what you know about key and other applications, determine your SLOs. An SLO is a target for a directory service level that you can measure quantitatively.

  What objectives will you set for your service? How will you measure the following?

  - Availability
  - Response times
  - Throughput
  - Support response

- **What are your availability requirements?** DS services are designed to run continuously, without interruption even during upgrade. Providing a highly available service of course comes with operational complexities and costs.

  If your deployment must be highly available, take care in your planning phase to avoid single points of failure. You will need to budget for redundancy in all cases, and good operational policies, procedures, and training to avoid downtime as much as possible.

  If your deployment does not require true high availability, however, you will benefit from taking this into account during the planning stages of your deployment as well. You may be able to find significant cost savings as a trade for lower availability.

- **What are your security requirements?** DS services build in security in depth, as described in Security.

  Understand the specific requirements of your deployment in order to use only the security features you really need. If you have evaluated DS software by setting up servers with the evaluation setup profile, be aware that access control settings for Example.com data in the evaluation setup profile are very lenient.

- **Are all stakeholders engaged starting in the planning phase?** This effort includes but is not limited to delivery resources, such as project managers, architects, designers, implementers, testers, and service resources, such as service managers, production transition managers, security, support, and sustaining personnel. Input from all stakeholders ensures all viewpoints are considered at project inception, rather than downstream, when it may be too late.

Planning steps

Follow these steps to a successful deployment.

Project initiation

The project initiation phase begins by defining the overall scope and requirements of the deployment. Plan the following items:

- Determine the scope, roles and responsibilities of key stakeholders and resources required for the deployment.
- Determine critical path planning including any dependencies and their assigned expectations.
- Run a pilot to test the functionality and features of AM and uncover any possible issues early in the process.
- Determine training for administrators of the environment and training for developers, if needed.

Design

The design phase involves defining the deployment architecture. Plan the following items:

- Determine the use of products, map requirements to features, and ensure the architecture meets the functional requirements.
- Ensure that the architecture is designed for ease of management and scale. TCO is directly proportional to the complexity of the deployment.
- Define the directory data model.
- Determine how client applications will access directory data, and what data they have access to.
- Determine which, if any, custom DS server plugins must be developed. Derive specifications and project plans for each plugin.
- Determine the replication configuration.
- Define backup and recovery procedures, including how to recover all the servers, should disaster occur.
- Define monitoring and audit procedures, and how the directory service integrates with your tools.
- Determine how to harden DS servers for a secure deployment.
- Define the change management process for configurations and custom plugins.
- Define the test criteria to validate that the service meets your objectives.
- Define the operations required to maintain and support the running service.
- Define how you will roll out the service into production.
- Determine how many of each DS server type to deploy in order to meet SLOs. In addition, define the systems where each of the servers will run.

Implementation

The implementation phase involves deploying directory services. Plan the following items:

- Provision the DS servers.
- Maintain a record and history of the deployment for consistency across the project.
- Monitor and maintain the running service.

Automation and testing

The automation and continuous integration phase involves using tools for testing. Plan the following items:

- Use a continuous integration server, such as Jenkins, to ensure that changes have the expected impact, and no change causes any regressions.
- Ensure your custom plugins follow the same continuous integration process.
- Test all functionality to deliver the solution without any failures. Ensure that all customizations and configurations are covered in the test plan.

- Non-functionally test failover and disaster recovery procedures. Run load testing to determine the demand of the system and measure its responses. During this phase, anticipate peak load conditions.

Supportability

The supportability phase involves creating the runbook for system administrators and operators. This includes procedures for backup and restore operations, debugging, change control, and other processes.

If you have a ForgeRock Support contract, it ensures everything is in place prior to your deployment.

Comprehensive plans

Your comprehensive deployment plan should cover the following themes.

Team training

Training provides a common understanding, vocabulary, and basic skills for those working together on the project. Depending on previous experience with access management and with DS software, both internal teams and project partners might need training.

The type of training team members need depends on their involvement in the project:

- All team members should take at least some training that provides an overview of DS software. This helps to ensure a common understanding and vocabulary for those working on the project.
- Team members planning the deployment should take an DS training before finalizing their plans, and ideally before starting to plan the deployment.

  DS training pays for itself as it helps you to make the right initial choices to deploy more quickly and successfully.
- Team members involved in designing and developing DS client applications or custom plugins should take training in DS development in order to help them make the right choices.
- Team members who have already had been trained in the past might need to refresh their knowledge if your project deploys newer or significantly changed features, or if they have not worked with DS software for some time.

ForgeRock University regularly offers training courses for DS topics. For a current list of available courses, see the ForgeRock web site.

When you have determined who needs training and the timing of the training during the project, prepare a training schedule based on team member and course availability. Include the scheduled training plans in your deployment project plan.

ForgeRock also offers an accreditation program for partners, including an in-depth assessment of business and technical skills for each ForgeRock product. This program is open to the partner community and ensures that best practices are followed during the design and deployment phases.

Customization

DS servers provide a Java plugin API that allows you to extend and customize server processing. A server plugin is a library that you plug into an installed server and configure for use. The DS server calls the plugin as described in Plugin types.

DS servers have many features that are implemented as server plugin extensions. This keeps the core server processing focused on directory logic, and loosely coupled with other operations.

When you create your own custom plugin, be aware you must at a minimum recompile and potentially update your plugin code for every DS server update. The plugin API has interface stability: *Evolving*. A plugin built with one version of a server is not guaranteed to run or even to compile with a subsequent version. Only create your own custom plugin when you require functionality that the server cannot be configured to provide. The best practice is to deploy DS servers with a minimum of custom plugins.

> **Note**
>
> ForgeRock supports customers using standard plugins delivered as part of DS software.
> If you deploy with custom plugins and need support in production, contact info@forgerock.com in advance to determine how your deployment can be supported.

Although some custom plugins involve little development work, they can require additional scheduling and coordination. The more you customize, the more important it is to test your deployment thoroughly before going into production. Consider each custom plugin as sub-project with its own acceptance criteria. Prepare separate plans for unit testing, automation, and continuous integration of each custom plugin. For details, see Tests.

When you have prepared plans for each custom plugin sub-project, you must account for those plans in your overall deployment project plan.

Plugin types

Plugin types correspond to the points where the server invokes the plugin.

For the full list of plugin invocation points, see the Javadoc for PluginType. The following list summarizes the plugin invocation points:

- At server startup and shutdown
- Before and after data export and import
- Immediately after a client connection is established or is closed
- Before processing begins on an LDAP operation (to change an incoming request before it is decoded)
- Before core processing for LDAP operations (to change the way the server handles the operation)
- After core processing for LDAP operations (where the plugin can access all information about the operation including the impact it has on the targeted entry)
- When a subordinate entry is deleted as part of a subtree delete, or moved or renamed as part of a modify DN operation
- Before sending intermediate and search responses
- After sending a result

A plugin's types are specified in its configuration, and can therefore be modified at runtime.

Plugin configuration

Server plugin configuration is managed with the same configuration framework that is used for DS server configuration.

The DS configuration framework has these characteristics:

- LDAP schemas govern what attributes can be used in plugin configuration entries.

  For all configuration attributes that are specific to a plugin, the plugin should have its own object class and attributes defined in the server LDAP schema. Having configuration entries governed by schemas makes it possible for the server to identify and prevent configuration errors.

  For plugins, having schema for configuration attributes means that an important part of plugin installation is making the schema definitions available to the DS server.

- The plugin configuration is declared in XML files.

  The XML specifies configuration properties and their documentation, and inheritance relationships.

  The XML Schema Definition files (.xsd files) for the namespaces used are not published externally. For example, the namespace identifier `http://opendj.forgerock.org/admin` is not an active URL. An XML configuration definition has these characteristics:

  - The attributes of the `<managed-object>` element define XML namespaces, a (singular) name and plural name for the plugin, and the Java-related inheritance of the implementation to generate. A *managed object* is a configurable component of DS servers.

    A managed object definition covers the object's structure and inheritance, and is like a class in Java. The actual managed object is like an instance of an object in Java. Its configuration maps to a single LDAP entry in the configuration backend `cn=config`.

    Notice that the `<profile>` element defines how the whole object maps to an LDAP entry in the configuration. The `<profile>` element is mandatory, and should include an LDAP profile.

    The `name` and `plural-name` properties are used to identify the managed object definition. They are also used when generating Java class names. Names must be a lowercase sequence of words separated by hyphens.

    The `package` property specifies the Java package name for generated code.

    The `extends` property identifies a parent definition that the current definition inherits.

  - The mandatory `<synopsis>` element provides a brief description of the managed object.

    If a longer description is required, add a `<description>`. The `<description>` is used in addition to the synopsis, so there is no need to duplicate the synopsis in the description.

  - The `<property>` element defines a property specific to the plugin, including its purpose, its default value, its type, and how the property maps to an LDAP attribute in the configuration entry.

    The `name` attribute is used to identify the property in the configuration.

  - The `<property-override>` element sets the pre-defined property `java-class` to the fully qualified implementation class.

- Compilation generates the server-side and client-side APIs to access the plugin configuration from the XML. To use the server-side APIs in a plugin project, first generate and compile them, and then include the classes on the project classpath.

  When a plugin is loaded in the DS server, the client-side APIs are available to configuration tools like the `dsconfig` command. Directory administrators can configure a custom plugin in the same way they configure other server components.

- The framework supports internationalization.

  The plugin implementation selects appropriate messages from the resource bundle based on the server locale. If no message is available for the server locale, the plugin falls back to the default locale.

A complete plugin project includes LDAP schema definitions, XML configuration definitions, Java plugin code, and Java resource bundles. For examples, see the sample plugins delivered with DS software.

Pilot projects

Unless you are planning a maintenance upgrade, consider starting with a pilot implementation, which is a long-term project that is aligned with your specific requirements.

A pilot shows that you can achieve your goals with DS software plus whatever custom plugins and companion software you expect to use. The idea is to demonstrate feasibility by focusing on solving key use cases with minimal expense, but without ignoring real-world constraints. The aim is to fail fast, before investing too much, so you can resolve any issues that threaten the deployment.

Do not expect the pilot to become the first version of your deployment. Instead, build the pilot as something you can afford to change easily, and to throw away and start over if necessary.

The cost of a pilot should remain low compared to overall project cost. Unless your concern is primarily the scalability of your deployment, you run the pilot on a much smaller scale than the full deployment. Scale back on anything not necessary to validating a key use case.

Smaller scale does not necessarily mean a single-server deployment, though. If you expect your deployment to be highly available, for example, one of your key use cases should be continued smooth operation when part of your deployment becomes unavailable.

The pilot is a chance to experiment with and test features and services before finalizing your plans for deployment. The pilot should come early in your deployment plan, leaving appropriate time to adapt your plans based on the pilot results. Before you can schedule the pilot, team members might need training. You might require prototype versions of functional customizations.

Plan the pilot around the key use cases that you must validate. Make sure to plan the pilot review with stakeholders. You might need to iteratively review pilot results as some stakeholders refine their key use cases based on observations.

Directory data model

Before you start defining how to store and access directory data, you must know what data you want to store, and how client applications use the data. You must have or be able to generate representative data samples for planning purposes. You must be able to produce representative client traffic for testing.

When defining the directory information tree (DIT) and data model for your service, answer the following questions:

- What additional schema definitions does your directory data require?

  See LDAP schema extensions.

- What are the appropriate base DNs and branches for your DIT?

  See The DIT.

- How will applications access the directory service? Over LDAP? Over HTTP?

See Data views.

- Will a single team manage the directory service and the data? Will directory data management be a shared task, delegated to multiple administrators?

  See Data management.

- What groups will be defined in your directory service?

  See Groups.

- What sort of data will be shared across many directory entries? Should you define virtual or collective attributes to share this data?

  See Shared data.

- How should you cache data for appropriate performance?

  See Caching.

- How will identities be managed in your deployment?

  See Identity management.

LDAP schema extensions

As described in LDAP schema, DS servers ship with many standard LDAP schema definitions. In addition, you can update LDAP schema definitions while the server is online.

This does not mean, however, that you can avoid schema updates for your deployment. Instead, unless the data for your deployment requires only standard definitions, you must add LDAP schema definitions before importing your data.

Follow these steps to prepare the schema definitions to add:

1. If your data comes from another LDAP directory service, translate the schema definitions used by the data from the existing directory service. Use them to start an LDIF modification list of planned schema updates, as described in Update LDAP Schema.

   The schema definitions might not be stored in the same format as DS definitions. Translating from existing definitions should be easier than creating new ones, however.

   As long as the existing directory service performs schema checking for updates, the directory data you reuse already conforms to those definitions. You must apply them to preserve data integrity.

2. If your data comes from applications that define their own LDAP schema, add those definitions to your list of planned schema updates.

3. Match as much of your data as possible to the standard LDAP schema definitions listed in the Schema Reference.

4. Define new LDAP schema definitions for data that does not fit existing definitions. This is described in About LDAP Schema, and Update LDAP Schema.

   Add these new definitions to your list.

   Avoid any temptation to modify or misuse standard definitions, as doing so can break interoperability.

Once your schema modifications are ready, use comments to document your choices in the source LDIF. Keep the file under source control. Apply a change control process to avoid breaking schema definitions in the future.

Perhaps you can request object identifiers (OIDs) for new schema definitions from an OID manager in your organization. If not, either take charge of OID assignment, or else find an owner who takes charge. OIDs must remain globally unique, and must not be reused.

The DIT

When defining the base DNs and hierarchical structure of the DIT, keep the following points in mind:

- For ease of use, employ short, memorable base DNs with RDNs using well-known attributes.

  For example, you can build base DNs that correspond to domain names from domain component (`dc`) RDNs. The sample data for Example.com uses `dc=example,dc=com` .

  Well-known attributes used in base DNs include the following:

  - `c` : country, a two-letter ISO 3166 country code
  - `dc` : component of a DNS domain name
  - `l` : locality
  - `o` : organization
  - `ou` : organizational unit
  - `st` : state or province name

- For base DNs and hierarchical structures, depend on properties of the data that do not change.

  For example, the sample data places all user entries under `ou=People,dc=example,dc=com` . There is no need to move a user account when the user changes status, role in the organization, location, or any other property of their account.

- Introduce hierarchical branches in order to group similar entries.

  As an example of grouping similar entries, the following branches separate apps, devices, user accounts, and LDAP group entries:

  - `ou=Apps,dc=example,dc=com`
  - `ou=Devices,dc=example,dc=com`
  - `ou=Groups,dc=example,dc=com`
  - `ou=People,dc=example,dc=com`

  In this example, client application accounts belong under `ou=Apps` . A user account under `ou=People` for a device owner or subscriber can have an attribute referencing devices under `ou=Devices` . Device entries can reference their `owner` in `ou=People` . Group entries can include members from any branch. Their members' entries would reference the groups with `isMemberOf` .

- Otherwise, use hierarchical branches only as required for specific features. Such features include the following:

  - Access control

- Data distribution
- Delegated administration
- Replication
- Subentries

Use delegated administration when multiple administrators share the directory service. Each has access to manage a portion of the directory service or the directory data. By default, ACIs and subentries apply to the branch beneath their entry or parent. If a delegated administrator must be able to add or modify such operational data, the DIT should prevent the delegated administrator from affecting a wider scope than you intend to delegate.

As described in About Replication, the primary unit of replication is the base DN. If necessary, you can split a base DN into multiple branches. For example use cases, read Deployment patterns.

Once you have defined your DIT, arrange the directory data you import to follow its structure.

Data views

If client applications only use LDAP to access the directory service, you have few choices to make when configuring connection handlers. The choices involve how to secure connections, and whether to limit access with client hostnames or address masks. The client applications all have the same LDAP view of the directory data.

This is true for DSML applications as well if you use the DSML gateway.

If client applications use RESTful HTTP APIs to access the directory service, then you have the same choices as for LDAP. In addition, you must define how the HTTP JSON resources map to LDAP entries.

Figure 1. Mapping JSON Resources to LDAP Entries

As shown above, you can define multiple versioned APIs providing alternative views of the underlying LDAP data if required for your deployment. The basic sample API that ships with DS servers is likely not sufficient. For details, see REST to LDAP reference.

Data management

In a shared or high-scale directory service, service management—installation and configuration, backup, and recovery—may be the responsibility of only a few specialists. These tasks may be carefully scripted.

Directory data management is, however, often a task shared by multiple users. Many of these tasks may be performed manually. In addition, users may be responsible for profile data in their own entry, including passwords, for example. You can arrange the DIT hierarchically to make it easier to scope control of administrative access.

Your plan must define who should have what access to which data, and list the privileges and access controls to grant such access. Read Administrative roles to review the alternatives.

Groups

As described in Groups, DS directory servers offer dynamic, static, and virtual static group implementations:

- Dynamic groups identify members with an LDAP URL.

  An entry belongs to a dynamic group when it matches the base DN, scope, and filter defined in a member URL of the group. Changes to the entry can modify its dynamic group membership.

- Static groups enumerate each member. The size of a static group entry can grow very large in a high-scale directory.

- Virtual static groups are like dynamic groups, but the server can be configured to have them return a list of members when read.

Consider your data and client applications. Use dynamic or virtual static groups whenever possible. When you cannot use dynamic or virtual static groups, use static groups and consider caching them in memory, as described in Cache for Large Groups.

Shared data

As described in Virtual attributes, and Collective attributes, DS servers support virtual and collective attributes that let entries share attribute values. Sharing attribute values where it makes sense can significantly reduce data duplication, saving space and avoiding maintenance updates.

Consider your directory data. You can use virtual or collective attributes to replace attributes that repeat on many entries and can remain read-only on those entries. Familiar use cases include postal addresses that are the same for everyone in a given location, and class of service properties that depend on a service level attribute.

Caching

A directory server is an object-oriented database. It will therefore exhibit its best performance when its data is cached in memory. This is as true for large static groups mentioned in Groups as it is for all directory data.

A disadvantage of caching all data is that systems with enough RAM are more expensive. Consider the suggestions in Database Cache Settings, testing the results for your data when planning your deployment.

Identity management

DS servers have the following features that make them well-suited to serve identity data:

- LDAP entries provide a natural model for identity profiles and accounts.

  LDAP entries associate a unique name with a flat, extensible set of profile attributes such as credentials, location or contact information, descriptions, and more. LDAP schemas define what entries can contain, and are themselves extensible at runtime.

  Because they are defined and accessible in standard ways, and because fine-grained access controls can protect all attributes of each entry, the profiles can be shared by all network participants as the single source of identity information.

Profile names need not be identified by LDAP DNs. For HTTP access, DS servers offer several ways to map to a profile, including mapping an HTTP user name to an LDAP name, or using an OAuth 2.0 access token instead. For devices and applications, DS servers can also map public key certificates to profiles.

- Directory services are optimized to support common authentication mechanisms.

  LDAP entries easily store and retrieve credentials, keys, PKI metadata, and more. Where passwords are used, directory services support multiple secure and legacy password storage schemes. You can also configure directory servers to upgrade password storage when users authenticate.

  Each individual server can process thousands of authentication requests per second.

  ForgeRock® Access Management integrates directory authentication into full access management services, including making directory authentication part of a flow that potentially involves multiple authentication steps.

- Directory services support user self-service operations and administrator intervention.

  Directory services let you protect accounts automatically or manually by locking accounts after repeated authentication failure, expiring old passwords, and tracking authentication times to distinguish active and inactive accounts. Directory services can then notify applications and users about account-related events, such as account lockout, password expiration, and other events.

  Users can be granted access to update their own profiles and change their passwords securely. If necessary, administrators can also be granted access to update profiles and to reset passwords.

  ForgeRock Identity Management integrates directory account management features into full identity management services.

Further Reading on Managing Identities

Topics	References
Account Management	• Accounts • Active Accounts
Authentication	• Authentication mechanisms • Authentication (binds) • Certificate-Based Authentication • Pass-Through Authentication • DS REST APIs
Authorization	• Configure HTTP Authorization • Proxied authorization
Password Management	• Password management • Changing passwords over LDAP • Changing passwords over HTTP

Directory access

Consider these topics when designing the access model for your deployment.

Separation of duties (SoD)

The fewer restrictions you place on an administrative account, the greater the danger the account will be misused.

As described in Administrative Access, you can avoid using directory superuser accounts for most operations. Instead, limit administrator privileges and access to grant only what their roles require. The first high-level distinction to make is between operational staff who manage the service, and those users who manage directory data. Read the section cited for fine-grained distinctions.

When your deployment involves delegated administration, it is particularly important to grant only required access to the delegates. This is easier if your DIT supports appropriate access scopes by default, as described in The DIT.

Immutable and mutable configuration

An immutable configuration does not change at runtime. A mutable configuration does change at runtime.

With an immutable configuration, you maintain the server configuration as an artifact under source control, and manage changes by applying the same process you use for other source code. This approach helps prevent surprises in production configurations. If properly applied, there is little risk of rolling out an untested change.

With a mutable configuration, operational staff have more flexibility to make changes. This approach requires even more careful change management for test and production systems.

DS server configurations can be immutable, except for the portion devoted to replication, which evolves as peer servers come and go.

DS directory data, however, must remain mutable to support write operations. As long as you separate directory data from the configuration, this does not prevent you from replacing directory server replicas. As described in Manual Initialization, new replicas can start with existing data sets.

Fine-grained access

DS servers provide both HTTP and LDAP access to directory data. HTTP access to directory data eventually translates to LDAP access internally. At the LDAP level, DS servers provide powerful, fine-grained access control.

The default server behavior is to refuse all access. All DS servers therefore grant some level of access through privileges, and through access controls. For details, see Access control.

Access control instructions (ACIs) in directory data take the form of `aci` LDAP attributes, or `global-aci` properties in the server configuration. You write ACIs in a domain-specific language. The language lets you describe concisely who has access to what under what conditions. When configuring access control, notice that access controls apply beneath their location in the directory information tree. As a result, some ACIs, such as those granting access to LDAP controls and extended operations, must be configured for the entire server rather than a particular location in the data.

Privileges

Administrative privileges provide a mechanism that is separate from access control to restrict what administrators can do.

You assign privileges to users as values of the `ds-privilege-name` LDAP attribute. You can assign privileges to multiple users with collective attribute subentries. For details, see Administrative Privileges.

Take care when granting privileges, especially the following privileges:

- `bypass-acl`: The holder is not subject to access control.
- `config-write`: The holder can edit the server configuration.
- `modify-acl`: The holder can edit access control instructions.
- `privilege-change`: The holder can edit administrative privileges.
- `proxied-auth`: The holder can make requests on behalf of another user, including directory superusers such as `uid=admin`.

Authentication

DS servers support a variety of authentication mechanisms.

When planning your service, use the following guidelines:

- Limit anonymous access to public data.
- Allow simple (username and password) authentication only over secure connections.
- Require client applications to authenticate based on public key certificates (EXTERNAL SASL mechanism) rather than simple authentication where possible.

For details, see Authentication mechanisms.

Proxy layer

DS directory proxy servers, and DS DSML and REST to LDAP gateway applications encapsulate DS directory services, and offer access to other directory services.

Unlike directory servers, directory proxy servers do not hold directory data, and so use global access policies rather than ACIs. You define global access policies as server configuration objects. For details, see Access control.

As mentioned in System Resources, be aware that for high-performance services you may need to deploy as many proxy servers or gateways as directory servers.

For details about DS LDAP proxy services, see LDAP proxy.

HTTP access

As described in HTTP access, you can configure DS servers to provide RESTful HTTP access.

If you deploy this capability, you must configure how HTTP resources map to LDAP entries. This is explained in Data Views. For details, see REST to LDAP reference.

Higher-level abstraction

Although LDAP and RESTful HTTP access ensure high performance, your deployment may require a higher level of abstraction than LDAP or HTTP can provide.

Other ForgeRock® Identity Platform components offer such higher-level abstractions. For example, ForgeRock Access Management software lets you plug into directory services for authentication and account profiles, and then orchestrate powerful authentication and authorization scenarios. ForgeRock Identity Management software can plug into directory services to store configuration and account profiles, to provide user self-services, and to synchronize data with a wide variety of third-party systems.

For an introduction to the alternatives, read the Identity Platform Guide.

Data replication

Replication is the process of synchronizing data updates across directory servers. Replication is the feature that makes the directory a highly available distributed database.

Consistency and availability

Replication is designed to provide high availability with tolerance for network partitions. In other words, the service continues to allow both read and write operations when the network is down. Replication provides eventual consistency, not immediate consistency.

According to what is called the CAP theorem, it appears to be impossible to guarantee consistency, availability, and partition tolerance when network problems occur. The CAP theorem makes the claim that distributed databases can guarantee at most two of the following three properties:

Consistency

Read operations reflect the latest write operation (or result in errors).

Availability

Every correct operation receives a non-error response.

Partition Tolerance

The service continues to respond even when the network between individual servers is down or operating in degraded mode.

When the network connection is down between two replicas, replication is temporarily interrupted. Client applications continue to receive responses to their requests, but clients making requests to different servers will not have the same view of the latest updates. The discrepancy in data on different replicas also arises temporarily when a high update load takes time to fully process across all servers.

Eventual consistency can be a trap for the unwary. The client developer who tests software only with a single directory server might not notice problems that become apparent when a load balancer spreads requests evenly across multiple servers. A single server is immediately consistent for its own data. Implicit assumptions about consistency therefore go untested.

For example, a client application that implicitly assumes immediate consistency might perform a write quickly followed by a read of the same data. Tests are all successful when only one server is involved. In deployment, however, a load balancer distributes requests across multiple servers. When the load balancer sends the read to a replica that has not yet processed the write, the client application appears to perform a successful write, followed by a successful read that is inconsistent with the write that succeeded!

When deploying replicated DS servers, keep this eventual consistency trap in mind. Educate developers about the trade off, review patches, and test and fix client applications under your control. In deployments with client applications that cannot be fixed, use affinity load balancing in DS directory proxy servers to work around broken clients. For details, see Load Balancing.

Deploying replication

In DS software, the role of a replication server is to transmit messages about updates. Directory servers receive replication messages from replication servers, and apply updates accordingly, meanwhile serving client applications.

Deploy at least two replication servers per local network in case one fails, and deploy more if you have many directory servers per replication server. For LAN-based deployments, each directory server can double as a replication server. For large, WAN-based deployments, consider using standalone replication servers and directory servers.

In a widely distributed deployment, be aware that replication servers all communicate with each other. Directory servers always communicate through replication servers, even if the replication service runs in the same server process as the directory server. By assigning servers to replication groups, you can ensure that directory servers only connect to local replication servers until they need to fail over to remote replication servers. This limits the WAN replication traffic to messages between replication servers, except when all replication servers on the LAN are down. For details, see Install standalone servers and Replication Groups.

Deploy the replication servers first. You can think of them as providing a network service (replication) in the same way DNS provides a network service (name resolution). You therefore install and start replication servers before you add directory servers.

After you install a directory server and configure it as a replica, you must initialize it to the current replication state. There are a number of choices for this, as described in Manual Initialization. Once a replica has been initialized, replication eventually brings its data into a consistent state with the other replicas. As described in Consistency and availability, give a heavy update load or significant network latency, temporary inconsistency is expected. You can monitor the replication status to estimate when replicas will converge on the same data set.

Client applications can adopt best practices that work with eventual consistency, as described in Best practices, Optimistic Concurrency (MVCC), and Update. To work around broken client applications that assume immediate consistency, use affinity load balancing in directory proxy servers. For details, see Load Balancing.

Some client applications need notifications when directory data changes. Client applications cannot participate in replication itself, but can get change notifications. For details, see Changelog for notifications.

Scaling replication

When scaling replicated directory services, keep the following rules in mind:

- Read operations affect only one replica.

  To add more read performance, use more powerful servers or add servers.

- Write operations affect all replicas.

  To add more write performance, use more powerful servers or add separate replication domains.

When a replica writes an update to its directory data set, it transmits the change information to its replication server for replay elsewhere. The replication server transmits the information to connected directory servers, and to other replication servers replicating the same data. Those replication servers transmit the message to others until all directory servers have received the change information. Each directory server must process the change, reconciling it with other change information.

As a result, you cannot scale up write capacity by adding servers. Each server must replay all the writes.

If necessary, you can scale up write capacity by increasing the capacity of each server (faster disks, more powerful servers), or by splitting the data into separate sets that you replicate independently (data distribution).

High availability

In shared directory service deployments, the directory must continue serving client requests during maintenance operations, including service upgrades, during network outage recovery, and in spite of system failures.

DS replication lets you build a directory service that is always online. DS directory proxy capabilities enable you to hide maintenance operations from client applications. You must still plan appropriate use of these features, however.

As described above, replication lets you use redundant servers and systems that tolerate network partitions. Directory server replicas continue to serve requests when peer servers fail or become unavailable. Directory proxy servers route around directory servers that are down for maintenance or down due to failure. When you upgrade the service, you roll out one upgraded DS server at a time. New servers continue to interoperate with older servers, so the whole service never goes down. All of this depends on deploying redundant systems, including network links, to eliminate single points of failure. For more, see High Availability.

As shown in that section, your deployment may involve multiple locations, with servers communicating locally over LANs and remotely over WANs. Your deployment can also use separate replication topologies, for example, in order to sustain very high write loads, or to separate volatile data from more static data. Carefully plan your load balancing strategy to offer good service at a reasonable cost. By using replication groups, you can limit most replication traffic over WAN links to communications between replication servers. Directory proxy servers can direct client traffic to local servers until it becomes necessary to failover to remote servers.

Sound operational procedures play as important a role in availability as good design. Operational staff maintaining the directory service must be well-trained and organized so that someone is always available to respond if necessary. They must have appropriate tools to monitor the service in order to detect situations that need attention. When maintenance, debugging, or recovery is required, they should have a planned response in most cases. Your deployment plans should therefore cover the requirements and risks that affect your service.

Before finalizing deployment plans, make sure that you understand key availability features in detail. For details about replication, read Replication. For details about proxy features, read LDAP proxy.

Backup and recovery

Make sure your plans define how you:

- Back up directory data
- Safely store backup files
- Recover your directory service from backup

DS servers store data in *backends*. A backend is a private server repository that can be implemented in memory, as a file, or as an embedded database. DS servers use local backends to store directory data, server configuration, LDAP schema, and administrative tasks. Directory proxy servers implement a type of backend for non-local data, called a proxy backend, which forwards LDAP requests to a remote directory service.

For performance reasons, DS servers store directory data in a local database backend, which is a backend implemented using an embedded database. Database backends are optimized to store directory data. Database backends hold data sets as key-value pairs. LDAP objects fit the key-value model very effectively, with the result that a single database backend can serve hundreds of millions of LDAP entries. Database backends support indexing and caching for fast lookups in large data sets. Database backends do not support relational queries or direct access by other applications. For more information, see Data storage.

Backup and restore procedures are described in Backup and restore. When planning your backup and recovery strategies, be aware of the following key features:

- Backups are *not guaranteed to be compatible* across major and minor server releases. *Restore backups only on directory servers of the same major or minor version.*
- Backup and restore tasks can run while the server is online. They can, however, have a significant impact on server performance.

  For deployments with high performance requirements, consider dedicating a replica to perform only backup operations. This prevents other replicas from stealing cycles to back up data that could otherwise be used to serve client applications.
- When you restore replicated data from backup, the replication protocol brings the replica up to date with others after the restore operation.

  This requires, however, that the backup is recent enough. Backup files older than the replication purge delay (default: 3 days) are stale and should be discarded.
- Directory data replication ensures that all servers converge on the latest data. If your data is affected by a serious accidental deletion or change, you must restore the entire directory service to an earlier state.

  For details, see Recover From User Error.
- When you restore encrypted data, the server must have the same shared master key as the server that performed the backup.

  Otherwise, the directory server cannot decrypt the symmetric key used to decrypt the data. For details, see Data encryption.
- For portability across versions, and to save directory data in text format, periodically export directory data to LDIF.

  The LDIF serves as an alternative backup format. In a disaster recovery situation, restore directory data by importing the version saved in LDIF.

  > **Important**
  >
  > LDIF stores directory data in text format. It offers no protection of the data.
  > Use an external tool to encrypt the LDIF you export to protect against data leaks and privacy breaches.

  If you have stored passwords with a reversible encryption password storage scheme, be aware that the server must have the same shared master key as the server that encrypted the password.

  For details, see Import and export, and Manual initialization.
- You can perform a file system backup of your servers instead of using the server tools.

  You must, however, *stop the server before taking a file system backup*. Running DS directory servers cannot guarantee that database backends will be recoverable unless you back them up with the DS tools.

Monitoring and auditing

When monitoring DS servers and auditing access, be aware that you can obtain some but not all data remotely.

The following data sources allow remote monitoring:

- HTTP connection handlers expose a `/metrics/api` endpoint that offers RESTful access to monitoring data, and a `/metrics/prometheus` endpoint for Prometheus monitoring software.

  For details, see Use Administrative APIs.

- LDAP connection handlers expose a `cn=monitor` branch that offers LDAP access to monitoring data.

  For details, see LDAP-based monitoring.

- JMX connection handlers offer remote access.

  For details, see JMX-based monitoring.

- SNMP connection handlers enable remote access.

  For details, see SNMP-based monitoring.

- You can configure alerts to be sent over JMX or SMTP (mail).

  For details, see Alerts.

- Replication conflicts are found in the directory data.

  For details, see Replication Conflicts.

- Server tools, such as the `status` command, can run remotely.

  For details, see Status and tasks.

The following data sources require access to the server system:

- Server logs, as described in Logging.

  DS servers write log files to local disk subsystems. In your deployment, plan to move access logs that you want to retain. Otherwise the server will eventually remove logs according to its retention policy to avoid filling up the disk.

- Index verification output and statistics, as described in Rebuild Indexes, and Verify Indexes.

When defining how to monitor the service, use the following guidelines:

- Your service level objectives (SLOs) should reflect what your stakeholders expect from the directory service for their key client applications.

  If SLOs reflect what stakeholders expect, and you monitor them in the way key client applications would experience them, your monitoring system can alert operational staff when thresholds are crossed, before the service fails to meet SLOs.

- Make sure you keep track of resources that can expire, such as public key certificates and backup files from directory server replicas, and resources that can run out, such as system memory and disk space.

- Monitor system and network resources in addition to the directory service.

  Make sure operational staff can find and fix problems with the system or network, not only the directory.

- Monitor replication delay, so you can take action when it remains high and continues to increase over the long term.

In order to analyze server logs, use other software, such as Splunk, which indexes machine-generated logs for analysis.

If you require integration with an audit tool, plan the tasks of setting up logging to work with the tool, and analyzing and monitoring the data once it has been indexed. Consider how you must retain and rotate log data once it has been consumed, as a high-volume service can produce large volumes of log data.

Hardening and security

When you first set up DS servers with the evaluation profile, the configuration favors ease of use over security for Example.com data.

All other configurations and setup profiles leave the server hardened for more security by default. You explicitly grant additional access if necessary.

For additional details, see Security.

Tests

In addition to planning tests for each custom plugin, test each feature you deploy. Perform functional and non-functional testing to validate that the directory service meets SLOs under load in realistic conditions. Include acceptance tests for the actual deployment. The data from the acceptance tests help you to make an informed decision about whether to go ahead with the deployment or to roll back.

Functional tests

Functional testing validates that specified test cases work with the software considered as a black box.

As ForgeRock already tests DS servers and gateways functionally, focus your functional testing on customizations and service level functions. For each key capability, devise automated functional tests. Automated tests make it easier to integrate new deliveries to take advantage of recent bug fixes, and to check that fixes and new features do not cause regressions.

As part of the overall plan, include not only tasks to develop and maintain your functional tests, but also to provision and to maintain a test environment in which you run the functional tests before you significantly change anything in your deployment. For example, run functional tests whenever you upgrade any server or custom component, and analyze the output to understand the effect on your deployment.

Performance tests

With written SLOs, even if your first version consists of guesses, you turn performance plans from an open-ended project to a clear set of measurable goals for a manageable project with a definite outcome. Therefore, start your testing with service level objectives clear definitions of success.

Also, start your testing with a system for load generation that can reproduce the traffic you expect in production, and underlying systems that behave as you expect in production. To run your tests, you must therefore generate representative load data and test clients based on what you expect in production. You can then use the load generation system to perform iterative performance testing.

Iterative performance testing consists of identifying underperformance, and the bottlenecks that cause it, and discovering ways to eliminate or work around those bottlenecks. Underperformance means that the system under load does not meet service level objectives. Sometimes resizing or tuning the system can help remove bottlenecks that cause underperformance.

Based on SLOs and availability requirements, define acceptance criteria for performance testing, and iterate until you have eliminated underperformance.

Tools for running performance testing include the tools listed in Performance Tests, and Gatling, which uses a domain specific language for load testing. To mimic the production load, examine the access patterns, and the data that DS servers store. The representative load should reflect the distribution of client access expected in production.

Although you cannot use actual production data for testing, you can generate similar test data using tools, such as the `makeldif` command.

As part of the overall plan, include not only tasks to develop and maintain performance tests, but also to provision and to maintain a pre-production test environment that mimics your production environment. Security measures in your test environment must also mimic your production environment, as security measures can impact performance.

Once you are satisfied that the baseline performance is acceptable, run performance tests again when something in your deployment changes significantly with respect to performance. For example, if the load or number of clients changes significantly, it could cause the system to underperform. Also, consider the thresholds that you can monitor in the production system to estimate when your system might start to underperform.

Deployment tests

Here, deployment testing is a description rather than a term. It refers to the testing implemented within the deployment window after the system is deployed to the production environment, but before client applications and users access the system.

Plan for minimal changes between the pre-production test environment and the actual production environment. Then test that those changes have not cause any issues, and that the system generally behaves as expected.

Take the time to agree upfront with stakeholders regarding the acceptance criteria for deployment tests. When the production deployment window is small, and you have only a short time to deploy and test the deployment, you must trade off thorough testing for adequate testing. Make sure to plan enough time in the deployment window for performing the necessary tests and checks.

Include preparation for this exercise in your overall plan, as well as time to check the plans close to the deployment date.

Configuration changes

Make sure your plan defines the change control process for configuration. Identify the ways that the change is likely to affect your service. Validate your expectations with appropriate functional, integration, and stress testing. The goal is to adapt how you maintain the service before, during, and after the change. Complete your testing before you subject all production users to the change.

Review the configuration options described here, so that you know what to put under change control.

Server configuration

DS servers store configuration in files under the server's `config` directory. When you set up a server, the setup process creates the initial configuration files based on templates in the server's `template` directory. File layout describes the files.

When a server starts, it reads its configuration files to build an object view of the configuration in memory. This view holds the configuration objects, and the constraints and relationships between objects. This view of the configuration is accessible over client-side and server-side APIs. Configuration files provide a persistent, static representation of the configuration objects.

Configuration tools use the client-side API to discover the server configuration and to check for constraint violations and missing relationships. The tools prevent you from breaking the server configuration structurally by validating structural changes before applying them. The server-side API allows the server to validate configuration changes, and to synchronize the view of the configuration in memory with the file representation on disk. If you make changes to the configuration files on disk while the server is running, the server can neither validate the changes beforehand, nor guarantee that they are in sync with the view of the configuration in memory.

DS Server Configuration

Method	Notes
Tools (`dsconfig` and others)	Stable, supported, public interfaces for editing server configurations. Most tools work with local and remote servers, both online and offline.
Files	Internal interface to the server configuration, subject to change without warning in any release. If you must make manual changes to configuration files, always stop the DS server before editing the files. If the changes break the configuration, compare with the `var/config.ldif.startok` file, and with the compressed snapshots of the main configuration in the `var/archived-configs/` directory.
REST (HTTP) API	Internal interface to the server configuration, subject to change without warning in any release. Useful for enabling browser-based access to the configuration.

Once a server begins to replicate data with other servers, the part of the configuration pertaining to replication is specific to that server. As a result, a server effectively cannot be cloned once it has begun to participate in data replication. When deploying servers, do not initialize replication until you have deployed the server.

Gateway configuration

You edit files to configure DS DSML and REST to LDAP gateway web applications.

The gateways do not have external configuration APIs, and must be restarted after you edit configuration files for the changes to take effect.

Documentation

The DS product documentation is written for readers like you, who are architects and solution developers, as well as for DS developers and for administrators who have had DS training. The people operating your production environment need concrete documentation specific to your deployed solution, with an emphasis on operational policies and procedures.

Procedural documentation can take the form of a runbook with procedures that emphasize maintenance operations, such as backup, restore, monitoring and log maintenance, collecting data pertaining to an issue in production, replacing a broken server or web application, responding to a monitoring alert, and so forth. Make sure you document procedures for taking remedial action in the event of a production issue.

Furthermore, to ensure that everyone understands your deployment and to speed problem resolution in the event of an issue, changes in production must be documented and tracked as a matter of course. When you make changes, always prepare to roll back to the previous state if the change does not perform as expected.

Maintenance and support

If you own the architecture and planning, but others own the service in production, or even in the labs, then you must plan coordination with those who own the service.

Start by considering the service owners' acceptance criteria. If they have defined support readiness acceptance criteria, you can start with their acceptance criteria. You can also ask yourself the following questions:

- What do they require in terms of training in DS software?
- What additional training do they require to support your solution?
- Do your plans for documentation and change control, as described in Documentation, match their requirements?
- Do they have any additional acceptance criteria for deployment tests, as described in Deployment tests?

Also, plan back line support with ForgeRock or a qualified partner. The aim is to define clearly who handles production issues, and how production issues are escalated to a product specialist if necessary.

Include a task in the overall plan to define the hand off to production, making sure there is clarity on who handles monitoring and issues.

Rollout

In addition to planning for the hand off of the production system, also prepare plans to roll out the system into production. Rollout into production calls for a well-choreographed operation, so these are likely the most detailed plans.

Take at least the following items into account when planning the rollout:

- Availability of all infrastructure that DS software depends on, such as the following:
  - Server hosts and operating systems
  - Web application containers for gateways
  - Network links and configurations
  - Persistent data storage
  - Monitoring and audit systems
- Installation for all DS servers.
- Final tests and checks.
- Availability of the personnel involved in the rollout.

In your overall plan, leave time and resources to finalize rollout plans toward the end of the project.

Ongoing change

To succeed, your directory service must adapt to changes, some that you can predict, some that you cannot.

In addition to the configuration changes covered in Configuration changes, predictable changes include the following:

Increases and decreases in use of the service

For many deployments, you can predict changes in the use of the directory service, and in the volume of directory data.

If you expect cyclical changes, such as regular batch jobs for maintenance or high traffic at particular times of the year, test and prepare for normal and peak use of the service. For deployments where the peaks are infrequent but much higher than normal, it may be cost effective to dedicate replicas for peak use that are retired in normal periods.

If you expect use to increase permanently, then decide how much headroom you must build into the deployment. Plan to monitor progress and add capacity as necessary to maintain headroom, and to avoid placing DS servers under so much stress that they stop performing as expected.

If you expect use to decrease permanently, at some point you will retire the directory service. Make sure all stakeholders have realistic migration plans, and that their schedules match your schedule for retirement.

Depending on the volume of directory data and the growth you expect for the directory service, you may need to plan for scalability beyond your initial requirements.

As described in Scaling replication, you can increase read performance by adding servers. To increase write performance, adding servers is not the solution. Instead, you must split the data into sets that you replicate separately.

Luckily, single directory services can already support thousands of replicated write operations per second, meaning millions of write operations per hour. It may well be possible to achieve appropriate performance by deploying on more powerful servers, and by using higher performance components, such as dedicated SSD disks instead of traditional disks.

When scaling up the systems is not enough, you must instead organize the DIT to replicate different branches separately. Deploy the replicas for each branch on sets of separate systems. For details, see High Scalability.

Directory service upgrades

ForgeRock regularly offers new releases of DS software. These include maintenance and feature releases. Supported customers may also receive patch releases for particular issues.

Patch and maintenance releases are generally fully compatible. Plan to test and roll out patch and maintenance releases swiftly, as they include important updates such as fixes for security issues or bugs that you must address quickly.

Plan to evaluate feature releases as they occur. Even if you do not intend to use new features immediately, you might find important improvements that you should roll out. Furthermore, by upgrading regularly you apply fewer changes at a time than you would by waiting until the end of support life and then performing a major upgrade.

Key rotation

Even if you do not change the server configuration, the signatures eventually expire on certificates used to secure connections. You must at minimum replace the certificates. You could also change the key pair in addition to getting a new certificate.

If you encrypt directory data for confidentiality, you might also choose to rotate the symmetric encryption key.

Unpredictable changes include the following:

Disaster recovery

As described in High Availability, assess the risks. In light of the risks, devise and test disaster recovery procedures.

For details, refer to Disaster recovery.

New security issues

Time and time again, security engineers have found vulnerabilities in security mechanisms that could be exploited by attackers. Expect this to happen during the lifetime of your deployment.

You might need to change the following at any time:

- Keys used to secure connections
- Keys used to encrypt directory data
- Protocol versions used to secure connections
- Password storage schemes
- Deployed software that has a newly discovered security bug

In summary, plan to adapt your service to changing conditions. To correct security bugs and other issues and to recover from minor or major disasters, be prepared to patch, upgrade, roll out, and roll back changes as part of your regular operations.

Deployment patterns

Use these patterns in your deployments.

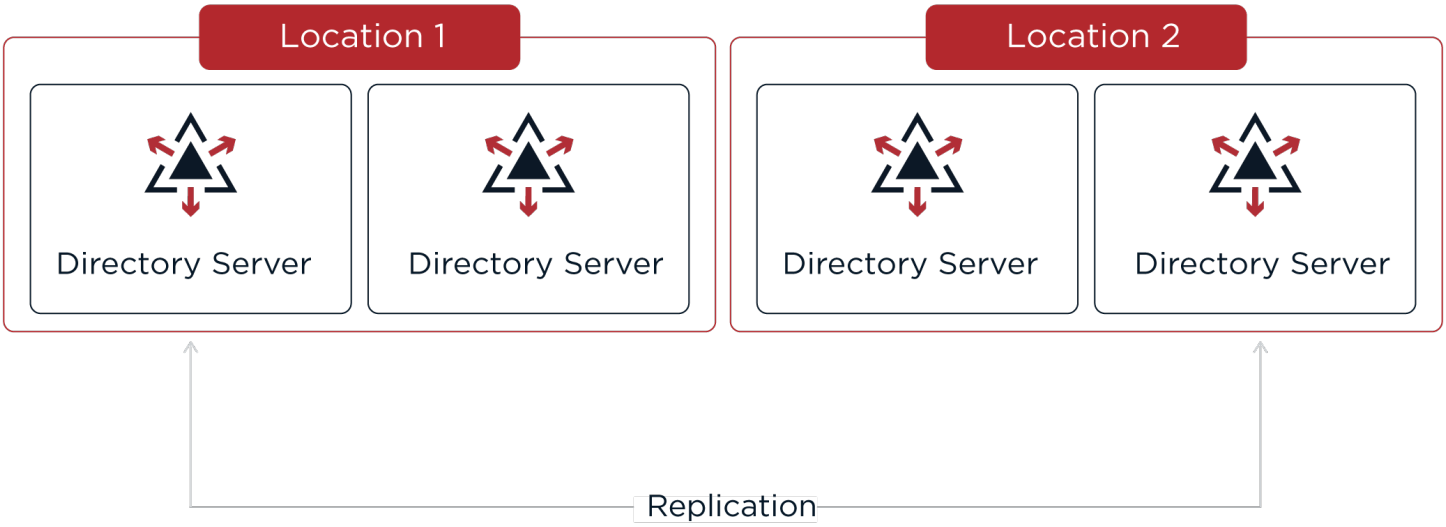
High availability

When you deploy DS servers into a highly available directory service, you are implementing the primary use case for which DS software is designed:

- Data replication lets you eliminate single points of failure.

  When using replication, keep in mind the trade off, described in Consistency and Availability.

- DS upgrade capabilities let you perform rolling upgrades without ever taking the whole service offline.
- If desired, DS proxy capabilities help you provide a single point of entry for directory applications, hiding the fact that individual servers do go offline.



You build a highly available directory service by using redundant servers in multiple locations. If possible, use redundant networks within and between locations to limit network partitions.

When you install or upgrade a highly available directory service, bring component servers online in the following order:

1. **Standalone Replication Servers**

   If you have a large DS service with standalone replication servers, they provide the foundation for high availability. They communicate change messages to directory server replicas, and they also let other servers discover available replicas.

2. **Directory Servers**

   Directory server replicas ultimately respond to client application requests. They hold an eventually convergent copy of the directory data. They require a replication service to communicate with other replicas about changes to their copy of the directory data.

3. **Directory Proxy Servers**

   If you use DS directory proxy servers for a unified view of the service, they discover DS replicas by querying the replication service. They forward requests to the replicas, and responses to the client applications.

In addition to redundant server components, avoiding downtime depends on being able operationally to recover quickly and effectively. Prepare and test your plans. Even if disaster strikes, you will be able to repair the service promptly.

Plan how you store backup files both onsite and offsite. Make sure you have safe copies of the master keys that let directory servers decrypt encrypted data. For details, see Backup and restore.

When defining disaster recovery plans, consider at least the following situations:

- **The entire service is down.**

It is important to distinguish whether the situation is temporary and easily recoverable, or permanent and requires implementation of disaster recovery plans.

If an accident, such as a sudden power cut at a single-site deployment, brought all the servers down temporarily, restart them when the power returns. As described in Server Recovery, directory servers might have to replay their transaction logs before they are ready. However, this operation happens automatically when you restart the server.

In a disaster, the entire service could go offline permanently. Be prepared to rebuild the entire service. For details, refer to Disaster recovery.

- **Part of the service is down.**

  Failover client applications to servers still in operation, and restart or rebuild servers that are down.

  You can configure directory proxy servers to fail over automatically, and to retry requests for certain types of failure. For details, see LDAP proxy.

- **The network is temporarily down between servers.**

  By default, you do not need to take immediate action for a temporary network outage. As long as client applications can still communicate with local servers, replication is designed to catch up when the network connections are reestablished.

  By default, when a directory server replica cannot communicate with a replication server, the `isolation-policy` setting prevents the directory server replica from accepting updates.

  In any case, if the network is partitioned longer than the replication purge delay (default: 3 days), then replication will have purged older data, and might not be able to catch up. For longer network outages, you will have to reinitialize replication.

When defining procedures to rebuild a service that is permanently offline, the order of operations is the same as during an upgrade:

1. Redirect client applications to a location where the service is still running.

   If the proxy layer is still running, directory proxy servers can automatically fail requests over to remote servers that are still running.

2. Rebuild replication servers.

3. Rebuild directory servers.

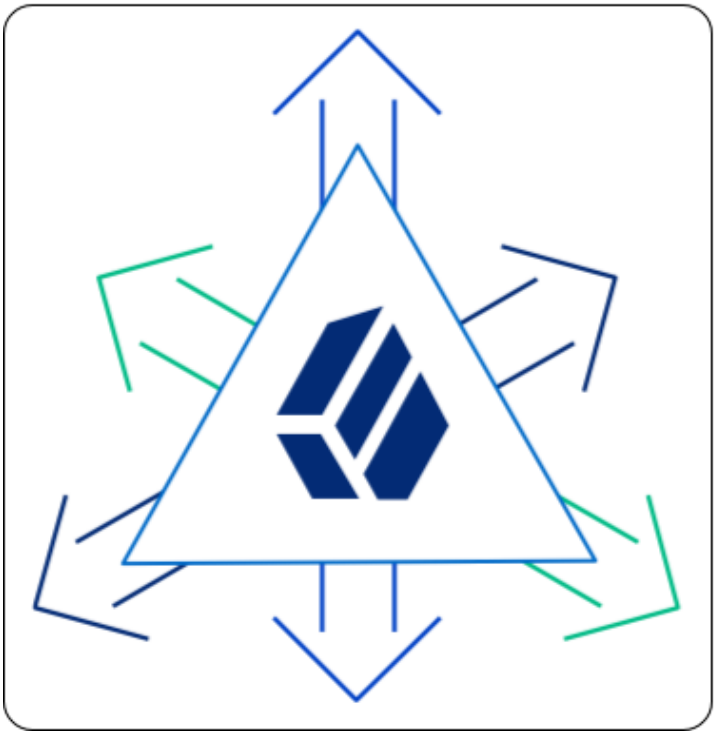
4. Rebuild directory proxy servers.

High scalability

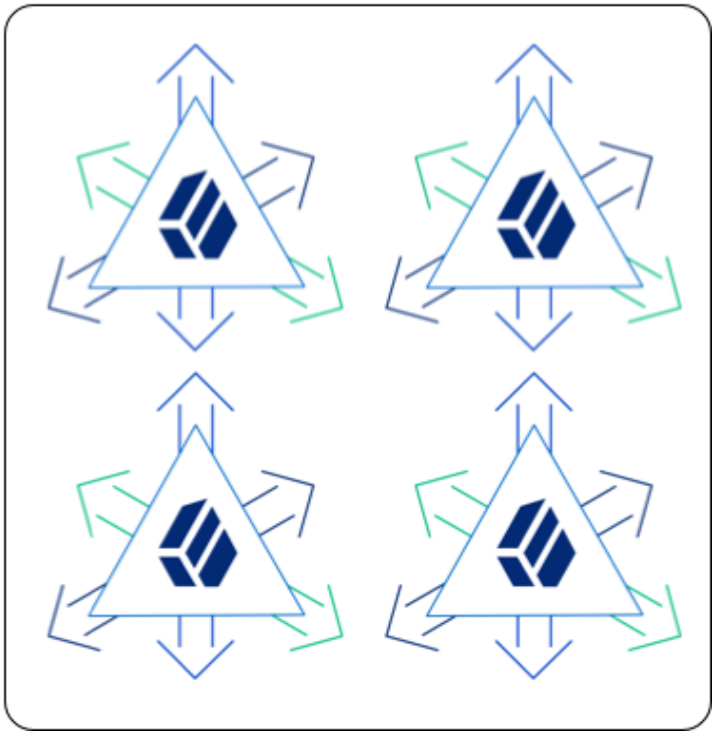
A high-scale directory service is one that requires high performance. For example, very high throughput or very low response times, or both. Or, it has a large data set, such as 100 million entries. When building a high-scale directory, the fundamental question is whether to scale up or scale out.

Scaling up means deploying more powerful server systems. *Scaling out* means deploying many more server systems.

Scale Up



Scale Out



Scale Up or Scale Out

	Scaling Up	Scaling Out
Why Choose…?	• Simpler architecture • Cannot distribute or shard data	• Very high update load • Can distribute or shard data
Advantages	• Simpler architecture • No need to distribute or shard data	• Not limited by underlying platform • Smaller server systems • Better isolation of issues • High update scalability
Disadvantages	• Limited by underlying platform • Powerful (expensive) server systems • Less isolation of issues • Limited write scalability	• Complex architecture • Must distribute/shard data somehow

Plan to scale

Before building a test directory service, start sizing systems by considering service level objectives (SLOs) and directory data.

Define SLOs as described in Performance Requirements. Once you have defined the SLOs, model directory client traffic to test them using your own tools, or the tools described in Performance Tests.

Estimate the disk space needed for each server. This depends on the traffic you modelled to meet SLOs, and on directory data that represents what you expect in production:

1. Import a known fraction of the expected initial data with the server configured for production.

   For help, see Generate Test Data. Make sure you adapt the template for *your* data. Do not rely only on the default template for the `makeldif` command.

2. Check the size of the database.

   Divide by the fraction used in the previous step to estimate the total starting database size.

3. Multiply the result to account for replication metadata.

   To estimate the volume of replication metadata, set up replication with multiple servers as expected in production, and run the estimated production load that corresponds to the data you used. Keep the load running until the replication purge delay. After the purge delay, measure the size of the databases on a directory server, and the size of the changelog database on a replication server. Assuming the load is representative of the production load including expected peaks and normal traffic, additional space used since the LDIF import should reflect expected growth due to replication metadata.

4. Multiply the result to account for the overall growth that you expect for the directory service during the lifetime of the current architecture.

5. To complete the estimate, add 2 GB for default access log files, and space for any backups or LDIF exports you expect to store on local disk.

For a directory server, make sure the system has enough RAM available to cache the database. By default, database files are stored under the `/path/to/opendj/db` directory. Ideally, the RAM available to the server should be at least 1.5 to 2 times the total size of the database files on disk.

Scale up

When scaling up on appropriately powerful server systems, each system must have the resources to run a high-scale DS server. As described in Scaling Replication, a directory server replica is only required to absorb its share of the full read load. But each replica must be able to absorb the full write load for the service.

Make sure that the estimates you arrived at in Plan to scale remain within the capabilities of each server and system.

In addition to the recommendations in Hardware, and the tips in Performance Settings, consider the following points to avoid resource contention:

- For best performance, use dedicated servers.
- Run as few additional system services as possible.
- Run standalone replication servers, directory servers, and directory proxy servers on separate systems.
- In addition to using fast disks with good IOPS, put logs, databases, and backup files on separate disk subsystems.
- Keep resource limitations for client applications to acceptable minimums.
- Schedule backups and maintenance for minimum service impact.

Scale out

When scaling out onto multiple server systems, you must find a usable way to distribute or shard the data into separate replication domains. In some cases, each replication domain holds a branch of the DIT with a similar amount of traffic, and an equivalent amount of data. Entries could then be distributed based on location or network or some other attribute. Branches could join at a base DN that brings all the entries together in the same logical view.

Separate at least the directory server replicas in each replication domain, so that they share only minimal and top-level entries. To achieve this, use subtree replication, which is briefly described in Subtree Replication. Each replica can hold minimal and top-level entries in one database backend, but its primary database backend holds only the branch it shares with others in the domain.

If the data to scale out is all under a single DN, consider using a DS proxy server layer to perform the data distribution, as described in Data Distribution.

When building a scaled-out architecture, be sure to consider the following questions:

- How will you distribute the data to allow the service to scale naturally, for example, by adding a replication domain?
- How will you manage what are essentially multiple directory services?

  All of your operations, from backup and recovery to routine monitoring, must take the branch data into account, always distinguishing between replication domains.
- How will you automate operations?
- How will you simplify access to the service?

  Consider using DS proxy servers for a single point of entry, as described in Single Point of Access.

Data sovereignty

In many countries, how you store and process user accounts and profile information is subject to regulations and restrictions that protect users' privacy. Data sovereignty legislation is beyond the scope of this document. However, DS servers do include features to help you build services in compliance with data sovereignty requirements:

- Data replication
- Subtree replication
- Fractional replication

The deployments patterns described below address questions of data storage. When planning your deployment, also consider how client applications access and process directory data. By correctly configuring access controls, as described in Access control, you can restrict network access by hostname or IP address, but not generally by physical location of a mobile client application, for example.

Consider developing a dedicated service layer to manage policies that define what clients can access and process based on their location. If your deployment calls for more dynamic access management, use DS together with ForgeRock Access Management software.

Replication and data sovereignty

Data replication is critical to a high-scale, highly available directory service. For deployments where data protection is also critical, you must, however, make sure you do not replicate data outside locations where you can guarantee compliance with local regulations.

As described in Deploying Replication, replication messages flow from directory servers through replication servers to other directory servers. Replication messages contain data that has changed, including data governed by privacy regulations:

- For details on replicating data that must not leave a given location, see Subtree replication.
- For details on replicating only part of the data set outside a given location, see Fractional replication.

Subtree replication

As described in Replication Per Base DN, the primary unit of replication is the base DN. Subtree replication refers to putting different subtrees (branches) in separate backends, and then replicating those subtrees only to specified servers. For example, you can ensure that the data replicates only to locations where you can guarantee compliance with the regulations in force.

For subtree replication, the RDN of the subtree base DN identifies the subtree. This leads to a hierarchical directory layout. The directory service retains the logical view of a flatter layout, because the branches all join at a top-level base DN.

The following example shows an LDIF outline for a directory service with top-level and local backends:

- The `userData` backend holds top-level entries, which do not directly reference users in a particular region.
- The `region1` backend holds entries under the `ou=Region 1,dc=example,dc=com` base DN.
- The `region2` backend holds entries under the `ou=Region 2,dc=example,dc=com` base DN.

The example uses nested groups to avoid referencing local accounts at the top level, but still allowing users to belong to top-level groups:

```
# %<--- Start of LDIF for userData --->%
# Base entries are stored in the userData backend:
dn: dc=example,dc=com                        # Base DN of userData backend
...

dn: ou=groups,dc=example,dc=com              # Stored in userData backend
...

dn: ou=Top-level Group,ou=groups,dc=example,dc=com
...
member: ou=R1 Group,ou=groups,ou=Region 1,dc=example,dc=com
member: ou=R2 Group,ou=groups,ou=Region 2,dc=example,dc=com

dn: ou=people,dc=example,dc=com              # Stored in userData backend
...

# %<--- End of LDIF for userData --->%
# %<--- Start of LDIF for Region 1 --->%
# Subtree entries are stored in a country or region-specific backend.
dn: ou=Region 1,dc=example,dc=com            # Base DN of region1 backend
...

dn: ou=groups,ou=Region 1,dc=example,dc=com  # Stored in region1 backend
...

dn: ou=R1 Group,ou=groups,ou=Region 1,dc=example,dc=com
...
member: uid=aqeprfEUXIEuMa7M,ou=people,ou=Region 1,dc=example,dc=com
...

dn: ou=people,ou=Region 1,dc=example,dc=com  # Stored in region1 backend
...

dn: uid=aqeprfEUXIEuMa7M,ou=people,ou=Region 1,dc=example,dc=com
uid: aqeprfEUXIEuMa7M
...

# %<--- End of LDIF for Region 1 --->%
# %<--- Start of LDIF for Region 2 --->%
dn: ou=Region 2,dc=example,dc=com            # Base DN of region2 backend
...

dn: ou=groups,ou=Region 2,dc=example,dc=com  # Stored in region2 backend
...

dn: ou=groups,ou=R2 Group,ou=Region 2,dc=example,dc=com
...
member: uid=8EvlfE0rRa3rgbX0,ou=people,ou=Region 2,dc=example,dc=com
...

dn: ou=people,ou=Region 2,dc=example,dc=com  # Stored in region2 backend
...

dn: uid=8EvlfE0rRa3rgbX0,ou=people,ou=Region 2,dc=example,dc=com
uid: 8EvlfE0rRa3rgbX0
...

# %<--- End of LDIF for Region 2 --->%
```

The deployment for this example has the following characteristics:

- The LDIF is split at the comments about where to cut the file:

  `# %<--- Start|End of LDIF for ... --->%`

- All locations share the LDIF for `dc=example,dc=com`, but the data is not replicated.

  If DS replicates `dc=example,dc=com`, it replicates all data for that base DN, which includes *all* the data from *all* regions.

  Instead, minimize the shared entries, and manually synchronize changes across all locations.

- The local LDIF files are constituted and managed only in their regions:
  - Region 1 data is only replicated to servers in region 1.
  - Region 2 data is only replicated to servers in region 2.
- The directory service only processes information for users in their locations according to local regulations.

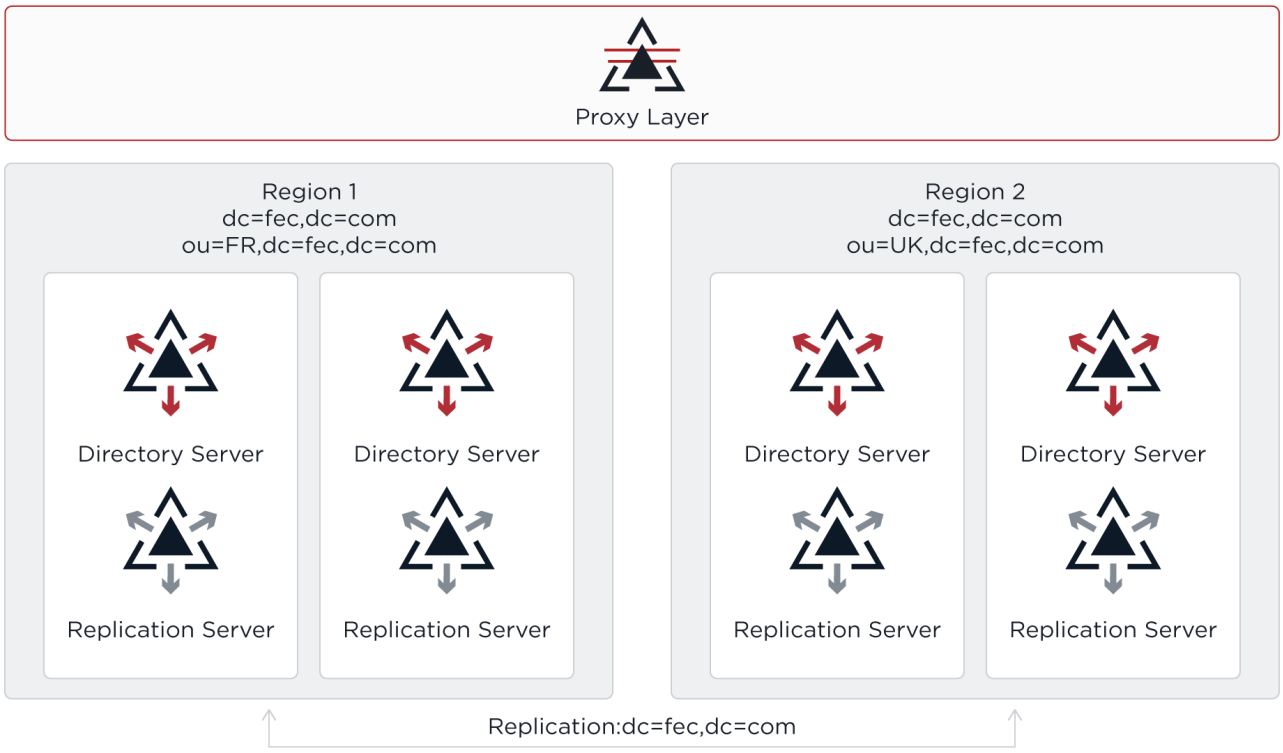


Figure 1. Separate Replication Domains for Data Sovereignty

In a variation on the deployment shown above, consider a deployment with the following constraints:

- Region 1 regulations allow region 1 user data to be replicated to region 2.

  You choose to replicate the region 1 base DN in both regions for availability.

- Region 2 regulations do not allow region 2 user data to be replicated to region 1.

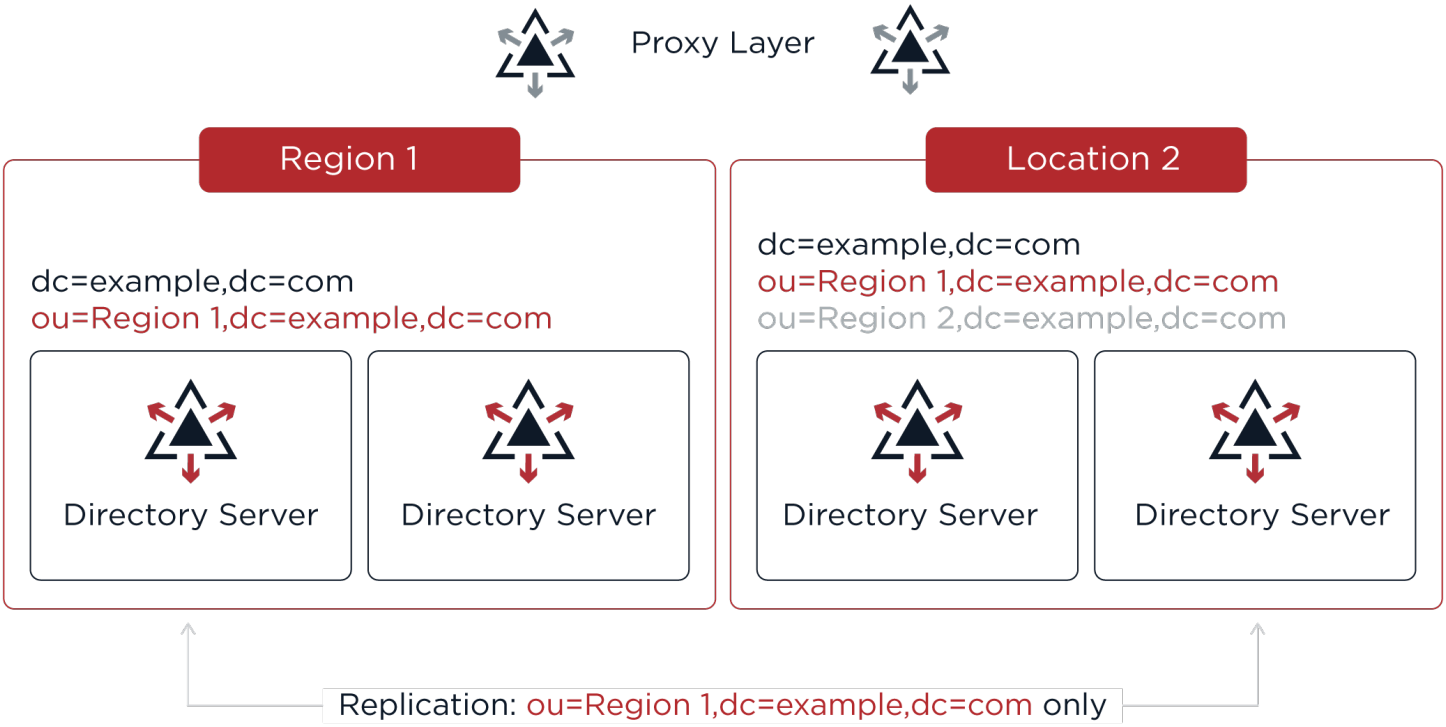


Figure 2. Mixed Replication Domains for Data Sovereignty

When you use subtree replication in this way, client applications can continue to read and update directory data as they normally would. Directory servers only return data that is locally available.

For additional information, see Subtree Replication, and Split Data.

Fractional replication

In some deployments, regulations let you replicate some user attributes. For example, consider a deployment where data sovereignty regulations in one region let you replicate UIDs and class of service levels everywhere, but do not let personally identifiable information leave the user's location.

Consider the following entry where you replicate only the `uid` and `classOfService` attributes outside the user's region:

```
dn: uid=aqeprfEUXIEuMa7M,ou=people,ou=Region 1,dc=example,dc=com
objectClass: top
objectClass: cos
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
cn: Babs Jensen
cn: Barbara Jensen
facsimiletelephonenumber: +1 408 555 1992
gidNumber: 1000
givenname: Barbara
homeDirectory: /home/bjensen
l: Region 1
mail: bjensen@example.com
manager: uid=2jD5NanzOZGjMmcz,ou=people,ou=Region 1,dc=example,dc=com
ou: People
ou: Product Development
preferredLanguage: en, ko;q=0.8
roomnumber: 0209
sn: Jensen
telephonenumber: +1 408 555 1862
uidNumber: 1076
userpassword: {PBKDF2-HMAC-SHA256}10000:<hash>
# Outside the user's region, you replicate only these attributes:
uid: aqeprfEUXIEuMa7M
classOfService: bronze
```

To let you replicate only a portion of each entry, DS servers implement fractional replication. You configure fractional replication by updating the directory server configuration to specify which attributes to include or exclude in change messages from replication servers to the directory server replica.

The replication server must remain located with the directory server replicas that hold full entries which include all attributes. The replication server can receive updates from these replicas, and from replicas that hold fractional entries. Each replication server must therefore remain within the location where the full entries are processed. Otherwise, replication messages describing changes to protected attributes would be sent outside the location where the full entries are processed.

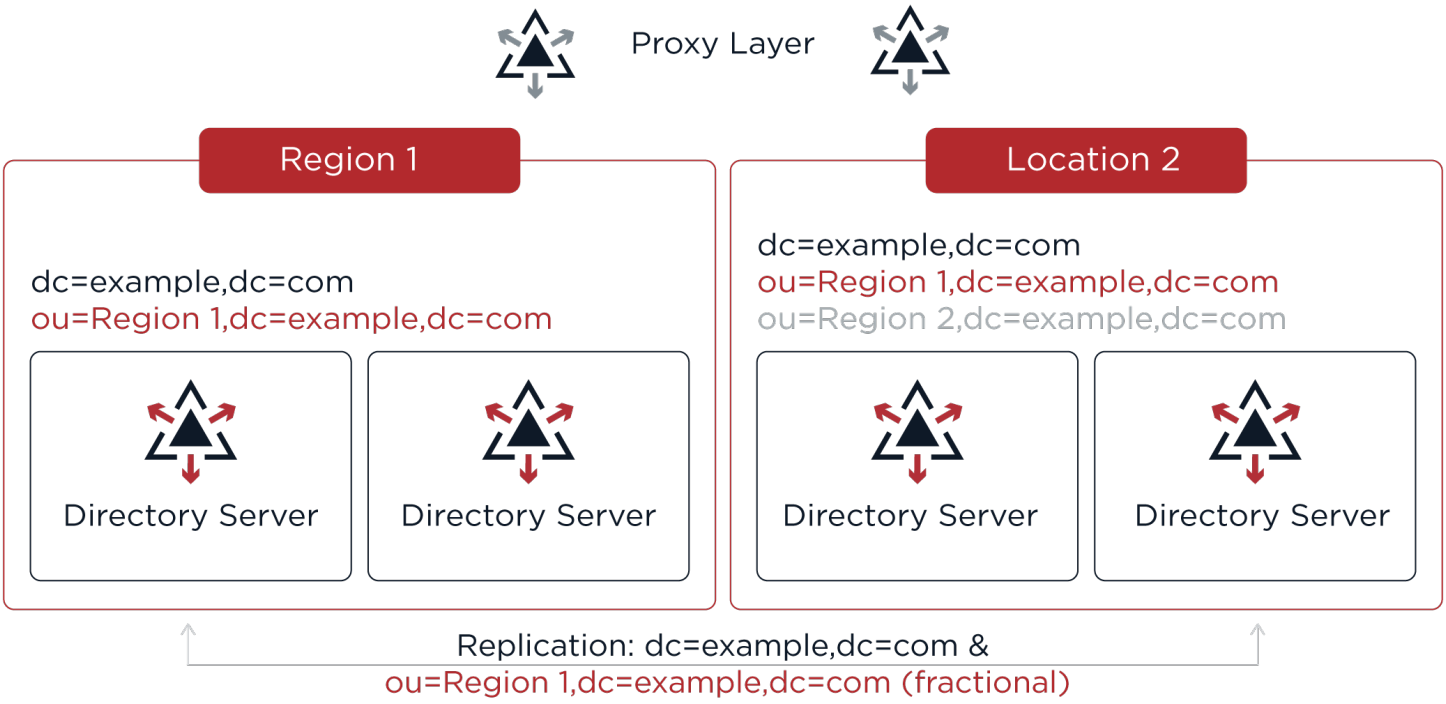


Figure 3. Fractional Replication for Protected Data

To leave schema checking enabled on the replicas that receive fractional updates, portions of entries that are replicated must themselves be complete entries. In other words, in the example above, the entry's structural object class would have to allow `classOfService` and `uid`. This would require editing the schema, and the `objectClass` values of the entries. For details, see LDAP schema.

For additional information, see Fractional Replication.

Interoperability

Common use cases involve interoperability with other directory software.

Use Case	See...
More than one directory service	Proxy layer
Credentials in another directory service	Pass-through authentication
Must sync changes across directory services	Data synchronization and migration
Web clients need alternate data views	Alternative views

Proxy layer

Adding a directory proxy layer can help you deploy alongside an existing directory service. The proxy layer lets you provide a single entry point to both new and old directory services.

You configure a directory proxy server to connect to servers in each directory. DS proxy servers can discover DS directory servers by connecting to DS replication servers. For other directories, you must statically enumerate the directory server to contact. DS proxy servers work with any LDAP directory server that supports the standard proxied authorization control defined in RFC 4370.

Each DS proxy server forwards client requests to the directory service based on the target DN of the operation. As long as the base DNs for each directory service differ, the proxy layer can provide a single entry point to multiple directory services.

For details, see Single Point of Access.

Pass-through authentication

For cases where an existing directory service holds authentication credentials, DS servers provide a feature called pass-through authentication.

With pass-through authentication, the DS server effectively redirects LDAP bind operations to a remote LDAP directory service. If the DS and remote user accounts do not have the same DN, you configure the DS server to automatically map local entries to the remote entries. Pass-through authentication can cache passwords if necessary for higher performance with frequent authentication.

For details, see Pass-Through Authentication.

Data synchronization and migration

You may need to continually synchronize changes across multiple services, or to migrate data from an existing directory service.

For ongoing data synchronization across multiple services, consider ForgeRock Identity Management software or a similar solution. ForgeRock Identity Management software supports configurable data reconciliation and synchronization at high scale, and with multiple data sources, including directory services.

For one-time upgrade and data migration to DS software, the appropriate upgrade and migration depends on your deployment:

- **Offline Migration**

  When downtime is acceptable, you can synchronize data, then migrate applications to the DS service and retire the old service.

  Depending on the volume of data, you might export LDIF from the old service and import LDIF into the DS service during the downtime period. In this case, stop the old service at the beginning of the downtime period to avoid losing changes.

  If the old service has too much data to fit the export/import operation into the downtime period, you can perform an export/import operation before the downtime starts, but you must then implement ongoing data synchronization from the old service to the DS service. Assuming you can keep the new DS service updated with the latest changes, the DS service will be ready to use. You can stop the old service after migrating the last client application.

- **Online Migration**

  When downtime is not acceptable, both services continue running concurrently. You must be able to synchronize data, possibly in both directions. ForgeRock Identity Management software supports bi-directional data synchronization.

  Once you have bi-directional synchronization operating correctly, migrate applications from the old service to the DS service. You can stop the old service after migrating the last client application.

Alternative views

Not all directory clients expect the same directory data. Clients might even expect completely different identity objects.

DS servers expose the same LDAP data view to all directory clients. (You can adjust this behavior somewhat for update operations, as described in Change Incoming Updates.)

The RESTful views of directory data for HTTP clients are fully configurable, however. By developing alternative REST to LDAP mappings and exposing multiple APIs, or different versions of the same API, you can present directory data in different ways to different applications. For details, see Configure HTTP User APIs, and REST to LDAP reference.

Provisioning systems

Before running Directory Services software in production, review the Requirements page of the *Release Notes*, and the following information.

Sizing systems

Given availability requirements and estimates on sizing for services, estimate the required capacity for individual systems, networks, and storage. Sizing described here only accounts for DS servers. Monitoring and audit tools, backup storage, and client applications require additional resources.

CPU, memory, network, and storage requirements depend in large part on the services you plan to provide. The indications in Hardware are only starting points for your sizing investigation.

For details about how each component uses system resources, see DS software.

CPU

Directory servers consume significant CPU resources when processing username-password authentications where the password storage scheme is computationally intensive (Bcrypt, PBKDF2, PKCS5S2).

> **Warning**
>
> Using a computationally intensive password storage scheme such as Bcrypt will have a severe impact on performance. Before you deploy a computationally intensive password storage scheme in production, you *must* complete sufficient performance testing and size your deployment appropriately. Provision enough CPU resources to keep pace with the peak rate of simple binds. If you do not complete this testing and sizing prior to deploying in production, you run the risk of production outages due to insufficient resources.

DS servers also use CPU resources to decode requests and encode responses, and to set up secure connections. LDAP is a connection-oriented protocol, so the cost of setting up a connection may be small compared to the lifetime of the connection.

HTTP, however, requires a new connection for each operation. If you have a significant volume of HTTPS traffic, provision enough CPU resources to set up secure connections.

Memory

DS uses system memory to cache:

- Directory database nodes.

Caching *all* directory data requires 1.5–2 times as much available RAM as the total size of the database files on disk.

For most deployments, caching all data is a costly and poor tradeoff. Instead, cache all internal database nodes. Let the file system cache hold database leaf nodes.

Small directory data sets can fit in the JVM heap, which can improve performance in some cases.

- ACIs.

  This makes a difference in deployments where applications routinely create ACIs programmatically.

- Static groups.

  This makes a difference in deployments with many or large static groups.

- LDAP subentries, such as replicated password policies or collective attribute definitions.

DS also uses system memory for:

- Argon2 password storage schemes.
- Maintaining active connections and processes.

Learn more in Memory.

Network connections

When sizing network connections, account for all requests and responses, including replication traffic. When calculating request and response traffic, base your estimates on your key client applications. When calculating replication traffic, be aware that all write operations must be communicated over the network, and replayed on each directory server. Each write operation results in at least *N-1* replication messages, where *N* is the total number of servers. Be aware that all DS servers running a replication service are fully connected, including those servers that are separated by WAN links.

For deployments in multiple regions, account especially for traffic over WAN links, as this is much more likely to be an issue than traffic over LAN links.

Make sure to size enough bandwidth for peak throughput, and do not forget redundancy for availability.

Disk I/O and storage

The largest disk I/O loads for DS servers arise from logging and writing directory data. You can also expect high disk I/O when performing a backup operation or exporting data to LDIF.

I/O rates depend on the service levels that the deployment provides. When you size disk I/O and disk space, you must account for peak rates and leave a safety margin when you must briefly enable debug logging to troubleshoot any issues that arise.

Also, keep in mind the possible sudden I/O increases that can arise in a highly available service when one server fails and other servers must take over for the failed server temporarily.

DS server access log files grow more quickly than other logs. Default settings prevent each access logger's files from growing larger than 2 GB before removing the oldest. If you configure multiple access loggers at once, multiply 2 GB by their number.

Directory server database backend size grows as client applications modify directory data. Even if data set's size remains constant, the size of the backend grows. Historical data on modified directory entries increases until purged by the directory server when it reaches the replication purge delay (default: 3 days). In order to get an accurate disk space estimate, follow the process described in Plan to Scale.

Replication server changelog backend size is subject to the same growth pattern as historical data. Run the service under load until it reaches the replication purge delay to estimate disk use.

For highest performance, use fast SSD disk and separate disk subsystems logging, backup, and database backends.

Portability

DS client and server code is pure Java, and depends only on the JVM. This means you can run clients and servers on different operating systems, and copy backup files and archives from one system to another.

Server portability

DS servers and data formats are portable across operating systems. When using multiple operating systems, nevertheless take the following features into account:

Command-Line Tool Locations

DS server and command-line tools are implemented as scripts. The path to the scripts differ on UNIX/Linux and Windows systems. Find UNIX/Linux scripts in the `bin` directory. Find Windows scripts in the `bat` folder.

Native Packaging

When you download DS software, you choose between cross-platform and native packages.

- Cross-platform .zip packaging facilitates independence from the operating system. You manage the server software in the same way, regardless of the operating system.
- Native packaging facilitates integration with the operating system. You use the operating system tools to manage the software.

Both packaging formats provide scripts to help register the server as a service of the operating system. These scripts are `create-rc-script` (UNIX/Linux) and `windows-service` (Windows).

Gateway portability

The only persistent state for gateway applications is in their configuration files. The gateway configuration files are portable across web application containers and operating systems.

Deployment checklists

Use these checklists when deploying your directory service:

Initiate the project

Task	Done?
Understand the business requirements for your DS deployment	☐
Identify key client applications	☐

Task	Done?
Identify project stakeholders	☐
Define SLOs based on business requirements	☐
Define project scope	☐
Define project roles and responsibilities	☐
Schedule DS training for deployment team members	☐

Prepare supportability

Task	Done?
Find out how to get help and support from ForgeRock and partners	☐
Find out how to get training from ForgeRock and partners	☐
Find out how to keep up to date with new development and new releases	☐
Find out how to report problems	☐

Design the service

Task	Done?
Understand the roles of directory components	☐
Define architecture, mapping requirements to component features	☐
Define the directory data model	☐
Define the directory access model	☐
Define the replication model	☐
Define how to backup, restore, and recover data	☐
Define how you will monitor and audit the service	☐
Determine how to harden and secure the service	☐

Develop the service

Task	Done?
Engage development of custom server plugins as necessary	☐
Apply configuration management	☐
Create a test plan	☐
Engage automation, continuous integration	☐
Create a documentation plan	☐
Create a maintenance and support plan	☐
Pilot the implementation	☐
Size systems to provision for production	☐
Execute test plans	☐
Execute documentation plans	☐
Create a rollout plan in alignment with all stakeholders	☐
Prepare patch and upgrade plans	☐

Implement the service

Task	Done?
Ensure appropriate support for production services	☐
Execute the rollout plan	☐
Engage ongoing monitoring and auditing services	☐
Engage ongoing maintenance and support	☐

Maintain the service

Task	Done?
Execute patch and upgrade plans as necessary	☐

Task	Done?
Plan how to adapt the deployment to new and changing requirements	☐

Installation

This guide shows you how to install and remove Directory Services software.

Evaluate DS

Try DS software.

DS for CTS

Store AM CTS tokens.

DS for Identities

Store AM identities.

DS as IDM Repo

Store IDM data.

Setup Hints

Review setup options.

DS Proxy

Install an LDAP proxy.

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.
Java APIs	Java server-side APIs for server plugins that extend directory services. All Java APIs have Interface Stability: *Evolving*. *Be prepared for incompatible changes in both major and minor releases.*

Read the Release notes before installing DS software.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

The common REST API provides ForgeRock Identity Platform software common ways to access web resources and collections of resources.

Unpack files

The following procedures only unpack the server files. You must then run the `setup` command to set up the server:

Unpack the cross-platform zip

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

1. Review requirements for installation.
2. Unpack the cross-platform .zip file in the file system directory where you want to install the server.

   Perform this step as a user with the same file system permissions as the user who will run the `setup` command.

   The `setup` command uses the directory where you unzipped the files as the installation directory. It 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-7.2.5` directory from the folder you specify.

Use the Debian package

On Debian and related Linux distributions, such as Ubuntu, you can unpack files using the Debian package:

1. Review requirements for installation.

   In particular, install a Java runtime environment (JRE) if none is installed yet. The following example uses the `java11-runtime` virtual package:

   ```
   $ sudo apt-get install java11-runtime
   ```

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 the `/usr/share/doc/opendj` directory.
   - Adds man pages under the `/opt/opendj/share/man` directory.

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

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

Use the RPM package

On Red Hat and related Linux distributions, such as Fedora and CentOS, you can unpack files using the RPM package:

1. Review requirements for installation.

   In particular, install a Java runtime environment (JRE) 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.
- Adds man pages under the `/opt/opendj/share/man` directory.

The files are owned by root by default, making it easier to have the server listen on privileged 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.

Use the Windows MSI

Important

Prevent antivirus and intrusion detection systems from interfering with DS software.
Before using DS 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 DS file access, particularly write access.
Antivirus and intrusion detection software have incorrectly marked DS files as suspect to infection, because they misinterpret normal DS processing.
Prevent antivirus and intrusion detection systems from scanning DS files, except these folders:

`/path/to/opendj/bat/`
: Windows command-line tools

`/path/to/opendj/bin/`
: UNIX/Linux command-line tools

`/path/to/opendj/extlib/`
: Optional additional `.jar` files used by custom plugins

`/path/to/opendj/lib/`
: Scripts and libraries shipped with DS servers

Port blocking

Antivirus and intrusion detection software can block ports that DS uses to provide directory services.
Make sure that your software does not block the ports that DS software uses. For details, see Administrative access.

Negative performance impact

Antivirus software consumes system resources, reducing resources available to other services including DS servers.
Running antivirus software can therefore have a significant negative impact on DS server performance. Make sure that you test and account for the performance impact of running antivirus software before deploying DS software on the same systems.

1. Review requirements for installation.
2. Install the server files in one of the following ways:
    1. Run the GUI installer:
        - Double-click the Windows installer package, `DS-7.2.5.msi`, to start the install wizard.
        - In the **Destination Folder** screen, set the folder location for 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\`.

          The native executable is a 32-bit application, though you can run the server in a 64-bit Java environment.
        - Start the installation.

          When installation is finished, the server files are found in the location you specified.

2. Use the Microsoft `msiexec.exe` command to install the files.

   The following example installs the server files under `C:\opendj-7.2.5`, writing an installation log file, `install.log`, in the current folder:

   ```
   C:\> msiexec /i DS-7.2.5.msi /l* install.log /q OPENDJ=C:\opendj-7.2.5
   ```

Setup hints

The following table provides extensive hints for using `setup` command options in the order they are presented in interactive mode, when you run the command without options.

For reference information, see setup:

Parameter	Description	Option(s)
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.	`--instancePath`
Unique server ID	A server identifier string that is unique for your deployment. Choose a relatively short string, as the value is recorded repeatedly in replicated historical data.	`--serverId`
Deployment ID	The *deployment ID* is a random string generated using the `dskeymgr` command. It is paired with a *deployment ID password*, which is a random string that you choose, and that you must keep secret. Together, the deployment ID and password serve to generate the shared master key that DS servers in the deployment require for protecting shared encryption secrets. By default, they also serve to generate a private CA and keys for TLS to protect communication between DS servers. When you deploy multiple servers together, reuse the same deployment ID and password for each server installation. For details, see Deployment IDs.	`--deploymentId`

Parameter	Description	Option(s)
Deployment ID password	This is a random string that you choose, and that you must keep secret. It is paired with the deployment ID.	`--deploymentIdPassword[:env|:file]`
Root user DN	The root user DN identifies the initial *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 name used in the documentation is the default name: `uid=admin`. For additional security in production environments, use a different name.	`-D, --rootUserDn`
Root user password	The root user authenticates with simple, password-based authentication. Use a strong password here unless this server is only for evaluation.	`-j, --rootUserPassword[:env|:file]`
Monitor user DN	The monitor user DN identifies a user with the privilege to read monitoring data (`monitor-read`). The account is replicated by default, so use the same DN on each server. The name used in the documentation is the default name: `uid=monitor`.	`--monitorUserDn`
Monitor user password	The monitor user authenticates with simple, password-based authentication. The account is replicated by default, so use the same password on each server.	`--monitorUserPassword[:env|:file]`
Fully qualified directory server domain name	The server uses the fully qualified domain name (FQDN) 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 `localhost`. Otherwise, use an FQDN that other hosts can resolve to reach your server, and that matches the FQDN in the server certificate.	`-h, --hostname`
Administration port	This is the service port used to configure the server and to run tasks. The port used in the documentation is `4444`. 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. *Configure the firewall to allow access to this port from all connecting DS servers.*	`--adminConnectorPort`

Parameter	Description	Option(s)

Securing the deployment	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 the private PKI derived from the deployment ID and passwords. For details, see Deployment IDs.
	You can also 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.
	Public key security is often misunderstood. Before making security choices for production systems, read Cryptographic keys.
		`--useJavaKeyStore`
		`--useJceKeyStore`
		`--usePkcs11KeyStore`
		`--usePkcs12KeyStore`
		`-W, --keyStorePassword[:env|:file]`
		`--keyStorePasswordFilePath`
		`-N, --certNickname`
		`--useJavaTrustStore`
		`--useJceTrustStore`
		`--usePkcs11TrustStore`
		`--usePkcs12TrustStore`
		`-T, --trustStorePassword[:env|:file]`
		`--trustStorePasswordFilePath`

Start the server	By default, the `setup` command does not start the server. Finish configuring the server, then use the `/path/to/opendj/bin/start-ds` command. If no further configuration is required, use the `setup --start` option.	`-s, --start`

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 negotiates a secure connection starting on the insecure LDAP port.
		`-p, --ldapPort`
		`-q, --enableStartTls`
		`-Z, --ldapsPort`

Parameter	Description	Option(s)
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 HTTPS on port `8443`. When you enable HTTP or HTTPS at setup time, only the administrative endpoints are enabled, `/admin/config`, `/metrics/api`, and `/metrics/prometheus`, allowing applications to configure and monitor the server. For access to user data in a directory server, see Configure HTTP User APIs.	`--httpPort` `--httpsPort`
Replication port	Port used for data replication messages. This port must be accessible externally from other DS servers. If this port is configured, the server acts as a replication server. It maintains a replication change log, which it exposes as an external change log by default. 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 `8989`.	`-r, --replicationPort`

Parameter	Description	Option(s)
Bootstrap replication servers	Specify bootstrap server `host:port` pairs, where port is the server's replication port. The current server contacts the bootstrap servers to discover other servers in the deployment. The `host:port` pair may represent the current server if it is a bootstrap server. Specify the same list of bootstrap servers each time you set up a replica or standalone replication server. This option interacts with the `-r, --replicationPort` option as follows: • If both options are set, the server acts as a replication server. It connects to the specified bootstrap replication server(s) to discover other servers. • If only the `-r, --replicationPort` option is set, the server acts as a replication server. It counts only itself as the bootstrap replication server. In production, specify the same list of at least two bootstrap servers every time, including when you set up the bootstrap servers. • If only the `--bootstrapReplicationServer` option is set, the server acts as a standalone directory server. It connects to the specified bootstrap replication server(s). • If neither option is set, the server is not configured for replication at setup time.	`--bootstrapReplicationServer`
Configure the server for use with other applications	For details, see Setup profiles.	`--profile` `--set`

Setup profiles

A *setup profile* lets you configure a server for a specific use case. Profiles greatly simplify the directory server setup process for such use cases, such as preparing a directory server to serve another ForgeRock® Identity Platform component product.

You can configure a setup profile using the `setup` command, or the `setup-profile` command after initial setup. The `setup-profile` command runs on a server that is offline.

Select a profile with the `--profile` option. Each profile has its own parameters, some of which have default values. You specify profile parameters with `--set` options.

The profile selection option takes the form `--profile profileName[:version]`. If you do not specify the optional `:version` portion of the argument, the `setup` command uses the current DS software version, falling back to the previous version if the current version does not match an available profile. Repeat the `--profile` option to apply multiple setup profiles.

An option to set a parameter takes the form `--set[:env|:file] parameterName:value` where:

- `profileName/` indicates which profile the parameter applies to.

  This name is required when you specify multiple profiles, and the parameter is available in more than one of the specified profiles.

  The `profileName` is case-insensitive.

- `parameterName` specifies the parameter to set.
- `value` specifies the value the parameter takes when the `setup` command applies the profile.

Use the `setup --help-profiles` or `setup-profile --help` command to list available profiles.

Use the `--help-profile profileName[:version]` option to list the parameters for the specified profile.

Check profiles

The `opendj/profiles.version` file lists the profiles selected at setup time:

```
$ cat /path/to/opendj/config/profiles.version
ds-evaluation:7.2.5
```

Default Setup Profiles

This page lists default profiles with their parameters.

AM Configuration Data Store 6.5.0

The `am-config:6.5.0` profile has the following parameters:

`backendName`

Name of the backend for storing config
Default: `--set am-config/backendName:cfgStore`
Syntax: Name

`baseDn`

The base DN to use to store AM's configuration in
Default: `--set am-config/baseDn:ou=am-config`
Syntax: DN

`amConfigAdminPassword`

Password of the administrative account that AM uses to bind to OpenDJ
Syntax: Password

AM CTS Data Store 6.5.0

The `am-cts:6.5.0` profile has the following parameters:

backendName

Name of the backend for storing tokens
Default: `--set am-cts/backendName:amCts`
Syntax: Name

baseDn

The base DN to use to store AM's tokens in
Default: `--set am-cts/baseDn:ou=tokens`
Syntax: DN

amCtsAdminPassword

Password of the administrative account that AM uses to bind to OpenDJ
Syntax: Password

tokenExpirationPolicy

Token expiration and deletion
Default: `--set am-cts/tokenExpirationPolicy:am`

This parameter takes one of the following values:

- `am`: AM CTS reaper manages token expiration and deletion
- `am-sessions-only`: AM CTS reaper manages SESSION token expiration and deletion. DS manages expiration and deletion for all other token types. AM continues to send notifications about session expiration and timeouts to agents
- `ds`: DS manages token expiration and deletion. AM session-related functionality is impacted and notifications are not sent

AM Identity Data Store 7.2.0

The `am-identity-store:7.2.0` profile has the following parameters:

backendName

Name of the backend for storing identities
Default: `--set am-identity-store/backendName:amIdentityStore`
Syntax: Name

baseDn

The base DN to use to store identities in
Default: `--set am-identity-store/baseDn:ou=identities`
Syntax: DN

amIdentityStoreAdminPassword

Password of the administrative account that AM uses to bind to OpenDJ
Syntax: Password

AM Identity Data Store 7.1.0

The `am-identity-store:7.1.0` profile has the following parameters:

`backendName`

Name of the backend for storing identities
Default: `--set am-identity-store/backendName:amIdentityStore`
Syntax: Name

`baseDn`

The base DN to use to store identities in
Default: `--set am-identity-store/baseDn:ou=identities`
Syntax: DN

`amIdentityStoreAdminPassword`

Password of the administrative account that AM uses to bind to OpenDJ
Syntax: Password

AM Identity Data Store 7.0.0

The `am-identity-store:7.0.0` profile has the following parameters:

`backendName`

Name of the backend for storing identities
Default: `--set am-identity-store/backendName:amIdentityStore`
Syntax: Name

`baseDn`

The base DN to use to store identities in
Default: `--set am-identity-store/baseDn:ou=identities`
Syntax: DN

`amIdentityStoreAdminPassword`

Password of the administrative account that AM uses to bind to OpenDJ
Syntax: Password

AM Identity Data Store 6.5.0

The `am-identity-store:6.5.0` profile has the following parameters:

`backendName`

Name of the backend for storing identities
Default: `--set am-identity-store/backendName:amIdentityStore`
Syntax: Name

`baseDn`

The base DN to use to store identities in
Default: `--set am-identity-store/baseDn:ou=identities`
Syntax: DN

`amIdentityStoreAdminPassword`

Password of the administrative account that AM uses to bind to OpenDJ
Syntax: Password

DS Evaluation 7.2.0

The `ds-evaluation:7.2.0` profile has the following parameters:

`generatedUsers`

Specifies the number of generated user entries to import. The evaluation profile always imports entries used in documentation examples, such as uid=bjensen. Optional generated users have RDNs of the form uid=user.%d, yielding uid=user.0, uid=user.1, uid=user.2 and so on. All generated users have the same password, "password". Generated user entries are a good fit for performance testing with tools like addrate and searchrate
Default: `--set ds-evaluation/generatedUsers:100000`
Syntax: Number

DS Proxied Server 7.0.0

The `ds-proxied-server:7.0.0` profile has the following parameters:

`proxyUserDn`

The proxy user service account DN. This will be be used for authorization and auditing proxy requests.
Default: `--set ds-proxied-server/proxyUserDn:uid=proxy`
Syntax: DN

`proxyUserCertificateSubjectDn`

The subject DN of the proxy user's certificate. The proxy must connect using mutual TLS with a TLS client certificate whose subject DN will be mapped to the proxy service account.
Default: `--set ds-proxied-server/proxyUserCertificateSubjectDn:CN=DS,O=ForgeRock.com`
Syntax: DN

`baseDn`

Base DN for user information in the server. Multiple base DNs may be provided by using this option multiple times. If no base DNs are defined then the server will allow proxying as any user, including administrator accounts.
Syntax: DN

DS Proxy Server 7.0.0

The `ds-proxy-server:7.0.0` profile has the following parameters:

backendName

Name of the proxy backend for storing proxy configuration
Default: `--set ds-proxy-server/backendName:proxyRoot`
Syntax: Name

bootstrapReplicationServer

Bootstrap replication server(s) to contact periodically in order to discover remote servers
Syntax: host:port or configuration expression

rsConnectionSecurity

Connection security type to use to secure communication with remote servers
Default: `--set ds-proxy-server/rsConnectionSecurity:ssl`

This parameter takes one of the following values:

- `ssl` : Use SSL
- `start-tls` : Use Start TLS

keyManagerProvider

Name of the key manager provider used for authenticating the proxy in mutual-TLS communications with backend server(s)
Default: `--set ds-proxy-server/keyManagerProvider:PKCS12`
Syntax: Name or configuration expression

trustManagerProvider

Name of the trust manager provider used for trusting backend server(s) certificate(s)
Syntax: Name or configuration expression

certNickname

Nickname(s) of the certificate(s) that should be sent to the server for SSL client authentication.
Default: `--set ds-proxy-server/certNickname:ssl-key-pair`
Syntax: Name or configuration expression

primaryGroupId

Replication domain group ID 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 (i.e all remote servers are primary)
Syntax: String or configuration expression

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
Syntax: DN or configuration expression

DS User Data Store 7.0.0

The `ds-user-data:7.0.0` profile has the following parameters:

`backendName`

Name of the backend to be created by this profile
Default: `--set ds-user-data/backendName:userData`
Syntax: Name

`baseDn`

Base DN for your users data.
Syntax: DN

`ldifFile`

Path to an LDIF file containing data to import. Use this option multiple times to specify multiple LDIF files
Syntax: File or directory path

`addBaseEntry`

Create entries for specified base DNs when the 'ldifFile' parameter is not used. When this option is set to 'false' and the 'ldifFile' parameter is not used, create an empty backend.
Default: `--set ds-user-data/addBaseEntry:true`

This parameter takes one of the following values:

- `false`
- `true`

IDM External Repository 7.2.0

The `idm-repo:7.2.0` profile has the following parameters:

`backendName`

IDM repository backend database name
Default: `--set idm-repo/backendName:idmRepo`
Syntax: Name

`domain`

Domain name translated to the base DN for IDM external repository data. Each domain component becomes a "dc" (domain component) of the base DN. This profile prefixes "dc=openidm" to the result. For example, the domain "example.com" translates to the base DN "dc=openidm,dc=example,dc=com".
Default: `--set idm-repo/domain:example.com`
Syntax: Domain name

IDM External Repository 7.1.0

The `idm-repo:7.1.0` profile has the following parameters:

backendName

IDM repository backend database name
Default: `--set idm-repo/backendName:idmRepo`
Syntax: Name

domain

Domain name translated to the base DN for IDM external repository data. Each domain component becomes a "dc" (domain component) of the base DN. This profile prefixes "dc=openidm" to the result. For example, the domain "example.com" translates to the base DN "dc=openidm,dc=example,dc=com".
Default: `--set idm-repo/domain:example.com`
Syntax: Domain name

IDM External Repository 7.0.0

The `idm-repo:7.0.0` profile has the following parameters:

backendName

IDM repository backend database name
Default: `--set idm-repo/backendName:idmRepo`
Syntax: Name

domain

Domain name translated to the base DN for IDM external repository data. Each domain component becomes a "dc" (domain component) of the base DN. This profile prefixes "dc=openidm" to the result. For example, the domain "example.com" translates to the base DN "dc=openidm,dc=example,dc=com".
Default: `--set idm-repo/domain:example.com`
Syntax: Domain name

IDM External Repository 6.5.0

The `idm-repo:6.5.0` profile has the following parameters:

backendName

IDM repository backend database name
Default: `--set idm-repo/backendName:idmRepo`
Syntax: Name

domain

Domain name translated to the base DN for IDM external repository data. Each domain component becomes a "dc" (domain component) of the base DN. This profile prefixes "dc=openidm" to the result. For example, the domain "example.com" translates to the base DN "dc=openidm,dc=example,dc=com".
Default: `--set idm-repo/domain:example.com`
Syntax: Domain name

Create your own

If you have changes that apply to each server you set up, you can create and maintain your own setup profile.

> **Important**
>
> The custom setup profile interface has stability: *Evolving*.
> Be prepared to adapt your custom profiles to changes in each new release.

- Add custom setup profiles under the `opendj/template/setup-profiles/` directory.

  The `setup` and `setup-profile` commands look for profiles in that location. The default profiles provide examples that you can follow when building custom profiles.

- Add custom setup profiles after unpacking the DS files, but before running the `setup` or `setup-profile` command.

- Each setup profile strictly follows the supported file layout.

  The base path, version directories, and the `.groovy` files are required. The other files are shown here as examples:

```
opendj/template/setup-profiles/base-path/
├── version
│   ├── base-entries.ldif
│   ├── parameters.groovy
│   ├── profile.groovy
│   └── schema
│       └── schema-file-name.ldif
└── name.txt
```

File or Directory	Description
`base-path` (required)	The base path distinguishes the profile from all other profiles, and defines the profile name. Path separator characters are replaced with dashes in the name. For example, the base path `DS/evaluation` yields the profile name `ds-evaluation`.
`version` (required)	The profile version, including either two or three numbers. Numbers can be separated by dots (`.`) or dashes (`-`). Set this to the version of the software that the profile is for. For example, if you are writing a profile for Transmogrifier 2.0, use `2.0`. Add multiple versions of your profile when using the same DS version with different versions of your application.
`base-entries.ldif`	Optional LDIF file that templates the entries this profile adds to the directory. This is an example of a file used by `profile.groovy`.
`parameters.groovy` (required)	Groovy script defining profile parameters that users can set. See Parameters.
`profile.groovy` (required)	Groovy script that makes changes to the server. See Profile script.

File or Directory	Description
`schema-file-name.ldif`	Optional LDAP schema file required for entries added by this profile. This is an example of a file used by `profile.groovy`.
`name.txt`	If this file is present, it must hold the human-readable form of the profile name, *not including the version*, in a single-line text file.

At setup time, the user cannot select more than one version of the same setup profile. The user can select multiple setup profiles for the same server. You must ensure that your profile is not incompatible with other available profiles.

Parameters

You let users set parameters through the `parameters.groovy` script. The profile uses the parameters as variables in the `profile.groovy` script, and resource files.

The `parameters.groovy` script lists all parameter definitions for the profile. It includes only parameter definitions. Each parameter definition is resolved at runtime, and so must not be provided programmatically. Parameter definitions start with `define`, and can have the following methods:

```
define.type "name"                              \
  advanced()                                    \
  defaultValueFromSetupTool global-setup-option \
  defaultValue default                          \
  description "short-description"               \
  help "help-message"                           \
  multivalued()                                 \
  multivalued "help message(s)"                 \
  optional()                                    \
  optional "help message(s)"                    \
  descriptionIfNoValueSet "short-description"   \
  property "property-name"                      \
  prompt "prompt message(s)"                    \
  expressionAllowed()
```

Element	Description
`type` (required)	This mandatory parameter type is one of the following. The profile mechanism converts the string input internally into a Java class: • `booleanParameter` • `dnParameter` (a DN) • `domainParameter` The input is a domain that the profile mechanism converts to a DN. The domain `example.com` becomes `dc=example,dc=com`. • `hostPortParameter` (a hostname:port pair) • `doubleParameter` (a Double number) • `floatParameter` (a Float number) • `integerParameter` (an Integer number) • `longParameter` (a Long number) • `passwordParameter` The input is a password, and encoded with a password storage scheme to avoid exposing the plain text. • `pathParameter` • `stringParameter`
`name` (required)	This mandatory parameter name must be a valid Groovy name string.
`advanced()`	This is an advanced parameter, meaning interactive mode does not show the parameter or prompt for input. When using `advanced()`, you must set a default parameter value. Interactive mode sets this parameter to the default value.
`defaultValueFromSetupTool global-setup-option`	This parameter takes its default from the value of the specified the global `setup` option. The `defaultValueFromSetupTool` method only applies when the profile is used by the `setup` command. The *global-setup-option* is the option name without leading dashes.
`defaultValue default`	This parameter's default must match the type.
`description "short-description"`	This provides a brief summary of what the parameter does. The "short-description" is a single paragraph with no trailing punctuation.
`help "help-message"`	The message, used in online help, provides further explanation on how to use the parameter. The "help-message" is a single paragraph with no trailing punctuation.
`multivalued()`	This parameter takes multiple values, and no help message is required. Use this, for example, when the property is `advanced()`.

Element	Description
`multivalued "help message(s)"`	This parameter takes multiple values, and interactive mode prompts the user for each one. Each help message string is a single paragraph, and the final help message has no trailing punctuation. Help message arguments are separated with commas.
`optional()`	This parameter is optional, and no help message is required.
`optional "help message(s)"`	This parameter is optional, and interactive mode prompts the user for input. Each help message string is a single paragraph, and the final help message has no trailing punctuation. Help message arguments are separated with commas.
`descriptionIfNoValueSet "short-description"`	The description is displayed when the parameter is optional, and the user has not set a value. The "short-description" is a single paragraph with no trailing punctuation.
`property "property-name"`	The profile replaces `&{property-name}` in resource files with the value of this property. The `&{property-name}` expressions follow the rules described in Property value substitution.
`prompt "prompt message(s)"`	In interactive mode, display one or more paragraphs when prompting the user for input. Each prompt message string is a single paragraph. If no default value is set the final prompt message takes a trailing colon. Prompt message arguments are separated with commas.

Profile script

When a user requests a profile, the `profile.groovy` script controls what the profile does.

When developing your profile script, you can use these classes and methods, which are bound into the execution context of your script before it executes:

In addition, see the Javadoc for the setup model.

Default imports

The profile mechanism imports the following classes and methods, making them available by default in the context of your profile script:

```
import static org.forgerock.i18n.LocalizableMessage.raw
import static org.forgerock.opendj.setup.model.Profile.ParameterType.of
import static java.nio.file.Files.*

import org.forgerock.i18n.LocalizableMessage
import org.forgerock.opendj.ldap.Dn
import org.forgerock.opendj.setup.model.SetupException

import java.nio.file.Paths
```

Server methods

A `ds` object is bound into the execution context of your profile script.

All its methods throw a `SetupException` on failure. On failure, the setup process removes the server's `db` and `config` directories. This allows the user to start over, applying the same profiles again.

All the `ds` methods run with the server offline:

Method	Description
`void ds.addBackend(String backendName, String entryDn)`	Creates the backend, adds it to the server, and sets it as the working backend. When you use other methods to import LDIF and create indexes, they operate on the working backend. Use `importBaseEntry` to add only the base entry, or `importLdif` to add entries to the backend.
`void ds.addBackend(String backendName, Dn entryDn)`	
`void ds.addBackendWithDefaultUserIndexes(String backendName, String entryDn)`	
`void ds.addBackendWithDefaultUserIndexes(String backendName, Dn entryDn)`	
`void ds.setWorkingBackend(String backendName)`	Set the specified backend as the one to operate on when importing LDIF and creating indexes.
`void ds.importBaseEntry(Dn baseDn)`	Import the entry with the specified base DN as the base entry of the working backend. The import operation erases any previous content of the backend before importing new content.
`void ds.importLdifWithSampleEntries(Dn sampleEntryBaseDn, int nbSampleEntries, String… ldifFilePaths)`	Import the specified number of sample entries with the specified base DN, based on the LDIF file templates provided, to the working backend. The LDIF must contain the base entry. This method replaces `&{property-name}` in the LDIF with the property values before import. The import operation erases any previous content of the backend before importing new content.

Method	Description
`void ds.importLdifTemplate(String… ldifFilePaths)`	Add the entries from the LDIF files provided to the working backend. The LDIF must contain the base entry. This method replaces `&{property-name}` in the LDIF with the property values before import. The import operation erases any previous content of the backend before importing new content.
`void ds.importLdifTemplate(Collection<Path> ldifFilePaths)`	
`void ds.importLdif(String… ldifFilePaths)`	Add the entries from the LDIF files provided to the working backend. The LDIF must contain the base entry. If there are multiple files, each entry must appear only once. The import operation erases any previous content of the backend before importing new content.
`void ds.importLdif(Collection<Path> ldifFilePaths)`	
`void ds.addSchemaFiles()`	Copy LDIF-format schema files from the `schema` directory of the profile to the `db/schema` directory of the server. If no `schema` directory is present for the current version of the profile, this method uses schema from a previous version of the profile.
`void ds.config(List<String> cliArgs)`	Run the `dsconfig` command in offline mode with the specified arguments. Use this method only for additional configuration, not when creating backends or indexes.
`void ds.addIndex(String attributeName, String… types)`	Create indexes of the specified types in the working backend for the specified attribute. For a list of available index types, see index-type.
`void ds.failSetup(String message)`	Cause the profile to fail with a `SetupException` having the specified message.

Resource file methods

A `resource` object is bound into the execution context of your profile script. The `resource` methods let you retrieve arbitrary files from the profile, and replace configuration expressions in resource files:

Method	Description
`Path resource.file(String path)`	Return the path to the specified resource file. If the specified path is relative, the method first returns the path of the file in the current profile version, or the path of the file in the previous profile version if none is present in the current version. If the specified path is absolute, the method only converts the string to a path.

Method	Description
`void resource.applyTemplate(String template, String target)`	Convert the relative template path as `resource.file(template)`,the relative target path as an absolute path, and copy the template file to the target file, replacing `&{property-name}` with the property values. The `&{property-name}` expressions follow the rules described in Property value substitution.

Logging methods

Use the `console` object to write log messages when your profile script runs.

The `console` object implements SetupConsole, and so provides all the methods documented for that interface.

The `setup` and `setup-profile` commands log any exceptions that occur when your profile script runs, so there is no need to catch exceptions just for logging purposes.

Install DS for evaluation

To set up the server, use the setup command-line tool.

When used without options, the command is interactive.

The following `setup` options are mandatory. When performing a non-interactive, silent installation, specify at least all mandatory options as part of the command. 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 specified or generated at setup time:

- `--adminConnectorPort {port}` (conventional port number: `4444`)
- `--hostname {hostname}`
- `--rootUserDN {rootUserDN}` (default: `uid=admin`)
- `--rootUserPassword {rootUserPassword}`

1. Before proceeding, install the server files.
   For details, see Unpack files.
2. Generate a deployment ID unless you already have one:

   ```
   $ /path/to/opendj/bin/dskeymgr create-deployment-id --deploymentIdPassword password

   your-deployment-id
   ```

   Save the deployment ID and its deployment password. Keep the ID and the password safe, and keep the password secret. Use the same deployment ID and password for all the servers in the same environment.

   A *deployment ID* is a random string generated using the `dskeymgr` command. It is a deployment identifier, not a key, but it is used with a password to generate keys.

A *deployment ID password* is a secret string at least 8 characters long that you choose.

The two are a pair. You must have the deployment ID password to use the deployment ID.

Each deployment requires a *single, unique deployment ID and its password*. DS uses the pair to:

- Protect the keys to encrypt and decrypt backup files and directory data.
- Generate the TLS key pairs to protect secure connections, unless you provide your own.

Store your deployment ID and password in a safe place, and reuse them when configuring other servers in the same deployment.

The DS `setup` and `dskeymgr` commands use the pair to generate the following:

- (Required) A shared master key for the deployment.

  DS replicas share secret keys for data encryption and decryption. DS servers encrypt backend data, backup files, and passwords, and each replica must be able to decrypt data encrypted on another peer replica.

  To avoid exposing secret keys, DS servers encrypt secret keys with a shared master key. DS software uses a deployment ID and its password to derive the master key.

- (Optional) A private PKI for trusted, secure connections.

  A PKI serves to secure network connections from clients and other DS servers. The PKI is a trust network, requiring trust in the CA that signs public key certificates.

  Building a PKI can be complex. You can use self-signed certificates, but you must distribute each certificate to each server and client application. You can pay an existing CA to sign certificates, but that has a cost, and leaves control of trust with a third party. You can set up a CA or certificate management software, but that can be a significant effort and cost. As a shortcut to setting up a private CA, DS software uses deployment IDs and passwords.

  DS software uses the deployment ID and its password to generate key pairs without storing the CA private key.

For additional details, see Deployment IDs.

3. Set the deployment ID as the value of the environment variable, `DEPLOYMENT_ID`:

   ```
   $ export DEPLOYMENT_ID=your-deployment-id
   ```

   Examples in the documentation show this environment variable as a reminder to use your own key. Other options are available, as described by the `setup --help` command.

4. Run the `setup` command to install a directory server replica with the evaluation profile:

```
# Set up a directory server for evaluation.
$ /path/to/opendj/setup \
 --serverId evaluation-only \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword password \
 --monitorUserPassword password \
 --hostname localhost \
 --adminConnectorPort 4444 \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --replicationPort 8989 \
 --bootstrapReplicationServer localhost:8989 \
 --profile ds-evaluation \
 --start \
 --acceptLicense
```

- The `setup` command is located where you installed the files.
- The setup process uses the deployment ID you generated, and its password.

  If you specify only a deployment password, and no deployment ID, the `setup` command generates a deployment ID and displays it in the output.
- This example prepares a single server for evaluation, so the hostname is `localhost`.

  In production, use fully qualified domain names, such as `ds.example.com`.
- The server is ready to replicate sample data with other servers, but there are no other replicas, yet.

  For now, the server points to itself as a bootstrap replication server. To get started with replication, see Learn replication.
- It sets a password for the default monitoring user account, `uid=Monitor`.
- The server listens for requests on the ports used in examples throughout the documentation.
- For evaluation purposes, no further configuration is required.

  The `--start` option forces the server to start as part of the setup process.

Learn about the evaluation setup profile

The evaluation setup profile helps you learn and demonstrate directory services. Unlike other setup profiles, which use secure, production-ready access control settings, the evaluation setup profile provides easy access to sample data with the following features:

- Sample Example.com data.

  The sample data has the base DN `dc=example,dc=com`. It includes more than 100 hand-written entries for users, groups, and devices.

By default, it also includes 100,000 generated users, with DNs from `uid=user.0,ou=people,dc=example.dc=com` to `uid=user.99999,ou=people,dc=example.dc=com`.

Use the `--set ds-evaluation/generatedUsers:number` option to generate a different number of additional entries. Each generated user has the same password, which is `password`.

The hand-written sample Example.com data includes a group of directory administrators, `cn=Directory Administrators,ou=Groups,dc=example,dc=com`. Members of this group, such as `kvaughan`, have full access to directory data.

Examples throughout the documentation demonstrate features using this sample data.

- Global permission to perform operations over insecure connections.
- REST to LDAP enabled by default.
- Additional schema for examples demonstrating class of service and JSON attributes.
- Custom matching rule providers for JSON attributes.
- Many permissions, such as anonymous read and search access, listed in the table that follows.

The evaluation setup profile lets you learn and demonstrate most directory features without adding any ACIs.

Name	Description	ACI definition
Anonymous extended operation access	Anonymous and authenticated users can request the LDAP extended operations that are specified by OID or alias. Modification or removal may affect applications.	`(targetcontrol="Assertion||AuthorizationIdentity||MatchedValues||NoOp||PasswordPolicy||PasswordQualityAdvice||PermissiveModify||PostRead||PreRead||RealAttrsOnly||SimplePagedResults||TransactionId||VirtualAttrsOnly||Vlv") (version 3.0; acl "Anonymous extended operation access"; allow(read) userdn="ldap:///anyone";)`
Anonymous extended operation access	Anonymous and authenticated users can request the LDAP extended operations that are specified by OID or alias. Modification or removal may affect applications.	`(extop="Cancel||GetSymmetricKey||PasswordModify||StartTls||WhoAmI") (version 3.0; acl "Anonymous extended operation access"; allow(read) userdn="ldap:///anyone";)`

Name	Description	ACI definition
Anonymous read and search access	Anonymous and authenticated Example.com users can read the user data attributes that are specified by their names.	`(targetattr!="userPassword||authPassword||debugsearchindex") (version 3.0; acl "Anonymous read and search access"; allow (read,search,compare) userdn="ldap:///anyone";)`
Authenticated control use	Authenticated Example.com users can proxy and examine CSNs.	`(targetcontrol="ProxiedAuth||Csn") (version 3.0; acl "Authenticated control use"; allow(read) userdn="ldap:///all";)`
Authenticated users extended operation access	Authenticated users can request the LDAP extended operations that are specified by OID or alias. Modification or removal may affect applications.	`(targetcontrol="ManageDsaIt||RelaxRules||ServerSideSort||SubEntries||SubtreeDelete") (version 3.0; acl "Authenticated users extended operation access"; allow(read) userdn="ldap:///all";)`
Authenticated users extended operation access	Authenticated users can request the LDAP extended operations that are specified by OID or alias. Modification or removal may affect applications.	`(extop="PasswordPolicyState") (version 3.0; acl "Authenticated users extended operation access"; allow(read) userdn="ldap:///all";)`
Directory administrator full access	Example.com directory administrators have access to read and write Example.com directory data, rename and move entries, and use proxied authorization.	`(targetattr="*") (version 3.0; acl "Directory administrator full access"; allow (all,export,import,proxy) groupdn="ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)`
Proxied authorization for apps	Example.com applications can make requests on behalf of other users.	`(targetattr="*") (version 3.0; acl "Proxied authorization for apps"; allow (all,proxy) (userdn="ldap:///cn=*,ou=Apps,dc=example,dc=com");)`

Name	Description	ACI definition
Self entry modification	Authenticated users can modify the specified attributes on their own entries.	`(targetattr=" audio || authPassword || description || displayName || givenName || homePhone || homePostalAddress || initials || jpegPhoto || labeledURI || mobile || pager || postalAddress || postalCode || preferredLanguage || telephoneNumber || userPassword") (version 3.0; acl "Self entry modification"; allow (write) userdn="ldap:///self";)`
Self entry read for passwords	Authenticated users can read the password values on their own entries. By default, the server applies a one-way hash algorithm to the password value before writing it to the entry, so it is computationally difficult to recover the plaintext version of the password from the stored value.	`(targetattr="userPassword|| authPassword") (version 3.0; acl "Self entry read for passwords"; allow (read,search,compare) userdn="ldap:///self";)`
Self service group creation	Authenticated Example.com users can create self service groups.	`(targattrfilters="add=objectClass:(objectClass=groupOfNames)") (version 3.0; acl "Self service group creation";allow (add) (userdn="ldap:///uid=*,ou=People,dc=example,dc=com");)`
Self service group deletion	The authenticated owner of a self service group can delete the group.	`(version 3.0; acl "Self service group deletion";allow (delete) (userattr="owner#USERDN");)`
Self service group registration	Authenticated Example.com users can sign themselves up as members of self service groups.	`(targetattr="member") (version 3.0; acl "Self service group registration"; allow (selfwrite) (userdn="ldap:///uid=*,ou=People,dc=example,dc=com");)`
User-Visible Monitor Attributes	Authenticated users can read monitoring information if they have the monitor read privilege. Modification or removal may affect applications.	`(target="ldap:///cn=monitor") (targetattr="*||+") (version 3.0; acl "User-Visible Monitor Attributes"; allow (read,search,compare) userdn="ldap:///all";)`

Name	Description	ACI definition
User-visible operational attributes	Anonymous and authenticated users can read attributes that identify entries and that contain information about modifications to entries.	`(targetattr=" createTimestamp || creatorsName || modifiersName || modifyTimestamp || entryDN || entryUUID || subschemaSubentry || etag || governingStructureRule || structuralObjectClass || hasSubordinates || numSubordinates || isMemberOf") (version 3.0; acl "User-visible operational attributes"; allow (read,search,compare) userdn="ldap:///anyone";)`
User-Visible Root DSE Operational Attributes	Anonymous and authenticated users can read attributes that describe what the server supports. Modification or removal may affect applications.	`(target="ldap:///") (targetscope="base") (targetattr="objectClass|| namingContexts|| subSchemaSubEntry|| supportedAuthPasswordSchemes|| supportedControl|| supportedExtension|| supportedFeatures|| supportedLDAPVersion|| supportedSASLMechanisms|| supportedTLSCiphers|| supportedTLSProtocols|| vendorName||vendorVersion|| fullVendorVersion||alive|| healthy")(version 3.0; acl "User-Visible Root DSE Operational Attributes"; allow (read,search,compare) userdn="ldap:///anyone";)`
User-Visible Schema Operational Attributes	Authenticated users can read LDAP schema definitions. Modification or removal may affect applications.	`(target="ldap:///cn=schema") (targetscope="base") (targetattr="objectClass|| attributeTypes||dITContentRules|| dITStructureRules ||ldapSyntaxes|| matchingRules||matchingRuleUse|| nameForms||objectClasses||etag|| modifiersName||modifyTimestamp") (version 3.0; acl "User-Visible Schema Operational Attributes"; allow (read,search,compare) userdn="ldap:///all";)`

Install DS for AM CTS

1. Before proceeding, install the server files.
   For details, see Unpack files.

2. Run the appropriate `setup` command with the `--profile am-cts` option.

   Installation settings depend on AM token expiration and session capability requirements:

   1. AM reaper manages all token expiration:

```
$ /path/to/opendj/setup \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --monitorUserPassword str0ngMon1torPa55word \
 --hostname ds.example.com \
 --adminConnectorPort 4444 \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --replicationPort 8989 \
 --bootstrapReplicationServer rs1.example.com:8989 \
 --bootstrapReplicationServer rs2.example.com:8989 \
 --profile am-cts \
 --set am-cts/amCtsAdminPassword:5up35tr0ng \
 --acceptLicense
```

   2. AM reaper manages only SESSION token expiration:

```
$ /path/to/opendj/setup \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --monitorUserPassword str0ngMon1torPa55word \
 --hostname ds.example.com \
 --adminConnectorPort 4444 \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --replicationPort 8989 \
 --bootstrapReplicationServer rs1.example.com:8989 \
 --bootstrapReplicationServer rs2.example.com:8989 \
 --profile am-cts \
 --set am-cts/amCtsAdminPassword:5up35tr0ng \
 --set am-cts/tokenExpirationPolicy:am-sessions-only \
 --acceptLicense
```

   3. DS manages all token expiration:

```
$ /path/to/opendj/setup \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --monitorUserPassword str0ngMon1torPa55word \
 --hostname ds.example.com \
 --adminConnectorPort 4444 \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --replicationPort 8989 \
 --bootstrapReplicationServer rs1.example.com:8989 \
 --bootstrapReplicationServer rs2.example.com:8989 \
 --profile am-cts \
 --set am-cts/amCtsAdminPassword:5up35tr0ng \
 --set am-cts/tokenExpirationPolicy:ds \
 --acceptLicense
```

For details about the mechanism DS uses to expire tokens, see Entry expiration.

In each of the preceding example commands:

- The deployment ID for installing the server is stored in the environment variable `DEPLOYMENT_ID`. Install all servers in the same deployment with the same deployment ID and deployment ID password. For details, read Deployment IDs.
- The service account to use in AM when connecting to DS has:
  - Bind DN: `uid=openam_cts,ou=admins,ou=famrecords,ou=openam-session,ou=tokens`.
  - Password: The password you set with `am-cts/amCtsAdminPassword`.
- The base DN for AM CTS tokens is `ou=famrecords,ou=openam-session,ou=tokens`.
- The `am-cts` profile excludes the base DN from change number indexing.

For the full list of profiles and parameters, see Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, see Install DS for custom cases.

4. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

Install DS for AM configuration

1. Before proceeding, install the server files.
   For details, see Unpack files.

2. Run the `setup` command with the `--profile am-config` option:

```
$ /path/to/opendj/setup \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --monitorUserPassword str0ngMon1torPa55word \
 --hostname ds.example.com \
 --adminConnectorPort 4444 \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --replicationPort 8989 \
 --bootstrapReplicationServer rs1.example.com:8989 \
 --bootstrapReplicationServer rs2.example.com:8989 \
 --profile am-config \
 --set am-config/amConfigAdminPassword:5up35tr0ng \
 --acceptLicense
```

   - The deployment ID for installing the server is stored in the environment variable `DEPLOYMENT_ID`. Install all servers in the same deployment with the same deployment ID and deployment ID password. For details, read Deployment IDs.
   - The service account to use in AM when connecting to DS has:
     - Bind DN: `uid=am-config,ou=admins,ou=am-config`.
     - Password: The password you set with `am-config/amConfigAdminPassword`.
   - The base DN for AM configuration data is `ou=am-config`.
   - AM does not require change number indexing, which involves resource-intensive processing. If AM is the only application using this data, disable change number indexing. For details, see Disable change number indexing.

   For the full list of profiles and parameters, see Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, see Install DS for custom cases.

4. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

Install DS for AM identities

Use this profile when setting up DS as an AM identity repository and user data store. It includes the additional LDAP schema and indexes required to store AM identities:

1. Before proceeding, install the server files.
   For details, see Unpack files.

2. Run the `setup` command with the `--profile am-identity-store` option:

```
$ /path/to/opendj/setup \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --monitorUserPassword str0ngMon1torPa55word \
 --hostname ds.example.com \
 --adminConnectorPort 4444 \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --replicationPort 8989 \
 --bootstrapReplicationServer rs1.example.com:8989 \
 --bootstrapReplicationServer rs2.example.com:8989 \
 --profile am-identity-store \
 --set am-identity-store/amIdentityStoreAdminPassword:5up35tr0ng \
 --acceptLicense
```

   - The deployment ID for installing the server is stored in the environment variable `DEPLOYMENT_ID`. Install all servers in the same deployment with the same deployment ID and deployment ID password. For details, read Deployment IDs.
   - The service account to use in AM when connecting to DS has:
     - Bind DN: `uid=am-identity-bind-account,ou=admins,ou=identities`.
     - Password: The password you set with `am-identity-store/amIdentityStoreAdminPassword`.
   - The base DN for AM identities is `ou=identities`.
   - AM does not require change number indexing, which involves resource-intensive processing. If AM is the only application using this data, disable change number indexing. For details, see Disable change number indexing.

     For the full list of profiles and parameters, see Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, see Install DS for custom cases.

4. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

Install DS as an IDM repository

1. Before proceeding, install the server files.
   For details, see Unpack files.

2. Run the `setup` command with the `--profile idm-repo` option:

   ```
   $ /path/to/opendj/setup \
    --deploymentId $DEPLOYMENT_ID \
    --deploymentIdPassword password \
    --rootUserDN uid=admin \
    --rootUserPassword str0ngAdm1nPa55word \
    --hostname localhost \
    --adminConnectorPort 34444 \
    --ldapPort 31389 \
    --enableStartTls \
    --profile idm-repo \
    --set idm-repo/domain:forgerock.com \
    --acceptLicense
   ```

   - The deployment ID for installing the server is stored in the environment variable `DEPLOYMENT_ID`. Install all servers in the same deployment with the same deployment ID and deployment ID password. For details, read Deployment IDs.
   - The administrative account to use in IDM when connecting to DS has:
     - Bind DN: The DN set with the `--rootUserDN` option.
     - Password: The password set with the `--rootUserPassword` option.
   - The base DN for IDM data is `dc=openidm,dc=forgerock,dc=com`.
   - IDM requires change number indexing with the default settings.

   For the full list of profiles and parameters, see Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, see Install DS for custom cases.

4. If all access to DS goes through IDM, IDM manages password policy.

   In this case, relax the default password policy settings:

```
$ dsconfig \
 set-password-policy-prop \
 --policy-name "Default Password Policy" \
 --reset password-validator \
 --offline \
 --no-prompt

$ dsconfig \
 set-password-policy-prop \
 --policy-name "Root Password Policy" \
 --reset password-validator \
 --offline \
 --no-prompt
```

5. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

Install DS for user data

This profile includes indexes for `inetOrgPerson` entries. It is not intended for deployments with AM or IDM identities.

It does not include the additional LDAP schema and indexes required to store AM identities. To set up a user data store for AM or for sharing between AM and IDM, see Install DS for AM identities instead.

To import generated sample user data, see Install DS for evaluation instead:

1. Before proceeding, install the server files.

   For details, see Unpack files.

2. Run the `setup` command with the `--profile ds-user-data` option:

```
$ /path/to/opendj/setup \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --monitorUserPassword str0ngMon1torPa55word \
 --hostname ds.example.com \
 --adminConnectorPort 4444 \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --replicationPort 8989 \
 --bootstrapReplicationServer rs1.example.com:8989 \
 --bootstrapReplicationServer rs2.example.com:8989 \
 --profile ds-user-data \
 --set ds-user-data/baseDn:dc=example,dc=com \
 --set ds-user-data/ldifFile:/tmp/user-data.ldif \
 --acceptLicense
```

In this example, the `/tmp/user-data.ldif` file contains the user data entries to import. This is just a placeholder. When you run the command, replace it with your LDIF file containing your own user data.

- The deployment ID for installing the server is stored in the environment variable `DEPLOYMENT_ID`. Install all servers in the same deployment with the same deployment ID and deployment ID password. For details, read Deployment IDs.
- The data is stored in the `userData` backend.

For the full list of profiles and parameters, see Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, see Install DS for custom cases.

4. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

This setup profile creates the following indexes for user data:

Index	Approx.	Equality	Ordering	Presence	Substring	Entry Limit
`aci`	-	-	-	Yes	-	4000
`cn`	-	Yes	-	-	Yes	4000
`dn2id`	Non-configurable internal index					
`ds-certificate-fingerprint`	-	Yes	-	-	-	4000

Index	Approx.	Equality	Ordering	Presence	Substring	Entry Limit
`ds-certificate-subject-dn`	-	Yes	-	-	-	4000
`ds-sync-conflict`	-	Yes	-	-	-	4000
`ds-sync-hist`	-	-	Yes	-	-	4000
`entryUUID`	-	Yes	-	-	-	4000
`givenName`	-	Yes	-	-	Yes	4000
`id2children`	Non-configurable internal index					
`id2subtree`	Non-configurable internal index					
`mail`	-	Yes	-	-	Yes	4000
`member`	-	Yes	-	-	-	4000
`objectClass`	-	Yes	-	-	-	4000
`sn`	-	Yes	-	-	Yes	4000
`telephoneNumber`	-	Yes	-	-	Yes	4000
`uid`	-	Yes	-	-	-	4000
`uniqueMember`	-	Yes	-	-	-	4000

Install DS for custom cases

Follow these steps to install a DS replica with your own custom configuration:

1. Before proceeding, install the server files.

   For details, see Unpack files.

2. Run the `setup` command with any required setup profiles.

3. Finish configuring the server.

   Perform any of the following optional steps *before starting the server*.

   Use the `--offline` option with commands instead of the credentials and connection information shown in many examples:

   - Add custom syntaxes and matching rules.

     For examples, see Custom indexes for JSON.

- Configure password storage.

  For details, see Configure password policies.

  Take care to configure the password policy import plugin as well. For details on the settings, see Password Policy Import Plugin.

- Add custom LDAP schema.

  For details, see LDAP schema.

- Configure one or more backends for your data.

  For details, see Create a backend. When you create the backend, unless you choose *not* to replicate the data, follow each step of the procedure, adapting the example commands for offline use:

  - Configure the new backend using the `dsconfig create-backend` as shown.
  - Verify that replication is enabled using the `dsconfig get-synchronization-provider-prop` command as shown.
  - Let the server replicate the base DN of the new backend, using the `dsconfig create-replication-domain` command as shown to configure the replication domain.
  - If you have existing data for the backend, make appropriate plans to initialize replication, as described in Manual initialization.

- Configure indexes for the backends you configured.

  For details, see Indexes.

- Make sure the server has the shared master key for encrypted data and backups.

  If you set up the servers with a known deployment ID and password, you have nothing to do.

  If you do not know the deployment ID and password, see Replace deployment IDs.

- Initialize replication.

  For example, import the data from LDIF, or restore the data from backup.

  For details, see Manual initialization, Import LDIF, or Restore.

4. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

When you start the server, it generates initial state identifiers (generation IDs) for its replicated base DNs. If you perform the above configuration steps on replicas separately after starting them, their generation IDs can be out of sync.

When generation IDs do not match on different replicas for a particular base DN, DS must assume that the replicas do not have the same data. As a result, replication cannot proceed. To fix the mismatch of this replica's generation IDs with other replicas, stop the server and clear all replication data:

```
$ /path/to/opendj/bin/stop-ds
$ /path/to/opendj/bin/dsrepl clear-changelog
```

Important

Clearing the changelog before all the changes have been sent to other replication servers can cause you to lose data. Use the `dsrepl clear-changelog` command only when initially setting up the replica, unless specifically instructed to do so by a qualified ForgeRock technical support engineer.

Complete any further configuration necessary while the replica is stopped to align it with other replicas. When you start the replica again with the `start-ds` command, other replication servers update it with the data needed to resume replication.

For details on replication, see Replication.

Install a directory proxy

Directory proxy servers forward LDAP requests for user data to remote directory servers. Proxy servers make it possible to provide a single point of access to a directory service, and to hide implementation details from client applications.

Check compatibility

DS proxy servers connect to remote LDAP directory servers using proxied authorization. The proxied authorization control (OID: `2.16.840.1.113730.3.4.18`) is defined by RFC 4370 *Lightweight Directory Access Protocol (LDAP) Proxied Authorization Control*. If the LDAP directory server does not support proxied authorization, *it cannot be used with DS directory proxy server*.

The following list of LDAP servers *do not support* proxied authorization, and so, do not work with DS directory proxy server:

- Microsoft Active Directory
- Oracle Internet Directory

The following list of LDAP servers support proxied authorization according to their documentation. ForgeRock does not test all servers listed:

- ForgeRock Directory Services
- ApacheDS
- NetIQ eDirectory
- OpenDJ
- OpenLDAP
- Oracle Directory Server Enterprise Edition
- PingDirectory
- Red Hat Directory Server

If your LDAP server does not appear in the lists above, check its documentation regarding support for proxied authorization. Alternatively, check the list of `supportedControl` values on the server's root DSE.

Try DS directory proxy

Before installing DS directory proxy server in production, or with a non-DS directory server, try it on your computer.

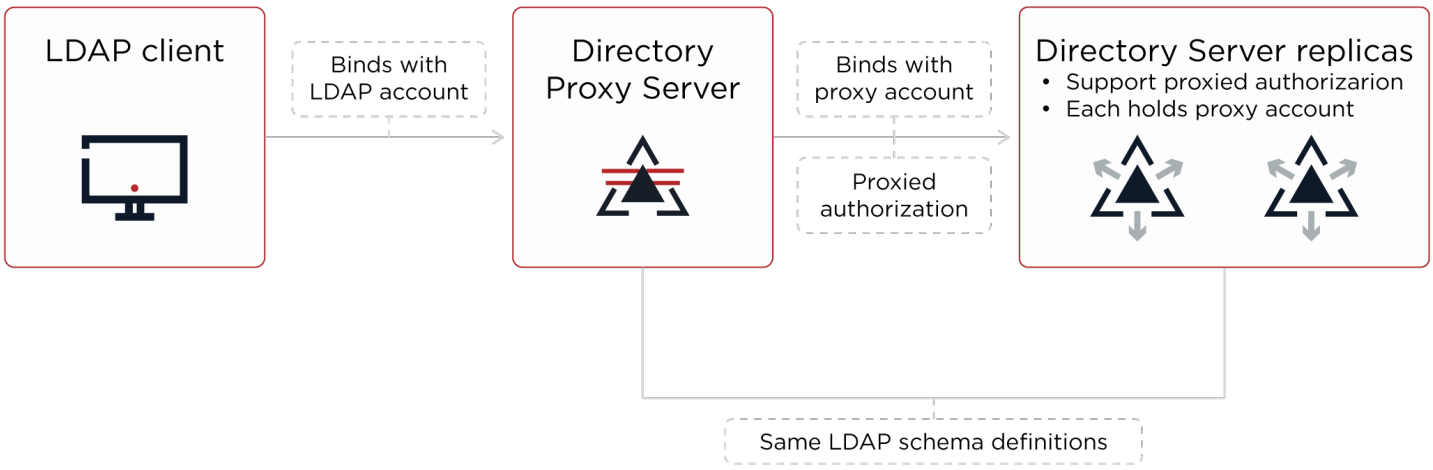


Figure 1. Proxy Configuration

The following examples demonstrate DS directory proxy server forwarding LDAP requests to two DS replicas on your computer. This demonstration includes the following high-level tasks:

- Install two DS directory server replicas as proxied servers.
- Set up the DS directory proxy server to forward requests to the DS directory servers.
- Send LDAP requests to the DS directory proxy server, and observe the results.

The deployment ID for installing the server is stored in the environment variable `DEPLOYMENT_ID`. Install all servers in the same deployment with the same deployment ID and deployment ID password. For details, read Deployment IDs.

> **Tip**
>
> The DS directory proxy server does not have backup files or directory data to encrypt and decrypt. But it does open secure connections to the remote directory servers, and so must trust the certificates that the remote directory servers present to negotiate TLS.
> By default, DS deployments use TLS keys and a CA generated from the deployment ID and deployment ID password. This is the same deployment ID and password used to configure the DS directory servers. Therefore, use the same deployment ID and password when configuring DS directory proxy servers, so they can trust the directory server certificates.

Install two DS directory server replicas with the evaluation and proxied server profiles:

```
# Unpack server files:
unzip -q ~/Downloads/DS-7.2.5.zip -d /tmp

# Copy server files before setting up each replica:
mkdir /path/to/opendj && cp -r /tmp/opendj/* /path/to/opendj
mkdir /path/to/replica && cp -r /tmp/opendj/* /path/to/replica

# Set up the servers as replicas of each other
# with StartTLS support for the proxy connections:
/path/to/opendj/setup \
  --deploymentId $DEPLOYMENT_ID \
  --deploymentIdPassword password \
  --hostname localhost \
  --ldapPort 11389 \
  --enableStartTls \
  --ldapsPort 11636 \
  --adminConnectorPort 14444 \
  --rootUserDN uid=admin \
  --rootUserPassword password \
  --profile ds-evaluation \
  --profile ds-proxied-server \
  --set ds-proxied-server/baseDn:dc=example,dc=com \
  --replicationPort 18989 \
  --bootstrapReplicationServer localhost:28989 \
  --acceptLicense \
  --start \
  --quiet

/path/to/replica/setup \
  --deploymentId $DEPLOYMENT_ID \
  --deploymentIdPassword password \
  --hostname localhost \
  --ldapPort 21389 \
  --enableStartTls \
  --ldapsPort 21636 \
  --adminConnectorPort 24444 \
  --rootUserDN uid=admin \
  --rootUserPassword password \
  --profile ds-evaluation \
  --profile ds-proxied-server \
  --set ds-proxied-server/baseDn:dc=example,dc=com \
  --replicationPort 28989 \
  --bootstrapReplicationServer localhost:18989 \
  --acceptLicense \
  --start \
  --quiet

# Update PATH to include DS tools:
export PATH=/path/to/opendj/bin:${PATH}
```

Notice that the examples apply two setup profiles to each replica:

- The DS evaluation setup profile adds sample Example.com data.

  For details, see Install DS for evaluation.

- The DS proxied server setup profile adds a service account for the proxy server, and sets ACIs to grant the account permission to use proxied authorization. The proxy authenticates to the directory servers with its certificate, whose subject DN is `CN=DS, O=ForgeRock.com`.

  For details, see Install DS for use with DS proxy.

Set up a directory proxy server to forward requests to the replicas:

```
# Copy server files before setting up the proxy:
mkdir /path/to/proxy && cp -r /tmp/opendj/* /path/to/proxy

# Set up the proxy server to access the replicas:
/path/to/proxy/setup \
 --serverId proxy \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword password \
 --hostname localhost \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --adminConnectorPort 4444 \
 --profile ds-proxy-server \
 --set ds-proxy-server/bootstrapReplicationServer:"localhost:14444" \
 --set ds-proxy-server/bootstrapReplicationServer:"localhost:24444" \
 --set ds-proxy-server/rsConnectionSecurity:start-tls \
 --set ds-proxy-server/certNickname:ssl-key-pair \
 --set ds-proxy-server/keyManagerProvider:PKCS12 \
 --set ds-proxy-server/trustManagerProvider:PKCS12 \
 --start \
 --acceptLicense

# Grant access to data through the proxy server:
dsconfig \
 create-global-access-control-policy \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Authenticated access to example.com data" \
 --set authentication-required:true \
 --set request-target-dn-equal-to:"dc=example,dc=com" \
 --set request-target-dn-equal-to:"**,dc=example,dc=com" \
 --set permission:read \
 --set permission:write \
 --set allowed-attribute:"*" \
 --set allowed-attribute:isMemberOf \
 --set allowed-attribute-exception:authPassword \
 --set allowed-attribute-exception:userPassword \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

As you set up only DS servers which all use the same default schema, there is no need to manually align the proxy LDAP schema with the directory server schema.

Send LDAP requests to the DS directory proxy server, and observe the results.

The following example searches the directory through the proxy:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN "ou=people,dc=example,dc=com" \
 "(|(cn=Babs Jensen)(cn=Sam Carter))" \
 cn

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen

dn: uid=scarter,ou=People,dc=example,dc=com
cn: Sam Carter
```

The following example modifies directory data through the proxy:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword hifalutin << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Modified by Babs Jensen
EOF

# MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com
```

Notice that the bind DNs and passwords are those of the users in the remote directory service.

For more background on each high-level task, read the rest of this page.

Create a service account

When preparing to use DS directory proxy servers with directory servers that are not DS servers, create a service account for the proxy to connect to the non-DS remote directory service.

The directory proxy server binds with this service account, and then forwards LDAP requests on behalf of other users.

For DS directory servers, use the proxied server setup profile if possible. For details, see Install DS for use with DS proxy.

The service account must have the following on all remote directory servers:

- The same bind credentials.

  If possible, use mutual TLS to authenticate the proxy user with the backend servers.

- The right to perform proxied authorization.

  Make sure the LDAP servers support proxied authorization (control OID: `2.16.840.1.113730.3.4.18`).

  For details, see RFC 4370, *Lightweight Directory Access Protocol (LDAP) Proxied Authorization Control*.

- When using a replication discovery mechanism with remote DS directory servers, the service account requires the `config-read` and `monitor-read` privileges for the service discovery mechanism. It requires the `proxied-auth` privilege and an ACI to perform proxied authorization.

The following listing shows an example service account that you could use with DS replicas. Adapt the account as necessary for your directory service:

```
dn: uid=proxy
objectClass: top
objectClass: account
objectClass: ds-certificate-user
uid: proxy
ds-certificate-subject-dn: CN=DS, O=ForgeRock.com
ds-privilege-name: config-read
ds-privilege-name: monitor-read
ds-privilege-name: proxied-auth
aci: (targetcontrol="ProxiedAuth")
  (version 3.0; acl "Allow proxied authorization";
  allow(read) userdn="ldap:///uid=proxy";)
```

Set up a directory proxy

The deployment ID for installing the server is stored in the environment variable `DEPLOYMENT_ID`. Install all servers in the same deployment with the same deployment ID and deployment ID password. For details, read Deployment IDs.

Proxy to DS servers

1. Before proceeding, install the server files.
   For details, see Unpack files.

2. Run the `setup --profile ds-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 based on the DS replication topology. It works with replicas set up using the `ds-proxied-server` setup profile.

   This feature works only with DS servers. If the remote LDAP servers in your deployment are not DS servers, see Proxy to non-DS servers.

   This proxy 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:

```
$ /path/to/opendj/setup \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --hostname ds.example.com \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --adminConnectorPort 4444 \
 --profile ds-proxy-server \
 --set ds-proxy-server/bootstrapReplicationServer:"rs.example.com:4444" \
 --set ds-proxy-server/rsConnectionSecurity:start-tls \
 --set ds-proxy-server/certNickname:ssl-key-pair \
 --set ds-proxy-server/keyManagerProvider:PKCS12 \
 --set ds-proxy-server/trustManagerProvider:PKCS12 \
 --start \
 --acceptLicense
```

This example uses mutual TLS with a certificate generated with a deployment ID and password. Adjust the security settings as required for your deployment.

Proxy to non-DS servers

1. Before proceeding, install the server files.
   For details, see Unpack files.

2. Run the `setup --profile ds-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 has a static list of remote servers to connect to. It forwards only requests targeting `dc=example,dc=com`:

```
# Initially configure the server with a fake replication service discovery mechanism:
$ /path/to/opendj/setup \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --hostname ds.example.com \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --adminConnectorPort 4444 \
 --profile ds-proxy-server \
 --set ds-proxy-server/bootstrapReplicationServer:"fake-rs.example.com:4444" \
 --set ds-proxy-server/rsConnectionSecurity:start-tls \
 --start \
 --acceptLicense
# Create a static service discovery mechanism:
$ dsconfig \
 create-service-discovery-mechanism \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword str0ngAdm1nPa55word \
 --mechanism-name "Static Service Discovery Mechanism" \
 --type static \
 --set primary-server:local1.example.com:636 \
 --set primary-server:local2.example.com:636 \
 --set secondary-server:remote1.example.com:636 \
 --set secondary-server:remote2.example.com:636 \
 --set ssl-cert-nickname:ssl-key-pair \
 --set key-manager-provider:PKCS12 \
 --set trust-manager-provider:"JVM Trust Manager" \
 --set use-ssl:true \
 --set use-sasl-external:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
# Replace the fake replication service discovery mechanism with the static one:
$ dsconfig \
 set-backend-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword str0ngAdm1nPa55word \
 --backend-name proxyRoot \
 --set shard:"Static Service Discovery Mechanism" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

This example uses mutual TLS with a certificate generated with a deployment ID and password. Adjust the security settings as required for your deployment.

Configure access control

1. Explicitly grant appropriate access to remote data.

   The following example grants authenticated users access to data under `dc=example,dc=com`:

```
$ dsconfig \
 create-global-access-control-policy \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword str0ngAdm1nPa55word \
 --policy-name "Authenticated access to example.com data" \
 --set authentication-required:true \
 --set request-target-dn-equal-to:"dc=example,dc=com" \
 --set request-target-dn-equal-to:"**,dc=example,dc=com" \
 --set permission:read \
 --set permission:write \
 --set allowed-attribute:"*" \
 --set allowed-attribute:isMemberOf \
 --set allowed-attribute-exception:authPassword \
 --set allowed-attribute-exception:userPassword \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

DS proxy servers do not use ACIs for access control. Instead, they use global access control policies. By default, the access rights are configured the same as the default settings for a directory server. You no doubt need to adapt these policies for your deployment. For additional examples, see Access control.

2. Make sure the backend directory servers are properly prepared, as described Create a service account.

For more background on LDAP proxy features, see LDAP proxy.

Default global policies

Access control rules are defined using individual access control policy entries. A user's access is defined as the union of all access control rules that apply to that user. In other words, an individual access control rule can only grant additional access and can not remove rights granted by another rule. This approach results in an access control policy which is easier to understand and audit, since all rules can be understood in isolation.

Policy	Settings
Anonymous extended operation and control access	**authentication-required** • `false` **allowed-extended-operation** • `Cancel` • `GetSymmetricKey` • `PasswordModify` • `StartTLS` • `WhoAmI` **allowed-control** • `Assertion` • `MatchedValues` • `NoOp` • `PasswordQualityAdvice` • `PermissiveModify` • `PostRead` • `PreRead` • `RealAttrsOnly` • `SimplePagedResults` • `VirtualAttrsOnly` • `AuthorizationIdentity` • `PasswordPolicy` • `TransactionId` • `Vlv`
Authenticated extended operation and control access	**authentication-required** • `true` **allowed-extended-operation** • `PasswordPolicyState` **allowed-control** • `ManageDsaIt` • `SubEntries` • `RelaxRules` • `SubtreeDelete` • `ServerSideSort`

Policy	Settings
Schema access	**authentication-required** • `true` **request-target-dn-equal-to** • `cn=schema` **permission** • `read` **allowed-attribute** • `objectClass` • `@subschema` • `etag` • `ldapSyntaxes` • `modifiersName` • `modifyTimestamp`
Root DSE access	**authentication-required** • `false` **request-target-dn-equal-to** • `""` **permission** • `read` **allowed-attribute** • `objectClass` • `namingContexts` • `subSchemaSubEntry` • `supportedAuthPasswordSchemes` • `supportedControl` • `supportedExtension` • `supportedFeatures` • `supportedLDAPVersion` • `supportedSASLMechanisms` • `supportedTLSCiphers` • `supportedTLSProtocols` • `vendorName` • `vendorVersion` • `fullVendorVersion` • `alive` • `healthy`

Policy	Settings
Monitor access	**authentication-required** • `true` **request-target-dn-equal-to** • `cn=monitor` **permission** • `read` **allowed-attribute** • `*` • `+`

Align LDAP schema

Directory servers can reject LDAP change requests that do not comply with LDAP schema. LDAP client applications read LDAP schema definitions from directory servers to determine in advance whether a change request complies with LDAP schema.

When an LDAP client requests LDAP schema from the proxy, the proxy returns *its* LDAP schema. Ideally, the LDAP schema definitions on the proxy match the LDAP schema definitions on the remote directory servers. Otherwise, client applications might check their requests against the proxy's LDAP schema, and yet still see their requests fail with schema violations when the proxy forwards a request to a directory server.

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

For more information, see LDAP schema. Schema definitions on a non-DS remote directory server might require translation from another format before you add them on DS directory proxy servers.

Troubleshooting

Common errors with DS directory proxy server installations include the following:

49 (Invalid Credentials)

When LDAP bind requests through the proxy invariably result in an invalid credentials error, but bind requests to the directory server with the same credentials do not, the problem lies with the proxy service account.

The proxy service account must allow bind requests to the directory server. The following example demonstrates a request sent directly to a directory server. The command makes a bind request and then a search request. The directory server is set up according to the instructions in Try DS directory proxy:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN "ou=people,dc=example,dc=com" \
 "(|(cn=Babs Jensen)(cn=Sam Carter))" \
 cn

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen

dn: uid=scarter,ou=People,dc=example,dc=com
cn: Sam Carter
```

Start with the filtered directory server access log, `logs/filtered-ldap-access.audit.json`, to debug bind failures.

123 (Authorization Denied)

Make sure that access control on the remote LDAP servers allows the proxy service account to use the proxied authorization control.

Proxied authorization does not allow operations on remote LDAP servers as the directory superuser (`uid=admin`). Do not connect as directory superuser when trying to access a directory server through the proxy. For administrative requests on remote LDAP servers, access the servers directly. This includes monitoring requests.

It is possible to configure proxied authorization so that an anonymous user (no bind DN, no bind password) can make a request through the proxy server to the remote directory server. ForgeRock recommends that you do not do this, however, as it is less secure.

Many applications perform some operations anonymously, such as reading the root DSE or LDAP schema. These operations are in fact requests to the proxy, not forwarded to remote LDAP servers. For applications to receive an appropriate response for LDAP schema requests, align LDAP schema on the proxy with LDAP schema on the remote LDAP servers as described above.

Install DS for use with DS proxy

1. Before proceeding, install the server files.
   For details, see Unpack files.

2. Run the `setup` command with the `--profile ds-proxied-server` option.

   The example shows the profile used with the evaluation profile. *Add this profile to the list* so proxy servers can access other profiles' data:

```
$ /path/to/opendj/setup \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDN uid=admin \
 --rootUserPassword str0ngAdm1nPa55word \
 --monitorUserPassword str0ngMon1torPa55word \
 --hostname ds.example.com \
 --adminConnectorPort 4444 \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --replicationPort 8989 \
 --bootstrapReplicationServer rs1.example.com:8989 \
 --bootstrapReplicationServer rs2.example.com:8989 \
 --profile ds-evaluation \
 --profile ds-proxied-server \
 --set ds-proxied-server/baseDn:dc=example,dc=com \
 --acceptLicense
```

- The deployment ID for installing the server is stored in the environment variable `DEPLOYMENT_ID`. Install all servers in the same deployment with the same deployment ID and deployment ID password. For details, read Deployment IDs.
- The account the DS proxy can use to connect to DS replicas has:
  - Bind DN: The DN from the `--set ds-proxied-server/proxyUserDn` option.

    Default: `uid=proxy`.
  - Certificate subject DN: The DN from the `--set ds-proxied-server/proxyUserCertificateSubjectDn` option.

    Default: `CN=DS, O=ForgeRock.com`.
  - Access to use proxied authorization in the base DNs specified by the multivalued `--set ds-proxied-server/baseDn` option.

    If you do not specify any values for `ds-proxied-server/baseDn`, the proxy user can perform operations with any account as authorization identity. This includes administrator accounts.

    To understand what this means, read Proxied authorization.
- The DS proxy server binds using certificate-based authentication with the SASL EXTERNAL mechanism.

  Make sure that the DS replicas' truststores lets them trust the proxy's certificate.
- The DS proxy server uses proxied authorization to perform operations on the DS replicas.

  The authorization identity for the operations must have appropriate access to the data on the DS replicas.

For the full list of profiles and parameters, see Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, see Install DS for custom cases.

4.

Start the server:

```
$ /path/to/opendj/bin/start-ds
```

Install standalone servers

Standalone replication servers have no application data backends. They store only changes to directory data. They are dedicated to transmitting replication messages, and to maintaining a replication change log.

Standalone directory servers store replicated copies of application data. These replicas send updates to and receive updates from replication servers. They connect to one replication server at a time, and do not maintain a replication change log.

Each replication server in a deployment connects to all other replication servers. The total number of replication connections, $Total_{conn}$, increases like the square of the number of replication servers. Large deployments that span WAN links can therefore benefit from having fewer replication servers.

$Total_{conn} = (N_{RS} * (N_{RS}-1))/2 + N_{DS}$

Here, N_{RS} is the number of replication servers (standalone or running in a directory server), and N_{DS} is the number of standalone directory servers.

A deployment with only a few standalone replication servers and many standalone directory servers, significantly limits the number of WAN connections for replication:

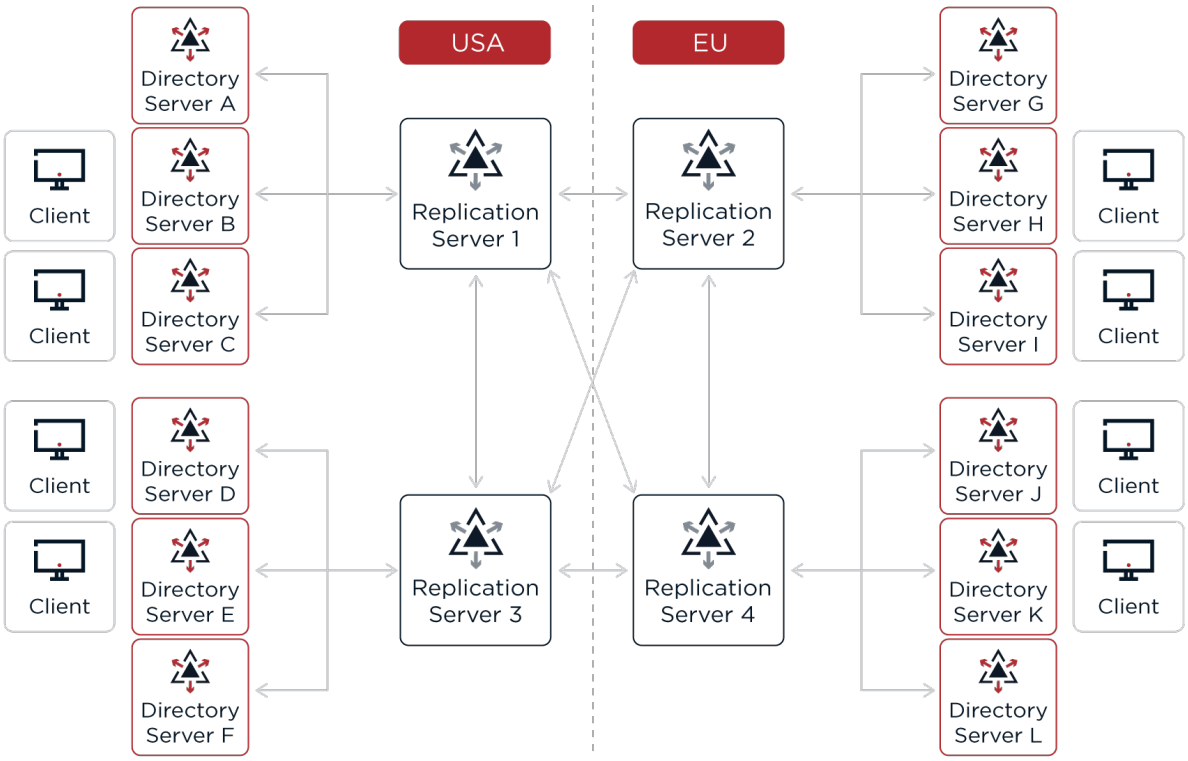


Figure 1. Deployment For Multiple Data Centers

The deployment ID for installing the server is stored in the environment variable `DEPLOYMENT_ID`. Install all servers in the same deployment with the same deployment ID and deployment ID password. For details, read Deployment IDs.

Set up standalone replication servers

1. Before proceeding, install the server files.
   For details, see Unpack files.

2. Set up a server as a standalone replication server:

   ```
   $ /path/to/opendj/setup \
    --deploymentId $DEPLOYMENT_ID \
    --deploymentIdPassword password \
    --rootUserDN uid=admin \
    --rootUserPassword password \
    --hostname rs-only.example.com \
    --adminConnectorPort 4444 \
    --replicationPort 8989 \
    --bootstrapReplicationServer rs-only.example.com:8989 \
    --bootstrapReplicationServer rs-only2.example.com:8989 \
    --acceptLicense
   ```

   The standalone replication server has no application data.

   It does have LDAP schema and changelog data. If you plan to add any additional schema to the replicas as part of the setup process, also add the schema to this server before starting it.

3. Start the server:

   ```
   $ /path/to/opendj/bin/start-ds
   ```

4. Repeat the previous steps on additional systems until you have sufficient replication servers to meet your availability requirements.

   To ensure availability, add at least one additional replication server per location. The following example adds a second standalone replication server:

   ```
   $ /path/to/opendj/setup \
    --deploymentId $DEPLOYMENT_ID \
    --deploymentIdPassword password \
    --rootUserDN uid=admin \
    --rootUserPassword password \
    --hostname rs-only2.example.com \
    --adminConnectorPort 4444 \
    --replicationPort 8989 \
    --bootstrapReplicationServer rs-only.example.com:8989 \
    --bootstrapReplicationServer rs-only2.example.com:8989 \
    --acceptLicense
   ```

   The standalone replication servers use each other as bootstrap servers to discover other servers in the deployment.

Set up standalone directory servers

1. Before proceeding, install the server files.
   For details, see Unpack files.

2. Set up the server as a directory server.

   Notice that the `--bootstrapReplicationServer` references the replication servers set up according to the steps in Set up standalone replication servers.

   The `--replicationPort` option is not used, because this is a standalone directory server:

   ```
   $ /path/to/opendj/setup \
    --serverId evaluation-only \
    --deploymentId $DEPLOYMENT_ID \
    --deploymentIdPassword password \
    --rootUserDN uid=admin \
    --rootUserPassword password \
    --adminConnectorPort 4444 \
    --hostname ds-only.example.com \
    --ldapPort 1389 \
    --enableStartTls \
    --ldapsPort 1636 \
    --httpsPort 8443 \
    --bootstrapReplicationServer rs-only.example.com:8989 \
    --bootstrapReplicationServer rs-only2.example.com:8989 \
    --profile ds-evaluation \
    --acceptLicense
   ```

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, see Install DS for custom cases.

4. Start the server:

   ```
   $ /path/to/opendj/bin/start-ds
   ```

5. Repeat the previous steps on additional systems until you have sufficient directory servers to meet your availability and performance requirements.

   To ensure availability, add at least one additional directory server per location.

Install with existing cryptographic keys

> **Note**
>
> When you set up a DS server with your own keys for PKI, you must still use a deployment ID and password.
> In this case, the deployment ID and password generate the shared master key used to protect secret keys for encrypted data, such as backup files. If you have not yet learned about deployment IDs, read Deployment IDs.
> If you plan to store the shared master key in an HSM, read the documentation carefully *before you install DS*. When you set up the server, you must avoid accidentally encrypting data while using the wrong shared master key. For details, see PKCS#11 hardware security module.

The `setup` command has options to simplify setting up a server with existing keys:

For…	Use…

Keystores containing server key pairs	`--useJavaKeyStore`
	`--useJceKeyStore`
	`--usePkcs11KeyStore`
	`--usePkcs12KeyStore`
	`-W, --keyStorePassword[:env|:file]`
	`-N, --certNickname`

Truststores containing trusted CA or self-signed certificates	`--useJavaTrustStore`
	`--useJceTrustStore`
	`--usePkcs11TrustStore`
	`--usePkcs12TrustStore`
	`-T, --trustStorePassword[:env|:file]`

Important features to be aware of:

- If the keystore file that holds the server key pair protects the server key with a password, that password must match the password for the entire store.

  DS does not support separate keystore and key passwords in keystore files.

- If you are using an HSM, also see PKCS#11 hardware security module.
- If you are using PEM format keys, see PEM format keys instead.

Follow steps similar to these to install a DS replica with existing cryptographic keys:

1. Before proceeding, install the server files.
   For details, see Unpack files.
2. Run the `setup` command with the appropriate options.

   The following example uses a PKCS#12 keystore file with the server's key pair, and a PKCS#12 truststore file with the CA's certificate.

   This example installs the server with the evaluation setup profile. Adapt the command for your use:

```
# Set up a directory server for evaluation using existing keys:
$ /path/to/opendj/setup \
 --serverId evaluation-only \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --usePkcs12TrustStore /path/to/truststore \
 --trustStorePassword password \
 --certNickname ssl-key-pair \
 --usePkcs12KeyStore /path/to/keystore \
 --keyStorePassword password \
 --rootUserDN uid=admin \
 --rootUserPassword password \
 --monitorUserPassword password \
 --hostname localhost \
 --adminConnectorPort 4444 \
 --ldapPort 1389 \
 --enableStartTls \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --replicationPort 8989 \
 --bootstrapReplicationServer localhost:8989 \
 --profile ds-evaluation \
 --acceptLicense
```

3. Finish configuring the server.

4. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

When you set up the server to use existing keystore files, the server configuration directly references those files. If you read the server configuration, you find that a Key Manager Provider references the keystore, and that a Trust Manager Provider references the truststore.

If you provide keystore and truststore passwords as strings, the `setup` command records them in files in the `opendj/config` directory.

For details on using variables instead, see Property value substitution.

Install a REST to LDAP gateway

A REST to LDAP gateway web application translates an HTTP request into one or more LDAP requests. The translation depends on the specific REST to LDAP gateway configuration.

An identity mapper translates the user identity into an LDAP identity for the bind. For background information, see Identity mappers.

Then the REST to LDAP mapping defines how the REST JSON resource corresponds to LDAP entries. The gateway handles the mapping configuration in the same way as an HTTP connection handler.

Figure 1. Requests Through a 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 Configure HTTP user APIs.

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 LDAP connections.

`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 LDAP features.

`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 API configuration.

Follow these steps to install the REST to LDAP gateway:

1. Review requirements for installation.
2. Deploy the .war file according to the instructions for your web application container.

   If you are using Wildfly, you must unzip the .war file into the deployment directory.
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 hostnames and port numbers.
   - `authentication`: Set the correct simple bind credentials.

     The LDAP account used to authenticate needs to perform proxied authorization, as described in Proxied authorization.

     The default sample configuration works with generated example data, and with the sample data imported when you set up the directory server for evaluation, as shown in Install DS for evaluation. If your data is different, then you must also change the JSON resource to LDAP entry mapping settings, as described in API configuration.

     For details regarding the configuration, see REST to LDAP 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 Key management for an example of how to use the Java `keytool` command to import a server certificate into a truststore file.

4. 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:

   1. 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`.

   2. If the REST to LDAP gateway runs in a different container, set the log level as described in the documentation.

      Messages are written to the container log.

5. Restart the REST to LDAP gateway or the web application container 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 set up for evaluation, as shown in Install DS for evaluation. In this example, the gateway is deployed under `/rest2ldap`:

```
$ curl \
 --user bjensen:hifalutin \
 --cacert ca-cert.pem \
 https://localhost:8443/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 `https://user.0:password@localhost:8443/rest2ldap/api/users/user.0`.

7. Configure your web application container to use HTTPS for secure connections to the gateway.

   See your web application container documentation for details.

Install a DSML gateway

The DSML gateway web application translates each HTTP request into one or more LDAP requests. The translation depends on the DSML protocol. For authentication, you must configure how HTTP user IDs map to LDAP identities.

Figure 1. Requests through a 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. Review requirements for installation.

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

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

   For details, see Configure DSML access.

4. Configure your web application container to use HTTPS for secure connections to the gateway.

   See your web application container documentation for details.

5. Restart the web application according to the instructions for your web application container.

Configure DSML access

Directory Services Markup Language (DSML) client access is implemented as a servlet web application. You edit the `WEB-INF/web.xml` file after deploying the web application.

The list of DSML configuration parameters are the following:

`ldap.host`

The hostname of the underlying directory service.

Default: `localhost`

`ldap.port`

The LDAP port number of the underlying directory service.

Default: `389`

`ldap.userdn`

Optional parameter specifying the DN to bind to the underlying directory service.

Default: anonymous bind

`ldap.userpassword`

Optional parameter specifying the password to bind to the underlying directory service.

Default: anonymous bind

`ldap.authzidtypeisid`

Use this parameter to set up the DSML gateway to do HTTP Basic Access Authentication, given the appropriate mapping between the user ID, and the user's entry in the directory.

This takes a boolean parameter specifying whether the HTTP Authorization header field's Basic credentials in the request hold a plain ID, rather than a DN.

If set to `true`, the gateway performs an LDAP SASL bind using SASL plain, enabled by default in DS servers to look for an exact match between a `uid` in the server, and the plain ID from the header.

In other words, if the plain ID is `bjensen`, then the bind DN is `uid=bjensen,ou=people,dc=example,dc=com`.

Configure DS identity mappers as necessary to use a different attribute than `uid`. For background information, see Identity Mappers.

Default: `false`

`ldap.usessl`

Whether `ldap.port` uses LDAPS.

Default: `false`

`ldap.usestarttls`

Whether to use StartTLS when connecting to `ldap.port`.

Default: `false`

`ldap.trustall`

Whether to blindly trust all server certificates when using LDAPS or StartTLS.

Default: `false`

`ldap.truststore.path`

The truststore used to verify server certificates when using LDAPS or StartTLS.

Required when using LDAPS or StartTLS and `ldap.trustall` is `false`.

`ldap.truststore.password`

The password to read the truststore.

Required when using a truststore with a password.

For initial testing purposes, try JXplorer, where the DSML Service is: `/webapp-dir/DSMLServlet`. The `webapp-dir` refers to the name of the directory holding the DSML `.war`.

Uninstallation

Uninstall .zip

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 the server:

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

3. Delete the files manually:

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

Uninstall the Debian package

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

1. Purge the package from your system:

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

2. Remove any remaining server configuration files and directory data:

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

Uninstall the RPM package

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

1. Remove the package from your system:

```
# rpm -e opendj
```

2. Remove the server configuration files and any directory data:

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

Uninstall the Windows MSI

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

1. Remove installed files in one of the following ways:
    1. Use Windows Control Panel.
        - Open Windows Control Panel and browse to the page to uninstall a program.
        - Find the ForgeRock directory service in the list and uninstall it.
    2. Use the `msiexec` command.

       The following command quietly removes installed files:

       ```
       C:\> msiexec /x DS-7.2.5.msi /q
       ```
2. Manually remove the server configuration files and any directory data.

File layout

DS software installs and creates the following files and directories. The following table is not meant to be exhaustive.

File or directory	Description
`bak`	Directory intended for local backup data.
`bat`	Windows command-line tools.
`bin`	UNIX/Linux command-line tools.
`changelogDb`	Backend for replication changelog data.
`classes`	Directory added to the server classpath, permitting individual classes to be patched.
`config`	(Optionally) immutable server configuration files.
`config/audit-handlers`	Templates for configuring external Common Audit event handlers.
`config/config.ldif`	LDIF representation of current DS server configuration.
`config/keystore` `config/keystore.pin`	Keystore and password (`.pin`) files for servers using PKI based on a deployment ID and password.
`config/MakeLDIF`	Templates for use with the `makeldif` LDIF generation tool.
`db`	Default directory for backend database files. For details, see Data storage.
`extlib`	Directory for additional .jar files used by your custom plugins. If the instance path is not the same as the binaries, copy additional files into the `instance-path/extlib/` directory.
`import-tmp`	Working directory used when importing LDIF data.
`ldif`	Directory for saving LDIF export files.
`legal-notices`	License information.
`lib`	Scripts and libraries shipped with DS servers.
`lib/extensions`	Directory for custom plugins.
`locks`	Lock files that prevent more than one process from using the same backend.
`logs`	Access, errors, audit, and replication logs.

File or directory	Description
`logs/server.pid`	Contains the process ID for a running server.
`opendj_logo.png`	DS splash logo.
`README`	About DS servers.
`samples` [1]	Samples for use with DS servers, such as: • A sample `Dockerfile` and related files for building custom DS Docker images. • A sample Grafana dashboard demonstrating how to graph DS server metrics stored in a Prometheus database. • Sample server plugins and extensions.
`setup`	UNIX/Linux setup tool.
`setup.bat`	Windows setup tool.
`snmp`	SNMP support files.
`template`	Templates for setting up a server instance.
`template/setup-profiles`	Profile scripts to configure directory servers for specific use cases.
`upgrade`	UNIX/Linux upgrade tool.
`upgrade.bat`	Windows upgrade tool.
`var`	Files the DS server writes to during operation. Do not modify or move files in the `var` directory.
`var/archived-configs`	Snapshots of the main server configuration file, `config/config.ldif`. The server writes a compressed snapshot file when the configuration is changed.
`var/config.ldif.startok`	The most recent version of the main server configuration file that the server successfully started with.

[1] The samples are provided on an "as is" basis. ForgeRock does not guarantee the individual success developers may have in implementing the samples on their development platforms or in production configurations.

For details about how to try the samples, see the accompanying `README.md` files.

Upgrade

This guide shows you how to upgrade Directory Services software.

About Upgrades

Read this first.

Directory Server

Upgrade a directory server.

Directory Proxy

Upgrade a directory proxy server.

Replication Server

Upgrade a standalone replication server.

Read the Release notes before you upgrade DS software.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

The common REST API provides ForgeRock Identity Platform software common ways to access web resources and collections of resources.

About upgrades

DS 7 is a major release, much more cloud-friendly than ever before, and different in significant ways from earlier releases.

To upgrade successfully from DS 6.5 and earlier, make sure you understand the key differences beforehand. With these in mind, plan the upgrade, how you will test the upgraded version, and how you will recover if the upgrade process does not go as expected:

Fully compatible replication

Some things never change. The replication protocol remains fully compatible with earlier versions back to OpenDJ 3.

This means you can still upgrade servers while the directory service is online, but the process has changed.

Key configuration differences

DS 6.5 and earlier	DS 7.0 and later
You configure replication after installation.	You configure replication during installation, before starting the server.
You configure which servers replicate.	You configure bootstrap replication servers. Replicas discover other servers through them.
You configure trust and TLS when configuring replication.	By default, you install servers with a shared deployment ID and password that enables trust and TLS.
Before retiring a server, you unconfigure replication for the server.	After retiring a bootstrap replication server, you remove it from other servers' configurations. Otherwise, no unconfiguration is necessary.
Use the `dsreplication` command.	Use the `dsrepl` command.
Replicas share secret keys through `cn=admin data`.	Replicas protect secret keys with the shared deployment ID and password.

In 6.5 and earlier, you set up DS servers that did not yet replicate. Then, when enough of them were online, you configured replication.

In 7, you configure replication at setup time *before you start the server*. For servers that will have a changelog, you use the `setup --replicationPort` option for the replication server port. For all servers, you use the `setup --bootstrapReplicationServer` option to specify the replication servers that the server will contact when it starts up.

The bootstrap replication servers maintain information about the servers in the deployment. The servers learn about the other servers in the deployment by reading the information that the bootstrap replication server maintains. Replicas initiate replication when they contact the first bootstrap replication server.

As directory administrator, *you no longer have to configure and initiate replication* for a pure DS 7 deployment. DS 7 servers can start in any order as long as they initiate replication before taking updates from client applications.

Furthermore, *you no longer have to actively purge replicas you removed from other servers' configurations*. The other servers "forget" a replica that disappears for longer than the replication purge delay, meaning they eventually purge its state from memory and from their changelogs. (DS servers do not "forget" bootstrap replication servers, because each server's configuration explicitly references its bootstrap replication servers.) With earlier DS versions, you had to purge replicas from other servers' configurations after they were removed. DS servers do this automatically now. No administrative action is required.

These new capabilities bring you more deployment flexibility than ever before. As a trade off, you must now think about configuring replication at setup time, and you must migrate scripts and procedures that used older commands to the new `dsrepl` command.

Unique string-based server IDs

By default, DS 7 servers use unique string-based server IDs.

In prior releases, servers had multiple numeric server IDs. Before you add a new DS 7 server to a deployment of older servers, you must assign it a "numeric" server ID.

Secure by default

The `setup --production-mode` option is gone. All setup options and profiles are secure by default.

DS 7 servers require:

- Secure connections.
- Authentication for nearly all operations, denying most anonymous access by default.
- Additional access policies when you choose to grant access beyond what setup profiles define.
- Stronger passwords.

  New passwords must not match known compromised passwords from the default password dictionary. Also in 7, only secure password storage schemes are enabled by default, and reversible password storage schemes are deprecated.
- Permission to read log files.

Furthermore, DS 7 encrypts backup data by default. As a result of these changes, *all deployments* now require cryptographic keys.

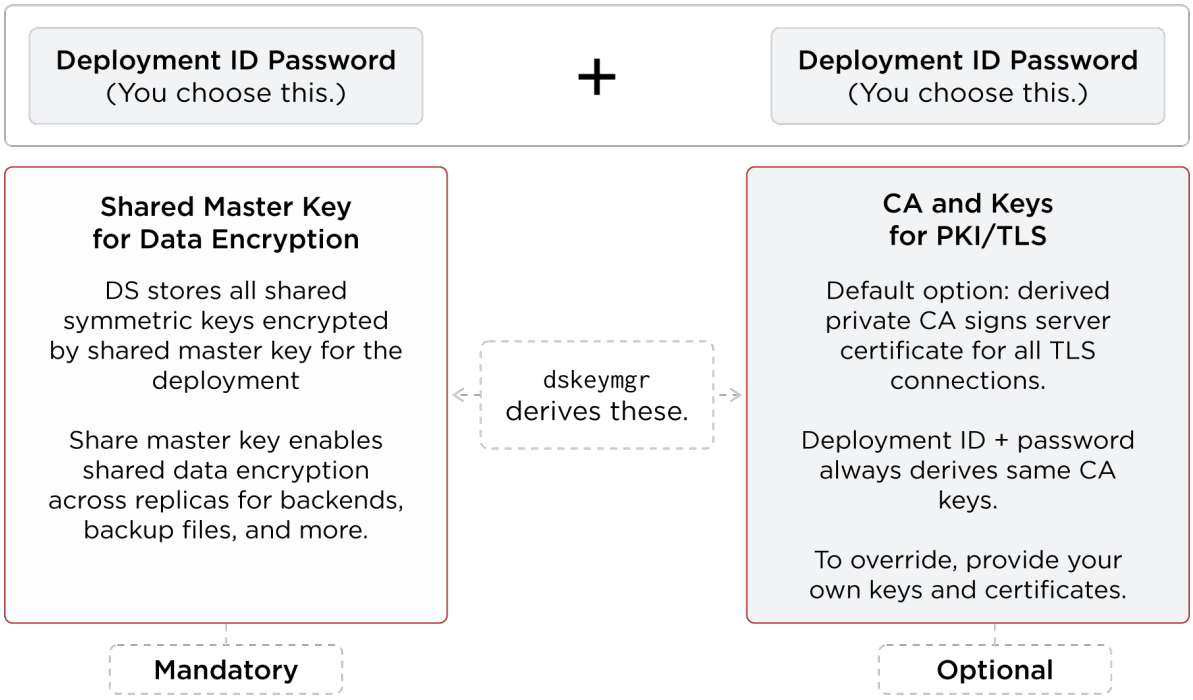
Deployment ID required

DS 7 deployments require cryptographic keys. Secure connections require asymmetric keys (public key certificates and associated private keys). Encryption requires symmetric (secret) keys that each replica shares.

To simplify key management and distribution, and especially to simplify disaster recovery, DS 7 uses a shared master key to protect secret keys. DS 7 stores the encrypted secret keys with the replicated and backed up data. This is new in DS 7, and replaces `cn=admin data` and the keys for that backend.

A deployment ID is a random string generated using the `dskeymgr` command. A deployment ID password is a secret string at least 8 characters long that you choose. The two are a pair. You must have a deployment ID's password to use the ID.

You generate a shared master key to protect encryption secrets, and optionally, asymmetric key pairs to protect communications, with the `dskeymgr` command using your deployment ID and password. Even if you provide your own asymmetric keys for securing connections, you must use the deployment ID and password to generate the shared master key.



When you upgrade, or add a DS 7 server to a deployment of pre-7 servers, you must intervene to move from the old model to the new, and unlock all the capabilities of DS 7.

New backup

As before, backups are not guaranteed to be compatible across major and minor server releases. If you must roll back from an unsuccessful upgrade, roll back the data as well as the software.

When you back up DS 7 data, the backup format is different. The new format *always* encrypts backup data. The new format allows you to back up and restore data directly in cloud storage if you choose.

Backup operations are now incremental by design. The initial backup operation copies all the data, incrementing from nothing to the current state. All subsequent operations back up data that has changed.

Restoring a backup no longer involves restoring files from the full backup archive, and then restoring files from each incremental backup archive. You restore any backup as a single operation.

The previous backup and restore tools are gone. In their place is a single `dsbackup` command for managing backup and restore operations, for verifying backup archives, and for purging outdated backup files.

For additional details, refer to the rest of the DS 7 documentation.

Important

To the extent possible, separate the upgrade process from the process of adopting new features. The DS `upgrade` command encourages this by maintaining compatibility where possible.
Once you have validated that the upgrade has completed successfully, take advantage of the new features available. Be sure to review the suggestions in After you upgrade.

Activate new features after upgrade

When you upgrade DS, the upgrade process preserves the existing configuration as much as possible. This maintains compatibility, but it means you do not have access to all new features immediately after upgrade.

You must take additional steps to complete the process, including activating new features. For details, refer to After you upgrade.

Supported upgrades

From…	To…	Important Notes
Official ForgeRock release, version 3.0 or later	Official ForgeRock release, same edition of directory server or replication server	Supported.
Official ForgeRock release, OEM edition, version 3.0 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, the OEM edition uses PDB backends for local data. This release removes support for PDB backend databases. The upgrade process only converts PDB backend configuration entries to JE backend configuration entries. It renames the PDB backend database directories, appending a `.bak` suffix, but does not change the format of the databases. *The PDB backend database content is no longer accessible after upgrade.* Backup files for PDB backend databases are also no longer usable after upgrade. You must export data from any PDB backend databases to LDIF before upgrading, and then import the data into the new JE backend databases after upgrade. For instructions on exporting and importing LDIF, refer to Import and export. After upgrading, configure backup tasks for the new JE backend databases as you had done previously for PDB backend databases.
Official ForgeRock release, version 2.6	Official ForgeRock release, directory server or replication server	Not supported. Workaround: First, upgrade all servers in the deployment to 6.5 before upgrading further. For details on upgrading to 6.5, refer to the *DS 6.5 Installation Guide*.
Official ForgeRock release, version 2.4 or 2.5	Official ForgeRock release, directory server or replication server	Not supported. Workaround: Upgrade all servers in the deployment to use at least 2.6.0 before upgrading further. For details on upgrading to that version, refer to *Upgrading to OpenDJ 2.6.0*.

From…	To…	Important Notes
Evaluation release, version 5.0 or later	Official ForgeRock release	Not supported. The evaluation version includes an additional server plugin and configuration. Official releases do not have an upgrade task to remove the plugin and its configuration.
Unofficial build, version 2.6.0 or later	Official ForgeRock release	Not supported.

Upgrade strategies

When you upgrade to a new DS version, you choose between in-place upgrade, unpacking the new software over old, then running the `upgrade` command, or upgrade by adding new servers and retiring old ones.

DS software provides an `upgrade` command to simplify the process of upgrading a server in place.

> **Important**
>
> For some scenarios, like upgrading Docker images in a Kubernetes deployment, in-place upgrade is the only kind that works.

Upgrade in place

The most straightforward option when upgrading DS servers is to upgrade in place. One by one, you stop, upgrade, and restart each server individually, leaving the service running during upgrade:

Advantages	Disadvantages
No additional systems to manage.	During upgrade, the host system must meet the requirements for both the older version and the new release. For example, you might need to have more than one Java environment installed. The operating system must also be supported for both releases.
Simpler to understand.	Slower to roll back. Rollback involves restoring each server to its pre-upgrade state. Once a replica's databases have been upgraded, they cannot be rolled back.
Easier to maintain compatibility. To the extent possible, the `upgrade` command leaves the configuration as is.	You must manually enable new features after upgrade.

On upgrading replicas

> **Important**
>
> The in-place 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.

When upgrading in place, follow these steps for each replica:

1. Direct client application traffic away from the server to upgrade.
2. Upgrade the replica.
3. Direct client application traffic back to the upgraded server.

Add new servers

Adding new servers and then retiring old ones is an alternative to upgrading in place. You replicate data between old and new systems, leaving the service running during upgrade:

Advantages	Disadvantages
Smoothly phase out old host systems. After successfully completing the upgrade, you gradually retire the old systems.	New host systems to manage.
Faster to roll back. Old servers remain in operation until upgrade completes successfully.	Harder to maintain compatibility. You must manually configure new servers to be fully compatible with existing servers, rather than relying on the `upgrade` command. This requires an in-depth understanding of both your existing configuration and the new configuration. Some new default settings may be incompatible with the old default settings, for example.
	Requires initializing the new replicas. Depending on the volume of data to synchronize, you can initialize at least the first new replica online. For deployments with medium to large data sets, initialize from exported LDIF, or from backup files created using an upgraded DS server. In either case, you must plan the operation.
	While the upgrade is in progress, replication monitoring is split between the older servers, which use `dsreplication status`, and the newer servers, which use `dsrepl status`. Run both commands to get a more complete picture of replication status.
	You must manually enable new features after upgrade.

Before you upgrade

Fulfill these requirements before upgrading Directory Services software, especially before upgrading the software in a production environment. Also refer to the requirements listed in release notes.

Supported Java

Important

- Always use a JVM 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, use one of the following solutions:
  - Edit the `default.java-home` setting in the `opendj/config/java.properties` file.
  - Set `OPENDJ_JAVA_HOME` to the path to the correct Java environment.
  - Set `OPENDJ_JAVA_BIN` to the absolute path of the `java` command.
- When running the `dskeymgr` and `setup` commands, use the same Java environment everywhere in the deployment.
  Due to a change in Java APIs, the same DS deployment ID generates different CA key pairs with Java 11 and Java 17.
  Using different Java versions is a problem if you use deployment ID-based CA certificates. Replication breaks, for example, when you use the `setup` command for a new server with a more recent version of Java than was used to set up existing servers.
  For details on resolving the issue, refer to Incompatible Java versions.

DS software supports the following Java environments:

Supported Java Versions

Vendor	Versions
OpenJDK, including OpenJDK-based distributions: • AdoptOpenJDK/Eclipse Temurin Java Development Kit (Adoptium) • Amazon Corretto • Azul Zulu • Red Hat OpenJDK ForgeRock tests most extensively with AdoptOpenJDK/Eclipse Temurin. ForgeRock recommends using the HotSpot JVM.	11[1], 17[2]
Oracle Java	11[1], 17[2]

[1] DS requires Java 11.0.6 or later. Earlier Java 11 updates lack required cryptography fixes. To use TLS 1.3 with PKCS#11, DS requires Java 11.0.8 or later. Use Java 11.0.12 or later for compatibility with third-party cryptographic tools.

[2] DS requires Java 17.0.3 or later. Earlier Java 17 updates lack required cryptography fixes.

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.

Back up first

Before upgrading, perform a full file system backup of the current server so that you can revert on failure. Make sure you stop the directory server and *back up the file system directory where the current server is installed*.

Backup archives are *not guaranteed to be compatible* across major and minor server releases. *Restore backups only on directory servers of the same major or minor version.*

Disable Windows service

If you are upgrading a server registered as a Windows service, disable the Windows service before upgrade:

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

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

When adding new servers

When upgrading by adding new DS 7 and later servers to a DS 6.5 or earlier deployment, add the new directory servers or replication servers as described on this page.

If all the servers are DS 7 or later and use the new security model based on deployment IDs, not `cn=admin data`*, then skip these instructions. Install the new servers as described in the pages on Installation, and rebuild indexes as necessary.*

Important

- Set up replication before upgrade. Do not set up replication for the first time between servers of different versions.
- The new server you add must first connect to an existing replica that is a directory server, not a standalone replication server.
- Newer directory servers update LDAP schema definitions to add support for new features. The newer schema definitions are not all compatible with older servers.

1. Install and set up a new server, but do not start it, yet.

   Because replication is now configured at setup time, you may need to create the new server with some specific arguments. The following table indicates which arguments are needed for which kind of server:

New server is a...	Use this replication setup option
Combined DS/RS	`--replicationPort port`
Standalone DS	N/A
Standalone RS	`--replicationPort port`

- Do not use the `setup --bootstrapReplicationServer` option. In a later step of this procedure, you will use the `dsrepl add-local-server-to-pre-7-0-topology` command. That command configures the bootstrap replication server settings for the new server based on the existing deployment.
- Do not use the `setup --start` option. In a later step of this procedure, you will start the server.

For details about setup options, refer to Setup hints, and many of the examples that use the `setup` command.

2. Configure the new server settings to be compatible with the settings of the existing servers.

   Examples of incompatible default settings include:

   - Password storage schemes not available in earlier versions.
   - String-based server IDs. Server IDs were limited to numbers between 1 and 65535.

     Remove leading `0` (zero) characters when setting a numeric server ID. DS servers classify a server ID with a leading `0` as a string, not a number.
   - String-based group IDs. Group IDs were also limited to numbers.
   - TLS protocols and cipher suites.

   For changes in the release, refer to Incompatible changes. If the existing servers run a release older than 6.5, refer to similar pages in the previous release notes.

3. Configure the new server as a replica of an existing server that is a directory server, and not a standalone replication server:

```
$ dsrepl \
 add-local-server-to-pre-7-0-topology \
 --hostname pre-7-ds.example.com \
 --port 4444 \
 --bindDn "cn=admin,cn=Administrators,cn=admin data" \
 --bindPassword password \
 --baseDn dc=example,dc=com \
 --trustAll \
 --no-prompt
```

   The existing server in this example is a directory server, as suggested by the `ds` in the hostname. The `dsrepl add-local-server-to-pre-7-0-topology` command does not support connecting to a standalone replication server.

   The command configures the new server, discovering the replication servers in the deployment, and setting the bootstrap replication servers.

The command also generates one or more `dsrepl initialize` commands. Copy those commands, and add required credentials for use when initializing the new server.

In the example command shown here:

- The `--bindDn` and `--bindPassword` options reflect either the UID and password of the existing servers' global replication administrator, or the DN and password of any user with sufficient access to act as global administrator on all servers.
- The insecure `--trustAll` option is used to simplify this procedure.

  To avoid using this option, add the remote server's CA or signing certificate to the new server's keystore, and use the appropriate keystore options.

4. Start the new server.
5. Initialize the new server with the `dsrepl initialize` command(s) from the previous step:

New server is a...	Initialize these base DNs
Combined DS/RS	`cn=admin data`, `cn=schema`, all directory data DNs
Standalone DS	`cn=admin data`, `cn=schema`, all directory data DNs
Standalone RS	`cn=admin data`

6. Rebuild indexes as necessary for changes to schema definitions for the RFC 2703bis Internet-Draft, *An Approach for Using LDAP as a Network Information Service*.

   The definitions for DS 7.2 and later align with those of the latest Internet-Draft. The change does not affect the data, but does affect indexes for the following attributes:

   - `automountInformation`
   - `automountKey`
   - `automountMapName`
   - `gecos`
   - `ipHostNumber`
   - `ipNetworkNumber`
   - `memberNisNetGroup`
   - `memberUid`
   - `nisMapEntry`
   - `nisNetgroupTriple`

Use the new schema	Use the old schema
When you add a new server, it replicates the new schema. Rebuild "degraded" indexes on existing, older servers. For details, on rebuilding degraded indexes, refer to Automate index rebuilds.	Before upgrading, save a copy of the schema file, `db/schema/04-rfc2307bis.ldif`, from an existing server. After upgrading: ◦ Replace the newer schema file with your saved copy. ◦ Rebuild "degraded" indexes on the newer servers.

Directory server

This page shows how to upgrade a directory server in place.

If you are adding a new server to an existing deployment, see When adding new servers instead.

Important

Before upgrading, make sure you stop the server. Once you have unpacked the new server files, do not modify the server configuration until after you have completed the upgrade process.
Failure to follow the upgrade instructions can result in the loss of all user data.

1. Prepare for upgrade as described in Before you upgrade.
2. Stop the server.
3. Proceed to upgrade the server:
    1. When upgrading a server installed from the cross-platform ZIP distribution:
        - Unpack the new files over the old files as described in Unpack files.
        - Run the upgrade command to bring the server 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.
    2. When upgrading a server installed from native packages, use the system package management tools.
4. When the mutable data mounted at runtime differs from that of the instance where you first run the `upgrade` command, upgrade only mutable data by running the command again with the `--dataOnly` option at runtime.

   The `--dataOnly` option can be useful when running the server in a Docker container, for example.

This improvement is available when upgrading from DS 6.0.0 or later releases.

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.

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

   When you upgrade from OpenDJ 3.0, the upgrade process leaves the HTTP connection handler disabled.

   The newer REST to LDAP configuration is not necessarily compatible with the previous configuration. You must rewrite your configuration according to REST to LDAP reference, and then configure the server to use the new configuration.

6. If you disabled the Windows service to upgrade, enable it again:

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

Directory proxy

This page shows how to upgrade a directory proxy server in place. A directory proxy server has no local user data.

> **Important**
>
> Before upgrading, make sure you stop the server. Once you have unpacked the new server files, do not modify the server configuration until after you have completed the upgrade process.

1. Prepare for upgrade as described in Before you upgrade.
2. Stop the server.
3. Proceed to upgrade the server:
   1. When upgrading a server installed from the cross-platform ZIP distribution:
      - Unpack the new files over the old files as described in Unpack files.
      - Run the upgrade command to bring the server 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.
   2. 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.

5. If you disabled the Windows service to upgrade, enable it again:

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

Replication server

This page shows how to upgrade a standalone replication server in place. A standalone replication server has no local user data. If the server holds user data, see Directory server instead.

If you are adding a new server to an existing deployment, see When adding new servers instead.

> **Important**
>
> Before upgrading, make sure you stop the server. Once you have unpacked the new server files, do not modify the server configuration until after you have completed the upgrade process.

1. Prepare for upgrade as described in Before you upgrade.
2. Stop the server.
3. Proceed to upgrade the server:
   1. When upgrading a server installed from the cross-platform ZIP distribution:
      - Unpack the new files over the old files as described in Unpack files.
      - Run the upgrade command to bring the server 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.
   2. 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.

5. If you disabled the Windows service to upgrade, enable it again:

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

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.

DSML gateway

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

After you upgrade

The DS server upgrade process preserves the existing configuration as much as possible. This maintains compatibility, but there are additional steps you must take.

Checklist

Use this checklist to make sure you don't miss these important post-upgrade tasks:

Task	Done?
Back up your directory data. Backup files are *not* compatible between versions.	☐
Update your scripts to account for incompatible changes.	☐
Plan your move away from deprecated features.	☐
Move to dedicated service accounts for your directory applications. You would not run all your UNIX applications as root, or all your Windows applications as Administrator. Stop using administrator accounts like `cn=Directory Manager` as service accounts. Many DS setup profiles create service accounts for applications to use when authenticating to DS. For examples of AM service accounts, see the `base-entries.ldif` files in setup profiles under the `opendj/template/setup-profiles/AM` directory.	☐
Manually review and purge the DS server configurations for stale references to old servers. You can read the `opendj/config/config.ldif` file to find stale references, but always use the dsconfig command to make changes to the configuration.	☐
After you upgrade by adding new servers, but before you retire old servers, update bootstrap replication server settings to remove the old servers, and add the new, DS 7 servers.	☐

Task	Done?
Review what's new and changed in the intervening releases to identify useful changes, and take advantage of new features and improvements.	☐
Use the new security model (in-place upgrades only, required).	☐
Eliminate outdated password storage	☐
Clean up admin data (required for upgrades from DS 6.5 and earlier). Before starting this task, complete the work to Use the new security model.	☐
Add a monitor user account.	☐
Update LDAP schema.	☐
Tune settings.	☐
Use string-based server IDs.	☐
Use the entity tag plugin for ETags.	☐
Activate cloud storage for backup (in-place upgrades only, optional).	☐

Note

Many example commands in this page use `cn=Directory Manager` as the name of the directory superuser account. This was the default before DS 7.
Here, `cn=Directory Manager` stands for the name of the directory superuser account in DS 6.5 and earlier.

Use the new security model

Note

If you have upgraded DS 6.5 or earlier servers in place, enable upgraded servers to use the new security model. *You cannot add new servers using normal procedures until you have completed these steps*. Some new features depend on the new model.
If you started by adding DS 7 or later servers, as described in When adding new servers, then the new DS servers already have the keys. In that case, you can skip these steps.

DS release 7 changes the security model to let you configure replication at setup time, to make disaster recovery more straightforward, and to simplify symmetric key distribution:

- In prior releases, trust and symmetric key distribution in a replication topology depends on the replicated `cn=admin data` base DN. DS servers prior to release 7 reference each others' *instance keys*, and use them to protect symmetric keys in `cn=admin data` entries.

- DS servers now rely on a deployment ID and password to derive a shared master key, and provide a default PKI to trust each others' certificates. DS servers protect symmetric keys using the shared master key to encrypt and decrypt them. For details, see Deployment IDs.

The following examples demonstrate the process of creating keys and updating the configuration for replicas installed with the DS 6.5 evaluation profile:

1. Make sure you have upgraded all DS servers to version 7 or later.
2. Generate a deployment ID for the deployment:

```
###
# Generate a deployment ID for the topology.
# Do this once and SAVE THE DEPLOYMENT ID:
$ dskeymgr create-deployment-id --deploymentIdPassword password
<deployment-id>
```

   For more command options, refer to dskeymgr. The default validity for the deployment ID is 10 years.

3. For each upgraded server, add at least the shared master key generated using the deployment ID:

```
\###
# Use the same deployment ID on each server:
export DEPLOYMENT_ID=<deployment-id>

# Add a shared master key based on the deployment ID:
dskeymgr \
 export-master-key-pair \
 --alias master-key \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --keyStoreFile /path/to/opendj/config/keystore \
 --keyStorePassword:file /path/to/opendj/config/keystore.pin

# Deployment ID-based PKI?
# Add a deployment ID CA certificate:
dskeymgr \
 export-ca-cert \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --keyStoreFile /path/to/opendj/config/keystore \
 --keyStorePassword:file /path/to/opendj/config/keystore.pin

# Deployment ID-based PKI?
# Add a deployment ID-based TLS certificate:
dskeymgr \
 create-tls-key-pair \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --keyStoreFile /path/to/opendj/config/keystore \
 --keyStorePassword:file /path/to/opendj/config/keystore.pin \
 --hostname localhost \
 --hostname opendj.example.com \
 --subjectDn CN=DS,O=ForgeRock
```

The default validity for the certificate is one year.

4. For each upgraded server, start the server, if necessary.
5. For each upgraded server, update the configuration to use the new keys.

   The following example uses the private PKI keys based on the deployment ID and password. At minimum, even if you use your own keys for PKI, update the Crypto Manager to use the shared master key:

```
# Copy the keys used to protect secret keys and replication traffic
# to the default key manager keystore.
# This makes the keys available for trust and decryption
# after you switch to the default key and trust managers:
keytool \
 -importkeystore \
 -srckeystore /path/to/opendj/db/ads-truststore/ads-truststore \
 -srcstorepass:file /path/to/opendj/db/ads-truststore/ads-truststore.pin \
 -destkeystore /path/to/opendj/config/keystore \
 -deststoretype PKCS12 \
 -deststorepass:file /path/to/opendj/config/keystore.pin

# Configure the server to wrap new secret keys
# using the new shared master key:
dsconfig \
 set-crypto-manager-prop \
 --set key-manager-provider:"Default Key Manager" \
 --set master-key-alias:master-key \
 --reset digest-algorithm \
 --reset mac-algorithm \
 --reset key-wrapping-transformation \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --trustAll \
 --no-prompt

# Deployment ID-based PKI?
dsconfig \
 create-trust-manager-provider \
 --set enabled:true \
 --set trust-store-file:config/keystore \
 --set trust-store-pin:\&{file:config/keystore.pin} \
 --set trust-store-type:PKCS12 \
 --type file-based \
 --provider-name PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

# Switch to the new keys to secure
# administrative and replication communications:
dsconfig \
 set-administration-connector-prop \
 --set ssl-cert-nickname:ssl-key-pair \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 set-synchronization-provider-prop \
 --provider-name "Multimaster Synchronization" \
```

```
 --set key-manager-provider:"Default Key Manager" \
 --set ssl-cert-nickname:ssl-key-pair \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

# Switch to the new keys for other secure communications:
dsconfig \
 set-connection-handler-prop \
 --handler-name HTTPS \
 --set ssl-cert-nickname:ssl-key-pair \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 set-connection-handler-prop \
 --handler-name LDAP \
 --set ssl-cert-nickname:ssl-key-pair \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 set-connection-handler-prop \
 --handler-name LDAPS \
 --set ssl-cert-nickname:ssl-key-pair \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt
```

6. For each upgraded server, restart the server, causing it to generate new secret keys, wrapped using the shared master key:

```
stop-ds --restart
```

Eliminate outdated password storage

Reversible password storage schemes (3DES, AES, Blowfish, RC4) are deprecated since DS 7.0. Many password storage schemes are no longer enabled by default for new installations.

After upgrading to DS 7 and later, migrate active accounts away from the following deprecated and outdated password storage schemes:

- 3DES
- AES
- Base64
- Blowfish
- CRYPT
- Clear
- PBKDF2
- PKCS5S2
- SHA-1
- Salted SHA-1
- Salted SHA-256
- Salted SHA-384
- Salted SHA-512

For instructions on migrating accounts away from outdated storage schemes, refer to How do I change a password storage scheme and apply a new password policy to users in PingDS?

Clean up admin data

These steps are required after you upgrade from DS 6.5 and earlier to ensure servers share secret keys according to the new security model. You must follow the upgrade procedures to add new DS 7 servers until you have completed these steps.

> **Important**
>
> If, after cleanup, your deployment still stores secret keys under the replicated `cn=admin data` base DN, *do not disable* `cn=admin data` *or remove the* `adminRoot` *database*.
> This applies, for example, to deployments that use (deprecated) reversible password storage schemes (3DES, AES, Blowfish, RC4). It also applies to deployments where servers were set up in production mode, and use keys with automatically generated, self-signed certificates to protect replication connections.
> If you do choose to disable `cn=admin data` and remove the `adminRoot` database, you must first *manually* ensure that admin data is no longer used, and then remove references to it from your configuration.

1. Make sure you have upgraded all DS servers to version 7 or later.

   If you upgraded by adding new servers, and still have DS 6.5 or earlier servers, retire them before continuing.

   As explained in [overview] at the top of this page, this means purging stale references to retired servers from the new servers' configurations, and updating bootstrap replication server settings to reference only the new, DS 7 servers.

2. If you upgraded in place, make sure you have followed the steps in Use the new security model.

3. Run the cleanup command.

   For example, run the cleanup command on each server with directory superuser credentials. If the credentials are the same on every server, it is sufficient to run the command once:

   After upgrade in place

   ```
   $ dsrepl \
    cleanup-migrated-pre-7-0-topology \
    --bindDn "cn=Directory Manager" \
    --bindPassword password \
    --hostname localhost \
    --port 4444 \
    --trustAll \
    --no-prompt
   ```

   After adding new servers

   ```
   $ dsrepl \
    cleanup-migrated-pre-7-0-topology \
    --bindDn uid=admin \
    --bindPassword password \
    --hostname localhost \
    --port 4444 \
    --trustAll \
    --no-prompt
   ```

   The command is idempotent. You can run it multiple times if the initial run cannot fully complete the cleanup process.

4. Remove unused configuration settings:

After upgrade in place

```
dsconfig \
 delete-key-manager-provider \
 --provider-name "Crypto Manager Key Manager" \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 delete-key-manager-provider \
 --provider-name "Replication Key Manager" \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 delete-trust-manager-provider \
 --provider-name "Replication Trust Manager" \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

# Skip this command if the deployment has passwords stored
# with reversible password storage schemes:
dsconfig \
 delete-backend \
 --backend-name adminRoot \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt
```

After adding new servers

```
dsconfig \
 delete-key-manager-provider \
 --provider-name "Crypto Manager Key Manager" \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 delete-key-manager-provider \
 --provider-name "Replication Key Manager" \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 delete-trust-manager-provider \
 --provider-name "Replication Trust Manager" \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --trustAll \
 --bindPassword password \
 --no-prompt

# Skip this command if the deployment has passwords stored
# with reversible password storage schemes:
dsconfig \
 delete-backend \
 --backend-name adminRoot \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --trustAll \
 --bindPassword password \
 --no-prompt
```

5. Replace references to `Admin Data` in the server configuration.

   Find all references to admin data in your configuration:

   ```
   $ grep -i "admin data" /path/to/opendj/config/config.ldif
   ```

   How you replace or remove these references depends on your deployment.

6. Remove unused files:

```
# Skip these commands if the deployment has passwords stored
# with reversible password storage schemes:
rm -rf /path/to/opendj/db/adminRoot
rm -rf /path/to/opendj/db/ads-truststore
```

Note

After cleanup, the `dsrepl status` command can show `cn=admin data` status even if you have removed the backend.

Add a monitor user account

The `dsrepl status` command, and general server monitoring require an account with the `monitor-read` privilege. Since DS 6, you can create a monitor user account at setup time. However, the setup process does not *require* that you create such an account, and earlier versions do not offer the option.

If no such account exists, do one of the following:

- Add the `monitor-read` privilege to an existing, replicated user entry, as demonstrated in Monitor privilege.
- Add a separate, replicated monitor user account, as demonstrated in How do I create a dedicated user for monitoring in PingDS?

Use this replicated account when monitoring DS servers, and when running the `dsrepl status` command.

Update LDAP schema

Update LDAP schema definitions to support new features.

When you upgrade servers, the servers inherit existing LDAP schema definitions. This ensures compatibility between the newer and older servers during upgrade. However, upgrade does not apply changes that new features depend on.

Once all servers run the latest software, add LDAP schema definitions required to use additional features:

1. Make sure you have upgraded all DS servers to version 7 or later.
2. Compare current schema definitions with the schema templates.

   The following example summarizes the differences for a new server added to a 6.5 deployment:

   ```
   $ cd /path/to/opendj
   $ diff -q db/schema template/db/schema
   Files db/schema/00-core.ldif and template/db/schema/00-core.ldif differ
   Files db/schema/03-pwpolicyextension.ldif and template/db/schema/03-pwpolicyextension.ldif differ
   Files db/schema/04-rfc2307bis.ldif and template/db/schema/04-rfc2307bis.ldif differ
   Only in db/schema: 60-ds-evaluation-schema.ldif
   Only in db/schema: 99-user.ldif
   ```

   The following table summarizes the changes in detail:

Schema File	Notes	Action
`00-core.ldif`	Cosmetic changes due to schema replication: ◦ Each definition in `db/schema/00-core.ldif` has `X-SCHEMA-FILE '00-core.ldif'`. No definitions in `template/db/schema/00-core.ldif` have the `X-SCHEMA-FILE` extension. ◦ Some object classes in `db/schema/00-core.ldif` are explicitly defined as `STRUCTURAL`. Other minor differences: ◦ In 7, some attribute definitions have minimum upper bounds. ◦ The schema for collective attributes is extended.	**Replace** with template file
`03-pwpolicyextension.ldif`	The new version was rewritten to support fully featured replicated password policies.	**Replace** with template file
`04-rfc2307bis.ldif`	In DS 7.2 and later, the new version aligns schema definitions with those of the latest RFC 2703bis Internet-Draft, *An Approach for Using LDAP as a Network Information Service*.	**Replace** with template file
`60-ds-evaluation-schema.ldif`	Added to existing version by the evaluation setup profile.	**Keep** existing file
`99-user.ldif`	Contains replication metadata.	**Keep** existing file
Any schema file missing in `template/db/schema`	This includes schema from setup profiles, and any custom schema definitions for the deployment.	**Keep** existing file

3. For each upgraded server, update the schema to the latest version.

   The following example updates the schema on a single server. Always stop a server before making changes to its files:

   ```
   $ cd /path/to/opendj
   $ ./bin/stop-ds
   $ cp template/db/schema/00-core.ldif db/schema
   $ cp template/db/schema/03-pwpolicyextension.ldif db/schema
   $ cp template/db/schema/04-rfc2307bis.ldif db/schema
   $ ./bin/start-ds
   ```

4. Rebuild indexes, if they exist, for the following attributes, which DS considers degraded:

   ◦ `automountInformation`

   ◦ `automountKey`

   ◦ `automountMapName`

   ◦ `gecos`

- `ipHostNumber`
- `ipNetworkNumber`
- `memberNisNetGroup`
- `memberUid`
- `nisMapEntry`
- `nisNetgroupTriple`

For details, see Automate index rebuilds.

Tune settings

Major software releases include significant changes that can render existing tuning settings obsolete. When upgrading to a new major release of DS or Java software, revisit the system configuration, server configuration, and Java settings. Adjust the settings appropriately for your deployment as part of the upgrade process.

For information and suggestions on tuning, read the Release notes and Performance tuning.

Use string-based server IDs

After upgrading from earlier releases, you can change server IDs to strings:

1. Make sure you have upgraded all DS servers to version 7 or later.
2. For each server, change the global server ID to the desired string.

   The following example shows a command that changes a server's global ID to a string:

   ```
   $ dsconfig \
    set-global-configuration-prop \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --set server-id:ds-us-west-1 \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

3. Restart the server for the change to take effect.

Use the entity tag plugin for ETags

The ETag plugin generates ETag attribute values more efficiently. For compatibility, the plugin is configured by default only on new servers.

After upgrading all servers, you can configure the plugin manually on each server:

1. Make sure you have upgraded all DS servers.

2. For each server, configure the plugin:

```
$ dsconfig \
 create-plugin \
 --plugin-name "Entity Tag" \
 --type entity-tag \
 --set enabled:true \
 --set invoke-for-internal-operations:true \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The plugin generates real ETag attributes for new and updated entries.

Activate cloud storage for backup

When upgrading in place from DS 6.5.x and earlier, the upgrade process does not unpack the libraries required to store backup files in the cloud, and does not configure the plugin used for the feature. You must activate cloud storage for backup if you want to use the feature.

> **Note**
>
> If you started by adding DS 7 or later servers, as described in When adding new servers, then the new DS servers already have the libraries and plugin configuration. In that case, you can skip these steps.

1. Unpack the libraries required to store backup files in the cloud:

```
$ cd /path/to/opendj
$ unzip -o -q extensions/backup-cloud-extension.zip
```

1. Restart DS.

2. Configure the cloud storage plugin to use the libraries:

```
$ dsconfig \
create-plugin \
--plugin-name Cloud\ Storage\ Plugin \
--type custom \
--set enabled:true \
--set java-class:com.forgerock.opendj.server.backup.cloud.CloudStoragePlugin \
--set plugin-type:initialization \
--hostname localhost \
--port 4444 \
--bindDn "cn=Directory Manager" \
--trustAll \
--no-prompt
```

Configuration

This guide shows you how to configure DS server features.

HTTP

Access DS over HTTP.

LDAP

Access DS over LDAP.

Storage

Manage DS data.

Indexes

Index DS data.

Replication

Replicate DS data.

LDAP Proxy

Configure proxy features.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

The common REST API provides ForgeRock Identity Platform software common ways to access web resources and collections of resources.

HTTP access

Set the HTTP port

The following steps demonstrate how to set up an HTTP port if none was configured at setup time with the `--httpPort` option:

1. Create an HTTP connection handler:

```
$ dsconfig \
 create-connection-handler \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name HTTP \
 --type http \
 --set enabled:true \
 --set listen-port:8080 \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

2. Enable an HTTP access log.

    1. The following command enables JSON-based HTTP access logging:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

    2. The following command enables HTTP access logging:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

3. After you set up an HTTP port, enable an HTTP endpoint.

   For details, see Configure HTTP user APIs or Use administrative APIs.

Set the HTTPS port

At setup time use the `--httpsPort` option.

Later, follow these steps to set up an HTTPS port:

1. Create an HTTPS connection handler.

   The following example sets the port to `8443` and uses the default server certificate:

   ```
   $ dsconfig \
    create-connection-handler \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --handler-name HTTPS \
    --type http \
    --set enabled:true \
    --set listen-port:8443 \
    --set use-ssl:true \
    --set key-manager-provider:PKCS12 \
    --set trust-manager-provider:"JVM Trust Manager" \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

   If the key manager provider has multiple key pairs that DS could use for TLS, where the secret key was generated with the same key algorithm, such as `EC` or `RSA`, you can specify which key pairs to use with the `--set ssl-cert-nickname:server-cert` option. The *server-cert* is the certificate alias of the key pair. This option is not necessary if there is only one server key pair, or if each secret key was generated with a different key algorithm.

2. Enable the HTTP access log.

   1. The following command enables JSON-based HTTP access logging:

      ```
      $ dsconfig \
       set-log-publisher-prop \
       --hostname localhost \
       --port 4444 \
       --bindDN uid=admin \
       --bindPassword password \
       --publisher-name "Json File-Based HTTP Access Logger" \
       --set enabled:true \
       --no-prompt \
       --usePkcs12TrustStore /path/to/opendj/config/keystore \
       --trustStorePassword:file /path/to/opendj/config/keystore.pin
      ```

   2. The following command enables HTTP access logging:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

3. If the deployment requires SSL client authentication, set the properties `ssl-client-auth-policy` and `trust-manager-provider` appropriately.

4. After you set up an HTTPS port, enable an HTTP endpoint.

   For details, see Configure HTTP user APIs, or Use administrative APIs.

Configure HTTP user APIs

The way directory data appears to client applications is configurable. You configure a Rest2ldap endpoint for each HTTP API to user data.

A Rest2ldap mapping file defines how JSON resources map to LDAP entries. The default mapping file is `/path/to/opendj/config/rest2ldap/endpoints/api/example-v1.json`. The default mapping works with Example.com data from the evaluation setup profile.

Edit or add mapping files for your own APIs. For details, see REST to LDAP reference.

If you have set up a directory server with the `ds-evaluation` profile, you can skip the first two steps:

1. If necessary, change the properties of the default Rest2ldap endpoint, or create a new endpoint.

   The default Rest2ldap HTTP endpoint is named `/api` after its `base-path`. The `base-path` must be the same as the name, and is read-only after creation. By default, the `/api` endpoint requires authentication.

   The following example enables the default `/api` endpoint:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --endpoint-name /api \
 --set authorization-mechanism:"HTTP Basic" \
 --set config-directory:config/rest2ldap/endpoints/api \
 --set enabled:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

Alternatively, you can create another Rest2ldap endpoint to expose a different HTTP API, or to publish data under an alternative base path, such as `/rest`:

```
$ dsconfig \
 create-http-endpoint \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --endpoint-name /rest \
 --type rest2ldap-endpoint \
 --set authorization-mechanism:"HTTP Basic" \
 --set config-directory:config/rest2ldap/endpoints/api \
 --set enabled:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

2. If necessary, adjust the endpoint configuration to use an alternative HTTP authorization mechanism.

   By default, the Rest2ldap endpoint maps HTTP Basic authentication to LDAP authentication. Change the `authorization-mechanism` setting as necessary. For details, see Configure HTTP Authorization.

3. Try reading a resource.

   The following example generates the CA certificate in PEM format from the server deployment ID and password. It uses the CA certificate to trust the server certificate. A CA certificate is only necessary if the CA is not well-known:

```
$ dskeymgr \
 export-ca-cert \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --outputFile ca-cert.pem

$ curl \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 https://localhost:8443/api/users/bjensen?_prettyPrint=true

{
  "_id" : "bjensen",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "userName" : "bjensen@example.com",
  "displayName" : [ "Barbara Jensen", "Babs Jensen" ],
  "name" : {
    "givenName" : "Barbara",
    "familyName" : "Jensen"
  },
  "description" : "Original description",
  "manager" : {
    "_id" : "trigden",
    "_rev" : "<revision>"
  },
  "groups" : [ {
    "_id" : "Carpoolers",
    "_rev" : "<revision>"
  } ],
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1862",
    "emailAddress" : "bjensen@example.com"
  },
  "uidNumber" : 1076,
  "gidNumber" : 1000,
  "homeDirectory" : "/home/bjensen"
}
```

Configure HTTP authorization

HTTP authorization mechanisms map HTTP credentials to LDAP credentials.

Multiple HTTP authorization mechanisms can be enabled simultaneously. You can assign different mechanisms to Rest2ldap endpoints and to the Admin endpoint.

By default, these HTTP authorization mechanisms are supported:

HTTP Anonymous

Process anonymous HTTP requests, optionally binding with a specified DN.

If no bind DN is specified (default), anonymous LDAP requests are used.

This mechanism is enabled by default.

HTTP Basic (enabled by default)

Process HTTP Basic authentication requests by mapping the HTTP Basic identity to a user's directory account.

By default, the exact match identity mapper with its default configuration is used to map the HTTP Basic user name to an LDAP `uid`. The DS server then searches in all local public naming contexts to find the user's entry based in the `uid` value. For details, see Identity Mappers.

HTTP OAuth2 CTS

Process OAuth 2.0 requests as a resource server, acting as an AM Core Token Service (CTS) store.

When the client bearing an OAuth 2.0 access token presents the token to access the JSON resource, the server tries to resolve the access token against the CTS data that it serves for AM. If the access token resolves correctly (is found in the CTS data and has not expired), the DS server extracts the user identity and OAuth 2.0 scopes. If the required scopes are present and the token is valid, it maps the user identity to a directory account.

This mechanism makes it possible to resolve access tokens by making an internal request, avoiding a request to AM. *This mechanism does not ensure that the token requested will have already been replicated to the replica where the request is routed.*

AM's CTS store is constrained to a specific layout. The `authzid-json-pointer` must use `userName/0` for the user identifier.

HTTP OAuth2 OpenAM

Process OAuth 2.0 requests as a resource server, sending requests to AM for access token resolution.

When the client bearing an OAuth 2.0 access token presents the token to access the JSON resource, the directory service requests token information from AM. If the access token is valid, the DS server extracts the user identity and OAuth 2.0 scopes. If the required scopes are present, it maps the user identity to a directory account.

Access token resolution requests should be sent over HTTPS. You can configure a truststore manager if necessary to trust the authorization server certificate, and a keystore manager to obtain the DS server certificate if the authorization server requires mutual authentication.

HTTP OAuth2 Token Introspection (RFC7662)

Handle OAuth 2.0 requests as a resource server, sending requests to an RFC 7662-compliant authorization server for access token resolution.

The DS server must be registered as a client of the authorization server.

When the client bearing an OAuth 2.0 access token presents the token to access the JSON resource, the DS server requests token introspection from the authorization server. If the access token is valid, the DS server extracts the user identity and OAuth 2.0 scopes. If the required scopes are present, it maps the user identity to a directory account.

Access token resolution requests should be sent over HTTPS. You can configure a truststore manager if necessary to trust the authorization server certificate, and a keystore manager to obtain the DS server certificate if the authorization server requires mutual authentication.

> **Note**
>
> The HTTP OAuth2 File mechanism is an internal interface intended for testing, and not supported for production use.

When more than one authentication mechanism is specified, mechanisms are applied in the following order:

- If the client request has an `Authorization` header, and an OAuth 2.0 mechanism is specified, the server attempts to apply the OAuth 2.0 mechanism.
- If the client request has an `Authorization` header, or has the custom credentials headers specified in the configuration, and an HTTP Basic mechanism is specified, the server attempts to apply the Basic Auth mechanism.
- Otherwise, if an HTTP anonymous mechanism is specified, and none of the previous mechanisms apply, the server attempts to apply the mechanism for anonymous HTTP requests.

There are many possibilities when configuring HTTP authorization mechanisms. *This procedure shows only one OAuth 2.0 example.*

The example below uses settings as listed in the following table. When using secure connections, make sure the servers can trust each other's certificates. Download ForgeRock Access Management software from the Backstage download site:

Setting	Value
OpenAM URL	`https://am.example.com:8443/openam` (When using HTTPS, make sure DS can trust the AM certificate.)
Authorization server endpoint	`/oauth2/tokeninfo` (top-level realm)
Identity repository	DS server configured by the examples that follow.
OAuth 2.0 client ID	`myClientID`
OAuth 2.0 client secret	`password`
OAuth 2.0 client scopes	`read`, `uid`, `write`
Rest2ldap configuration	Default settings. See Configure HTTP User APIs.

Read the ForgeRock Access Management documentation if necessary to install and configure AM. Then follow these steps to try the demonstration:

1. Update the default HTTP OAuth2 OpenAM configuration:

```
$ dsconfig \
 set-http-authorization-mechanism-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --mechanism-name "HTTP OAuth2 OpenAM" \
 --set enabled:true \
 --set token-info-url:https://am.example.com:8443/openam/oauth2/tokeninfo \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

2. Update the default Rest2ldap endpoint configuration to use HTTP OAuth2 OpenAM as the authorization mechanism:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --endpoint-name "/api" \
 --set authorization-mechanism:"HTTP OAuth2 OpenAM" \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

3. Obtain an access token with the appropriate scopes:

```
$ curl \
 --request POST \
 --user "myClientID:password" \
 --data "grant_type=password&username=bjensen&password=hifalutin&scope=read%20uid%20write" \
 https://am.example.com:8443/openam/oauth2/access_token
{
 "access_token": "token-string",
 "scope": "uid read write",
 "token_type": "Bearer",
 "expires_in": 3599
}
```

   Use HTTPS when obtaining access tokens.

4. Request a resource at the Rest2ldap endpoint using HTTP Bearer authentication with the access token:

```
$ curl \
 --header "Authorization: Bearer token-string" \
 --cacert ca-cert.pem \
 https://localhost:8443/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"
  }
}
```

Use HTTPS when presenting access tokens.

Use administrative APIs

The APIs for configuring and monitoring DS servers are under the following endpoints:

`/admin/config`

Provides a REST API to the server configuration under `cn=config`.

By default, this endpoint is protected by the HTTP Basic authorization mechanism. Users reading and editing the configuration must have appropriate privileges, such as `config-read` and `config-write`.

Each LDAP entry maps to a resource under `/admin/config`, with default values shown in the resource even if they are not set in the LDAP representation.

`/alive`

Provides an endpoint to check whether the server is currently *alive*, meaning that its internal checks have not found any errors that would require administrative action.

By default, this endpoint returns a status code to anonymous requests, and supports authenticated requests. For details, see Server is Alive (HTTP).

`/healthy`

Provides an endpoint to check whether the server is currently *healthy*, meaning that it is alive and any replication delays are below a configurable threshold.

By default, this endpoint returns a status code to anonymous requests, and supports authenticated requests. For details, see Server Health (HTTP).

`/metrics/api`

Provides read-only access through Forgerock Common REST to a JSON-based view of `cn=monitor` and the monitoring backend.

By default, the HTTP Basic authorization mechanism protects this endpoint. Users reading monitoring information must have the `monitor-read` privilege.

The endpoint represents a collection where each LDAP entry maps to a resource under `/metrics/api`. Use a REST Query with a `_queryFilter` parameter to access this endpoint. To return all resources, use `/metrics/api?_queryFilter=true`.

`/metrics/prometheus`

Provides an API to the server monitoring information for use with Prometheus monitoring software.

By default, this endpoint is protected by the HTTP Basic authorization mechanism. Users reading monitoring information must have the `monitor-read` privilege.

To use the Admin endpoint APIs, follow these steps:

1. Grant users access to the endpoints as appropriate:
    1. For access to `/admin/config`, assign `config-read` or `config-write` privileges, and a global ACI to read or update `cn=config`.

       The following example grants Kirsten Vaughan access to read the configuration:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: config-read
EOF
$ dsconfig \
 set-access-control-handler-prop \
 --add global-aci:\(target=\"ldap:///cn=config\"\)\(targetattr=\"*\"\)\ \(version\ 3.0\;\ acl\ \"Read\
configuration\"\;\ allow\ \(read,search\)\ userdn=\"ldap:///
uid=kvaughan,ou=People,dc=example,dc=com\"\;\) \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. For access to `/metrics` endpoints, if no account for monitoring was created at setup time, assign the `monitor-read` privilege.

   The following example adds the `monitor-read` privilege to Kirsten Vaughan's entry:

   ```
   $ ldapmodify \
    --hostname localhost \
    --port 1636 \
    --useSsl \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --bindDN uid=admin \
    --bindPassword password << EOF
   dn: uid=kvaughan,ou=People,dc=example,dc=com
   changetype: modify
   add: ds-privilege-name
   ds-privilege-name: monitor-read
   EOF
   ```

   For details, see Administrative privileges.

2. Adjust the `authorization-mechanism` settings for the Admin endpoint.

   By default, the Admin endpoint uses the HTTP Basic authorization mechanism. The HTTP Basic authorization mechanism default configuration resolves the user identity extracted from the HTTP request to an LDAP user identity as follows:

   1. If the request has an `Authorization: Basic` header for HTTP Basic authentication, the server extracts the username and password.

2. If the request has `X-OpenIDM-Username` and `X-OpenIDM-Password` headers, the server extracts the username and password.
3. The server uses the default exact match identity mapper to search for a unique match between the username and the UID attribute value of an entry in the local public naming contexts of the DS server.

   In other words, in LDAP terms, it searches under all user data base DNs for `(uid=http-username)`. The username `kvaughan` maps to the example entry with DN `uid=kvaughan,ou=People,dc=example,dc=com`.

   For details, see Identity mappers, and Configure HTTP authorization.

3. Test access to the endpoint as an authorized user.

   The following example reads the Admin endpoint resource under `/admin/config`. The following example generates the CA certificate in PEM format from the server deployment ID and password. It uses the CA certificate to trust the server certificate. A CA certificate is only necessary if the CA is not well-known:

```
$ dskeymgr \
 export-ca-cert \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --outputFile ca-cert.pem

$ curl \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 https://localhost:8443/admin/config/http-endpoints/%2Fadmin?_prettyPrint=true

{
  "_id" : "/admin",
  "_rev" : "<revision>",
  "_schema" : "admin-endpoint",
  "java-class" : "org.opends.server.protocols.http.rest2ldap.AdminEndpoint",
  "base-path" : "/admin",
  "authorization-mechanism" : [ {
    "_id" : "HTTP Basic",
    "_rev" : "<revision>"
  } ],
  "enabled" : true
}
```

   Notice how the path to the resource in the example above, `/admin/config/http-endpoints/%2Fadmin`, corresponds to the DN of the entry under `cn=config`, which is `ds-cfg-base-path=/admin,cn=HTTP Endpoints,cn=config`.

   The following example demonstrates reading everything under `/metrics/api`:

```
$ curl \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 https://localhost:8443/metrics/api?_queryFilter=true
```

LDAP access

Set the LDAP port

The reserved port number for LDAP is `389`. Most examples in the documentation use `1389`, which is accessible to non-privileged users:

1. The following example changes the LDAP port number to `11389`:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAP \
 --set listen-port:11389 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Restart the connection handler, and the change takes effect:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAP \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAP \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Enable StartTLS

StartTLS negotiations start on the unsecure LDAP port, and then protect communication with the client:

1. Activate StartTLS on the current LDAP port:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAP \
 --set allow-start-tls:true \
 --set key-manager-provider:PKCS12 \
 --set trust-manager-provider:"JVM Trust Manager" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

If the key manager provider has multiple key pairs that DS could use for TLS, where the secret key was generated with the same key algorithm, such as `EC` or `RSA`, you can specify which key pairs to use with the `--set ssl-cert-nickname:server-cert` option. The *server-cert* is the certificate alias of the key pair. This option is not necessary if there is only one server key pair, or if each secret key was generated with a different key algorithm.

The change takes effect. No need to restart the server.

Set the LDAPS port

At setup time, use the `--ldapsPort` option.

Later, follow these steps to set up an LDAPS port:

1. Configure the server to activate LDAPS access:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAPS \
 --set enabled:true \
 --set listen-port:1636 \
 --set use-ssl:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. If the deployment requires SSL client authentication, set the `ssl-client-auth-policy` and `trust-manager-provider` properties appropriately.

Set the LDAPS port

The reserved port number for LDAPS is `636`. Most examples in the documentation use `1636`, which is accessible to non-privileged users.

1. Change the port number using the `dsconfig` command.

The following example changes the LDAPS port number to `11636`:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAPS \
 --set listen-port:11636 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Restart the connection handler so the change takes effect.

   To restart the connection handler, you disable it, then enable it again:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAPS \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAPS \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

LDIF file access

The LDIF connection handler lets you change directory data by placing LDIF files in a file system directory. The DS server regularly polls for changes to the directory. The server deletes the LDIF file after making the changes.

1. Add the directory where you put LDIF to be processed:

```
$ mkdir /path/to/opendj/config/auto-process-ldif
```

This example uses the default value of the `ldif-directory` property.

2. Activate LDIF file access:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDIF \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The change takes effect immediately.

LDAP schema

About LDAP schema

Directory schema, described in RFC 4512, define the kinds of information you find in the directory, and how the information is related.

By default, DS servers conform strictly to LDAPv3 standards for schema definitions and syntax checking. This ensures that data stored is valid and properly formed. Unless your data uses only standard schema present in the server when you install, you must add additional schema definitions to account for the data specific to your applications.

DS servers include many standard schema definitions. You can update and extend schema definitions while DS servers are online. As a result, you can add new applications requiring additional data without stopping your directory service.

The examples that follow focus primarily on the following types of directory schema definitions:

- *Attribute type* definitions describe attributes of directory entries, such as `givenName` or `mail`.

  Here is an example of an attribute type definition:

```
# Attribute type definition
attributeTypes: ( 0.9.2342.19200300.100.1.3 NAME ( 'mail' 'rfc822Mailbox' )
  EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} X-ORIGIN 'RFC 4524' )
```

  Attribute type definitions start with an OID, and a short name or names that are easier to remember. The attribute type definition can specify how attribute values should be collated for sorting, and what syntax they use.

  The X-ORIGIN is an extension to identify where the definition originated. When you define your own schema, provide an X-ORIGIN to help track versions of definitions.

Attribute type definitions indicate whether the attribute is:

- A *user attribute* intended to be modified by external applications.

  This is the default, and you can make it explicit with `USAGE userApplications`.

  The user attributes that are required and permitted on each entry are defined by the entry's object classes. The server checks what the entry's object classes require and permit when updating user attributes.

- An *operational attribute* intended to be managed by the server for internal purposes.

  You can specify this with `USAGE directoryOperation`.

  The server does not check whether an operational attribute is allowed by an object class.

Attribute type definitions differentiate operational attributes with the following `USAGE` types:

- `USAGE directoryOperation` indicates a generic operational attribute.

  Use this type, for example, when creating a last login time attribute.

- `USAGE dSAOperation` indicates a DSA-specific operational attribute, meaning an operational attribute specific to the current server.

- `USAGE distributedOperation` indicates a DSA-shared operational attribute, meaning an operational attribute shared by multiple servers.

- *Object class* definitions identify the attribute types that an entry must have, and may have.

  Here is an example of an object class definition:

```
# Object class definition
objectClasses: ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL MUST ( sn $ cn )
  MAY ( userPassword $ telephoneNumber $ seeAlso $ description )
  X-ORIGIN 'RFC 4519' )
```

  Entries all have an attribute identifying their object classes(es), called `objectClass`.

  Object class definitions start with an object identifier (OID), and a short name that is easier to remember.

  The definition here says that the person object class inherits from the `top` object class, which is the top-level parent of all object classes.

  An entry can have one STRUCTURAL object class inheritance branch, such as `top` → `person` → `organizationalPerson` → `inetOrgPerson`. Entries can have multiple AUXILIARY object classes. The object class defines the attribute types that must and may be present on entries of the object class.

- An *attribute syntax* constrains what directory clients can store as attribute values.

  An attribute syntax is identified in an attribute type definition by its OID. String-based syntax OIDs are optionally followed by a number set between braces. The number represents a minimum upper bound on the number of characters in the attribute value. For example, in the attribute type definition shown above, the syntax is `1.3.6.1.4.1.1466.115.121.1.26{256}`, IA5 string. An IA5 string (composed of characters from the international version of the ASCII character set) can contain at least 256 characters.

You can find a table matching attribute syntax OIDs with their human-readable names in RFC 4517, Appendix A. Summary of Syntax Object Identifiers. The RFC describes attribute syntaxes in detail. You can list attribute syntaxes with the `dsconfig` command.

If you are trying unsuccessfully to import non-compliant data, clean the data before importing it. If cleaning the data is not an option, read Import Legacy Data.

When creating attribute type definitions, use existing attribute syntaxes where possible. If you must create your own attribute syntax, then consider the schema extensions in Update LDAP schema.

Although attribute syntaxes are often specified in attribute type definitions, DS servers do not always check that attribute values comply with attribute syntaxes. DS servers do enforce compliance by default for the following to avoid bad directory data:

- Certificates
- Country strings
- Directory strings
- JPEG photos
- Telephone numbers

- *Matching rules* define how to compare attribute values to assertion values for LDAP search and LDAP compare operations.

For example, suppose you search with the filter `(uid=bjensen)`. The assertion value in this case is `bjensen`.

DS servers have the following schema definition for the user ID attribute:

```
attributeTypes: ( 0.9.2342.19200300.100.1.1 NAME ( 'uid' 'userid' )
 EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} X-ORIGIN 'RFC 4519' )
```

When finding an equality match for your search, servers use the `caseIgnoreMatch` matching rule to check for user ID attribute values that equal `bjensen`.

You can read the schema definitions for matching rules that the server supports by performing an LDAP search:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDn cn=schema \
 --searchScope base \
 "(&)" \
 matchingRules
```

Notice that many matching rules support string collation in languages other than English. For the full list of string collation matching rules, see Supported locales. For the list of other matching rules, see Matching rules.

Matching rules enable directory clients to compare other values besides strings.

DS servers expose schema over protocol through the `cn=schema` entry. The server stores the schema definitions in LDIF format files in the `db/schema/` directory. When you set up a server, the process copies many definitions to this location.

Update LDAP schema

DS servers allow you to update directory schema definitions while the server is running. You can add support for new types of attributes and entries without interrupting the directory service. DS servers replicate schema definitions, propagating them to other replicas automatically.

You update schema either by:

- Performing LDAP modify operations while the server is running.
- Adding schema files to the `db/schema/` directory before starting the server.

Before adding schema definitions, take note of the following points:

- Define DS schema elements in LDIF.

  For examples, see the build in schema files in the `db/schema/` directory.

- Add your schema definitions in a file prefixed with a higher number than built-in files, such as `99-user.ldif`, which is the default filename when you modify schema over LDAP.

  On startup, the DS server reads schema files in order, sorted alphanumerically. Your definitions likely depend on others that the server should read first.

- Define a schema element before referencing it in other definitions.

  For example, make sure you define an attribute type before using it in an object class definition.

For an example, see Custom schema.

DS servers support the standard LDAP schema definitions described in RFC 4512, section 4.1. They also support the following extensions:

Schema Extensions for All Types

X-DEPRECATED-SINCE

This specifies a release that deprecates the schema element.

Example: `X-DEPRECATED-SINCE: 7.2.5`

X-ORIGIN

This specifies the origin of a schema element.

Examples:

- `X-ORIGIN 'RFC 4519'`

- `X-ORIGIN 'draft-ietf-ldup-subentry'`
- `X-ORIGIN 'DS Directory Server'`

X-SCHEMA-FILE

This specifies the relative path to the schema file containing the schema element.

Schema definitions are located in `/path/to/opendj/db/schema/*.ldif` files.

Example: `X-SCHEMA-FILE '00-core.ldif'`.

X-STABILITY

Used to specify the interface stability of the schema element.

This extension takes one of the following values:

- `Evolving`
- `Internal`
- `Removed`
- `Stable`
- `Technology Preview`

Schema Extensions for Syntaxes

Extensions to syntax definitions requires additional code to support syntax checking. DS servers support the following extensions for their particular use cases:

X-ENUM

This defines a syntax that is an enumeration of values.

The following attribute syntax description defines a syntax allowing four possible attribute values:

```
ldapSyntaxes: ( security-label-syntax-oid DESC 'Security Label'
 X-ENUM ( 'top-secret' 'secret' 'confidential' 'unclassified' ) )
```

X-PATTERN

This defines a syntax based on a regular expression pattern. Valid regular expressions are those defined for java.util.regex.Pattern.

The following attribute syntax description defines a simple, lenient SIP phone URI syntax check:

```
ldapSyntaxes: ( simple-sip-uri-syntax-oid DESC 'Lenient SIP URI Syntax'
  X-PATTERN '^sip:[a-zA-Z0-9.]+@[a-zA-Z0-9.]+(:[0-9]+)?$' )
```

X-SUBST

This specifies a substitute syntax to use for one that DS servers do not implement.

The following example substitutes Directory String syntax, OID `1.3.6.1.4.1.1466.115.121.1.15`, for a syntax that DS servers do not implement:

```
ldapSyntaxes: ( non-implemented-syntax-oid DESC 'Not Implemented in DS'
  X-SUBST '1.3.6.1.4.1.1466.115.121.1.15' )
```

Schema Extensions for Attributes

X-APPROX

`X-APPROX` specifies a non-default approximate matching rule for an attribute type.

The default is the double metaphone approximate match.

Custom schema

This example updates the LDAP schema while the server is online. It defines a custom enumeration syntax using the attribute type and a custom object class that uses the attribute:

- A custom enumeration syntax using the `X-ENUM` extension.
- A custom attribute type using the custom syntax.
- A custom object class for entries that have the custom attribute:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password <<EOF
dn: cn=schema
changetype: modify
add: ldapSyntaxes
ldapSyntaxes: ( temporary-syntax-oid
  DESC 'Custom enumeration syntax'
  X-ENUM ( 'bronze' 'silver' 'gold' )
  X-ORIGIN 'DS Documentation Examples'
  X-SCHEMA-FILE '99-user.ldif' )
-
add: attributeTypes
attributeTypes: ( temporary-attr-oid
  NAME 'myEnum'
  DESC 'Custom attribute type for'
  SYNTAX temporary-syntax-oid
  USAGE userApplications
  X-ORIGIN 'DS Documentation Examples'
  X-SCHEMA-FILE '99-user.ldif' )
-
add: objectClasses
objectClasses: ( temporary-oc-oid
  NAME 'myEnumObjectClass'
  DESC 'Custom object class for entries with a myEnum attribute'
  SUP top
  AUXILIARY
  MAY myEnum
  X-ORIGIN 'DS Documentation Examples'
  X-SCHEMA-FILE '99-user.ldif' )

EOF

# MODIFY operation successful for DN cn=schema
```

Notice the follow properties of this update to the schema definitions:

- The `ldapSyntaxes` definition comes before the `attributeTypes` definition that uses the syntax.

  The `attributeTypes` definition comes before the `objectClasses` definition that uses the attribute type.

- Each definition has a temporary OID of the form `temporary-*-oid`.

  While you develop new schema definitions, temporary OIDs are fine. Get permanent, correctly assigned OIDs before using schema definitions in production.

- Each definition has a `DESC` (description) string intended for human readers.

- Each definition specifies its origin with the extension `X-ORIGIN 'DS Documentation Examples'`.

- Each definition specifies its schema file with the extension `X-SCHEMA-FILE '99-user.ldif'`.

- The syntax definition has no name, as it is referenced internally by OID only.

- `X-ENUM ( 'bronze' 'silver' 'gold' )` indicates that the syntax allows three values, `bronze`, `silver`, `gold`.

  DS servers reject other values for attributes with this syntax.

- The attribute type named `myEnum` has these properties:
  - It uses the enumeration syntax, `SYNTAX temporary-syntax-oid`.
  - It can only have one value at a time, `SINGLE-VALUE`.

    The default, if you omit `SINGLE-VALUE`, is to allow multiple values.
  - It is intended for use by user applications, `USAGE userApplications`.
- The object class named `myEnumObjectClass` has these properties:
  - Its parent for inheritance is the top-level object class, `SUP top`.

    `top` is the abstract parent of all structural object class hierarchies, so all object classes inherit from it.
  - It is an auxiliary object class, `AUXILIARY`.

    Auxiliary object classes are used to augment attributes of entries that already have a structural object class.
  - It defines no required attributes (no `MUST`).
  - It defines one optional attribute which is the custom attribute, `MAY myEnum`.

After adding the schema definitions, you can add the attribute to an entry as shown in the following example:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password <<EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
add: objectClass
objectClass: myEnumObjectClass
-
add: myEnum
myEnum: silver

EOF

# MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com
```

As shown in the following example, the attribute syntax prevents users from setting an attribute value that is not specified in your enumeration:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password <<EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: myEnum
myEnum: wrong value

EOF

# The LDAP modify request failed: 21 (Invalid Attribute Syntax)
# Additional Information:  When attempting to modify entry uid=bjensen,ou=People,dc=example,dc=com to replace the set
of values for attribute myEnum, value "wrong value" was found to be invalid according to the associated syntax: The
provided value "wrong value" cannot be parsed because it is not allowed by enumeration syntax with OID "temporary-
syntax-oid"
```

For more examples, read the built-in schema definition files in the `db/schema/` directory.

Schema and JSON

DS software has the following features for working with JSON objects:

- **RESTful HTTP access to directory services**

  If you have LDAP data, but HTTP client applications want JSON over HTTP instead, DS software can expose your LDAP data as JSON resources over HTTP to REST clients. You can configure how LDAP entries map to JSON resources.

  There is no requirement to change LDAP schema definitions before using this feature. To get started, see DS REST APIs.

- **JSON syntax LDAP attributes**

  If you have LDAP client applications that store JSON in the directory, you can define LDAP attributes that have `Json` syntax.

  The following schema excerpt defines an attribute called `json` with case-insensitive matching:

  ```
  attributeTypes: ( json-attribute-oid NAME 'json'
   SYNTAX 1.3.6.1.4.1.36733.2.1.3.1 EQUALITY caseIgnoreJsonQueryMatch
   X-ORIGIN 'DS Documentation Examples' )
  ```

  Notice that the JSON syntax OID is `1.3.6.1.4.1.36733.2.1.3.1`. The definition above uses the (default) `caseIgnoreJsonQueryMatch` matching rule for equality. As explained later in this page, you might want to choose different matching rules for your JSON attributes.

  When DS servers receive update requests for `Json` syntax attributes, they expect valid JSON objects. By default, `Json` syntax attribute values must comply with *The JavaScript Object Notation (JSON) Data Interchange Format*, described in RFC 7159. You can use the advanced core schema configuration option `json-validation-policy` to have the server be more lenient in what it accepts, or to disable JSON syntax checking.

- **Configurable indexing for JSON attributes**

  When you store JSON attributes in the directory, you can index every field in each JSON attribute value, or you can index only what you use.

  As for other LDAP attributes, the indexes depend on the matching rule defined for the JSON syntax attribute.

DS servers treat JSON syntax attribute values as objects. Two JSON values may be considered equivalent despite differences in their string representations. The following JSON objects can be considered equivalent because each field has the same value. Their string representations are different, however:

```
[
 { "id": "bjensen", "given-name": "Barbara", "surname": "Jensen" },
 {"surname":"Jensen","given-name":"Barbara","id":"bjensen"}
]
```

Unlike other objects with their own LDAP attribute syntaxes, such as X.509 certificates, two JSON objects with completely different structures (different field names and types) are still both JSON. Nothing in the JSON syntax alone tells the server anything about what a JSON object must and may contain.

When defining LDAP schema for JSON attributes, it helps therefore to understand the structure of the expected JSON. Will the attribute values be arbitrary JSON objects, or JSON objects whose structure is governed by some common schema? If a JSON attribute value is an arbitrary object, you can do little to optimize how it is indexed or compared. If the value is a structured object, however, you can configure optimizations based on the structure.

For structured JSON objects, the definition of JSON object equality is what enables you to pick the optimal matching rule for the LDAP schema definition. The matching rule determines how the server indexes the attribute, and how the server compares two JSON values for equality. You can define equality in the following ways:

- Two JSON objects are equal if *all* fields have the same values.

  By this definition, `{"a": 1, "b": true}` equals `{"b":true,"a":1}`. However, `{"a": 1, "b": true}` and `{"a": 1, "b": "true"}` are different.

- Two JSON objects ` equal if *some* fields have the same values. Other fields are ignored when comparing for equality.

  For example, take the case where two JSON objects are considered equal if they have the same "_id" values. By this definition, `{"_id":1,"b":true}` equals `{"_id":1,"b":false}`. However, `{"_id":1,"b":true}` and `{"_id": 2,"b":true}` are different.

  DS servers have built-in matching rules for the case where equality means "_id" values are equal. If the fields to compare are different from "_id", you must define your own matching rule and configure a custom schema provider that implements it. This following table helps you choose a JSON equality matching rule:

JSON Content	Two JSON Objects are Equal if...	Use One of These Matching Rules
Arbitrary (any valid JSON is allowed)	All fields have the same values.	`caseExactJsonQueryMatch` `caseIgnoreJsonQueryMatch`
Structured	All fields have the same values.	`caseExactJsonQueryMatch` `caseIgnoreJsonQueryMatch`

JSON Content	Two JSON Objects are Equal if...	Use One of These Matching Rules
Structured	One or more other fields have the same values. Additional fields are ignored when comparing for equality.	When using this matching rule, create a custom `json-equality-matching-rule` Schema Provider. The custom schema provider must include all the necessary properties and reference the custom field(s). See JSON Equality Matching Rule Index.

When you choose an equality matching rule in the LDAP attribute definition, you are also choosing the default that applies in an LDAP search filter equality assertion. For example, `caseIgnoreJsonQueryMatch` works with filters such as `"(json=id eq 'bjensen')"`.

DS servers also implement JSON ordering matching rules for determining the relative order of two JSON values using a custom set of rules. You can select which JSON fields should be used for performing the ordering match. You can also define whether those fields that contain strings should be normalized before comparison by trimming white space or ignoring case differences. DS servers can implement JSON ordering matching rules on demand when presented with an extended server-side sort request, as described in Server-Side Sort. If, however, you define them statically in your LDAP schema, then you must implement them by creating a custom `json-ordering-matching-rule` Schema Provider. For details about the `json-ordering-matching-rule` object's properties, see JSON Ordering Matching Rule.

For examples showing how to add LDAP schema for new attributes, see Update LDAP Schema. For examples showing how to index JSON attributes, see Custom Indexes for JSON.

Import legacy data

By default, DS servers accept data that follows the schema for allowable and rejected data. You might have legacy data from a directory service that is more lenient, allowing non-standard constructions such as multiple structural object classes per entry, not checking attribute value syntax, or even not respecting schema definitions.

For example, when importing data with multiple structural object classes defined per entry, you can relax schema checking to warn rather than reject entries having this issue:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set single-structural-objectclass-behavior:warn \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

You can allow attribute values that do not respect the defined syntax:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set invalid-attribute-syntax-behavior:warn \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

You can even turn off schema checking altogether. Only turn off schema checking *when you are absolutely sure that the entries and attributes already respect the schema definitions*:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set check-schema:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Turning off schema checking can potentially boost import performance.

Standard schema

DS servers provide many standard schema definitions. For full descriptions, see the Schema Reference. Find the definitions in LDIF files in the `/path/to/opendj/db/schema/` directory:

File	Description
`00-core.ldif`	Schema definitions for the following Internet-Drafts, RFCs, and standards: • draft-boreham-numsubordinates • draft-findlay-ldap-groupofentries • draft-good-ldap-changelog • draft-howard-namedobject • draft-ietf-ldup-subentry • draft-wahl-ldap-adminaddr • RFC 1274 • RFC 2079 • RFC 2256 • RFC 2798 • RFC 3045 • RFC 3296 • RFC 3671 • RFC 3672 • RFC 4512 • RFC 4519 • RFC 4523 • RFC 4524 • RFC 4530 • RFC 5020 • X.501
`01-pwpolicy.ldif`	Schema for draft-behera-ldap-password-policy (Draft 09), which defines a mechanism for storing password policy information in an LDAP directory server.
`02-config.ldif`	This file contains the attribute type and objectclass definitions for use with the server configuration.
`03-changelog.ldif`	Schema for draft-good-ldap-changelog, which defines a mechanism for storing information about changes to directory server data.
`03-rfc2713.ldif`	Schema for RFC 2713, which defines a mechanism for storing serialized Java objects in the directory server.
`03-rfc2714.ldif`	Schema for RFC 2714, which defines a mechanism for storing CORBA objects in the directory server.
`03-rfc2739.ldif`	Schema for RFC 2739, which defines a mechanism for storing calendar and vCard objects in the directory server. Be aware that the definition in RFC 2739 contains a number of errors. This schema file has been altered from the standard definition to fix a number of those problems.
`03-rfc2926.ldif`	Schema for RFC 2926, which defines a mechanism for mapping between Service Location Protocol (SLP) advertisements and LDAP.

File	Description
`03-rfc3112.ldif`	Schema for RFC 3112, which defines the authentication password schema.
`03-rfc3712.ldif`	Schema for RFC 3712, which defines a mechanism for storing printer information in the directory server.
`03-uddiv3.ldif`	Schema for RFC 4403, which defines a mechanism for storing UDDIv3 information in the directory server.
`04-rfc2307bis.ldif`	Schema for draft-howard-rfc2307bis, which defines a mechanism for storing naming service information in the directory server.
`05-rfc4876.ldif`	Schema for RFC 4876, which defines a schema for storing Directory User Agent (DUA) profiles and preferences in the directory server.
`05-samba.ldif`	Schema required when storing Samba user accounts in the directory server.
`05-solaris.ldif`	Schema required for Solaris and OpenSolaris LDAP naming services.
`06-compat.ldif`	Backwards-compatible schema for use in the server configuration.

Indexes

About indexes

A basic, standard directory feature is the ability to respond quickly to searches.

An LDAP search specifies the information that directly affects how long the directory might take to respond:

- The base DN for the search.

  The more specific the base DN, the less information to check during the search. For example, a request with base DN `dc=example,dc=com` potentially involves checking many more entries than a request with base DN `uid=bjensen,ou=people,dc=example,dc=com`.

- The scope of the search.

  A subtree or one-level scope targets many entries, whereas a base search is limited to one entry.

- The search filter to match.

  A search filter asserts that for an entry to match, it has an attribute that corresponds to some value. For example, `(cn=Babs Jensen)` asserts that `cn` must have a value that equals `Babs Jensen`.

  A directory server would waste resources checking all entries for a match. Instead, directory servers maintain indexes to expedite checking for a match.

LDAP directory servers disallow searches that cannot be handled expediently using indexes. Maintaining appropriate indexes is a key aspect of directory administration.

Role of an index

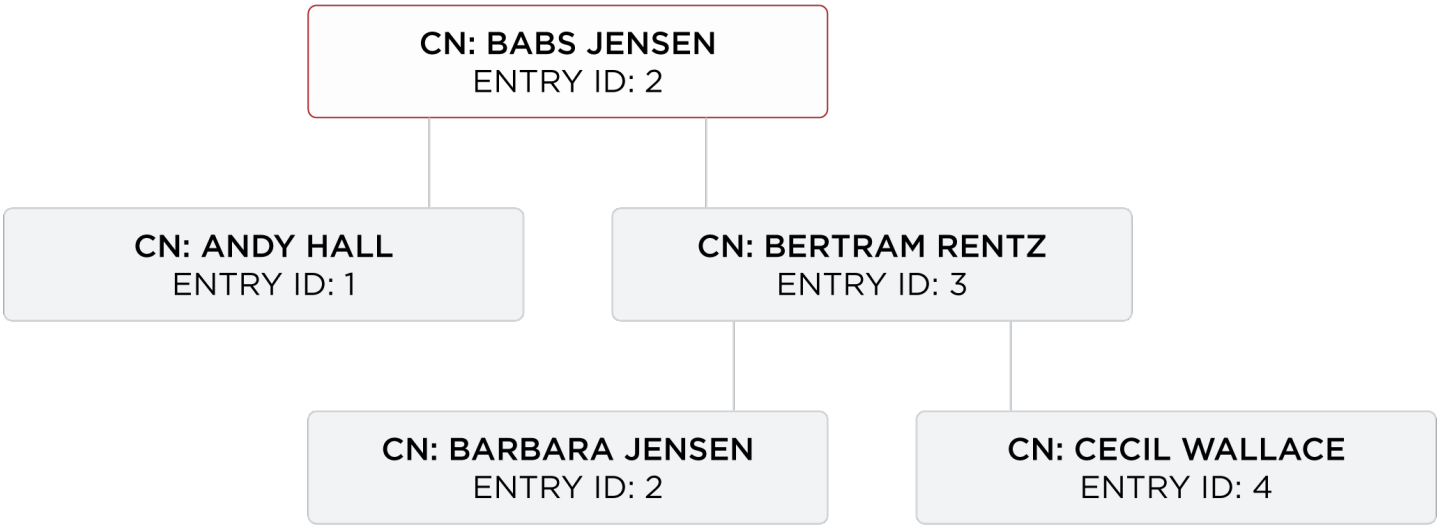
The role of an index is to answer the question, "Which entries have an attribute with this corresponding value?"

Each index is therefore specific to an attribute.

Each index is also specific to the comparison implied in the search filter. For example, a directory server maintains distinct indexes for exact (equality) matching and for substring matching. The types of indexes are explained in Index types. Furthermore, indexes are configured in specific directory backends.

Index implementation

An index is implemented as a tree of key-value pairs. The key is a form of the value to match, such as `babs jensen`. The value is a list of IDs for entries that match the key. The figure that follows shows an equality (case ignore exact match) index with five keys from a total of four entries. If the data set were large, there could be more than one entry ID per key:



How DS uses indexes

This example illustrates how DS uses an index.

When the search filter is `(cn=Babs Jensen)`, DS retrieves the IDs for entries whose CN matches `Babs Jensen` by looking them up in the equality index of the CN attribute. (For a complex filter, it might optimize the search by changing the order in which it uses the indexes.) A successful result is zero or more entry IDs. These are the candidate result entries.

For each candidate, DS retrieves the entry by ID from a system index called `id2entry`. As its name suggests, this index returns an entry for an entry ID. If there is a match, and the client application has the right to access to the data, DS returns the search result. It continues this process until no candidates are left.

Unindexed searches

If there are no indexes that correspond to a search request, DS must check for a match against every entry in the scope of the search. Evaluating every entry for a match is referred to as an *unindexed* search.

An unindexed search is an expensive operation, particularly for large directories. A server refuses unindexed searches unless the user has specific permission to make such requests. The permission to perform an unindexed search is granted with the `unindexed-search` privilege. This privilege is reserved for the directory superuser by default. It should not be granted lightly.

If the number of entries is smaller than the default resource limits, you can still perform what appear to be unindexed searches, meaning searches with filters for which no index appears to exist. That is because the `dn2id` index returns all user data entries without hitting a resource limit that would make the search unindexed.

Use cases that may call for unindexed searches include the following:

- An application must periodically retrieve a very large amount of directory data all at once through an LDAP search.

  For example, an application performs an LDAP search to retrieve everything in the directory once a week as part of a batch job that runs during off hours.

  Make sure the application has no resource limits. For details, see Resource limits.

- A directory data administrator occasionally browses directory data through a graphical UI without initially knowing what they are looking for or how to narrow the search.

  Big indexes let you work around this problem. They facilitate searches where large numbers of entries match. For example, big indexes can help when paging through all the employees in the large company, or all the users in the state of California. For details, see Big index and Indexes for attributes with few unique values.

  Alternatively, DS directory servers can use an appropriately configured VLV index to sort results for an unindexed search. For details, see VLV for paged server-side sort.

Index updates

When an entry is added, changed, or deleted, the directory server updates each affected index to reflect the change. This happens while the server is online, and has a cost. This cost is the reason to maintain indexes only those indexes that are used.

DS only updates indexes for the attributes that change. Updating an unindexed attribute is therefore faster than updating an indexed attribute.

What to index

DS directory server search performance depends on indexes. The default settings are fine for evaluating DS software, and they work well with sample data. The default settings do not necessarily fit your directory data, and the searches your applications perform:

- Configure necessary indexes for the searches you anticipate.
- Let DS optimize search queries to use whatever indexes are available.

  DS servers may use a presence index when an equality index is not available, for example.

- Use metrics and index debugging to check that searches use indexes and are optimized.

You cannot configure the optimizations DS servers perform. You can, however, review search metrics, logs, and search debugging data to verify that searches use indexes and are optimized.

- Monitor DS servers for indexes that are not used.

Necessary indexes

Index maintenance has its costs. Every time an indexed attribute is updated, the server must update each affected index to reflect the change. This is wasteful if the index is not used. Indexes, especially substring indexes, can occupy more memory and disk space than the corresponding data.

Aim to maintain only indexes that speed up appropriate searches, and that allow the server to operate properly. The former indexes depend on how directory users search, and require thought and investigation. The latter includes non-configurable internal indexes, that should not change.

Begin by reviewing the attributes of your directory data. Which attributes would you expect to see in a search filter? If an attribute is going to show up frequently in reasonable search filters, then index it.

Compare your guesses with what you see actually happening in the directory.

Unindexed searches

Directory users might complain their searches fail because they're unindexed.

By default, DS directory servers reject unindexed searches with a result code of 50 and additional information about the unindexed search. The following example attempts, anonymously, to get the entries for all users whose email address ends in `.com`:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=user.0,ou=People,dc=example,dc=com \
 --bindPassword password \
 --baseDN ou=people,dc=example,dc=com \
 "(&(mail=*.com)(objectclass=person))"
# The LDAP search request failed: 50 (Insufficient Access Rights)
# Additional Information:  You do not have sufficient privileges to perform an unindexed search
```

If they're unintentionally requesting an unindexed search, suggest ways to perform an indexed search instead. Perhaps the application needs a better search filter. Perhaps it requests more results than necessary. For example, a GUI application lets a user browse directory entries. The application could page through the results, rather than attempting to retrieve all the entries at once. To let the user page back and forth through the results, you could add a browsing (VLV) index for the application to get the entries for the current screen.

An application might have a good reason to get the full list of all entries in one operation. If so, assign the application's account the `unindexed-search` privilege. Consider other options before you grant the privilege, however. Unindexed searches cause performance problems for concurrent directory operations.

When an application has the privilege or binds with directory superuser credentials—by default, the `uid=admin` DN and password—then DS does not reject its request for an unindexed search. Check for unindexed searches using the `ds-mon-backend-filter-unindexed` monitoring attribute:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN cn=monitor \
 "(objectClass=ds-monitor-backend-db)" ds-mon-backend-filter-unindexed
```

If `ds-mon-backend-filter-unindexed` is greater than zero, review the access log for unexpected unindexed searches. The following example shows the relevant fields in an access log message:

```
{
  "request": {
    "protocol": "LDAP",
    "operation": "SEARCH"
  },
  "response": {
    "detail": "You do not have sufficient privileges to perform an unindexed search",
    "additionalItems": {
      "unindexed": null
    }
  }
}
```

Beyond the key fields shown in the example, messages in the access log also specify the search filter and scope. Understand the operation that led to each unindexed search. If the filter is appropriate and often used, add an index to ease the search. Either analyze the access logs to find how often operations use the search filter or monitor operations with the index analysis feature, described in Index analysis metrics.

In addition to responding to client search requests, a server performs internal searches. Internal searches let the server retrieve data needed for a request, and maintain internal state information. Sometimes, internal searches become unindexed. When this happens, the server logs a warning similar to the following:

```
The server is performing an unindexed internal search request
with base DN '%s', scope '%s', and filter '%s'. Unindexed internal
searches are usually unexpected and could impact performance.
Please verify that that backend's indexes are configured correctly
for these search parameters.
```

When you see a message like this in the server log, take these actions:

- Figure out which indexes are missing, and add them.

  For details, see Index analysis metrics, Debug search indexes, and Configure indexes.

- Check the integrity of the indexes.

For details, see Verify indexes.

- If the relevant indexes exist, and you have verified that they are sound, the index entry limit might be too low.

  This can happen, for example, in directory servers with more than 4000 groups in a single backend. For details, see Index entry limits.

- If you have made the changes described in the steps above, and problem persists, contact technical support.

Index analysis metrics

DS servers provide the index analysis feature to collect information about filters in search requests. This feature is useful, but not recommend to keep enabled on production servers, as DS maintains the metrics in memory.

You can activate the index analysis mechanism using the `dsconfig set-backend-prop` command:

```
$ dsconfig \
 set-backend-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --set index-filter-analyzer-enabled:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

The command causes the server to analyze filters used, and to keep the results in memory. You can read the results as monitoring information:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN ds-cfg-backend-id=dsEvaluation,cn=Backends,cn=monitor \
 --searchScope base \
 "(&)" \
 ds-mon-backend-filter-use-start-time ds-mon-backend-filter-use-indexed ds-mon-backend-filter-use-unindexed ds-mon-
backend-filter-use

dn: ds-cfg-backend-id=dsEvaluation,cn=backends,cn=monitor
ds-mon-backend-filter-use-start-time: <timestamp>
ds-mon-backend-filter-use-indexed: 2
ds-mon-backend-filter-use-unindexed: 3
ds-mon-backend-filter-use: {"search-filter":"(employeenumber=86182)","nb-hits":1,"latest-failure-
reason":"caseIgnoreMatch index type is disabled for the employeeNumber attribute"}
```

The `ds-mon-backend-filter-use` values include the following fields:

`search-filter`

The LDAP search filter.

`nb-hits`

The number of times the filter was used.

`latest-failure-reason`

A message describing why the server could not use any index for this filter.

The output can include filters for internal use, such as `(aci=*)`. In the example above, you see a filter used by a client application.

In the example, a search filter that led to an unindexed search, `(employeenumber=86182)`, had no matches because, "caseIgnoreMatch index type is disabled for the employeeNumber attribute". Some client application has tried to find users by employee number, but no index exists for that purpose. If this appears regularly as a frequent search, add an employee number index.

To avoid impacting server performance, turn off index analysis after you collect the information you need. Use the `dsconfig set-backend-prop` command:

```
$ dsconfig \
 set-backend-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --set index-filter-analyzer-enabled:false \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

Debug search indexes

Sometimes it is not obvious by inspection how a directory server processes a given search request. The directory superuser can gain insight with the `debugsearchindex` attribute.

The default global access control prevents users from reading the `debugsearchindex` attribute. To allow an administrator to read the attribute, add a global ACI such as the following:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --add global-aci:"(targetattr=\"debugsearchindex\")(version 3.0; acl \"Debug search indexes\"; \
  allow (read,search,compare) userdn=\"ldap:///uid=user.0,ou=people,dc=example,dc=com\";)" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Note

The format of `debugsearchindex` values has interface stability: Internal.
The values are intended to be read by human beings, not scripts. If you do write scripts that interpret `debugsearchindex` values, be aware that they are not stable. Be prepared to adapt your scripts for every upgrade or patch.

The `debugsearchindex` attribute value indicates how the server would process the search. The server use its indexes to prepare a set of candidate entries. It iterates through the set to compare candidates with the search filter, returning entries that match. The following example demonstrates this feature for a subtree search with a complex filter:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=user.0,ou=people,dc=example,dc=com \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(&(objectclass=person)(givenName=aa*))" \
 debugsearchindex | sed -n -e "s/^debugsearchindex: //p"

{
  "baseDn": "dc=example,dc=com",
  "scope": "sub",
  "filter": "(&(givenName=aa*)(objectclass=person))",
  "maxCandidateSize": 100000,
  "strategies": [
    {
      "name": "BaseObjectSearchStrategy",
      "diagnostic": "not applicable"
    },
    {
      "name": "VlvSearchStrategy",
      "diagnostic": "not applicable"
    },
    {
      "name": "AttributeIndexSearchStrategy",
      "filter": {
        "query": "INTERSECTION",
        "rank": "RANGE_MATCH",
        "filter": "(&(givenName=aa*)(objectclass=person))",
        "subQueries": [
          {
            "query": "ANY_OF",
            "rank": "RANGE_MATCH",
            "filter": "(givenName=aa*)",
            "subQueries": [
              {
                "query": "ANY_OF",
                "rank": "RANGE_MATCH",
                "filter": "(givenName=aa*)",
                "subQueries": [
                  {
                    "query": "RANGE_MATCH",
                    "rank": "RANGE_MATCH",
                    "index": "givenName.caseIgnoreMatch",
                    "range": "[aa,ab[",
                    "diagnostic": "indexed",
                    "candidates": 50
                  },
                  {
                    "query": "RANGE_MATCH",
                    "rank": "RANGE_MATCH",
                    "index": "givenName.caseIgnoreSubstringsMatch:6",
                    "range": "[aa,ab[",
                    "diagnostic": "skipped"
                  }
                ],
                "diagnostic": "indexed",
```

```
                "candidates": 50
              },
              {
                "query": "MATCH_ALL",
                "rank": "MATCH_ALL",
                "filter": "(givenName=aa*)",
                "index": "givenName.presence",
                "diagnostic": "skipped"
              }
            ],
            "diagnostic": "indexed",
            "candidates": 50,
            "retained": 50
          },
          {
            "query": "ANY_OF",
            "rank": "OBJECT_CLASS_EQUALITY_MATCH",
            "filter": "(objectclass=person)",
            "subQueries": [
              {
                "query": "OBJECT_CLASS_EQUALITY_MATCH",
                "rank": "OBJECT_CLASS_EQUALITY_MATCH",
                "filter": "(objectclass=person)",
                "subQueries": [
                  {
                    "query": "EXACT_MATCH",
                    "rank": "EXACT_MATCH",
                    "index": "objectClass.objectIdentifierMatch",
                    "key": "person",
                    "diagnostic": "not indexed",
                    "candidates": "[LIMIT-EXCEEDED]"
                  },
                  {
                    "query": "EXACT_MATCH",
                    "rank": "EXACT_MATCH",
                    "index": "objectClass.objectIdentifierMatch",
                    "key": "2.5.6.6",
                    "diagnostic": "skipped"
                  }
                ],
                "diagnostic": "not indexed",
                "candidates": "[LIMIT-EXCEEDED]"
              },
              {
                "query": "MATCH_ALL",
                "rank": "MATCH_ALL",
                "filter": "(objectclass=person)",
                "index": "objectClass.presence",
                "diagnostic": "skipped"
              }
            ],
            "diagnostic": "not indexed",
            "candidates": "[LIMIT-EXCEEDED]",
            "retained": 50
          }
        ],
        "diagnostic": "indexed",
        "candidates": 50
      },
      "scope": {
        "type": "sub",
        "diagnostic": "not indexed",
```

```
          "candidates": "[NOT-INDEXED]",
          "retained": 50
        },
        "diagnostic": "indexed",
        "candidates": 50
      }
    ],
    "final": 50
  }
```

The filter in the example matches person entries whose given name starts with `aa`. The search scope is not explicitly specified, so the scope defaults to the subtree including the base DN.

Notice that the `debugsearchindex` value has the following top-level fields:

- (Optional) `"vlv"` describes how the server uses VLV indexes.

  The VLV field is not applicable for this example, and so is not present.

- `"filter"` describes how the server uses the search filter to narrow the set of candidates.
- `"scope"` describes how the server uses the search scope.
- `"final"` indicates the final number of candidates in the set.

In the output, notice that the server uses the equality and substring indexes to find candidate entries whose given name starts with `aa`. If the filter indicated given names *containing* `aa`, as in `givenName=*aa*`, the server would rely only on the substring index.

Notice that the output for the `(objectclass=person)` portion of the filter shows `"candidates": "[LIMIT-EXCEEDED]"`. In this case, there are so many entries matching the value specified that the index is not useful for narrowing the set of candidates. The scope is also not useful for narrowing the set of candidates. Ultimately, however, the `givenName` indexes help the server to narrow the set of candidates. The overall search is indexed and the result is 50 matching entries.

The following example shows a subtree search for accounts with initials starting either with `aa` or with `zz`:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN dc=example,dc=com \
 --bindDN uid=user.0,ou=people,dc=example,dc=com \
 --bindPassword password \
 "(|(initials=aa*)(initials=zz*))" \
 debugsearchindex | sed -n -e "s/^debugsearchindex: //p"

{
  "baseDn": "dc=example,dc=com",
  "scope": "sub",
  "filter": "(|(initials=aa*)(initials=zz*))",
  "maxCandidateSize": 100000,
  "strategies": [
    {
      "name": "BaseObjectSearchStrategy",
      "diagnostic": "not applicable"
    },
    {
      "name": "VlvSearchStrategy",
      "diagnostic": "not applicable"
    },
    {
      "name": "AttributeIndexSearchStrategy",
      "filter": {
        "query": "UNION",
        "rank": "MATCH_ALL",
        "filter": "(|(initials=aa*)(initials=zz*))",
        "subQueries": [
          {
            "query": "ANY_OF",
            "rank": "MATCH_ALL",
            "filter": "(initials=aa*)",
            "subQueries": [
              {
                "query": "MATCH_ALL",
                "rank": "MATCH_ALL",
                "filter": "(initials=aa*)",
                "index": "initials.presence",
                "diagnostic": "not indexed"
              },
              {
                "query": "MATCH_ALL",
                "rank": "MATCH_ALL",
                "filter": "(initials=aa*)",
                "index": "initials.presence",
                "diagnostic": "not indexed"
              }
            ],
            "diagnostic": "not indexed"
          },
          {
            "query": "ANY_OF",
            "rank": "MATCH_ALL",
            "filter": "(initials=zz*)",
            "subQueries": [
```

```
              {
                "query": "MATCH_ALL",
                "rank": "MATCH_ALL",
                "filter": "(initials=zz*)",
                "index": "initials.presence",
                "diagnostic": "skipped"
              },
              {
                "query": "MATCH_ALL",
                "rank": "MATCH_ALL",
                "filter": "(initials=zz*)",
                "index": "initials.presence",
                "diagnostic": "skipped"
              }
            ],
            "diagnostic": "skipped"
          }
        ],
        "diagnostic": "not indexed"
      },
      "scope": {
        "type": "sub",
        "diagnostic": "not indexed",
        "candidates": "[NOT-INDEXED]",
        "retained": "[NOT-INDEXED]"
      },
      "diagnostic": "not indexed",
      "candidates": "[NOT-INDEXED]"
    },
    {
      "name": "BigIndexSearchStrategy",
      "filter": {
        "query": "UNION",
        "rank": "MATCH_ALL",
        "filter": "(|(initials=aa*)(initials=zz*))",
        "subQueries": [
          {
            "query": "ANY_OF",
            "rank": "MATCH_ALL",
            "filter": "(initials=aa*)",
            "subQueries": [
              {
                "query": "MATCH_ALL",
                "rank": "MATCH_ALL",
                "filter": "(initials=aa*)",
                "index": "initials.big.presence",
                "diagnostic": "not supported"
              },
              {
                "query": "MATCH_ALL",
                "rank": "MATCH_ALL",
                "filter": "(initials=aa*)",
                "index": "initials.big.presence",
                "diagnostic": "not supported"
              }
            ],
            "diagnostic": "not indexed"
          }
        ],
        "diagnostic": "not indexed"
      },
```

```
      "diagnostic": "not indexed"
    }
  ],
  "final": "[NOT-INDEXED]"
}
```

As shown in the output, the search is not indexed. To fix this, index the `initials` attribute:

```
$ dsconfig \
 create-backend-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name initials \
 --set index-type:equality \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ rebuild-index \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --baseDn dc=example,dc=com \
 --index initials \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

After configuring and building the new index, try the same search again:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN dc=example,dc=com \
 --bindDN uid=user.0,ou=people,dc=example,dc=com \
 --bindPassword password \
 "(|(initials=aa*)(initials=zz*))" \
 debugsearchindex | sed -n -e "s/^debugsearchindex: //p"

{
  "baseDn": "dc=example,dc=com",
  "scope": "sub",
  "filter": "(|(initials=aa*)(initials=zz*))",
  "maxCandidateSize": 100000,
  "strategies": [
    {
      "name": "BaseObjectSearchStrategy",
      "diagnostic": "not applicable"
    },
    {
      "name": "VlvSearchStrategy",
      "diagnostic": "not applicable"
    },
    {
      "name": "AttributeIndexSearchStrategy",
      "filter": {
        "query": "UNION",
        "rank": "RANGE_MATCH",
        "filter": "(|(initials=aa*)(initials=zz*))",
        "subQueries": [
          {
            "query": "ANY_OF",
            "rank": "RANGE_MATCH",
            "filter": "(initials=aa*)",
            "subQueries": [
              {
                "query": "ANY_OF",
                "rank": "RANGE_MATCH",
                "filter": "(initials=aa*)",
                "subQueries": [
                  {
                    "query": "RANGE_MATCH",
                    "rank": "RANGE_MATCH",
                    "index": "initials.caseIgnoreMatch",
                    "range": "[aa,ab[",
                    "diagnostic": "indexed",
                    "candidates": 378
                  },
                  {
                    "query": "RANGE_MATCH",
                    "rank": "RANGE_MATCH",
                    "index": "initials.caseIgnoreSubstringsMatch:6",
                    "range": "[aa,ab[",
                    "diagnostic": "skipped"
                  }
                ],
                "diagnostic": "indexed",
```

```
                "candidates": 378
              },
              {
                "query": "MATCH_ALL",
                "rank": "MATCH_ALL",
                "filter": "(initials=aa*)",
                "index": "initials.presence",
                "diagnostic": "skipped"
              }
            ],
            "diagnostic": "indexed",
            "candidates": 378
          },
          {
            "query": "ANY_OF",
            "rank": "RANGE_MATCH",
            "filter": "(initials=zz*)",
            "subQueries": [
              {
                "query": "ANY_OF",
                "rank": "RANGE_MATCH",
                "filter": "(initials=zz*)",
                "subQueries": [
                  {
                    "query": "RANGE_MATCH",
                    "rank": "RANGE_MATCH",
                    "index": "initials.caseIgnoreMatch",
                    "range": "[zz,z{[",
                    "diagnostic": "indexed",
                    "candidates": 26
                  },
                  {
                    "query": "RANGE_MATCH",
                    "rank": "RANGE_MATCH",
                    "index": "initials.caseIgnoreSubstringsMatch:6",
                    "range": "[zz,z{[",
                    "diagnostic": "skipped"
                  }
                ],
                "diagnostic": "indexed",
                "candidates": 26
              },
              {
                "query": "MATCH_ALL",
                "rank": "MATCH_ALL",
                "filter": "(initials=zz*)",
                "index": "initials.presence",
                "diagnostic": "skipped"
              }
            ],
            "diagnostic": "indexed",
            "candidates": 26
          }
        ],
        "diagnostic": "indexed",
        "candidates": 404
      },
      "scope": {
        "type": "sub",
        "diagnostic": "not indexed",
        "candidates": "[NOT-INDEXED]",
        "retained": 404
```

```
        },
        "diagnostic": "indexed",
        "candidates": 404
      }
    ],
    "final": 404
  }
```

Notice that the server can narrow the list of candidates using the equality index you created. The server would require a substring index instead of an equality index if the filter were not matching initial strings.

If an index already exists, but you suspect it is not working properly, see Verify indexes.

Unused indexes

DS maintains metrics about index use. The metrics indicate how often an index was accessed since the DS server started.

The following examples demonstrate how to read the metrics for all monitored indexes:

LDAP

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN cn=monitor \
 "(objectClass=ds-monitor-backend-index)" ds-mon-index ds-mon-index-uses
```

Prometheus

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null |
grep index_uses
```

If the number of index uses is persistently zero, then you can eventually conclude the index is unused. Of course, it is possible that an index is needed, but has not been used since the last server restart. Be sure to sample often enough that you know the indexed is unused before taking action.

You can remove unused indexes with one of the following commands, depending on the type of index:

- `dsconfig delete-backend-index`
- `dsconfig delete-backend-vlv-index`

Index types

DS directory servers support multiple index types, each corresponding to a different type of search.

View what is indexed by using the `backendstat list-indexes` command. For details about a particular index, you can use the `backendstat dump-index` command.

Presence index

A presence index matches an attribute that is present on the entry, regardless of the value. By default, the `aci` attribute is indexed for presence:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(aci=*)" \
 aci
```

A presence index takes up less space than other indexes. In a presence index, there is just one key with a list of IDs.

The following command examines the ACI presence index for a server configured with the evaluation profile:

```
$ stop-ds

$ backendstat \
 dump-index \
 --backendId dsEvaluation \
 --baseDn dc=example,dc=com \
 --indexName aci.presence

Key (len 1): PRESENCE
Value (len 3): [COUNT:2] 1 9

Total Records: 1
Total / Average Key Size: 1 bytes / 1 bytes
Total / Average Data Size: 3 bytes / 3 bytes
```

In this case, entries with ACI attributes have IDs `1` and `9`.

Equality index

An equality index matches values that correspond exactly (generally ignoring case) to those in search filters. An equality index requires clients to match values without wildcards or misspellings:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 mail

dn: uid=bjensen,ou=People,dc=example,dc=com
mail: bjensen@example.com
```

An equality index has one list of entry IDs for each attribute value. Depending on the backend implementation, the keys in a case-insensitive index might not be strings. For example, a key of `6A656E73656E` could represent `jensen`.

The following command examines the SN equality index for a server configured with the evaluation profile:

```
$ stop-ds

$ backendstat \
 dump-index \
 --backendID dsEvaluation \
 --baseDN dc=example,dc=com \
 --indexName sn.caseIgnoreMatch | grep -A 1 "jensen$"

Key (len 6): jensen
Value (len 26): [COUNT:17] 18 31 32 66 79 94 133 134 150 5996 19415 32834 46253 59672 73091 86510 99929
```

In this case, there are 17 entries that have an SN of Jensen.

Unless the keys are encrypted, the server can reuse an equality index for ordering and initial substring searches.

Approximate index

An approximate index matches values that "sound like" those provided in the filter. An approximate index on `sn` lets client applications find people even when they misspell surnames:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(&(sn~=Jansen)(cn=Babs*))" \
 cn

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
```

An approximate index squashes attribute values into a normalized form.

The following command examines an SN approximate index added to a server configured with the evaluation profile:

```
$ stop-ds

$ backendstat \
 dump-index \
 --backendID dsEvaluation \
 --baseDN dc=example,dc=com \
 --indexName sn.ds-mr-double-metaphone-approx | grep -A 1 "JNSN$"

Key (len 4): JNSN
Value (len 83): [COUNT:74] 18 31 32 59 66 79 94 133 134 150 5928 5939 5940 5941 5996 5997 6033 6034 19347 19358 19359
19360 19415 19416 19452 19453 32766 32777 32778 32779 32834 32835 32871 32872 46185 46196 46197 46198 46253 46254
46290 46291 59604 59615 59616 59617 59672 59673 59709 59710 73023 73034 73035 73036 73091 73092 73128 73129 86442
86453 86454 86455 86510 86511 86547 86548 99861 99872 99873 99874 99929 99930 99966 99967
```

In this case, there are 74 entries that have an SN that sounds like Jensen.

Substring index

A substring index matches values that are specified with wildcards in the filter. Substring indexes can be expensive to maintain, especially for large attribute values:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(cn=Barb*)" \
 cn

dn: uid=bfrancis,ou=People,dc=example,dc=com
cn: Barbara Francis

dn: uid=bhal2,ou=People,dc=example,dc=com
cn: Barbara Hall

dn: uid=bjablons,ou=People,dc=example,dc=com
cn: Barbara Jablonski

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen

dn: uid=bmaddox,ou=People,dc=example,dc=com
cn: Barbara Maddox
```

In a substring index, there are enough keys to match any substring in the attribute values. Each key is associated with a list of IDs. The default maximum size of a substring key is 6 bytes.

The following command examines an SN substring index for a server configured with the evaluation profile:

```
$ stop-ds

$ backendstat \
dump-index \
 --backendID dsEvaluation \
 --baseDN dc=example,dc=com \
 --indexName sn.caseIgnoreSubstringsMatch:6

...
Key (len 1): e
Value (len 25): [COUNT:22] ...
...
Key (len 2): en
Value (len 15): [COUNT:12] ...
...
Key (len 3): ens
Value (len 3): [COUNT:1] 147
Key (len 5): ensen
Value (len 10): [COUNT:9] 18 31 32 66 79 94 133 134 150
...
Key (len 6): jensen
Value (len 10): [COUNT:9] 18 31 32 66 79 94 133 134 150
...
Key (len 1): n
Value (len 35): [COUNT:32] ...
...
Key (len 2): ns
Value (len 3): [COUNT:1] 147
Key (len 4): nsen
Value (len 10): [COUNT:9] 18 31 32 66 79 94 133 134 150
...
Key (len 1): s
Value (len 13): [COUNT:12] 12 26 47 64 95 98 108 131 135 147 149 154
...
Key (len 2): se
Value (len 7): [COUNT:6] 52 58 75 117 123 148
Key (len 3): sen
Value (len 10): [COUNT:9] 18 31 32 66 79 94 133 134 150
...
```

In this case, the SN value Jensen shares substrings with many other entries. The size of the lists and number of keys make a substring index much more expensive to maintain than other indexes. This is particularly true for longer attribute values.

Ordering index

An ordering index is used to match values for a filter that specifies a range. For example, the `ds-sync-hist` attribute used by replication has an ordering index by default. Searches on that attribute often seek entries with changes more recent than the last time a search was performed.

The following example shows a search that specifies a range on the SN attribute value:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(sn>=zyw)" \
 sn

dn: uid=user.13401,ou=People,dc=example,dc=com
sn: Zywiel

dn: uid=user.26820,ou=People,dc=example,dc=com
sn: Zywiel

dn: uid=user.40239,ou=People,dc=example,dc=com
sn: Zywiel

dn: uid=user.53658,ou=People,dc=example,dc=com
sn: Zywiel

dn: uid=user.67077,ou=People,dc=example,dc=com
sn: Zywiel

dn: uid=user.80496,ou=People,dc=example,dc=com
sn: Zywiel

dn: uid=user.93915,ou=People,dc=example,dc=com
sn: Zywiel
```

In this case, the server only requires an ordering index if it cannot reuse the (ordered) equality index instead. For example, if the equality index is encrypted, an ordering index must be maintained separately.

Big index

A big index is designed for attributes where many, many entries have the same attribute value.

This can happen for attributes whose values all belong to a known enumeration. For example, if you have a directory service with an entry for each person in the United States, the `st` (state) attribute is the same for more than 30 million Californians (`st: CA`). With a regular equality index, a search for `(st=CA)` would be unindexed. With a big index, the search is indexed, and optimized for paging through the results. For an example, refer to Indexes for attributes with few unique values.

A big index can be easier to configure and to use than a virtual list view index. Consider big indexes when:

- Many[1] entries have the same value for a given attribute.

  DS search performance with a big index is equivalent to search performance with a standard index. For attributes with only a few unique values, big indexes support much higher modification rates.

- Modifications outweigh searches for the attribute.

  When attributes have a wide range of possible values, favor standard indexes, except when the attribute is often the target of modifications, and only sometimes part of a search filter.

[1] Many, but not all entries. **Do not create a big index for all values of the `objectClass` attribute, for example.** When all entries have the same value for an attribute, as is the case for `objectClass: top`, indexes consume additional system resources and disk space with no benefit. The DS server must still read every entry to return search results. In practice, the upper limit is probably somewhat less than half the total entries. In other words, if half the entries have the same value for an attribute, it will cost more to maintain the big index than to evaluate all entries to find matches for the search. Let such searches remain *unindexed* searches.

Virtual list view (browsing) index

A virtual list view (VLV) or browsing index is designed to help applications that list results. For example, a GUI application might let users browse through a list of users. VLVs help the server respond to clients that request server-side sorting of the search results.

VLV indexes correspond to particular searches. Configure your VLV indexes using the command line.

Extensible matching rule index

In some cases, you need an index for a matching rule other than those described above.

For example, a generalized time-based matching index lets applications find entries with a time-based attribute later or earlier than a specified time.

Indexing tools

Command	Use this to...
`backendstat`	Drill down into the details and inspect index contents during debugging.
`dsconfig`	Configure indexes, and change their settings.
`rebuild-index`	Build a new index. Rebuild an index after changing its settings, or if the index has errors.
`verify-index`	Check an index for errors if you suspect there's a problem.

Configure indexes

You modify index configurations by using the `dsconfig` command. Configuration changes take effect after you rebuild the index with the new configuration, using the `rebuild-index` command. The `dsconfig --help-database` command lists subcommands for creating, reading, updating, and deleting index configuration.

> **Tip**
>
> Indexes are per directory backend rather than per base DN. To maintain separate indexes for different base DNs on the same server, put the entries in different backends.

Standard indexes

New index

The following example creates a new equality index for the `description` attribute:

```
$ dsconfig \
 create-backend-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name description \
 --set index-type:equality \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Approximate index

The following example adds an approximate index for the `sn` (surname) attribute:

```
$ dsconfig \
 set-backend-index-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name sn \
 --add index-type:approximate \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Approximate indexes depend on the Double Metaphone matching rule.

Extensible match index

DS servers support matching rules defined in LDAP RFCs. They also define DS-specific extensible matching rules.

The following are DS-specific extensible matching rules:

Name: `ds-mr-double-metaphone-approx`

Double Metaphone Approximate Match described at http://aspell.net/metaphone/. The DS implementation always produces a single value rather than one or possibly two values.

Configure approximate indexes as described in Approximate index.

For an example using this matching rule, see Approximate match.

Name: `ds-mr-user-password-exact`

User password exact matching rule used to compare encoded bytes of two hashed password values for exact equality.

Name: `ds-mr-user-password-equality`

User password matching rule implemented as the user password exact matching rule.

Name: `partialDateAndTimeMatchingRule`

Partial date and time matching rule for matching parts of dates in time-based searches.

For an example using this matching rule, see Active accounts.

Name: `relativeTimeOrderingMatch.gt`

Greater-than relative time matching rule for time-based searches.

For an example using this matching rule, see Active accounts.

Name: `relativeTimeOrderingMatch.lt`

Less-than relative time matching rule for time-based searches.

For an example using this matching rule, see Active accounts.

The following example configures an extensible matching rule index for "later than" and "earlier than" generalized time matching on the `ds-last-login-time` attribute:

```
$ dsconfig \
 create-backend-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --set index-type:extensible \
 --set index-extensible-matching-rule:1.3.6.1.4.1.26027.1.4.5 \
 --set index-extensible-matching-rule:1.3.6.1.4.1.26027.1.4.6 \
 --index-name ds-last-login-time \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Notice that the `index-extensible-matching-rule` setting takes an OID, not the name of the matching rule.

Indexes for attributes with few unique values

As described in Big index, big indexes fit the case where many, many entries have the same attribute value.

The default DS evaluation profile generates 100,000 user entries with addresses in the United States, so some `st` (state) attributes are shared by 4000 or more users. With a regular equality index, searches for some states reach the index entry limit, causing unindexed searches. A big index avoids this problem.

The following commands configure an equality index for the state attribute, and then build the new index:

```
$ dsconfig \
 create-backend-index \
 --backend-name dsEvaluation \
 --index-name st \
 --set index-type:big-equality \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ rebuild-index \
 --baseDn dc=example,dc=com \
 --index st \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

Once the index is ready, a client application can page through all the users in a state with an indexed search:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 --simplePageSize 5 \
 "(st=CA)" \
 cn
dn: uid=user.9,ou=People,dc=example,dc=com
cn: Abbe Abbate

dn: uid=user.123,ou=People,dc=example,dc=com
cn: Aili Aksel

dn: uid=user.124,ou=People,dc=example,dc=com
cn: Ailina Akyurekli

dn: uid=user.132,ou=People,dc=example,dc=com
cn: Ainslee Alary

dn: uid=user.264,ou=People,dc=example,dc=com
cn: Alphen Angell

Press RETURN to continue
```

When creating a big index that uses an extensible matching rule, model your work on the example in Extensible match index, but use the following options to set the index type and the matching rule:

- `--set index-type:big-extensible`
- `--set big-index-matching-rule:OID`

  Notice that the `big-index-matching-rule` setting takes an OID, not the name of the matching rule.

  The *OID* must specify an equality matching rule for big indexes. For example, if you create a big index for the `sn` (surname) attribute with `caseIgnoreMatch`, use `--set big-index-matching-rule:2.5.13.2`.

Custom indexes for JSON

DS servers support attribute values that have JSON syntax. The following schema excerpt defines a `json` attribute with case-insensitive matching:

```
attributeTypes: ( json-attribute-oid NAME 'json'
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1 EQUALITY caseIgnoreJsonQueryMatch
  X-ORIGIN 'DS Documentation Examples' )
```

When you index a JSON attribute defined in this way, the default directory server behavior is to maintain index keys for each JSON field. Large or numerous JSON objects can result in large indexes, which is wasteful. If you know which fields are used in search filters, you can choose to index only those fields.

As described in Schema and JSON, for some JSON objects only a certain field or fields matter when comparing for equality. In these special cases, the server can ignore other fields when checking equality during updates, and you would not maintain indexes for other fields.

How you index a JSON attribute depends on the matching rule in the attribute's schema definition, and on the JSON fields you expect to be used as search keys for the attribute.

Index JSON attributes

The examples that follow demonstrate these steps:

1. Using the schema definition and the information in the following table, configure a custom schema provider for the attribute's matching rule, if necessary.

Matching Rule in Schema Definition	Fields in Search Filter	Custom Schema Provider Required?
`caseExactJsonQueryMatch` `caseIgnoreJsonQueryMatch`	Any JSON field	No
Custom JSON query matching rule	Specific JSON field or fields	Yes, see JSON query matching rule index
Custom JSON equality or ordering matching rule	Specific field(s)	Yes, see JSON equality matching rule index

   A custom schema provider applies to all attributes using this matching rule.

2. Add the schema definition for the JSON attribute.
3. Configure the index for the JSON attribute.
4. Add the JSON attribute values in the directory data.

JSON query matching rule index

This example illustrates the steps in Index JSON attributes.

> **Note**
>
> If you installed a directory server with the `ds-evaluation` profile, the custom index configuration is already present.

The following command configures a custom, case-insensitive JSON query matching rule. This only maintains keys for the `access_token` and `refresh_token` fields:

```
$ dsconfig \
 create-schema-provider \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --provider-name "Custom JSON Query Matching Rule" \
 --type json-query-equality-matching-rule \
 --set enabled:true \
 --set case-sensitive-strings:false \
 --set ignore-white-space:true \
 --set matching-rule-name:caseIgnoreOAuth2TokenQueryMatch \
 --set matching-rule-oid:1.3.6.1.4.1.36733.2.1.4.1.1 \
 --set indexed-field:access_token \
 --set indexed-field:refresh_token \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The following commands add schemas for a `oauth2Token` attribute that uses the matching rule:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( oauth2token-attribute-oid NAME 'oauth2Token'
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1 EQUALITY caseIgnoreOAuth2TokenQueryMatch
  SINGLE-VALUE X-ORIGIN 'DS Documentation Examples' )
-
add: objectClasses
objectClasses: ( oauth2token-attribute-oid NAME 'oauth2TokenObject' SUP top
  AUXILIARY MAY ( oauth2Token ) X-ORIGIN 'DS Documentation Examples' )
EOF
```

The following command configures an index using the custom matching rule implementation:

```
$ dsconfig \
 create-backend-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name oauth2Token \
 --set index-type:equality \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

For an example of how a client application could use this index, see JSON query filters.

JSON equality matching rule index

This example illustrates the steps in Index JSON attributes.

> **Note**
>
> If you installed a directory server with the `ds-evaluation` profile, the custom index configuration is already present.

The following command configures a custom, case-insensitive JSON equality matching rule, `caseIgnoreJsonTokenIdMatch`:

```
$ dsconfig \
 create-schema-provider \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --provider-name "Custom JSON Token ID Matching Rule" \
 --type json-equality-matching-rule \
 --set enabled:true \
 --set case-sensitive-strings:false \
 --set ignore-white-space:true \
 --set matching-rule-name:caseIgnoreJsonTokenIDMatch \
 --set matching-rule-oid:1.3.6.1.4.1.36733.2.1.4.4.1 \
 --set json-keys:id \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Notice that this example defines a matching rule with OID `1.3.6.1.4.1.36733.2.1.4.4.1`. In production deployments, use a numeric OID allocated for your own organization.

The following commands add schemas for a `jsonToken` attribute, where the unique identifier is in the "id" field of the JSON object:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( jsonToken-attribute-oid NAME 'jsonToken'
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1 EQUALITY caseIgnoreJsonTokenIDMatch
  SINGLE-VALUE X-ORIGIN 'DS Documentation Examples' )
-
add: objectClasses
objectClasses: ( json-token-object-class-oid NAME 'JsonTokenObject' SUP top
  AUXILIARY MAY ( jsonToken ) X-ORIGIN 'DS Documentation Examples' )
EOF
```

The following command configures an index using the custom matching rule implementation:

```
$ dsconfig \
 create-backend-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name jsonToken \
 --set index-type:equality \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

For an example of how a client application could use this index, see JSON assertions.

Virtual list view index

The following example shows how to create a VLV index. This example applies where GUI users browse user accounts, sorting on surname then given name:

```
$ dsconfig \
 create-backend-vlv-index \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name people-by-last-name \
 --set base-dn:ou=People,dc=example,dc=com \
 --set filter:"(|(givenName=*)(sn=*))" \
 --set scope:single-level \
 --set sort-order:"+sn +givenName" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Note

When referring to a VLV index after creation, you must add `vlv.` as a prefix. In other words, if you named the VLV index `people-by-last-name`, refer to it as `vlv.people-by-last-name` when rebuilding indexes, changing index properties such as the index entry limit, or verifying indexes.

VLV for paged server-side sort

A special VLV index lets the server return sorted results. For example, users page through an entire directory database in a GUI. The user does not filter the data before seeing what is available.

The VLV index must have the following characteristics:

- Its filter must be "always true," `(&)`.
- Its scope must cover the search scope of the requests.

- Its base DN must match or be a parent of the base DN of the search requests.
- Its sort order must match the sort keys of the requests in the order they occur in the requests, starting with the first sort key used in the request.

  For example, if the sort order of the VLV index is `+l +sn +cn`, then it works with requests having the following sort orders:

  - `+l +sn +cn`
  - `+l +sn`
  - `+l`
  - Or none for single-level searches.

  The VLV index sort order can include additional keys not present in a request.

The following example commands demonstrate creating and using a VLV index to sort paged results by locality, surname, and then full name. The `l` attribute is not indexed by default. This example makes use of the `rebuild-index` command described below. The directory superuser is not subject to resource limits on the LDAP search operation:

```
$ dsconfig \
 create-backend-vlv-index \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name by-name \
 --set base-dn:ou=People,dc=example,dc=com \
 --set filter:"(&)" \
 --set scope:subordinate-subtree \
 --set sort-order:"+l +sn +cn" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ rebuild-index \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --baseDn dc=example,dc=com \
 --index vlv.by-name \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --bindPassword password \
 --baseDn dc=example,dc=com \
 --sortOrder +l,+sn,+cn \
 --simplePageSize 5 \
 "(&)" \
 cn l sn

dn: uid=user.93953,ou=People,dc=example,dc=com
cn: Access Abedi
l: Abilene
sn: Abedi

dn: uid=user.40283,ou=People,dc=example,dc=com
cn: Achal Abernathy
l: Abilene
sn: Abernathy

dn: uid=user.67240,ou=People,dc=example,dc=com
cn: Alaine Alburger
l: Abilene
sn: Alburger

dn: uid=user.26994,ou=People,dc=example,dc=com
cn: Alastair Alexson
l: Abilene
sn: Alexson
```

```
dn: uid=user.53853,ou=People,dc=example,dc=com
cn: Alev Allen
l: Abilene
sn: Allen

Press RETURN to continue ^C
```

Rebuild indexes

When you first import directory data, the directory server builds the indexes as part of the import process. DS servers maintain indexes automatically, updating them as directory data changes.

Only rebuild an index manually when it is necessary to do so. Rebuilding valid indexes wastes server resources, and is disruptive for client applications.

> **Important**
>
> When you rebuild an index while the server is online, the index appears as degraded and unavailable while the server rebuilds it.
> A search request that relies on an index in this state *may temporarily fail as an unindexed search*.

However, you must manually intervene when you:

- Create a new index for a new directory attribute.
- Create a new index for existing directory attribute.
- Change the server configuration in a way that affects the index, for example, by changing Index entry limits.
- Verify an existing index, and find that it has errors or is not in a valid state.

Automate index rebuilds

To automate the process of rebuilding indexes, use the `--rebuildDegraded` option. This rebuilds only degraded indexes, and does not affect valid indexes:

```
$ rebuild-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --rebuildDegraded \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

Clear a new index for a new attribute

When you add a new attribute, as described in Update LDAP schema, and create an index for the new attribute, the new index appears as *degraded* or *invalid*. The attribute has not yet been used, and so the index is sure to be empty, not degraded.

In this case, you can safely use the `rebuild-index --clearDegradedState` command. The server can complete this operation quickly, because there is no need to scan the entire directory backend to rebuild a new, unused index.

In this example, an index has just been created for `newUnusedAttribute`. *If the newly indexed attribute has already been used, rebuild the index instead of clearing the degraded state.*

Before using the `rebuild-index` command, test the index status to make sure is has not been used:

1. DS servers must be stopped before you use the backendstat command:

```
$ stop-ds
```

2. Check the status of the index(es).

   The indexes are prefixed with `!` before the rebuild:

```
$ backendstat \
 show-index-status \
 --backendID dsEvaluation \
 --baseDN dc=example,dc=com \
 | grep newUnusedAttribute

!  newUnusedAttribute.caseIgnoreMatch             ...
!  newUnusedAttribute.caseIgnoreSubstringsMatch:6 ...
!  newUnusedAttribute.presence                    ...
```

3. Update the index information to fix the value of the unused index:

```
$ rebuild-index \
 --offline \
 --baseDN dc=example,dc=com \
 --clearDegradedState \
 --index newUnusedAttribute
```

   Alternatively, you can first start the server, and perform this operation while the server is online.

4. (Optional) With the server offline, check that the `!` prefix has disappeared:

```
$ backendstat \
 show-index-status \
 --backendID dsEvaluation \
 --baseDN dc=example,dc=com \
 | grep newUnusedAttribute

newUnusedAttribute.caseIgnoreMatch             ...
newUnusedAttribute.caseIgnoreSubstringsMatch:6 ...
newUnusedAttribute.presence                    ...
```

5. Start the server if you have not done so already:

```
$ start-ds
```

Rebuild an index

When you make a change that affects an index configuration, manually rebuild the index.

Individual indexes appear as degraded and are unavailable while the server rebuilds them. A search request that relies on an index in this state *may temporarily fail as an unindexed search*.

The following example rebuilds a degraded `cn` index immediately with the server online.

While the server is rebuilding the `cn` index, *search requests that would normally succeed may fail*:

```
$ rebuild-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --index cn \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

Avoid rebuilding all indexes at once

Rebuilding multiple non-degraded indexes at once is disruptive, and not recommended unless some change has affected all indexes.

If you use the `--rebuildAll` option, first take the backend offline, stop the server, or, at minimum, make sure that no applications connect to the server while it is rebuilding indexes.

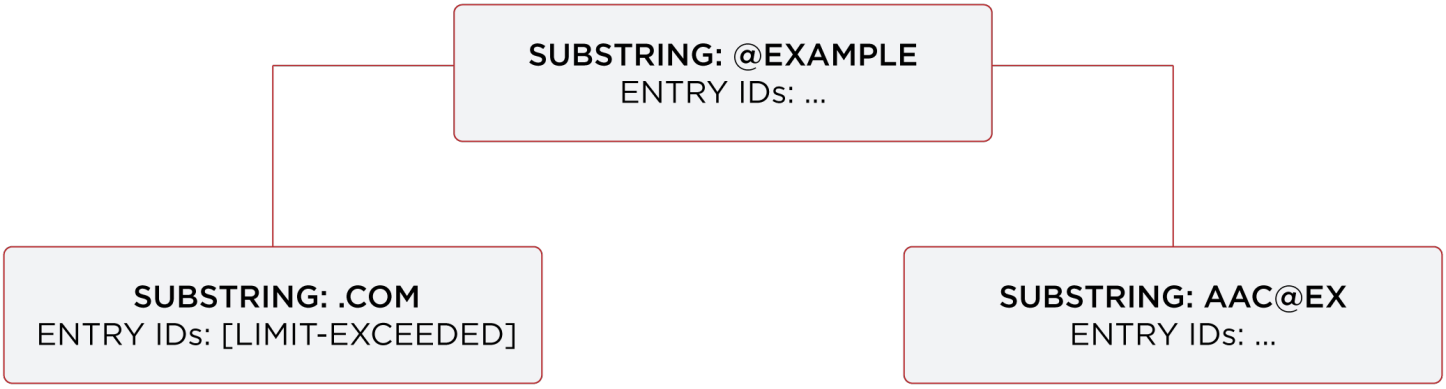
Index entry limits

An index is a tree of key-value pairs. The key is what the search is trying to match. The value is a list of entry IDs.

As the number of entries in the directory grows, the list of entry IDs for some keys can become very large. For example, every entry in the directory has `objectClass: top`. If the directory maintains a substring index for `mail`, the number of entries ending in `.com` could be huge.

A directory server therefore defines an *index entry limit*. When the number of entry IDs for a key exceeds the limit, the server stops maintaining a list of IDs for that key. The limit effectively means a search using only that key is unindexed. Searches using other keys in the same index are not affected.

The following figure shows a fragment from a substring index for the `mail` attribute. The number of email addresses ending in `com` has exceeded the index entry limit. For the other substring keys, the entry ID lists are still maintained. To save space, the entry IDs are not shown in the figure.



Ideally, the limit is set at the point where it becomes more expensive to maintain the entry ID list for a key, and to perform an indexed search than to perform an unindexed search. In practice, the limit is a tradeoff, with a default index entry limit value of 4000. Keep the default setting unless you have good reason to change it.

Check index entry limits

The following steps show how to get information about indexes where the index entry limit is exceeded for some keys. In this case, the directory server holds 100,000 user entries.

Use the `backendstat show-index-status` command:

1. Stop DS servers before you use the `backendstat` command:

   ```
   $ stop-ds
   ```

2. Non-zero values in the **Over** column of the output table indicate the number of keys for which the `index-entry-limit` setting has been exceeded. The keys that are over the limit are then listed below the table:

```
$ backendstat show-index-status --backendID dsEvaluation --baseDN dc=example,dc=com

  Index Name                                          ...Over  Entry Limit...
------------------------------------------------------...------------------...
  ...                                                 ...
  cn.caseIgnoreSubstringsMatch:6                      ...  14         4000...
  ...                                                 ...
  givenName.caseIgnoreSubstringsMatch:6               ...   9         4000...
  ...                                                 ...
  mail.caseIgnoreIA5SubstringsMatch:6                 ...  31         4000...
  ...                                                 ...
  objectClass.objectIdentifierMatch                   ...   4         4000...
  ...                                                 ...
  sn.caseIgnoreSubstringsMatch:6                      ...  14         4000...
  ...                                                 ...
  telephoneNumber.telephoneNumberSubstringsMatch:6    ...  10         4000...
  ...

Index: mail.caseIgnoreIA5SubstringsMatch:6
Over index-entry-limit keys: [.com] [0@exam] ...

Index: cn.caseIgnoreSubstringsMatch:6
Over index-entry-limit keys: [a] [an] [e] [er] [i] [k] [l] [n] [o] [on] [r] [s] [t] [y]

Index: givenName.caseIgnoreSubstringsMatch:6
Over index-entry-limit keys: [a] [e] [i] [ie] [l] [n] [na] [ne] [y]

Index: telephoneNumber.telephoneNumberSubstringsMatch:6
Over index-entry-limit keys: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

Index: sn.caseIgnoreSubstringsMatch:6
Over index-entry-limit keys: [a] [an] [e] [er] [i] [k] [l] [n] [o] [on] [r] [s] [t] [y]

Index: objectClass.objectIdentifierMatch
Over index-entry-limit keys: [inetorgperson] [organizationalperson] [person] [top]
```

For example, every user entry has the object classes listed, and every user entry has an email address ending in `.com`, so those values are not specific enough to be used in search filters.

A non-zero value in the **Over** column represents a tradeoff. As described above, this is usually a good tradeoff, not a problem to be solved.

For a detailed explanation of each column of the output, see backendstat show-index-status.

3. Start the server:

```
$ start-ds
```

On over index-entry-limit keys

The settings for this directory server are a good tradeoff. Unless you are seeing many unindexed searches that specifically target keys in the `Over index-entry-limit keys` lists, there's no reason to change the `index-entry-limit` settings.

A search might be indexed even though some keys are over the limit.

For example, as shown above, the `objectClass` value `inetorgperson` is over the limit. Yet, a search with a filter like `(&(cn=Babs Jensen)(objectclass=inetOrgPerson))` is indexed:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=user.1,ou=people,dc=example,dc=com \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(&(cn=Babs Jensen)(objectclass=inetOrgPerson))" cn
dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
```

The search is indexed because the equality index for `cn` is not over the limit, so the search term `(cn=Babs Jensen)` is enough for DS to find a match using that index.

If you look at the `debugsearchindex` output, you can see how DS uses the `cn` index, and skips the `objectclass` index. The overall search is clearly indexed:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(&(cn=Babs Jensen)(objectclass=inetOrgPerson))" \
 debugsearchindex | sed -n -e "s/^debugsearchindex: //p"
{
  "baseDn": "dc=example,dc=com",
  "scope": "sub",
  "filter": "(&(cn=Babs Jensen)(objectclass=inetOrgPerson))",
  "maxCandidateSize": 100000,
  "strategies": [{
    "name": "BaseObjectSearchStrategy",
    "diagnostic": "not applicable"
  }, {
    "name": "VlvSearchStrategy",
    "diagnostic": "not applicable"
  }, {
    "name": "AttributeIndexSearchStrategy",
    "filter": {
      "query": "INTERSECTION",
      "rank": "EXACT_MATCH",
      "filter": "(&(cn=Babs Jensen)(objectclass=inetOrgPerson))",
      "subQueries": [{
        "query": "ANY_OF",
        "rank": "EXACT_MATCH",
        "filter": "(cn=Babs Jensen)",
        "subQueries": [{
          "query": "EXACT_MATCH",
          "rank": "EXACT_MATCH",
          "filter": "(cn=Babs Jensen)",
          "index": "cn.caseIgnoreMatch",
          "key": "babs jensen",
          "diagnostic": "indexed",
          "candidates": 1
        }, {
          "query": "MATCH_ALL",
          "rank": "MATCH_ALL",
          "filter": "(cn=Babs Jensen)",
          "index": "cn.presence",
          "diagnostic": "skipped"
        }],
        "diagnostic": "indexed",
        "candidates": 1,
        "retained": 1
      }, {
        "query": "ANY_OF",
        "rank": "OBJECT_CLASS_EQUALITY_MATCH",
        "filter": "(objectclass=inetOrgPerson)",
        "subQueries": [{
          "query": "OBJECT_CLASS_EQUALITY_MATCH",
          "rank": "OBJECT_CLASS_EQUALITY_MATCH",
          "filter": "(objectclass=inetOrgPerson)",
          "subQueries": [{
            "query": "EXACT_MATCH",
```

```
              "rank": "EXACT_MATCH",
              "index": "objectClass.objectIdentifierMatch",
              "key": "inetorgperson",
              "diagnostic": "skipped"
            }, {
              "query": "EXACT_MATCH",
              "rank": "EXACT_MATCH",
              "index": "objectClass.objectIdentifierMatch",
              "key": "2.16.840.1.113730.3.2.2",
              "diagnostic": "skipped"
            }],
            "diagnostic": "skipped"
          }, {
            "query": "MATCH_ALL",
            "rank": "MATCH_ALL",
            "filter": "(objectclass=inetOrgPerson)",
            "index": "objectClass.presence",
            "diagnostic": "skipped"
          }],
          "diagnostic": "skipped"
        }],
        "diagnostic": "indexed",
        "candidates": 1
      },
      "diagnostic": "indexed",
      "candidates": 1
    }],
    "final": 1
  }
```


Index entry limit changes

In rare cases, the index entry limit might be too low for a certain key. This could manifest itself as a frequent, useful search becoming unindexed with no reasonable way to narrow the search.

You can change the index entry limit on a per-index basis. Do not do this in production unless you can explain and show why the benefits outweigh the costs.

> **Important**
>
> Changing the index entry limit significantly can result in serious performance degradation. Be prepared to test performance thoroughly before you roll out an index entry limit change in production.

To configure the `index-entry-limit` for an index or a backend:

- Use the `dsconfig set-backend-index-prop` command to change the setting for a specific backend index.
- (Not recommended) Use the `dsconfig set-backend-prop` command to change the setting for all indexes in the backend.

Verify indexes

You can verify that indexes correspond to current directory data, and do not contain errors. Use the verify-index command.

The following example verifies the `cn` index offline:

```
$ stop-ds

$ verify-index \
 --baseDN dc=example,dc=com \
 --index cn \
 --clean \
 --countErrors
```

The output indicates whether any errors are found in the index.

Debug a missing index

This example explains how you, as directory administrator, investigate an indexing problem.

How it looks to the application

In this example, an LDAP client application helps people look up names using mobile telephone numbers. The mobile numbers stored on the `mobile` attribute in the directory.

The LDAP client application sees a search for a mobile number failing with error 50:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=user.1,ou=people,dc=example,dc=com \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(mobile=14120300042)" cn
# The LDAP search request failed: 50 (Insufficient Access Rights)
# Additional Information:  You do not have sufficient privileges to perform an unindexed search
```

The application owner tells you there's a problem searching on mobile numbers.

How it looks to the administrator

As administrator, you can see the failures in the DS access logs. The following example includes only the relevant fields of an access log message with the failure:

```
{
  "request": {
    "operation": "SEARCH",
    "dn": "dc=example,dc=com",
    "scope": "sub",
    "filter": "(mobile=14120300042)"
  },
  "response": {
    "status": "FAILED",
    "statusCode": "50",
    "detail": "You do not have sufficient privileges to perform an unindexed search",
    "additionalItems": {
      "unindexed": null
    },
    "nentries": 0
  },
  "userId": "uid=user.1,ou=People,dc=example,dc=com"
}
```

For this simple filter, `(mobile=14120300042)`, if the search is unindexed, you can conclude that the attribute must not be indexed. As expected, the `mobile` attribute does not appear in the list of indexes for the backend:

```
$ dsconfig \
 list-backend-indexes \
 --backend-name dsEvaluation \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
Backend Index              : index-type             ...
---------------------------:------------------------...
aci                        : presence               ...
cn                         : equality, substring    ...
ds-certificate-fingerprint : equality               ...
ds-certificate-subject-dn  : equality               ...
ds-sync-conflict           : equality               ...
ds-sync-hist               : ordering               ...
entryUUID                  : equality               ...
givenName                  : equality, substring    ...
json                       : equality               ...
jsonToken                  : equality               ...
mail                       : equality, substring    ...
manager                    : equality, extensible   ...
member                     : equality               ...
oauth2Token                : equality               ...
objectClass                : big-equality, equality ...
sn                         : equality, substring    ...
telephoneNumber            : equality, substring    ...
uid                        : equality               ...
uniqueMember               : equality               ...
```

If the filter were more complex, you could use run the search with the `debugsearchindex` attribute to determine why it is unindexed.

You notice that `telephoneNumber` has equality and substring indexes, and decide to add the same for the `mobile` attribute. Adding a new index means adding the configuration for the index, and then building the index. An index is specific to a given server, so you do this for each DS replica. For example:

```
$ dsconfig \
 create-backend-index \
 --backend-name dsEvaluation \
 --index-name mobile \
 --type generic \
 --set index-type:equality \
 --set index-type:substring \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
$ rebuild-index \
 --index mobile \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

How the fix looks to the administrator

Once the index is built, you check that the search is now indexed by looking at the `debugsearchindex` output:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(mobile=14120300042)" \
 debugsearchindex | sed -n -e "s/^debugsearchindex: //p"
{
  "baseDn": "dc=example,dc=com",
  "scope": "sub",
  "filter": "(mobile=14120300042)",
  "maxCandidateSize": 100000,
  "strategies": [{
    "name": "BaseObjectSearchStrategy",
    "diagnostic": "not applicable"
  }, {
    "name": "VlvSearchStrategy",
    "diagnostic": "not applicable"
  }, {
    "name": "AttributeIndexSearchStrategy",
    "filter": {
      "query": "ANY_OF",
      "rank": "EXACT_MATCH",
      "filter": "(mobile=14120300042)",
      "subQueries": [{
        "query": "EXACT_MATCH",
        "rank": "EXACT_MATCH",
        "filter": "(mobile=14120300042)",
        "index": "mobile.telephoneNumberMatch",
        "key": "14120300042",
        "diagnostic": "indexed",
        "candidates": 1
      }, {
        "query": "MATCH_ALL",
        "rank": "MATCH_ALL",
        "filter": "(mobile=14120300042)",
        "index": "mobile.presence",
        "diagnostic": "skipped"
      }],
      "diagnostic": "indexed",
      "candidates": 1
    },
    "diagnostic": "indexed",
    "candidates": 1
  }],
  "final": 1
}
```

You tell the application owner to try searching on mobile numbers now that you have indexed the attribute.

How the fix looks to the application

The client application now sees that the search works:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=user.1,ou=people,dc=example,dc=com \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(mobile=14120300042)" cn
dn: uid=user.0,ou=People,dc=example,dc=com
cn: Aaccf Amar
```

More to do?

Must you do anything more? What about `index-entry-limit` settings, for example?

Stop the server, and run the `backendstat show-index-status` command:

```
$ stop-ds --quiet
$ backendstat show-index-status --backendID dsEvaluation --baseDN dc=example,dc=com
Index Name                                ...  Over  Entry Limit  Mean  ...
------------------------------------------...----------------------...--
...
mobile.telephoneNumberMatch               ...     0         4000     1  ...
mobile.telephoneNumberSubstringsMatch:6   ...    10         4000     2  ...
...

Index: mobile.telephoneNumberSubstringsMatch:6
Over index-entry-limit keys: [0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
...
```

Notice that the equality index `mobile.telephoneNumberMatch` has no keys whose entry ID lists are over the limit, and the average number of values is 1. This is what you would expect. Each mobile number belongs to a single user. DS uses this index for exact match search filters like `(mobile=14120300042)`.

Notice also that the substring index `mobile.telephoneNumberSubstringsMatch:6` has 10 keys whose lists are over the (default) limit of 4000 values. These are the keys for substrings that match only a single digit of a mobile number. For example, a search for all users whose mobile telephone number starts with `1` uses the filter `(mobile=1*)`. This search would be unindexed.

Should you raise the `index-entry-limit` for this substring index? Probably not, no.

The filter `(mobile=1*)` matches all mobile numbers for the United States and Canada, for example. Someone running this search is not looking up a user's name by their mobile phone number. They are scanning the directory database, even if it is not intentional. If you raise the `index-entry-limit` setting to prevent any `Over index-entry-limit keys`, the server must update enormous entry ID lists for these keys whenever a `mobile` attribute value changes. The impact on write performance could be a bad tradeoff.

If possible, suggest that the LDAP application refrains from searching until the user has provided enough digits of the mobile number to match, or that it prompts the user for more digits when it encounters an unindexed search.

If you cannot change the application, it might be acceptable that searches for a single mobile telephone digit simply fail. That might be a better tradeoff than impacting write performance due to a very high `index-entry-limit` setting.

Data storage

DS directory servers store data in *backends* . A backend is a private server repository that can be implemented in memory, as an LDIF file, or as an embedded database.

Embedded database backends have these characteristics:

Suitable for large numbers of entries

When creating a database backend, you choose the backend type. DS directory servers use JE backends for local data.

The JE backend type is implemented using B-tree data structures. It stores data as key-value pairs, which is different from the model used by relational databases.

JE backends are designed to hold hundreds of millions, or even billions of LDAP entries.

Fully managed by DS servers

Let the server manage its backends and their database files.

By default, backend database files are located under the `opendj/db` directory.

> **Warning**
>
> Do not compress, tamper with, or otherwise alter backend database files directly, unless specifically instructed to do so by a qualified technical support engineer. External changes to backend database files can render them unusable by the server.
> If you use snapshot technology for backup, read Back up using snapshots and Restore from a snapshot.

DS servers provide the `dsbackup` command for backing up and restoring database backends. For details, refer to Backup and restore.

Self-cleaning

A JE backend stores data on disk using append-only log files with names like `number.jdb` . The JE backend writes updates to the highest-numbered log file. The log files grow until they reach a specified size (default: 1 GB). When the current log file reaches the specified size, the JE backend creates a new log file.

To avoid an endless increase in database size on disk, JE backends clean their log files in the background. A cleaner thread copies active records to new log files. Log files that no longer contain active records are deleted.

Due to the cleanup processes, JE backends can actively write to disk, even when there are no pending client or replication operations.

Configurable I/O behavior

By default, JE backends let the operating system potentially cache data for a period of time before flushing the data to disk. This setting trades full durability with higher disk I/O for good performance with lower disk I/O.

With this setting, it is possible to lose the most recent updates that were not yet written to disk, in the event of an underlying OS or hardware failure.

You can modify this behavior by changing the advanced configuration settings for the JE backend. If necessary, you can change the advanced setting, db-durability, using the `dsconfig set-backend-prop` command.

Automatic recovery

When a JE backend is opened, it recovers by recreating its B-tree structure from its log files.

This is a normal process. It lets the backend recover after an orderly shutdown or after a crash.

Automatic verification

JE backends run checksum verification periodically on the database log files.

If the verification process detects backend database corruption, then the server logs an error message and takes the backend offline. If this happens, restore the corrupted backend from backup so that it can be used again.

By default, the verification runs every night at midnight local time. If necessary, you can change this behavior by adjusting the advanced settings, db-run-log-verifier and db-log-verifier-schedule, using the `dsconfig set-backend-prop` command.

Encryption support

JE backends support encryption for data confidentiality.

For details, refer to Data encryption.

Depending on the setup profiles used at installation time, DS servers have backends with the following default settings:

Backend	Type	Optional?(1)	Replicated?	Part of Backup?	Details
adminRoot	LDIF	No	If enabled	If enabled	Symmetric keys for (deprecated) reversible password storage schemes. Base DN: `cn=admin data` Base directory: `db/adminRoot`
amCts	JE	Yes	Yes	Yes	AM core token store (CTS) data. More information: Install DS for AM CTS. Base DN: `ou=tokens` Base directory: `db/amCts`
amIdentityStore	JE	Yes	Yes	Yes	AM identities. More information: Install DS for AM identities. Base DN: `ou=identities` Base directory: `db/amIdentityStore`

Backend	Type	Optional?[1]	Replicated?	Part of Backup?	Details
cfgStore	JE	Yes	Yes	Yes	AM configuration, policy, and other data. More information: Install DS for AM configuration. Base DN: `ou=am-config` Base directory: `db/cfgStore`
changelogDb	Changelog	Yes[2]	N/A	No	Change data for replication and change notifications. More information: Changelog for notifications. Base DN: `cn=changelog` Base directory: `changelogDb`
config	Config	No	N/A	No	File-based representation of this server's configuration. Do not edit `config/config.ldif` directly. Use the `dsconfig` command instead. Base DN: `cn=config` Base directory: `config`
dsEvaluation	JE	Yes	Yes	Yes	Example.com sample data. More information: Install DS for evaluation. Base DN: `dc=example,dc=com` Base directory: `db/dsEvaluation`
idmRepo	JE	Yes	Yes	Yes	IDM repository data. More information: Install DS as an IDM repository. Base DN: `dc=openidm,dc=forgerock,dc=com` Base directory: `db/idmRepo`
monitor	Monitor	No	N/A	N/A	Single entry; volatile monitoring metrics maintained in memory since server startup. More information: LDAP-based monitoring. Base DN: `cn=monitor` Base directory: N/A

Backend	Type	Optional?[1]	Replicated?	Part of Backup?	Details
monitorUser	LDIF	Yes	Yes	Yes	Single entry; default monitor user account. Base DN: `uid=Monitor` Base directory: `db/monitorUser`
rootUser	LDIF	No[3]	No	Yes	Single entry; default directory superuser account. Base DN: `uid=admin` Base directory: `db/rootUser`
root DSE	Root DSE	No	N/A	N/A	Single entry describing server capabilities. Use `ldapsearch --baseDn "" --searchScope base "(&)" +` to read all the (operational) attributes of this entry. Base DN: `""` (empty string) Base directory: N/A
schema	Schema	No	Yes	Yes	Single entry listing LDAP schema definitions. More information: LDAP schema. Base DN: `cn=schema` Base directory: `db/schema`
tasks	Task	No	N/A	Yes	Scheduled tasks for this server. Use the `manage-tasks` command. Base DN: `cn=tasks` Base directory: `db/tasks`
userData	JE	Yes	Yes	Yes	User entries imported from the LDIF you provide. More information: Install DS for user data. Base directory: `db/userData`

[1] Optional backends depend on the setup choices.

[2] The changelog backend is mandatory for servers with a replication server role.

[3] You must create a superuser account at setup time. You may choose to replace it later. For details, refer to Use a non-default superuser account.

Create a backend

When you create a new backend on a replicated directory server, let the server replicate the new data:

1. Configure the new backend.

   The following example creates a database backend for Example.org data. The backend relies on a JE database for data storage and indexing:

   ```
   $ dsconfig \
    create-backend \
    --hostname localhost \
    --port 4444 \
    --bindDn uid=admin \
    --bindPassword password \
    --backend-name exampleOrgBackend \
    --type je \
    --set enabled:true \
    --set base-dn:dc=example,dc=org \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

   When you create a new backend using the `dsconfig` command, DS directory servers create the following indexes automatically:

Index	Approx.	Equality	Ordering	Presence	Substring	Entry Limit
`aci`	-	-	-	Yes	-	4000
`dn2id`	Non-configurable internal index					
`ds-sync-conflict`	-	Yes	-	-	-	4000
`ds-sync-hist`	-	-	Yes	-	-	4000
`entryUUID`	-	Yes	-	-	-	4000
`id2children`	Non-configurable internal index					
`id2subtree`	Non-configurable internal index					
`objectClass`	-	Yes	-	-	-	4000

2. Verify that replication is enabled:

```
$ dsconfig \
 get-synchronization-provider-prop \
 --provider-name "Multimaster Synchronization" \
 --property enabled \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
Property : Value(s)
---------:---------
enabled  : true
```

If replication should be enabled but is not, use `dsconfig set-synchronization-provider-prop --set enabled:true` to enable it.

3. Let the server replicate the base DN of the new backend:

```
$ dsconfig \
 create-replication-domain \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --domain-name dc=example,dc=org \
 --type generic \
 --set enabled:true \
 --set base-dn:dc=example,dc=org \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

4. If you have existing data for the backend, follow the appropriate procedure to initialize replication.

   For details, refer to Manual initialization.

If you must temporarily disable replication for the backend, take care to avoid losing changes. For details, refer to Disable replication.

Import and export

The following procedures demonstrate how to import and export LDIF data.

For details on creating custom sample LDIF to import, refer to Generate test data.

Import LDIF

If you are initializing replicas by importing LDIF, refer to Initialize from LDIF for context.

Important

- Importing from LDIF overwrites all data in the target backend with entries from the LDIF data.
- If the LDIF was exported from another server, it may contain pre-encoded passwords. As long as DS supports the password storage schemes used to encode the passwords, you can enable the storage schemes in the configuration to migrate existing passwords into DS. For details, refer to Password storage.
  By default, password policies do not allow you to use pre-encoded passwords. You can change this behavior by changing the default password policy configuration property, `allow-pre-encoded-passwords`.
  The DS import process does not warn you about passwords that use disabled password storage schemes. Instead, search the LDIF to find all the password storage schemes in use, and make sure all the schemes are enabled in the server configuration. Users whose passwords are stored with a disabled scheme cannot bind successfully.
- LDIF from another server can include passwords encrypted with a reversible storage scheme, such as AES or Blowfish. To decrypt the passwords, the server must use the same deployment ID as the server that encrypted the passwords.

Perform either of the following steps to import `dc=example,dc=com` data into the `dsEvaluation` backend:

- To import offline, shut down the server before you run the `import-ldif` command:

```
$ stop-ds
$ import-ldif \
 --offline \
 --backendId dsEvaluation \
 --includeBranch dc=example,dc=com \
 --ldifFile example.ldif
```

- To import online, schedule a task:

```
$ start-ds
$ import-ldif \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --backendId dsEvaluation \
 --includeBranch dc=example,dc=com \
 --ldifFile example.ldif \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

You can schedule the import task to start at a particular time using the `--start` option.

Export LDIF

Perform either of the following steps to export `dc=example,dc=com` data from the `dsEvaluation` backend:

- To export offline, shut down the server before you run the `export-ldif` command:

```
$ stop-ds
$ export-ldif \
 --offline \
 --backendId dsEvaluation \
 --includeBranch dc=example,dc=com \
 --ldifFile backup.ldif
```

- To export online, schedule a task:

```
$ export-ldif \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --backendId dsEvaluation \
 --includeBranch dc=example,dc=com \
 --ldifFile backup.ldif \
 --start 0 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

The `--start 0` option tells the directory server to start the export task immediately.

You can specify a time for the task to start using the format yyyymmddHHMMSS. For example, `20250101043000` specifies a start time of 4:30 AM on January 1, 2025.

If the server is not running at the time specified, it attempts to perform the task after it restarts.

Split data

Splitting data impacts replication, and can require that you interrupt the service.

On one set of replicas, create the `exampleData` backend to hold all `dc=example,dc=com` data except `ou=People,dc=example,dc=com`:

```
# Create a backend for the data not under the subordinate base DN:
$ dsconfig \
 create-backend \
 --backend-name exampleData \
 --type je \
 --set enabled:true \
 --set base-dn:dc=example,dc=com \
 --offline \
 --no-prompt

# Let the server replicate the base DN of the new backend:
$ dsconfig \
 create-replication-domain \
 --provider-name "Multimaster Synchronization" \
 --domain-name dc=example,dc=com \
 --type generic \
 --set enabled:true \
 --set base-dn:dc=example,dc=com \
 --offline \
 --no-prompt

# Import data not under the subordinate base DN:
$ import-ldif \
 --backendId exampleData \
 --excludeBranch ou=People,dc=example,dc=com \
 --ldifFile example.ldif \
 --offline
```

On another set of replicas, create a `peopleData` backend to hold `ou=People,dc=example,dc=com` data:

```
# Create a backend for the data under the subordinate base DN:
$ dsconfig \
 create-backend \
 --backend-name peopleData \
 --type je \
 --set enabled:true \
 --set base-dn:ou=People,dc=example,dc=com \
 --offline \
 --no-prompt

# Let the server replicate the base DN of the new backend:
$ dsconfig \
 create-replication-domain \
 --provider-name "Multimaster Synchronization" \
 --domain-name ou=People,dc=example,dc=com \
 --type generic \
 --set enabled:true \
 --set base-dn:ou=People,dc=example,dc=com \
 --offline \
 --no-prompt

# Import data under the subordinate base DN:
$ import-ldif \
 --backendId peopleData \
 --includeBranch ou=People,dc=example,dc=com \
 --ldifFile example.ldif \
 --offline
```

After completing the data import process, start the replicas.

To split an *existing* backend, follow these high-level steps:

1. Stop the affected replica.
2. Create the new backend.
3. Export the data for the backend.
4. Import the data under the subordinate base DN into the new backend, using the `--includeBranch` option.
5. Delete all data under the subordinate base DN from the old backend.
6. Start the affected replica.

Disk space thresholds

Directory data growth depends on applications that use the directory. When directory applications add more data than they delete, the database backend grows until it fills the available disk space. The system can end up in an unrecoverable state if no disk space is available.

Database backends therefore have advanced properties, `disk-low-threshold` and `disk-full-threshold`. When available disk space falls below `disk-low-threshold`, the directory server only allows updates from users and applications that have the `bypass-lockdown` privilege. When available space falls below `disk-full-threshold`, the directory server stops allowing updates, instead returning an `UNWILLING_TO_PERFORM` error to each update request.

If growth across the directory service tends to happen quickly, set the thresholds higher than the defaults to allow more time to react when growth threatens to fill the disk. The following example sets `disk-low-threshold` to 10 GB `disk-full-threshold` to 5 GB for the `dsEvaluation` backend:

```
$ dsconfig \
 set-backend-prop \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --set "disk-low-threshold:10 GB" \
 --set "disk-full-threshold:5 GB" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The properties disk-low-threshold, and disk-full-threshold are listed as *advanced* properties.

To examine their values with the `dsconfig` command, use the `--advanced` option:

```
$ dsconfig \
 get-backend-prop \
 --advanced \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --property disk-low-threshold \
 --property disk-full-threshold \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Property            : Value(s)
--------------------:---------
disk-full-threshold : 5 gb
disk-low-threshold  : 10 gb
```

Entry expiration

If the directory service creates many entries that expire and should be deleted, you could find the entries with a time-based search and then delete them individually. That approach uses replication to delete the entries in all replicas. It has the disadvantage of generating potentially large amounts of replication traffic.

Entry expiration lets you configure the backend database to delete the entries as they expire. This approach deletes expired entries at the backend database level, without generating replication traffic. AM uses this approach when relying on DS to delete expired token entries, as demonstrated in Install DS for AM CTS.

Backend indexes for generalized time (timestamp) attributes have these properties to configure automated, optimized entry expiration and removal:

- ttl-enabled
- ttl-age

Configure this capability by performing the following steps:

1. Prepare an ordering index for a generalized time (timestamp) attribute on entries that expire.

   For details, refer to Configure indexes and Active accounts.

2. Using the `dsconfig set-backend-index-prop` command, set `ttl-enabled` on the index to true, and set `ttl-age` on the index to the desired entry lifetime duration.

3. Optionally enable the access log to record messages when the server deletes an expired entry.

   Using the `dsconfig set-log-publisher-prop` command, set `suppress-internal-operations:false` for the access log publisher. Note that this causes the server to log messages for all internal operations.

   When the server deletes an expired entry, it logs a message with `"additionalItems":{"ttl": null}` in the response.

Once you configure and build the index, the backend can delete expired entries. At intervals of 10 seconds, the backend automatically deletes entries whose timestamps on the attribute are older than the specified lifetime. Entries that expire in the interval between deletions are removed on the next round. Client applications should therefore check that entries have not expired, as it is possible for expired entries to remain available until the next round of deletions.

When using this capability, keep the following points in mind:

- Entry expiration is per index. The time to live depends on the value of the indexed attribute and the `ttl-age` setting, and all matching entries are subject to TTL.
- If multiple indexes' `ttl-enabled` and `ttl-age` properties are configured, as soon as one of the entry's matching attributes exceeds the TTL, the entry is eligible for deletion.
- The backend deletes the entries directly as an internal operation. The server only records the deletion when `suppress-internal-operations: false` for the access log publisher. Persistent searches do not return the deletion.

  Furthermore, this means that *deletion is not replicated*. To ensure expired entries are deleted on all replicas, use the same indexes with the same settings on all replicas.

- When a backend deletes an expired entry, the effect is a subtree delete. In other words, if a parent entry expires, the parent entry *and all the parent's child entries* are deleted.

  If you do not want parent entries to expire, index a generalized time attribute that is only present on its child entries.

- The backend deletes expired entries atomically.

  If you update the TTL attribute to prevent deletion and the update succeeds, then TTL has effectively been reset.

- Expiration fails when the `index-entry-limit` is exceeded. (For background information, refer to Index entry limits.)

  This only happens if the timestamp for the indexed attribute matches to the nearest millisecond on more than 4000 entries (for default settings). This corresponds to four million timestamp updates per second, which would be very difficult to reproduce in a real directory service.

It is possible, however, to construct and import an LDIF file where more than 4000 entries have the same timestamp. Make sure not to reuse the same timestamp for thousands of entries when artificially creating entries that you intend to expire.

Add a base DN to a backend

The following example adds the base DN `o=example` to the `dsEvaluation` backend, and creates a replication domain configuration to replicate the data under the new base DN:

```
$ dsconfig \
 set-backend-prop \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --add base-dn:o=example \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 get-backend-prop \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --property base-dn \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
Property : Value(s)
---------:------------------------------
base-dn  : "dc=example,dc=com", o=example

$ dsconfig \
 create-replication-domain \
 --provider-name "Multimaster Synchronization" \
 --domain-name o=example \
 --type generic \
 --set enabled:true \
 --set base-dn:o=example \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

When you add a base DN to a backend, the base DN must not be subordinate to any other base DNs in the backend. As in the commands shown here, you can add the base DN `o=example` to a backend that already has a base DN `dc=example,dc=com`. You cannot, however, add `o=example,dc=example,dc=com` as a base DN, because that is a child of `dc=example,dc=com`.

Delete a backend

When you delete a database backend with the `dsconfig delete-backend` command, the directory server does not actually remove the database files for these reasons:

- A mistake could potentially cause lots of data to be lost.
- Deleting a large database backend could cause severe service degradation due to a sudden increase in I/O load.

After you run the `dsconfig delete-backend` command, manually remove the database backend files, and remove replication domain configurations any base DNs you deleted by removing the backend.

If run the `dsconfig delete-backend` command by mistake, but have not yet deleted the actual files, recover the data by creating the backend again, and reconfiguring and rebuilding the indexes.

Groups

DS servers support several methods of grouping entries:

- Dynamic groups look up their membership based on an LDAP filter.
- Static groups list each member.

  Static groups are easy to start, but can become large and expensive to maintain.

  For static groups, you must have a mechanism to remove members whose entries are deleted, and members whose entries are modified in a way that ends their membership. DS servers use a *referential integrity* plugin for this.
- Virtual static groups use a dynamic group-style definition, but let applications list group members as if the group were static.

Tip

The examples that follow assume `ou=Groups,dc=example,dc=com` already exists. The `ds-evaluation` profile includes this entry by default. If you are using another profile, you can create the groups entry:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: ou=Groups,dc=example,dc=com
objectClass: organizationalunit
objectClass: top
ou: Groups
EOF
```

Dynamic groups

A *dynamic group* references members using LDAP URLs. Dynamic group entries take the `groupOfURLs` object class, with one or more `memberURL` values specifying LDAP URLs to identify group members.

Dynamic groups are a natural fit for directory servers. If you have a choice, choose dynamic groups over static groups for these reasons:

- Dynamic group membership is a property of a member's entry.

  There is no need to maintain a separate entry with the list of members.

- Determining dynamic group membership is as simple as applying the member URL criteria.

- Dynamic groups scale to any size without performance issues.

  Dynamic group entries remain small even when the group has a large number of members.

The following dynamic group includes entries matching the filter `"(l=San Francisco)"` (users whose location is San Francisco):

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=My Dynamic Group,ou=Groups,dc=example,dc=com
cn: My Dynamic Group
objectClass: top
objectClass: groupOfURLs
ou: Groups
memberURL: ldap:///ou=People,dc=example,dc=com??sub?l=San Francisco
EOF
```

Group membership changes dynamically as entries change to match the `memberURL` values:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(&(uid=*jensen)(isMemberOf=cn=My Dynamic Group,ou=Groups,dc=example,dc=com))" \
 1.1

dn: uid=bjensen,ou=People,dc=example,dc=com

dn: uid=rjensen,ou=People,dc=example,dc=com

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: uid=ajensen,ou=People,dc=example,dc=com
changetype: modify
replace: l
l: San Francisco
EOF

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(&(uid=*jensen)(isMemberOf=cn=My Dynamic Group,ou=Groups,dc=example,dc=com))" \
 1.1

dn: uid=ajensen,ou=People,dc=example,dc=com

dn: uid=bjensen,ou=People,dc=example,dc=com

dn: uid=rjensen,ou=People,dc=example,dc=com
```

Virtual static groups

DS servers let you create *virtual static groups*. Virtual static groups allow applications to see dynamic groups as if they had an enumerated list of members like a static group.

The virtual static group takes the auxiliary object class `ds-virtual-static-group`. Virtual static groups use the object class `groupOfNames`, or `groupOfUniqueNames`. Instead of `member` or `uniqueMember` attributes, they have `ds-target-group-dn` attributes pointing to other groups.

Generating the list of members can be resource-intensive for large groups. By default, you cannot retrieve the list of members. If you have an application that must read the list of members, change the configuration of `Virtual Static member` or `Virtual Static uniqueMember` to set `allow-retrieving-membership:true`:

```
$ dsconfig \
 set-virtual-attribute-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --name "Virtual Static member" \
 --set allow-retrieving-membership:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The following example creates a virtual static group, and reads the group entry with all members:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=Virtual Static,ou=Groups,dc=example,dc=com
cn: Virtual Static
objectclass: top
objectclass: groupOfNames
objectclass: ds-virtual-static-group
ds-target-group-dn: cn=My Dynamic Group,ou=Groups,dc=example,dc=com
EOF

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(cn=Virtual Static)"

dn: cn=Virtual Static,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
objectClass: ds-virtual-static-group
cn: Virtual Static
ds-target-group-dn: cn=My Dynamic Group,ou=Groups,dc=example,dc=com
member: uid=abergin,ou=People,dc=example,dc=com
member: uid=ajensen,ou=People,dc=example,dc=com
member: uid=aknutson,ou=People,dc=example,dc=com
member: uid=awalker,ou=People,dc=example,dc=com
member: uid=aworrell,ou=People,dc=example,dc=com
member: uid=bjensen,ou=People,dc=example,dc=com
member: uid=bplante,ou=People,dc=example,dc=com
member: uid=btalbot,ou=People,dc=example,dc=com
member: uid=cwallace,ou=People,dc=example,dc=com
member: uid=dakers,ou=People,dc=example,dc=com
member: uid=dthorud,ou=People,dc=example,dc=com
member: uid=ewalker,ou=People,dc=example,dc=com
member: uid=gfarmer,ou=People,dc=example,dc=com
member: uid=jbourke,ou=People,dc=example,dc=com
member: uid=jcampaig,ou=People,dc=example,dc=com
member: uid=jmuffly,ou=People,dc=example,dc=com
member: uid=jreuter,ou=People,dc=example,dc=com
member: uid=jwalker,ou=People,dc=example,dc=com
member: uid=kcarter,ou=People,dc=example,dc=com
member: uid=kschmith,ou=People,dc=example,dc=com
member: uid=mjablons,ou=People,dc=example,dc=com
member: uid=mlangdon,ou=People,dc=example,dc=com
member: uid=mschneid,ou=People,dc=example,dc=com
member: uid=mtalbot,ou=People,dc=example,dc=com
member: uid=mtyler,ou=People,dc=example,dc=com
member: uid=mwhite,ou=People,dc=example,dc=com
```

```
member: uid=pshelton,ou=People,dc=example,dc=com
member: uid=rjensen,ou=People,dc=example,dc=com
member: uid=smason,ou=People,dc=example,dc=com
member: uid=tlabonte,ou=People,dc=example,dc=com
member: uid=tschmith,ou=People,dc=example,dc=com
```

Static groups

A *static group* entry enumerates the entries in the group. Static group entries grow as their membership increases.

Large static groups are a performance bottleneck. If you have a choice, choose dynamic groups over static groups for these reasons:

- Static group membership is defined in a separate, potentially large LDAP entry.

  Large static group entries are expensive to maintain, cache, and replicate.

- Determining static group membership involves reading the group entry, or using virtual membership attributes.

- Static group performance tends to decrease with size.

  Reading and updating the group entry becomes more expensive as group size grows.

> **Important**
>
> When working with static groups, also read Verify membership and Referential integrity.
> If you have a deployment with large static groups that are updated regularly, use an entry cache. For details, see Cache for large groups.

Static group entries are based on one of the following standard object classes:

`groupOfNames`

The `groupOfNames` object class requires at least one `member` attribute.

Each value is the distinguished name of an entry.

`groupOfEntries`

The `groupOfEntries` object class requires zero or more `member` attributes.

`groupOfUniqueNames`

The `groupOfUniqueNames` object class has at least one `uniqueMember` attribute.

Each value follows Name and Optional UID syntax. Name and Optional UID syntax values are a DN, optionally followed by `\#BitString`. The BitString, such as `'0101111101'B`, serves to distinguish the entry from another entry with the same DN, which can occur when the original entry was deleted and a new entry was created with the same DN.

Like other LDAP attributes, each group member attribute value is unique. LDAP does not allow duplicate values for the same attribute on the same entry.

> **Tip**
>
> When creating a group entry, use `groupOfNames` or `groupOfEntries` where possible.

To create a static group, add a group entry such as the following to the directory:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=My Static Group,ou=Groups,dc=example,dc=com
cn: My Static Group
objectClass: groupOfNames
objectClass: top
ou: Groups
member: uid=ahunter,ou=People,dc=example,dc=com
member: uid=bjensen,ou=People,dc=example,dc=com
member: uid=tmorris,ou=People,dc=example,dc=com
EOF
```

To change group membership, modify the values of the membership attribute:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=My Static Group,ou=Groups,dc=example,dc=com
changetype: modify
add: member
member: uid=scarter,ou=People,dc=example,dc=com
EOF

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(cn=My Static Group)"

dn: cn=My Static Group,ou=Groups,dc=example,dc=com
objectClass: groupOfNames
objectClass: top
cn: My Static Group
member: uid=ahunter,ou=People,dc=example,dc=com
member: uid=bjensen,ou=People,dc=example,dc=com
member: uid=tmorris,ou=People,dc=example,dc=com
member: uid=scarter,ou=People,dc=example,dc=com
ou: Groups
```

RFC 4519 says a `groupOfNames` entry must have at least one member. Although DS servers allow you to create a `groupOfNames` without members, strictly speaking, that behavior is not standard. Alternatively, you can use the `groupOfEntries` object class as shown in the following example:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=Initially Empty Static Group,ou=Groups,dc=example,dc=com
cn: Initially Empty Static Group
objectClass: groupOfEntries
objectClass: top
ou: Groups
EOF

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=Initially Empty Static Group,ou=Groups,dc=example,dc=com
changetype: modify
add: member
member: uid=ahunter,ou=People,dc=example,dc=com
member: uid=bjensen,ou=People,dc=example,dc=com
member: uid=tmorris,ou=People,dc=example,dc=com
member: uid=scarter,ou=People,dc=example,dc=com
EOF
```

Nested groups

DS servers let you nest groups. The following example shows a group of groups of managers and administrators:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=The Big Shots,ou=Groups,dc=example,dc=com
cn: The Big Shots
objectClass: groupOfNames
objectClass: top
ou: Groups
member: cn=Accounting Managers,ou=groups,dc=example,dc=com
member: cn=Directory Administrators,ou=Groups,dc=example,dc=com
member: cn=HR Managers,ou=groups,dc=example,dc=com
member: cn=PD Managers,ou=groups,dc=example,dc=com
member: cn=QA Managers,ou=groups,dc=example,dc=com
EOF
```

Although not shown in the example above, DS servers let you nest groups within nested groups.

DS servers let you create dynamic groups of groups. The following example shows a group of other groups. The members of this group are themselves groups, not users:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=Group of Groups,ou=Groups,dc=example,dc=com
cn: Group of Groups
objectClass: top
objectClass: groupOfURLs
ou: Groups
memberURL: ldap:///ou=Groups,dc=example,dc=com??sub?ou=Groups
EOF
```

Use the `isMemberOf` attribute to determine what groups a member belongs to, as described in Verify membership. The following example requests the groups that Kirsten Vaughan belongs to:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 isMemberOf

dn: uid=kvaughan,ou=People,dc=example,dc=com
isMemberOf: cn=Directory Administrators,ou=Groups,dc=example,dc=com
isMemberOf: cn=HR Managers,ou=groups,dc=example,dc=com
isMemberOf: cn=The Big Shots,ou=Groups,dc=example,dc=com
```

Notice that Kirsten is a member of The Big Shots group.

Notice also that Kirsten does not belong to the Group of Groups. The members of that group are groups, not users. The following example requests the groups that the directory administrators group belongs to:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(cn=Directory Administrators)" \
 isMemberOf

dn: cn=Directory Administrators,ou=Groups,dc=example,dc=com
isMemberOf: cn=Group of Groups,ou=Groups,dc=example,dc=com
isMemberOf: cn=The Big Shots,ou=Groups,dc=example,dc=com
```

The following example shows which groups each group belong to. The search is unindexed, and so is performed here with directory administrator credentials:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(ou=Groups)" \
 isMemberOf

dn: ou=Groups,dc=example,dc=com

dn: cn=Accounting Managers,ou=groups,dc=example,dc=com
isMemberOf: cn=The Big Shots,ou=Groups,dc=example,dc=com
isMemberOf: cn=Group of Groups,ou=Groups,dc=example,dc=com

dn: cn=Directory Administrators,ou=Groups,dc=example,dc=com
isMemberOf: cn=The Big Shots,ou=Groups,dc=example,dc=com
isMemberOf: cn=Group of Groups,ou=Groups,dc=example,dc=com

dn: cn=Group of Groups,ou=Groups,dc=example,dc=com

dn: cn=HR Managers,ou=groups,dc=example,dc=com
isMemberOf: cn=The Big Shots,ou=Groups,dc=example,dc=com
isMemberOf: cn=Group of Groups,ou=Groups,dc=example,dc=com

dn: cn=Initially Empty Static Group,ou=Groups,dc=example,dc=com
isMemberOf: cn=Group of Groups,ou=Groups,dc=example,dc=com

dn: cn=My Dynamic Group,ou=Groups,dc=example,dc=com

dn: cn=My Static Group,ou=Groups,dc=example,dc=com
isMemberOf: cn=Group of Groups,ou=Groups,dc=example,dc=com

dn: cn=PD Managers,ou=groups,dc=example,dc=com
isMemberOf: cn=The Big Shots,ou=Groups,dc=example,dc=com
isMemberOf: cn=Group of Groups,ou=Groups,dc=example,dc=com

dn: cn=QA Managers,ou=groups,dc=example,dc=com
isMemberOf: cn=The Big Shots,ou=Groups,dc=example,dc=com
isMemberOf: cn=Group of Groups,ou=Groups,dc=example,dc=com

dn: cn=The Big Shots,ou=Groups,dc=example,dc=com
isMemberOf: cn=Group of Groups,ou=Groups,dc=example,dc=com
```

Notice that the group of groups is not a member of itself.

Verify membership

For a dynamic group, you can check membership directly on the candidate member's entry using the same search criteria as the dynamic group's member URL.

DS servers also let you verify which groups a user belongs to by requesting the virtual `isMemberOf` attribute. This is more efficient than reading a large static group to determine whether a candidate belongs to the group:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 isMemberOf

dn: uid=bjensen,ou=People,dc=example,dc=com
isMemberOf: cn=Initially Empty Static Group,ou=Groups,dc=example,dc=com
isMemberOf: cn=My Static Group,ou=Groups,dc=example,dc=com
isMemberOf: cn=My Dynamic Group,ou=Groups,dc=example,dc=com
isMemberOf: cn=Virtual Static,ou=Groups,dc=example,dc=com
isMemberOf: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
```

You must request `isMemberOf` explicitly.

Referential integrity

When you delete or rename an entry that belongs to static groups, that entry's DN must be removed or changed in each group it belongs to. You can configure the server to resolve membership changes by enabling referential integrity.

Referential integrity functionality is implemented as a plugin. The referential integrity plugin is disabled by default. To enable the plugin, use the `dsconfig` command:

```
$ dsconfig \
 set-plugin-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --plugin-name "Referential Integrity" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

With the plugin enabled, referential integrity resolves group membership automatically:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(cn=My Static Group)"

dn: cn=My Static Group,ou=Groups,dc=example,dc=com
objectClass: groupOfNames
objectClass: top
cn: My Static Group
member: uid=ahunter,ou=People,dc=example,dc=com
member: uid=bjensen,ou=People,dc=example,dc=com
member: uid=tmorris,ou=People,dc=example,dc=com
member: uid=scarter,ou=People,dc=example,dc=com
ou: Groups

$ ldapdelete \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 uid=scarter,ou=People,dc=example,dc=com

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(cn=My Static Group)"

dn: cn=My Static Group,ou=Groups,dc=example,dc=com
objectClass: groupOfNames
objectClass: top
cn: My Static Group
member: uid=ahunter,ou=People,dc=example,dc=com
member: uid=bjensen,ou=People,dc=example,dc=com
member: uid=tmorris,ou=People,dc=example,dc=com
ou: Groups
```

By default, the referential integrity plugin is configured to manage `member` and `uniqueMember` attributes. These attributes take values that are DNs, and are indexed for equality by default for the default backend. Before you add an additional attribute to manage, make sure that it has DN syntax and that it is indexed for equality. DS servers require indexes because an unindexed search can consume too many of the server's resources. For instructions on indexing attributes, see Configure indexes.

Consider these settings when configuring the referential integrity plugin:

- `check-references:true` checks that members' entries exist when added to a group.
- `check-references-filter-criteria` lets your members' entries match an LDAP filter.

  For example, `check-references-filter-criteria:member:(objectclass=person)` checks that members are person entries.
- `check-references-scope-criteria:naming-context` checks that members' entries are in the same naming context (base DN).

Virtual attributes

Virtual attributes augment directory entries with attribute values that the DS server computes or obtains dynamically. Virtual attribute values do not exist in persistent storage. They help to limit the amount of data that needs to be stored. They fit well in some use cases, such as determining the groups a users belongs to, or adding an ETag to an entry.

Important considerations

- Do not index virtual attributes.

  Virtual attribute values are generated by the server when they are read. They are not designed to be stored in a persistent index. Since you do not index virtual attributes, searching on a virtual attribute can result in an unindexed search. Unindexed searches are resource-intensive for large directories. By default, a directory server lets only the directory superuser perform unindexed searches.
- Avoid searches that use a simple filter with a virtual attribute.

  If you must use a virtual attribute in a search filter, use it in a complex search filter that first narrows the search by filtering on an indexed attribute. For example, the following filter first narrows the search based on the user's ID before checking group membership. Make sure that the user performing the search has access to read `isMemberOf` in the results:

  ```
  (&(uid=user-id)(isMemberOf=group-dn))
  ```

If you must use the `entryDN` and `isMemberOf` virtual attributes in a filter, use a simple equality filter. The following example shows how to add access to read `isMemberOf`, and then run a search that returns the common names for members of a group:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="isMemberOf")(version 3.0; acl "See isMemberOf"; allow (read,search,compare)
  groupdn= "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)
EOF

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)" \
 cn

dn: uid=hmiller,ou=People,dc=example,dc=com
cn: Harry Miller

dn: uid=kvaughan,ou=People,dc=example,dc=com
cn: Kirsten Vaughan

dn: uid=rdaugherty,ou=People,dc=example,dc=com
cn: Robert Daugherty
```

Virtual attributes are generally operational, and so, are returned only when explicitly requested. The searches in these examples are unindexed, are therefore performed with directory administrator credentials:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(dc=example)"

dn: dc=example,dc=com
dc: example
objectClass: domain
objectClass: top

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(dc=example)" \
 numSubordinates

dn: dc=example,dc=com
numSubordinates: 12
```

Default virtual attributes

DS servers define the following virtual attributes by default:

`entryDN`

The DN of the entry.

`entryUUID`

String. Provides a universally unique identifier for the entry.

`etag`

String entity tag. Defined in RFC 2616. Useful when checking whether an entry has changed since it was retrieved.

`hasSubordinates`

Boolean. Indicates whether the entry has children.

`numSubordinates`

The number of child entries directly beneath the entry.

`isMemberOf`

DN. Groups the entry belongs to.

By default, the server generates `isMemberOf` on entries having one of the following object classes:

- `groupOfEntries`
- `groupOfNames`
- `groupOfUniqueNames`
- `person`

To generate `isMemberOf` on entries with other object classes, edit the filter property of the `isMemberOf` virtual attribute configuration.

`member`

DN. Generated for virtual static groups.

`uniqueMember`

DN. Generated for virtual static groups.

`pwdPolicySubentry`

DN of the password policy that applies to the entry.

Default global access control settings prevent normal users from accessing this operational attribute.

`subschemaSubentry`

DN. References the schema definitions.

`collectiveAttributeSubentries`

DNs of applicable collective attribute definitions.

`governingStructureRule`

DN. References the rule specifying the type of subordinates the entry can have.

`structuralObjectClass`

DN. References the structural object class for the entry.

Static virtual attributes

You can use the existing virtual attribute types to create your own virtual attributes. You can also use the `user-defined` type to create your own virtual attribute types. The virtual attribute is defined by the server configuration, which is not replicated:

```
$ dsconfig \
 create-virtual-attribute \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --name "Served By Description" \
 --type user-defined \
 --set enabled:true \
 --set attribute-type:description \
 --set base-dn:dc=example,dc=com \
 --set value:"Served by my favorite directory server" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=wlutz)" \
 description

dn: uid=wlutz,ou=People,dc=example,dc=com
description: Description on ou=People
description: Served by my favorite directory server
```

Template-based virtual attributes

DS lets you create virtual attributes based on the values of non-virtual attributes. The following example overrides the `mail` attribute on user entries with a `user-template` virtual attribute:

```
$ dsconfig \
 create-virtual-attribute \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --name "Virtual email address" \
 --type user-template \
 --set enabled:true \
 --set attribute-type:mail \
 --set template:"{givenName}.{sn}@{domain|virtual.example.com}" \
 --set conflict-behavior:virtual-overrides-real \
 --set filter:"(objectClass=inetOrgPerson)" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 mail

dn: uid=bjensen,ou=People,dc=example,dc=com
mail: Barbara.Jensen@virtual.example.com
```

A template string, such as `{givenName}.{sn}@{domain|virtual.example.com}`, follows these rules:

- It has zero or more `{placeholder}` values, where *placeholder* is an attribute name.

  DS replaces the `{placeholder}` with the value of the attribute.

  When a `{placeholder}` references a multi-valued attribute, DS generates one virtual attribute value for each of the real values.

  When a `{placeholder}` references an attribute that does not exist on the entry, DS replaces the `{placeholder}` with an empty string.

- Template placeholders can have default values that DS uses when the attribute is missing.

  In this example, `{domain|virtual.example.com}` defaults to `virtual.example.com`.

- To escape placeholder values, use a backslash: `\{placeholder}`.

  When adding a sequence of backslashes to a template string with the `dsconfig` command, use single quotes to prevent your shell from interpreting backslashes as escape characters. For example, to set the template to `\\{placeholder}`, use `--set template:'\\{placeholder}'`.

You can read a virtual LDAP attribute over REST as you would any other attribute:

```
$ curl \
--user bjensen:hifalutin \
--cacert ca-cert.pem \
--silent \
--get \
--data-urlencode "_fields=contactInformation/emailAddress" \
--data-urlencode "_prettyPrint=true" \
"https://localhost:8443/api/users/bjensen"

{
  "_id" : "bjensen",
  "_rev" : "<revision>",
  "contactInformation" : {
    "emailAddress" : "Barbara.Jensen@virtual.example.com"
  }
}
```

Collective attributes

Collective attributes provide a standard mechanism for inheriting attribute values. DS servers support standard collective attributes, described in RFC 3671. The inherited values appear on all the entries in a subtree, optionally filtered by object class. Standard collective attribute type names have the prefix `c-`.

DS servers extend collective attributes to make them easier to use. You can define any DS attribute as collective with the `;collective` attribute option. Use LDAP filters in the subtree specification for fine-grained control over which entries inherit the values.

You establish an inheritance relationship between entries by specifying one of the following:

- Which attribute holds the DN of the entry to inherit attributes from.
- How to construct the RDN of the entry to inherit attributes from.

Class of service

This example defines attributes that depend on the user's service level.

This example shows collective attributes that depend on a `classOfService` attribute value:

- For entries with `classOfService: bronze`, `mailQuota` is 1 GB, and `diskQuota` is 10 GB.
- For entries with `classOfService: silver`, `mailQuota` is 5 GB, and `diskQuota` is 50 GB.
- For entries with `classOfService: gold`, `mailQuota` is 10 GB, and `diskQuota` is 100 GB.

You define collective attributes in the user data using an LDAP subentry. As a result, collective attribute settings are replicated. Collective attributes use attributes defined in the directory schema. The following LDIF shows the schema definitions:

```
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( example-class-of-service-attribute-type
NAME 'classOfService'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE
USAGE userApplications
X-ORIGIN 'DS Documentation Examples' )
-
add: attributeTypes
attributeTypes: ( example-class-of-service-disk-quota
NAME 'diskQuota'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
USAGE userApplications
X-ORIGIN 'DS Documentation Examples' )
-
add: attributeTypes
attributeTypes: ( example-class-of-service-mail-quota
NAME 'mailQuota'
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
USAGE userApplications
X-ORIGIN 'DS Documentation Examples' )
-
add: objectClasses
objectClasses: ( example-class-of-service-object-class
NAME 'cos'
SUP top
AUXILIARY
MAY ( classOfService $ diskQuota $ mailQuota )
X-ORIGIN 'DS Documentation Examples' )
```

The collective attribute definitions set the quotas depending on class of service:

```
dn: cn=Bronze Class of Service,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: Bronze Class of Service
diskQuota;collective: 10 GB
mailQuota;collective: 1 GB
subtreeSpecification: { base "ou=People", specificationFilter "(classOfService=bronze)" }

dn: cn=Silver Class of Service,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: Silver Class of Service
diskQuota;collective: 50 GB
mailQuota;collective: 5 GB
subtreeSpecification: { base "ou=People", specificationFilter "(classOfService=silver)" }

dn: cn=Gold Class of Service,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: Gold Class of Service
diskQuota;collective: 100 GB
mailQuota;collective: 10 GB
subtreeSpecification: { base "ou=People", specificationFilter "(classOfService=gold)" }
```

When the collective attributes are defined, you see the results on user entries. Before trying this example, set up a directory server with the `ds-evaluation` profile:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 classOfService mailQuota diskQuota

dn: uid=bjensen,ou=People,dc=example,dc=com
classOfService: bronze
mailQuota: 1 GB
diskQuota: 10 GB

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 classOfService mailQuota diskQuota

dn: uid=kvaughan,ou=People,dc=example,dc=com
classOfService: silver
mailQuota: 5 GB
diskQuota: 50 GB

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=scarter)" \
 classOfService mailQuota diskQuota

dn: uid=scarter,ou=People,dc=example,dc=com
classOfService: gold
mailQuota: 10 GB
diskQuota: 100 GB
```

Inherit from the manager

This example demonstrates how to set an employee's department number using the manager's department number.

The employee-manager relationship is defined by the employee's `manager` attribute. Each `manager` attribute specifies the DN of a manager's entry. The server looks up the manager's department number to set the attribute on the employee's entry.

The collective attribute subentry specifies that users inherit department number from their manager:

```
dn: cn=Inherit Department Number From Manager,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: inheritedCollectiveAttributeSubentry
objectClass: inheritedFromDNCollectiveAttributeSubentry
cn: Inherit Department Number From Manager
subtreeSpecification: { base "ou=People", specificationFilter "(objectClass=posixAccount)" }
inheritFromDNAttribute: manager
inheritAttribute: departmentNumber
```

Babs Jensen's manager is Torrey Rigden:

```
dn: uid=bjensen,ou=People,dc=example,dc=com
manager: uid=trigden, ou=People, dc=example,dc=com
```

Torrey's department number is 3001:

```
dn: uid=trigden,ou=People,dc=example,dc=com
departmentNumber: 3001
```

Babs inherits her department number from Torrey. Before trying this example, set up a directory server with the `ds-evaluation` profile:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 departmentNumber

dn: uid=bjensen,ou=People,dc=example,dc=com
departmentNumber: 3001
```

Inherit from the locality

This example demonstrates how to set a user's default language preferences and street address based on locality.

For this example, the relationship between entries is based on locality. The collective attribute subentry specifies how to construct the RDN of the entry with the values to inherit:

```
dn: cn=Inherit From Locality,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: inheritedCollectiveAttributeSubentry
objectClass: inheritedFromRDNCollectiveAttributeSubentry
cn: Inherit From Locality
subtreeSpecification: { base "ou=People", specificationFilter "(objectClass=posixAccount)" }
inheritFromBaseRDN: ou=Locations
inheritFromRDNAttribute: l
inheritFromRDNType: l
inheritAttribute: preferredLanguage
inheritAttribute: street
collectiveConflictBehavior: real-overrides-virtual
```

The RDN of the entry from which to inherit attributes is `l=localityName,ou=Locations`, where localityName is the value of the `l` (`localityName`) attribute on the user's entry.

In other words, if the user's entry has `l: Bristol`, then the RDN of the entry from which to inherit attributes starts with `l=Bristol,ou=Locations`. The actual entry looks like this:

```
dn: l=Bristol,ou=Locations,dc=example,dc=com
objectClass: top
objectClass: locality
objectClass: extensibleObject
l: Bristol
street: Broad Quay House, Prince Street
preferredLanguage: en-gb
```

The subentry specifies that the inherited attributes are preferred language and street address.

The object class `extensibleObject` allows the entry to take a preferred language. The object class `extensibleObject` means, "Let me add whatever attributes I want." The best practice is to add a custom auxiliary object class instead. The example uses shortcut `extensibleObject` for simplicity.

Notice the last line of the collective attribute subentry:

`collectiveConflictBehavior: real-overrides-virtual`

This line indicates that when a collective attribute clashes with a real attribute, the real value takes precedence over the virtual, collective value. You can set `collectiveConflictBehavior` to `virtual-overrides-real` for the opposite precedence, or to `merge-real-and-virtual` to keep both sets of values.

In this example, users can set their own language preferences. When users set language preferences manually, the user's settings override the locality-based setting.

Sam Carter is located in Bristol. Sam has specified no preferred languages:

```
dn: uid=scarter,ou=People,dc=example,dc=com
l: Bristol
```

Sam inherits the street address and preferred language from the Bristol locality. Before trying this example, set up a directory server with the `ds-evaluation` profile:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=scarter)" \
 preferredLanguage street

dn: uid=scarter,ou=People,dc=example,dc=com
preferredLanguage: en-gb
street: Broad Quay House, Prince Street
```

Babs's locality is San Francisco. Babs prefers English, but also knows Korean:

```
dn: uid=bjensen,ou=People,dc=example,dc=com
preferredLanguage: en, ko;q=0.8
l: San Francisco
```

Babs inherits the street address from the San Francisco locality, but keeps her language preferences:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 preferredLanguage street

dn: uid=bjensen,ou=People,dc=example,dc=com
preferredLanguage: en, ko;q=0.8
street: 201 Mission Street Suite 2900
```

Inherit from a parent entry

This example demonstrates how to inherit a description from the parent entry.

The following collective attribute subentry specifies that entries inherit a description from their parent unless they already have a description:

```
dn: cn=Inherit Description From Parent,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: inheritedCollectiveAttributeSubentry
objectClass: inheritedFromDNCollectiveAttributeSubentry
cn: Inherit Description From Parent
subtreeSpecification: { base "", minimum 2, specificationFilter "(objectClass=posixAccount)" }
inheritFromDNAttribute: entryDN
inheritFromDNParent: 1
inheritAttribute: description
collectiveConflictBehavior: real-overrides-virtual
```

In this example, `inheritFromDNAttribute` uses the virtual attribute `entryDN`. The setting `inheritFromDNParent: 1` indicates that the server should locate the parent entry by removing one leading RDN from the `entryDN`. (To inherit from the grandparent entry, you would specify `inheritFromDNParent: 2`, for example.)

Sam Carter's entry has no description attribute, and so inherits the parent's description:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=scarter)" \
 description

dn: uid=scarter,ou=People,dc=example,dc=com
description: Description on ou=People
description: Served by my favorite directory server
```

Babs Jensen's entry already has a description attribute, and so does not inherit from the parent:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 description

dn: uid=bjensen,ou=People,dc=example,dc=com
description: Original description
```

About subentry scope

LDAP subentries reside with the user data and so the server replicates them. Subentries hold operational data. They are not visible in search results unless explicitly requested. This section describes how a subentry's `subtreeSpecification` attribute defines the scope of the subtree that the subentry applies to.

An LDAP subentry's subtree specification identifies a subset of entries in a branch of the DIT. The subentry scope is these entries. In other words, these are the entries that the subentry affects.

The attribute value for a `subtreeSpecification` optionally includes the following parameters:

`base`

Indicates the entry, *relative to the subentry's parent*, at the base of the subtree.

By default, the base is the subentry's parent.

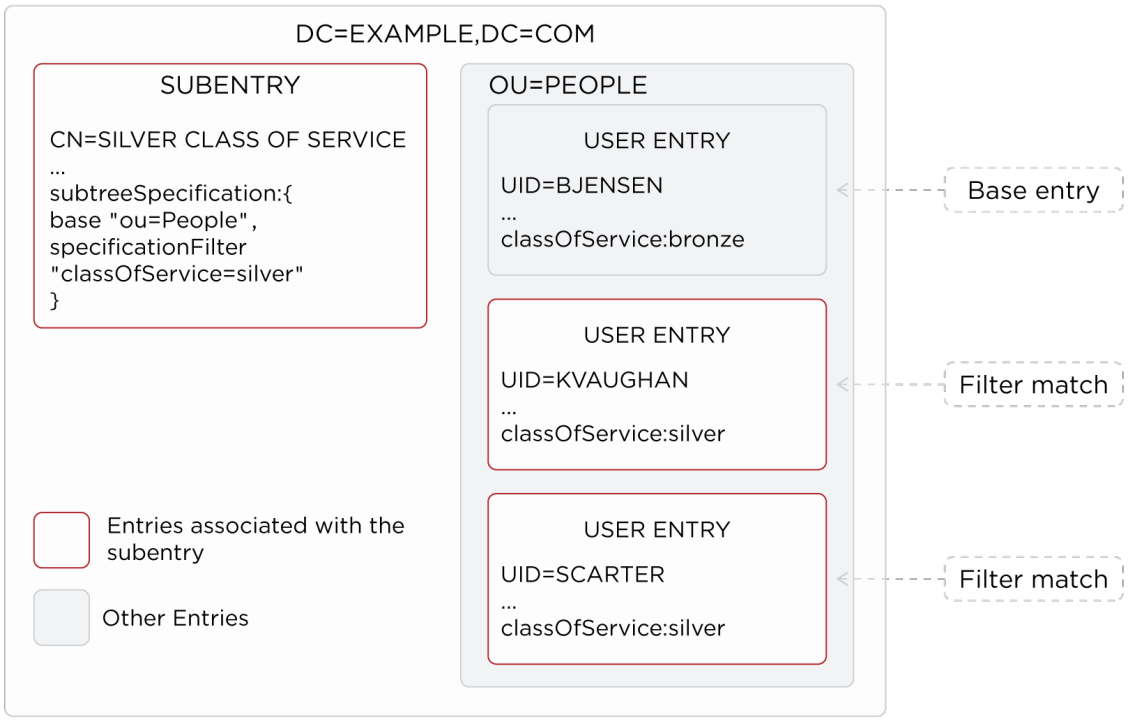
`specificationFilter`

Indicates an LDAP filter. Entries matching the filter are in scope.

DS servers extend the standard implementation to allow any search filter, not just an assertion about the `objectClass` attribute.

By default, all entries under the base entry are in scope.

The following illustration shows this for an example collective attribute subentry:



Notice that the base of `ou=People` on the subentry `cn=Silver Class of Service,dc=example,dc=com` indicates that the base entry is `ou=People,dc=example,dc=com`.

The filter `"(classOfService=silver)"` means that Kirsten Vaughan and Sam Carter's entries are in scope. Babs Jensen's entry, with `classOfService: bronze` does not match and is therefore not in scope. The `ou=People` organizational unit entry does not have a `classOfService` attribute, and so is not in scope, either.

Replication

Always-on directory services use replication to let applications continue reading and writing data, even when individual servers crash, and network connections are interrupted. DS replication is designed to make building highly available directory services straightforward.

About replication

Replication is the process of copying updates between DS servers so all directory servers eventually converge on identical copies of directory data. DS servers that replicate their data are *replicas*. Since replication is *eventually convergent*, different replicas can be momentarily out of sync. If you lose an individual replica, or even an entire data center, the remaining replicas continue to provide service. Applications can still write changes to the directory data. Replication brings the replicas back in sync when the problem is repaired.

Replication uses a DS-specific protocol that works only between DS replicas. It replays update operations quickly, storing historical change data to resolve most conflicts automatically. For example, if two client applications separately update a user entry to change the phone number, replication can identify the latest change, and apply it on all replicas without human intervention. To prevent the historical change data from growing forever, DS replicas purge historical data that is older than a configurable interval (default: three days).

DS software supports replication over LAN and WAN networks. If you have multiple sites with many replicas communicating over the WAN, consider standalone replication servers. For details, see Install standalone servers.

Replication is resilient to host clock anomalies. You should, however, aim to keep server clocks synchronized using `ntpd`, for example. Keeping replica clocks synchronized helps prevent issues when validating certificates for secure connections, and makes it easier to compare timestamps from multiple replicas. Replication is designed to overcome the following issues:

- Clock skew between different replicas.

  Replication adjusts for skew automatically, and using `ntpd` further mitigates this.

  Very large skew, such as replicating with a system whose clock was started at 0 (January 1, 1970), has been seen to cause problems.

- Forward and backward clock adjustments on a single replica.

  Replication adjusts for these automatically.

  Very large changes, such as abruptly setting the clock back an entire day, have been seen to cause problems.

- Timezone differences and adjustments.

  Replication uses UTC time.

Very large host clock anomalies can result in the following symptoms:

- SSL certificate validation errors, when the clocks are far enough apart that the validity dates cannot be correctly compared.

- Problems with time-based settings in access control instruction subjects, and with features that depend on timestamps, such as password age and last login attributes.
- Misleading changelog timestamps and replication-related monitoring data.
- Incorrect replication conflict resolution.
- Incorrect replication purge delay calculation.

Ready to replicate

When you set up a server, you can specify the following:

- The replication port.

  If specified, the setup process configures the server as a replication server.

- The bootstrap replication servers' host:port combinations.

  When the server starts, it contacts the bootstrap replication servers to discover other replicas and replication servers.

Setup profiles that create backends for schema and directory data configure replication domains for their base DNs. The server is ready to replicate that directory data when it starts up.

Replication initialization depends on the state of the data in the replicas.

DS replication shares changes, not data. When a replica applies an update, it sends a message to its replication server. The replication server forwards the update to all other replication servers, and to its replicas. The other replication servers do the same, so the update is eventually propagated to all replicas.

Each replica eventually converges on the same data by applying each update and resolving conflicts in the same way. As long as each replica starts from the same initial state, each replica eventually converges on the same state. It is crucial, therefore, for each replica to begin in the same initial state. Replicas cannot converge by following exactly the same steps from different initial states.

Internally, DS replicas store a shorthand form of the initial state called a generation ID. The generation ID is a hash of the first 1000 entries in a backend. If the replicas' generation IDs match, the servers can replicate data without user intervention. If the replicas' generation IDs do not match for a given backend, you must manually initialize replication between them to force the same initial state on all replicas.

If necessary, and before starting the servers, further restrict TLS protocols and cipher suites on all servers. This forces the server to use only the restricted set of protocols and cipher suites. For details, see TLS Settings.

Replication per base DN

The primary unit of replication is the *replication domain*. A replication domain has a base DN, such as `dc=example,dc=com`.

The set of DS replicas and replication servers sharing one or more replication domains is a *replication topology*. Replication among these servers applies to all the data under the domain's base DN.

The following example topology replicates `dc=example,dc=com`, `dc=example,dc=org`, and `dc=example,dc=net`. All the replication servers in a topology are fully connected, replicating the same data under each base DN:

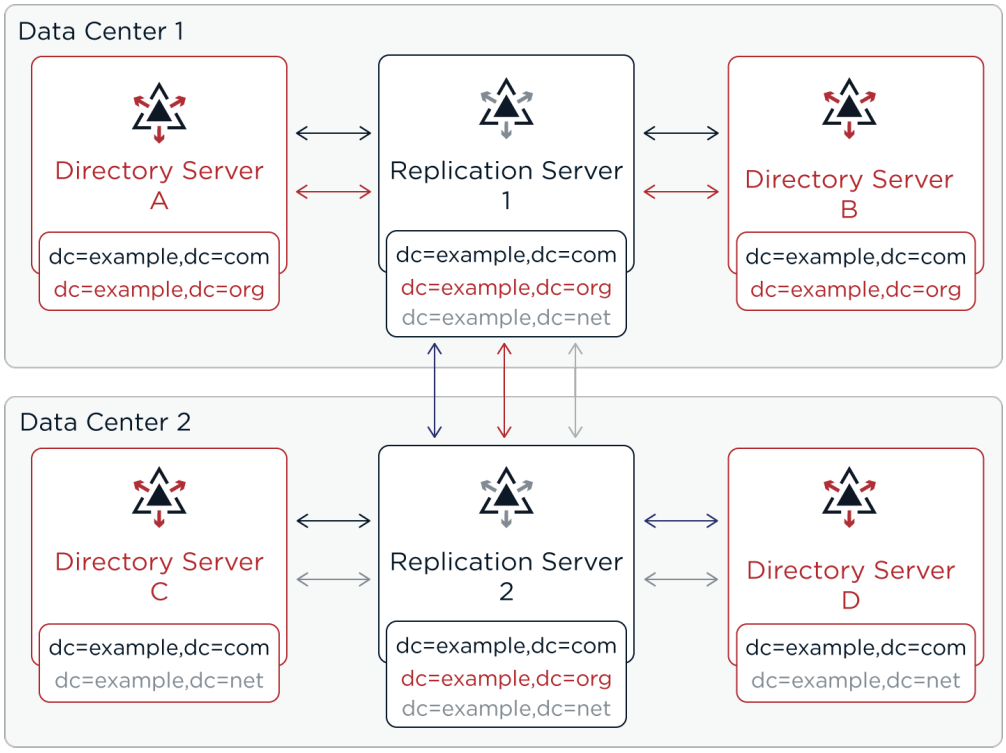


Figure 1. Correct replication configuration

Replication doesn't support separate, independent domains for the same base DN in the same topology. For example, you can't replicate two `dc=example,dc=org` domains with different data in the same topology:

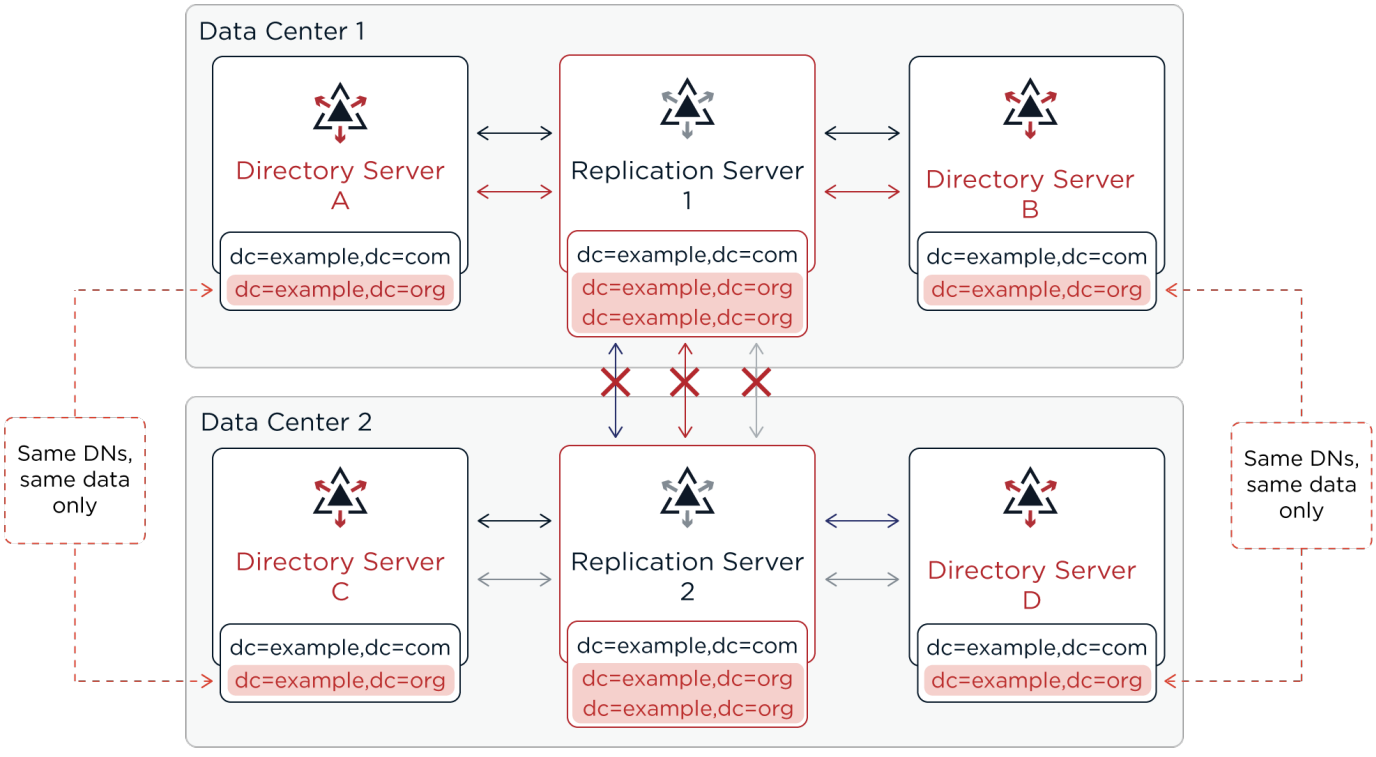


Figure 2. Incorrect replication configuration

When you set up a replication domain, replicate the data under the base DN among all the servers in the topology. If the data under a base DN is different on different servers, configure the servers appropriately:

Difference	Use this
Different data under the same base DN.	Separate replication topologies. In other words, put the replicas and replication servers in independent deployments unless the data matches.
Some replicas have only part of the data under a base DN.	Subtree replication
Some replicas have only a subset of LDAP attributes.	Fractional replication

Replication depends on the directory schema under `cn=schema`. If applications require specific, non-standard schema definitions or can update the LDAP schema online, replicate `cn=schema` with the other base DNs.

Port use and operations

DS servers listen on dedicated ports for administrative requests and for replication requests. These dedicated ports must remain open to remote requests from configuration tools and from other servers. *Make sure that firewall software allows connections to the administration and replication ports from all connecting servers.*

DS server configuration tools securely connect to administration ports. Administrative connections are short-lived.

DS replicas connect to DS replication ports for replication requests. A server listening on a replication port is called a *replication server*, whether it is running inside the same process as a directory server, or on a separate host system. Replication connections are long-lived. Each DS replica connects to a replication server upon initialization and then at startup time. The replica keeps the connection open to push and receive updates, although it can connect to another replication server.

The command to initialize replication uses the administrative port, and initialization uses the replication port:

Manual Replication Initialization

Figure 3. Manual Replication Initialization

DS replicas push updates to and receive updates from replication servers over replication connections. When processing an update, a replica (DS) pushes it to the replication server (RS) it is connected to. The replication server pushes the update to connected replicas and to other replication servers. Replicas always connect through replication servers.

A replica with a replication port and a changelog plays both roles (DS/RS), normally connecting to its own replication server. A standalone replica (DS) connects to a remote replication server (RS). The replication servers connect to each other. The following figure shows the flow of messages between standalone replicas and replication servers.

Data Replication

Figure 4. Data Replication

The command to monitor replication status uses the administration ports on multiple servers to connect and read monitoring information, as shown in the following sequence diagram:

status

Replication connection selection

DS servers can provide both directory services and replication services. The two services are not the same, even if they can run alongside each other in the same DS server in the same JVM.

Replication relies on the replication service provided by DS replication servers. DS directory servers (replicas) publish changes made to their data, and subscribe to changes published by other replicas. The replication service manages replication data only, sending and receiving replication messages. A replication server receives, sends, and stores only changes to directory data, not the data itself.

The directory service manages directory data. It responds to requests, and stores directory data and historical information. For each replicated base DN, such as `dc=example,dc=com` or `cn=schema`, the directory service publishes changes to and subscribes to changes from a replication service. The directory service resolves any conflicts that arise when reconciling changes from other replicas, using the historical information about changes to resolve the conflicts. (Conflict resolution is the responsibility of the directory server rather than the replication server.)

After a directory server connects to a replication topology, it connects to one replication server at a time for a given domain. The replication server provides the directory server with the list of all replication servers for that base DN. Given this list, the directory server selects its preferred replication server when starting up, when it loses the current connection, or when the connection becomes unresponsive.

For each replicated base DN, a directory server prefers to connect to a replication server:

1. In the same JVM as the directory server.
2. In the same group as the directory server.

   By default, if no replication server in the same group is available, the directory server chooses a replication server from any available group.

   To define the order of failover across replication groups, set the global configuration property, group-id-failover-order. When this property is set and no replication server is available in the directory server's group, the directory server chooses a replication server from the next group in the list.
3. With the same initial data under the base DN as the directory server.
4. If initial data was the same, a replication server with all the latest changes from the directory server.
5. With the most available capacity relative to other eligible replication servers.

   Available capacity depends on how many replicas in the topology are already connected to the replication server, and what proportion of all replicas ought to be connected to the replication server.

   To determine what proportion ought to be connected, a directory server uses replication server weight. When configuring a replication server, you can assign it a weight (default: 1). The weight property takes an integer that indicates capacity relative to other replication servers. For example, a weight of 2 indicates a replication server that can handle twice as many connected replicas as one with weight 1.

   The proportion that ought to be connected is `(replication server weight)/(sum of replication server weights)`. If there are four replication servers with weight 1, the proportion for each is 1/4.

Consider a `dc=example,dc=com` topology with five directory servers connected to replication servers A, B, and C, where:

- Two directory servers are connected to replication server A.

- Two directory servers are connected to replication server B.
- One directory server is connected to replication server C.

Replication server C is the server with the most available capacity. All other criteria being equal, replication server C is the server to connect to when another directory server joins the topology.

The directory server regularly updates the list of replication servers in case it must reconnect. As available capacity can change dynamically, a directory server can reconnect to another replication server to balance the replication load in the topology. For this reason, the server can also end up connected to different replication servers for different base DNs.

Manual initialization

Manual initialization is not always required. Replication can proceed automatically when replicas start from the same initial data.

If this is not the case, manually initialize replication. How you initialize replication depends on your situation:

Initialization Options

Use Cases	Recommendations
Replicas installed with same data	Nothing to do (no manual initialization required)
Evaluating DS software Developing a directory solution	Initialize over the network *Limitations:* • Transmits all data over the network, so requires ample bandwidth; can be a problem for WAN links. • Rebuilds indexes and consumes significant system resources; can impact service performance.
New directory service, medium to large data set (> 500,000 entries)	Initialize from LDIF *Limitations:* • Rebuilds indexes and consumes significant system resources; can impact service performance.
Existing directory service, medium to large data set (> 500,000 entries)	Initialize From backup *Limitations:* • All DS servers must be the same version. Backups are not guaranteed to be compatible across major and minor server releases.

Use Cases	Recommendations
New backend	Create a backend, then one of: • Initialize over the network • Initialize from LDIF • Back up the new backend, then Initialize From backup *Limitations:* • The limitations depend on how you initialize the new backend, and are described above.
Broken data set	Disaster recovery *Limitations:* • This method **permanently loses recent changes**. • Applications using the changelog must be reinitialized after you restore the data.

Initialize over the network

Review Initialization Options before following these steps:

1. Manually initialize replication using the replication protocol's total update capability in one of these ways:
    1. Overwrite the data in all replicas with the data from the replica where the command runs:

```
$ dsrepl \
 initialize \
 --baseDN dc=example,dc=com \
 --toAllServers \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --trustStorePath /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

    2. Initialize a single other replica, identified by its server ID, from the replica where the command runs:

```
$ dsrepl \
 initialize \
 --baseDN dc=example,dc=com \
 --toServer ds-1 \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --trustStorePath /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Initialize from LDIF

> **Important**
>
> If you aim to return to a previous state of the data, or to initialize replicas with LDIF from a non-replicated environment, refer to Disaster recovery.

Review Initialization Options before following these steps:

Initialize each replica with the same LDIF:

1. Stop the server.
2. If desired, enable data confidentiality.

   For details, see Data encryption and Encrypt External Changelog Data.
3. Import the LDIF.

   For details, see Import LDIF.
4. Start the server.

Initialize From backup

Review Initialization Options before following these steps:

1. Stop the replica.
2. Restore the backend from backup.

   For details, see Restore.
3. Start the replica.

   Replication replays changes from other replicas that have happened since the backup was created.

Replication status

The following command displays a snapshot of replication monitoring information:

```
$ dsrepl \
 status \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=monitor \
 --bindPassword password \
 --trustStorePath /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Base DN            Status  Receive     Replay
                           delay (ms)  delay (ms)
-------------------------------------------------
dc=example,dc=com  OK               0           0
uid=monitor        OK               0           0
cn=schema          OK               0           0
```

The command connects to each known server to read status information. It will eventually time out if other servers cannot be contacted.

To get a balanced view of replication delays, monitor them over time. You can do this with repeated use of the `dsrepl status` command, or by reading the monitoring information over LDAP or HTTP. For details, see Replication delay (Prometheus) or Replication delay (LDAP).

Manual purge

Over time, DS servers purge any historical data and changelog data older than the replication purge delay. To remove stale historical data, you can optionally trigger a purge task manually:

```
$ dsrepl \
 purge-meta-data \
 --baseDN dc=example,dc=com \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --trustStorePath /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

By default, this task runs for a maximum of one hour. If desired, set the maximum duration in seconds with the `--maximumDuration` option.

With earlier DS versions, you had to purge replicas from other servers' configurations after they were removed. DS servers do this automatically now. No administrative action is required.

Replication groups

Define replication groups so that replicas connect first to local replication servers, only going outside the group when no local replication servers are available.

For each group, set the appropriate group ID on the replication servers and the replicas. The following steps set up two replication groups, each with a replication server and a directory server. In a full-scale deployment, you would have multiple servers of each type in each group. The replicas and replication servers in the same location would belong to the same group:

1. Pick a group ID for each group.

   The default group ID is `default`. For mixed topologies with replicas running older DS versions that support only numeric group IDs, this is equivalent to `1`.

2. Set the group ID for each group on the directory servers:

```
$ dsconfig \
 set-global-configuration-prop \
 --set group-id:US-East \
 --hostname ds.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-global-configuration-prop \
 --set group-id:US-West \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. Set the group ID for each group on the replication servers:

```
$ dsconfig \
 set-global-configuration-prop \
 --set group-id:US-East \
 --hostname rs.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-global-configuration-prop \
 --set group-id:US-West \
 --hostname rs2.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Subtree replication

To configure subtree replication, split the data across multiple servers.

For example, use different servers for `ou=People,dc=example,dc=com`, and other `dc=example,dc=com` data. Set up one set of servers with everything in `dc=example,dc=com` except `ou=People,dc=example,dc=com`. Set up a separate set of servers for `ou=People,dc=example,dc=com`. The former replicate `dc=example,dc=com`, the latter `ou=People,dc=example,dc=com`. For details, see Split data.

Due to limitations described in Replication per base DN, the same servers cannot replicate both `dc=example,dc=com` and `ou=People,dc=example,dc=com` separately.

Fractional replication

With fractional replication, you specify the attributes to include and to exclude using `fractional-include` and `fractional-exclude` configuration properties. Fractional replicas must respect LDAP schemas. Attributes that are required by the relevant object classes are included whether you specify them or not. Excluded attributes must be optional attributes of the relevant object classes.

Each attribute must remain on at least one replica. When you configure a replica to exclude an attribute, the replica checks that the attribute is never added to the replica as part of any LDAP operation. If you exclude the attribute everywhere, it can never be added anywhere.

When using fractional replication, initialize replication from LDIF. The import process imports only the data allowed by fractional replication. Be aware that you cannot create a replica with a full data set from a replica with only a subset of the data.

Replication servers filter objects for fractional replication. If you must prevent data from being replicated across a national boundary, for example, keep standalone replication servers in locations where you can store full entries and their changes. Outside that location, set up standalone replicas that receive the fractional entries.

The following example configures a fractional replica with a subset of `inetOrgPerson` attributes:

```
$ dsconfig \
 set-replication-domain-prop \
 --provider-name "Multimaster Synchronization" \
 --domain-name "dc=example,dc=com" \
 --set fractional-include:inetorgperson:cn,givenname,mail,mobile,sn,telephonenumber \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The following example excludes a custom attribute, `sessionToken`, on the replica:

```
$ dsconfig \
 set-replication-domain-prop \
 --provider-name "Multimaster Synchronization" \
 --domain-name "dc=example,dc=com" \
 --set fractional-exclude:*:sessionToken \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

This example only applies if you have defined a `sessionToken` attribute in the LDAP schema.

Read-only replicas

By default, all directory servers in a replication topology are read-write.

The following command causes the replica to accept only replication updates, and to refuse updates from client applications:

```
$ dsconfig \
 set-global-configuration-prop \
 --set writability-mode:internal-only \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The following command resets the replica to the default behavior:

```
$ dsconfig \
 set-global-configuration-prop \
 --set writability-mode:enabled \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Trusted replicas

By default, all directory servers in a replication topology trust all replicas. If a replica allows an update, then other servers relay and replay the update without further verification. This simplifies deployments where you control all the replicas.

In deployments where you do not control all the replicas, you can configure replication servers to accept updates only from trusted replicas. The trust depends on the certificate that a replica presents to the replication server when connecting. Specifically, replication servers can verify trust when:

- Trusted certificates have the OID `1.3.6.1.4.1.36733.2.1.10.1` in their extended key usage certificate extension.

  Use this policy when you control the CA signing the replicas' certificates, and can enforce that only authorized replicas have certificates with this setting.

- They store fingerprints for all trusted certificates in their configurations.

  Use this policy when you do not control the CA.

You choose the policy by setting the replication server advanced property, allow-updates-policy. It takes the following values:

`all`

(Default) Trust all replicas.

`verify-certificate-key-usage`

Only trust updates from replicas whose certificates' `ExtendedKeyUsage` includes `1.3.6.1.4.1.36733.2.1.10.1`.

`verify-certificate-fingerprint`

Only trust updates from replicas with certificate fingerprints in the advanced property, allow-updates-server-fingerprints.

If a replication server does not trust an update, it logs an error message explaining why. The update is not replicated, and therefore may cause replication to diverge on the untrusted replica. Configure the untrusted replicas you control to be read-only, as described in Read-only replicas.

Trust extended key usage

1. For each trusted DS server, get the certificate for replication signed with the appropriate extended key usage.

   The following example demonstrates a certificate signing request with the appropriate extended key usage:

```
$ keytool \
 -certreq \
 -ext ExtendedKeyUsage:critical=clientAuth,serverAuth,1.3.6.1.4.1.36733.2.1.10.1 \
 -alias ssl-key-pair \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -file ssl-key-pair.csr
```

The full process for each server involves generating a certificate signing request, signing the certificate in the request with the CA signing certificate, and importing the CA-signed certificate on the server.

2. For each replication server, update the configuration to trust certificates with the appropriate extended key usage:

```
$ dsconfig \
 set-replication-server-prop \
 --provider-name "Multimaster Synchronization" \
 --set allow-updates-policy:verify-certificate-key-usage \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

At this point, the replication server can trust the other server's updates.

Trust fingerprints

1. For each trusted DS server, get the certificate fingerprint:

```
$ keytool \
 -list \
 -alias ssl-key-pair \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin \
ssl-key-pair, <date>, PrivateKeyEntry,
Certificate fingerprint (SHA-256): 05:55:BD:A5:E1:4C:35:A6:A5:4E:78:DD:3E:FD:EA:5A:66:5D:E0:DC:
9C:C5:18:7E:E9:CA:A9:1E:CD:87:4B:78
```

2. For each replication server, update the configuration to trust replicas by their certificate fingerprints:

```
$ dsconfig \
 set-replication-server-prop \
 --provider-name "Multimaster Synchronization" \
 --set allow-updates-policy:verify-certificate-fingerprint \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. For each replication server, update the configuration to recognize each certificate fingerprint.

   The following example demonstrates adding a trusted certificate fingerprint:

```
$ dsconfig \
 set-replication-server-prop \
 --provider-name "Multimaster Synchronization" \
 --add allow-updates-server-fingerprints:\
"{SHA-256}05:55:BD:A5:E1:4C:35:A6:A5:4E:78:DD:3E:FD:EA:5A:66:5D:E0:DC:9C:C5:18:7E:E9:CA:A9:1E:CD:87:4B:78" \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

   At this point, the replication server can trust the other server's updates.

4. Repeat the relevant steps each time a trusted certificate changes.

Listen addresses

When configuring a server on a multi-homed system with multiple IP addresses, you can specify the listen addresses. By default, the replication server listens on all network interfaces.

The following example configures the server to listen only on `192.168.0.10` for all connections:

```
$ dsconfig \
 set-global-configuration-prop \
 --set advertised-listen-address:192.168.0.10 \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

For details, see advertised-listen-address.

Disk space thresholds

Replication servers record changes in changelog database files under the `opendj/changelogDb` directory.

> **Warning**
>
> Do not compress, tamper with, or otherwise alter changelog database files directly, unless specifically instructed to do so by a qualified ForgeRock technical support engineer.
> External changes to changelog database files can render them unusable by the server.

Replication server configuration objects have changelog database properties, `disk-low-threshold`, and `disk-full-threshold`:

- When available disk space falls below `disk-low-threshold`, the replication server triggers a warning alert notification to let you know to free disk space.
- When available disk space falls below `disk-full-threshold`, the replication server triggers another warning alert notification, and *disconnects from the replication topology*.

  Connected directory servers fail over to another replication server until available disk space is above the `disk-full-threshold` again.

Set the thresholds high enough to allow time to react after the initial alert.

The following example sets `disk-low-threshold` to 10 GB and `disk-full-threshold` to 5 GB:

```
$ dsconfig \
 set-replication-server-prop \
 --provider-name "Multimaster Synchronization" \
 --set "disk-low-threshold:10 GB" \
 --set "disk-full-threshold:5 GB" \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The disk-low-threshold and disk-full-threshold properties are *advanced* properties. Examine their values with the `dsconfig --advanced` option.

Recover from user error

It is possible to restore accidentally deleted or changed data.

Changes to a replicated DS directory service are similar to those made with the Unix `rm` command, but with a twist. With the `rm` command, if you make a mistake you can restore your files from backup, and lose only the work done since the last backup. If you make a mistake with an update to the directory service, after you restore from backup, replication efficiently replays your mistake over the data you restored.

There is more than one way to recover from user error. None of the ways are limited to changing DS settings. All involve manually fixing mistakes.

Consider these alternatives:

- Encourage client applications to provide end users with undo capability.

  In this case, client applications take responsibility for maintaining an undo history.

- Maintain a record of each update to the service, so that you can manually "undo" mistakes.

  You can use the external changelog. The external changelog is enabled with replication, and does not use additional space.

  For instructions, see Changelog for notifications. In particular, see Include Unchanged Attributes on saving what is deleted as well as changed.

  DS servers can write to a file-based audit log. The audit log does not help with a general solution in this case. The DS audit log records only changes to the data. When you delete an entry, the audit log does not record the entry before deletion. The following example shows audit log records for changes made to Barbara Jensen's entry:

  ```
  # <datestamp>; conn=<number>; op=<number>
  dn: uid=bjensen,ou=People,dc=example,dc=com
  changetype: modify
  replace: description
  description: This is the description I want.
  -
  replace: modifiersName
  modifiersName: uid=admin
  -
  replace: modifyTimestamp
  modifyTimestamp: <timestamp>

  # <datestamp>; conn=<number>; op=<number>
  dn: uid=bjensen,ou=People,dc=example,dc=com
  changetype: modify
  replace: description
  description: I never should have changed this!
  -
  replace: modifiersName
  modifiersName: uid=admin
  -
  replace: modifyTimestamp
  modifyTimestamp: <timestamp>

  # <datestamp>; conn=<number>; op=<number>
  dn: uid=bjensen,ou=People,dc=example,dc=com
  changetype: delete
  ```

  You can use these records to fix the mistaken update to the description. However, the audit log lacks the information needed to restore Barbara Jensen's deleted entry.

- For administrative errors that involve directory data, use the external changelog if possible.

  If not, an alternative technique consists of restoring backup to a new server set up to not replicate. (Replication replays all updates, including mistakes.) Compare data on the separate restored server to the live replicas, and fix the mistakes manually.

A more drastic alternative is to rebuild the entire service from backup, as described in Disaster recovery. This alternative is only recommended in the case of a major error where you have a very fresh backup (taken immediately before the error), and no client applications are affected.

- For administrative configuration errors that prevent servers from starting, know that DS servers keep snapshots of the main configuration file, including the `opendj/var/config.ldif.startok` file, and the files in the `opendj/var/archived-configs/` directory.

  Compare the current configuration with the earlier configurations, stop the server, and repair mistakes manually. Take care to avoid trailing white space at the end of LDIF lines.

Replication conflicts

Replication is eventually consistent by design to support basic write availability. Changes are applied locally and then replayed to remote replicas. This means it is possible to have conflicts. A *replication conflict* arises when incompatible changes are made concurrently to multiple read-write replicas.

Two types of conflicts happen: *modify conflicts* and *naming conflicts*. Modify conflicts involve concurrent modifications to the same entry. Naming conflicts involve other operations that affect the DN of the entry.

Replication resolves modify conflicts, and many naming conflicts by replaying the changes in the correct order. To determine the relative order in which changes occurred, replicas retain historical information for each update. This information is stored in the target entry's `ds-sync-hist` operational attribute.

Replication resolves these conflicts automatically using the historical information to order changes correctly:

- The attributes of a given entry are modified concurrently in different ways on different replicas.
- An entry is renamed on one replica while being modified on another replica.
- An entry is renamed on one replica while being renamed in a different way on another replica.
- An entry is deleted on one replica while being modified on another replica.
- An entry is deleted and another entry with the same DN added on one replica while the same entry is being modified on another replica.

Replication cannot resolve these particular naming conflicts. You must resolve them manually:

- Different entries with the same DN are added concurrently on multiple replicas.
- An entry on one replica is moved (renamed) to use the same DN as a new entry concurrently added on another replica.
- A parent entry is deleted on one replica, while a child entry is added or renamed concurrently on another replica.

When replication cannot resolve naming conflicts automatically, the server renames the conflicting entry using its `entryUUID` operational attribute. The resulting conflicting entry has a DN with the following form:

```
entryuuid=entryUUID-value+original-RDN,original-parent-DN
```

For each conflicting entry named in this way, resolve the conflict manually:

1. Get the conflicting entry or entries, and the original entry if available.

   The following example shows the result on one replica of a naming conflict when a `newuser` entry was added concurrently on two replicas:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(uid=newuser)"

dn: uid=newuser,ou=People,dc=example,dc=com
objectClass: top
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
mail: newuser@example.com
sn: User
cn: New User
ou: People
description: Added on server 1
uid: newuser

dn: entryuuid=2f1b58c3-4bee-4215-88bc-88202a7bcb9d+uid=newuser,ou=People,dc=example,dc=com
objectClass: top
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
mail: newuser@example.com
sn: User
cn: New User
ou: People
description: Added on server 2
uid: newuser
```

2. To preserve changes made on the conflicting entry or entries, apply the changes manually.

   The following example shows a modification to preserve both description values:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --bindPassword password << EOF
dn: uid=newuser,ou=People,dc=example,dc=com
changetype: modify
add: description
description: Added on server 2
EOF
```

   For additional examples demonstrating how to apply changes to directory entries, see LDAP updates.

3. After making any necessary changes, manually delete the conflicting entry or entries.

The following example deletes the conflicting entry:

```
$ ldapdelete \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 entryuuid=2f1b58c3-4bee-4215-88bc-88202a7bcb9d+uid=newuser,ou=People,dc=example,dc=com
```

For additional examples, see Delete Entries.

Bootstrap replication servers

A *bootstrap replication server* is one of the replication servers in a deployment that a server should contact to discover all the other servers in the deployment.

Add a bootstrap replication server

After you add a replication server to a deployment, add it to the other servers' bootstrap-replication-server settings.

Apply these steps for each server whose configuration references the new replication server to add:

1. Add the bootstrap replication server to the server's configuration:

```
$ dsconfig \
 set-synchronization-provider-prop \
 --provider-name "Multimaster Synchronization" \
 --add bootstrap-replication-server:new-rs.example.com:8989 \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. If the server uses property value substitution to load the list of replication bootstrap servers from the environment, restart the server for the changes to take effect.

Remove a bootstrap replication server

After you remove a replication server from a deployment, remove it from other servers' `bootstrap-replication-server` settings.

Apply these steps for each server whose configuration references the replication server that you removed:

1. Remove the bootstrap replication server from the server's configuration:

```
$ dsconfig \
 set-synchronization-provider-prop \
 --provider-name "Multimaster Synchronization" \
 --remove bootstrap-replication-server:removed-rs.example.com:8989 \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. If the server uses property value substitution to load the list of replication bootstrap servers from the environment, restart the server for the changes to take effect.

Disable replication

Disable replication temporarily

> **Warning**
>
> Do not allow modifications on the replica for which replication is temporarily stopped. No record of such changes is kept, and the changes cause replication to diverge.

Follow these steps to disable replication temporarily for a replica and drop any changes that occur:

1. Prevent changes to the affected data.

   For details, see Read-only replicas.

2. Disable the replication mechanism:

```
$ dsconfig \
 set-synchronization-provider-prop \
 --provider-name "Multimaster Synchronization" \
 --set enabled:false \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. Perform whatever operations are required.

4. Enable the replication mechanism again:

```
$ dsconfig \
 set-synchronization-provider-prop \
 --provider-name "Multimaster Synchronization" \
 --set enabled:true \
 --hostname replica.example.com \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

5. Allow changes to the affected data.

   For details, see Read-only replicas.

Before removing a server from a group of replicated servers, disable replication as described. When the server you remove is a bootstrap replication server, also remove it from the configuration on all other servers.

Stop replicating permanently

You might remove a server from a replication topology because:

- The DS server is no longer needed.

  For example, you are scaling a deployment down, or retiring an old server that you replaced with a newer one.

- Someone configured replication between DS servers that should be independent.

  For example, at setup time, replication was configured to include all six replicas in three data centers, but the expected configuration was three separate directory service deployments with two replicas in each data center.

  In this case, you must permanently change which servers replicate with each other.

> **Note**
>
> The steps that follow only apply to deployments of DS 7 and later servers.
> If you are upgrading from older servers and have a mix of DS 7 and earlier servers, see the Upgrade documentation instead.

To remove a server that is no longer needed:

1. Uninstall the server.

   For details, see Uninstallation.

2. If the server is referenced in other servers' `bootstrap-replication-server` settings, remove it.

   For details, see Remove a bootstrap replication server.

3. The automated purge process eventually removes historical data and changelog data for old servers.

   You can optionally trigger a purge task manually, as described in Manual purge.

To change which servers replicate with each other:

1. Prevent changes to the affected data.

   For details, see Read-only replicas.

2. Perform these steps in parallel on all affected servers:

   1. Disable the replication mechanism.

      For details, see Disable replication temporarily.

   2. Adjust the `bootstrap-replication-server` settings to limit replication as desired.

   3. Enable the replication mechanism again.

   4. Restart the server for the changes to take effect.

3. Allow changes to the affected data.

4. Delete entries that were erroneously replicated.

   For details, see Delete Entries.

5. The automated purge process eventually removes historical data and changelog data for old servers.

   You can optionally trigger a purge task manually, as described in Manual purge.

Changelog for notifications

Some applications require notification when directory data updates occur. For example, an application might need to sync directory data with another database, or the application might need to kick off other processing when certain updates occur. DS replication servers provide an external changelog that replications can use to read changes. This mechanism is more scalable and robust than LDAP persistent searches.

By default, changelog database files are found under the `opendj/changelogDb` directory.

> ⚠ **Warning**
>
> Do not compress, tamper with, or otherwise alter changelog database files directly, unless specifically instructed to do so by a qualified ForgeRock technical support engineer.
> External changes to changelog database files can render them unusable by the server.

Enable the external changelog

DS servers that have a replication server port and an LDAP port publish an external changelog over LDAP:

Ports Configured	Examples	Notes
LDAP or LDAPS port	`1389`, `1636`	LDAP client applications use the LDAP or LDAPS port to read changelog data. A standalone replication server may not have an LDAP or LDAPS port configured.
Replication port	`8989`	Servers with replication ports maintain a changelog for their own use. The changelog is exposed over LDAP under the base DN, `cn=changelog`. Standalone directory servers do not maintain a changelog by default.

1. Make sure an LDAP or LDAPS port is configured so that LDAP client applications can read the changelog.
2. Depending on the server configuration, enable the changelog one of the following ways:
    1. For servers with replication ports, make sure replication is properly configured.

       The following example shows how the directory superuser can read the changelog:

       ```
       $ ldapsearch \
        --hostname localhost \
        --port 1636 \
        --useSsl \
        --usePkcs12TrustStore /path/to/opendj/config/keystore \
        --trustStorePassword:file /path/to/opendj/config/keystore.pin \
        --bindDN uid=admin \
        --bindPassword password \
        --baseDN cn=changelog \
        --searchScope base \
        "(&)"

       dn: cn=changelog
       objectclass: top
       objectclass: container
       cn: changelog
       ```

    2. For standalone directory servers without replication ports, you can manually enable the changelog. These steps cause the server to begin maintaining a replication changelog, which consumes disk space:
        - Update the default replication synchronization configuration entry as in the following example:

```
$ dsconfig \
 set-synchronization-provider-prop \
 --provider-name "Multimaster Synchronization" \
 --set bootstrap-replication-server:localhost:8989 \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

- Create a replication server configuration entry as in the following example:

```
$ dsconfig \
 create-replication-server \
 --provider-name "Multimaster Synchronization" \
 --set replication-port:8989 \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

- Create a replication domain configuration entry as in the following example:

```
$ dsconfig \
 create-replication-domain \
 --provider-name "Multimaster Synchronization" \
 --domain-name "Example.com" \
 --set base-dn:dc=example,dc=com \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. Make sure the user who needs to read changelog data has the `changelog-read` privilege, and has access to read entries under `cn=changelog`.

   For details, see Let a User Read the Changelog.

Encrypt changelog data

DS servers do not encrypt changelog data on disk by default. Any user with system access to read directory files can potentially read external changelog data.

In addition to preventing read access by other users, you can configure confidentiality for changelog data. When confidentiality is enabled, the server encrypts changelog records before storing them on disk. The server decrypts changelog records before returning them to client applications.

> **Important**
>
> Encrypting stored directory data does not prevent it from being sent over the network in the clear.
> Use secure connections to protect data sent over the network.

DS servers encrypt data using symmetric keys. Servers store symmetric keys, encrypted with the shared master key, with the data they encrypt. As long as servers have the same shared master key, any server can recover symmetric keys needed to decrypt data.

Symmetric keys for (deprecated) reversible password storage schemes are the exception to this rule. When you configure a reversible password storage scheme, enable the `adminRoot` backend, and configure a replication domain for `cn=admin data`.

Encrypting and decrypting data require cryptographic processing that reduces throughput, and extra space for larger encrypted values. Tests with default settings show that the cost of enabling confidentiality can be quite modest. Your results can vary based on the host system hardware, the JVM, and the settings for `cipher-transformation` and `cipher-key-length`. Make sure you test your deployment to qualify the impact of confidentiality before changing settings in production.

Follow these steps to enable confidentiality:

1. Before you enable confidentiality on a replication server for the changelog data, first enable confidentiality for data stored in directory backends.

   For details, see Data encryption.

2. Enable changelog confidentiality with the default encryption settings:

   ```
   $ dsconfig \
    set-replication-server-prop \
    --provider-name "Multimaster Synchronization" \
    --set confidentiality-enabled:true \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

   Encryption applies to the entire changelog regardless of the confidentiality settings for each domain.

   After confidentiality is enabled, new changelog records are encrypted. DS servers do not rewrite old records in encrypted form.

3. If necessary, adjust additional confidentiality settings.

   Use the same cipher suite for changelog confidentiality and data confidentiality.

The default settings for confidentiality are `cipher-transformation: AES/GCM/NoPadding`, and `cipher-key-length: 128`. This means the algorithm is the Advanced Encryption Standard (AES), and the cipher mode is Galois/Counter Mode (GCM). The syntax for the `cipher-transformation` is `algorithm/mode/padding`. You must specify the *algorithm*, *mode*, and *padding*. When the algorithm does not require a mode, use `NONE`. When the algorithm does not require padding, use `NoPadding`.

Let a user read the changelog

For a user to read the changelog, the user must have access to read, search, and compare changelog attributes, might have access to use the control to read the external changelog, and must have the `changelog-read` privilege.

1. Give the user access to read and search the changelog.

   The following example adds two global ACIs. The first ACI gives `My App` read access to root DSE attributes that hold information about the changelog. The second ACI gives `My App` read access to the changelog data:

   ```
   $ dsconfig \
    set-access-control-handler-prop \
    --add global-aci:"(target=\"ldap:///\")\
   (targetattr=\"changeLog||firstChangeNumber||lastChangeNumber||lastExternalChangeLogCookie\")\
   (version 3.0; acl \"My App can access changelog attributes on root DSE\"; \
   allow (read,search,compare) \
   userdn=\"ldap:///cn=My App,ou=Apps,dc=example,dc=com\";)" \
    --add global-aci:"(target=\"ldap:///cn=changelog\")(targetattr=\"*||+\")\
   (version 3.0; acl \"My App can access cn=changelog\"; \
   allow (read,search,compare) \
   userdn=\"ldap:///cn=My App,ou=Apps,dc=example,dc=com\";)" \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

   The IDM liveSync feature requires access to the root DSE attributes `changeLog`, `firstChangeNumber`, and `lastChangeNumber`.

2. Give the user access to use the public changelog exchange control.

   The following example adds a global ACI to give `My App` access to use the control:

```
$ dsconfig \
 set-access-control-handler-prop \
 --add global-aci:"(targetcontrol=\"Ecl\")\
(version 3.0; acl \"My App control access\"; \
allow (read) \
userdn=\"ldap:///cn=My App,ou=Apps,dc=example,dc=com\";)" \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. Give the user the `changelog-read` privilege:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: changelog-read
EOF
```

4. Check that the user can read the changelog:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN cn=changelog \
 --control "ecl:false" \
 "(&)" \
 changes changeLogCookie targetDN
```

Include unchanged attributes

The changes returned from a search on the external changelog include only what was actually changed. If you have applications that need additional attributes published with every changelog entry, regardless of whether the attribute itself has changed, specify those with the `ecl-include` and `ecl-include-for-deletes` properties:

1. Set the attributes to include for all update operations:

```
$ dsconfig \
 set-replication-domain-prop \
 --provider-name "Multimaster Synchronization" \
 --domain-name dc=example,dc=com \
 --set ecl-include:"@person" \
 --set ecl-include:entryUUID \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The `entryUUID` can be useful, for example, to ensure integrity when entries are renamed.

2. Set the attributes to include for deletes:

```
$ dsconfig \
 set-replication-domain-prop \
 --provider-name "Multimaster Synchronization" \
 --domain-name dc=example,dc=com \
 --add ecl-include-for-deletes:"*" \
 --add ecl-include-for-deletes:"+" \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

With the default configuration, the changelog records only the DN of the deleted entry.

Exclude a domain

Exclude domains to prevent applications that read the external changelog from having to process update notifications for entries that are not relevant to them:

1. Change the replication server configuration to exclude the domain by base DN.

   The following example prevents the changelog from sending notifications about Example.com entries:

```
$ dsconfig \
 set-replication-server-prop \
 --provider-name "Multimaster Synchronization" \
 --set changelog-enabled-excluded-domains:dc=example,dc=com \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Align draft change numbers

The external changelog can be used by applications that follow the Internet-Draft: Definition of an Object Class to Hold LDAP Change Records, and cannot process changelog cookies. Only default change number indexing is required to get the objects specified for this format, but there are steps you must perform to align change numbers across servers.

Change numbers described in the Internet-Draft are simple numbers, not cookies. When changelog numbers are aligned, applications can fail over from one server to another when necessary.

If you do not align the change numbers, each server keeps its own count. The same change numbers can refer to different changes on different servers.

For example, if you manually initialize a server from another, the last change numbers are likely to differ. The following example shows different last change numbers for two such servers:

```
$ ldapsearch \
 --hostname existing-ds.example.com \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN "" \
 --searchScope base \
 "(&)" lastChangeNumber

dn:
lastChangeNumber: 285924

Result Code:  0 (Success)

$ ldapsearch \
 --hostname new-ds.example.com \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN "" \
 --searchScope base \
 "(&)" lastChangeNumber

dn:
lastChangeNumber: 198643

Result Code:  0 (Success)
```

Follow these steps to align the change numbers with those of an existing server:

1. Make sure that the new server has the same replication configuration as the existing server.

   The change number calculations must be the same on both servers. Specifically, both servers must replicate data under the same base DNs. If the base DN configurations differ, the change numbers cannot be aligned.

2. If you must start the new servers's change numbering from a specific change, determine the `changeNumber` to use.

   The `changeNumber` must be from a change that has not yet been purged following the replication purge delay (default: 3 days).

3. Reset the change number on the new server to the change number from the existing server.

   The following example does not specify the change number to use. By default, the new server uses the last change number from the existing server:

```
$ dsrepl \
 reset-change-number \
 --sourceHostname existing-ds.example.com \
 --sourcePort 4444 \
 --sourceBindDn uid=admin \
 --sourceBindPasswordPassword password \
 --targetHostname new-ds.example.com \
 --targetPort 4444 \
 --targetBindDn uid=admin \
 --targetBindPasswordPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The new server's changelog now starts with the last change number from the existing server. Earlier change numbers are no longer present in the new servers's changelog.

Disable change number indexing

By default, a replication server indexes change numbers for replicated user data. This allows applications to get update notifications by change number. Indexing change numbers requires additional CPU, disk I/O, and storage. Disable it if none of your applications require change number-based browsing:

1. Adjust the replication server settings appropriately for your deployment:
    1. If applications need change notifications, but use changelog cookies rather than change numbers, set `changelog-enabled:enabled-cookie-mode-only` on each server.

       The replication server no longer indexes change numbers, and so has less work to do.
    2. If no applications need notifications, set `changelog-enabled:disabled` on each server after enabling replication.
    3. If applications need notifications for some domains but not for others, set `changelog-enabled-excluded-domains` on each server after enabling replication.

Referrals

Referrals point directory clients to another directory container, which can be another directory server running elsewhere, or another container on the same server. The client receiving a referral must use the other container to complete the request.

> **Note**
>
> Some clients follow referrals on your behalf by default. The DS commands do not follow referrals.

Referrals are used, for example, when directory data is temporarily unavailable due to maintenance. Referrals can also be used when a container holds only some of the directory data for a base DN, and points to other containers for branches whose data is not available locally.

Referrals are entries with https://www.rfc-editor.org/info/rfc4516 `ref` attribute values that point elsewhere. The `ref` attribute type is required by the `referral` object class. The `referral` object class is structural. By default, you cannot add it to an entry that already has a structural object class defined. (When adding a `ref` attribute to an entry with an existing structural object class, use the `extensibleObject` auxiliary object class.)

DS servers return referrals to requests that target the affected entry or its child entries. Client applications must be capable of following the referral returned. When the directory responds with a referral, the client can construct new operations and try them again.

The Manage DSAIT control lets you access the referral entry, rather than get a referral. It has OID `2.16.840.1.113730.3.4.2`. The control is described in RFC 3296.

Manage referrals

Suppose the entries below `ou=Subscribers,dc=example,dc=com` are stored on a different directory server at `referral.example.com:1636`. You can create a LDAP referral to reference the remote entries.

Before creating the LDAP referral, give directory administrators access to use the Manage DSAIT control, and to manage the `ref` attribute:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --add global-aci:"(targetcontrol=\"ManageDsaIt\")\
 (version 3.0; acl \"Allow Manage DSAIT control\"; allow(read)\
 groupdn=\"ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com\";)" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=example,dc=com")(targetattr="ref")
  (version 3.0; acl "Admins can manage referrals"; allow(all)
  (groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com");)
EOF
```

To create an LDAP referral, either create a referral entry, or add the `extensibleObject` object class and the `ref` attribute with an LDAP URL to an existing entry. The example above creates a referral entry at `ou=Subscribers,dc=example,dc=com`:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: ou=Subscribers,dc=example,dc=com
objectClass: top
objectClass: extensibleObject
objectClass: organizationalUnit
ou: Subscribers
ref: ldaps://referral.example.com:1636/ou=Subscribers,dc=example,dc=com
EOF
```

DS servers can now return a referral for operations under `ou=Subscribers`:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN ou=subscribers,dc=example,dc=com \
 "(uid=*)"

# The LDAP search request failed: 10 (Referral)
# Additional Information:  A referral entry ou=Subscribers,dc=example,dc=com indicates that the operation must be
processed at a different server
# Matched DN:  ou=Subscribers,dc=example,dc=com
```

To access the entry instead of the referral, use the Manage DSAIT control:

```
$ ldapsearch \
 --control 2.16.840.1.113730.3.4.2:true \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN ou=subscribers,dc=example,dc=com \
 "(&)" \
 ref

dn: ou=Subscribers,dc=example,dc=com
ref: ldaps://referral.example.com:1636/ou=Subscribers,dc=example,dc=com
```

You can use the Manage DSAIT control to change the referral with the `ldapmodify` command:

```
$ ldapmodify \
 --control 2.16.840.1.113730.3.4.2:true \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: ou=Subscribers,dc=example,dc=com
changetype: modify
replace: ref
ref: ldap://localhost:1636/ou=People,dc=example,dc=com
EOF
```

Attribute uniqueness

Some attribute values must remain unique. For example, if you use `uid` as the RDN for millions of user entries, you must avoid two or more identical `uid` values. As another example, if credit card or mobile numbers are stored on directory attributes, you want to be certain that neither is shared with another customer.

DS servers use the unique attribute plugin to ensure attribute value uniqueness. In a deployment with multiple replicas and unique attributes, direct updates for each unique attribute to a single replica at a time as described below.

Enable unique UIDs

By default, the unique attribute plugin is configured to ensure unique `uid` values:

1. Set the base DN where the `uid` should have unique values, and enable the plugin:

   ```
   $ dsconfig \
    set-plugin-prop \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --plugin-name "UID Unique Attribute" \
    --set base-dn:ou=people,dc=example,dc=com \
    --set enabled:true \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

   You can optionally specify unique values across multiple base DNs:

```
$ dsconfig \
 set-plugin-prop \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --plugin-name "UID Unique Attribute" \
 --set enabled:true \
 --add base-dn:ou=people,dc=example,dc=com \
 --add base-dn:ou=people,dc=example,dc=org \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Check your work:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=ajensen,ou=People,dc=example,dc=com
changetype: modify
add: uid
uid: bjensen
EOF

# The LDAP modify request failed: 19 (Constraint Violation)
# Additional Information:  A unique attribute conflict was detected for attribute uid: value bjensen already
exists in entry uid=bjensen,ou=People,dc=example,dc=com
```

If you have set up multiple base DNs, check your work as follows:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=bjensen,ou=People,dc=example,dc=org
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Babs
sn: Jensen
uid: bjensen
EOF

# The LDAP modify request failed: 19 (Constraint Violation)
# Additional Information:  A unique attribute conflict was detected for attribute uid: value bjensen already
exists in entry uid=bjensen,ou=People,dc=example,dc=com
```

Make other attributes unique

You can configure the unique attribute plugin for use with any attributes, not just `uid`:

1. Before you set up the plugin, index the attribute for equality.

   For instructions, see Configure Indexes.

2. Set up the plugin configuration for your attribute using one of the following alternatives:

   1. Add the attribute to an existing plugin configuration:

      ```
      $ dsconfig \
       set-plugin-prop \
       --hostname localhost \
       --port 4444 \
       --bindDN uid=admin \
       --bindPassword password \
       --plugin-name "UID Unique Attribute" \
       --add type:telephoneNumber \
       --usePkcs12TrustStore /path/to/opendj/config/keystore \
       --trustStorePassword:file /path/to/opendj/config/keystore.pin \
       --no-prompt
      ```

      Choose this alternative if you want each value to be unique across all configured attributes.

   2. Create a new plugin configuration:

```
$ dsconfig \
 create-plugin \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --plugin-name "Unique phone numbers" \
 --type unique-attribute \
 --set enabled:true \
 --set base-dn:ou=people,dc=example,dc=com \
 --set type:telephoneNumber \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Choose this alternative if values only need to be unique within the context of a particular attribute.

3. Check your work:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=ajensen,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 828 555 1212

dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: telephoneNumber
telephoneNumber: +1 828 555 1212
EOF

# MODIFY operation successful for DN uid=ajensen,ou=People,dc=example,dc=com

# The LDAP modify request failed: 19 (Constraint Violation)
# Additional Information:  A unique attribute conflict was detected for attribute telephoneNumber: value +1
828 555 1212 already exists in entry uid=ajensen,ou=People,dc=example,dc=com
```

Scope uniqueness

In some cases, attributes must be unique, but only in the context of a particular base DN. For example, `uid` values must be unique under `dc=example,dc=com` and under `dc=example,dc=org`. But it is okay to have `uid=bjensen,ou=people,dc=example,dc=com` and `uid=bjensen,ou=people,dc=example,dc=org`:

1. If the attribute you target is not indexed for equality by default, index the attribute for equality.

   See Configure Indexes for instructions.

2. For each base DN, set up a configuration entry that ensures the target attribute values are unique:

```
$ dsconfig \
 create-plugin \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --plugin-name "Unique Example.com UIDs" \
 --type unique-attribute \
 --set enabled:true \
 --set base-dn:dc=example,dc=com \
 --set type:uid \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
$ dsconfig \
 create-plugin \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --plugin-name "Unique Example.org UIDs" \
 --type unique-attribute \
 --set enabled:true \
 --set base-dn:dc=example,dc=org \
 --set type:uid \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. Check your work:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=unique,ou=People,dc=example,dc=com
uid: unique
givenName: Unique
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
cn: Unique Person
sn: Person
userPassword: 1Mun1qu3

dn: uid=unique,ou=People,dc=example,dc=org
uid: unique
givenName: Unique
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
cn: Unique Person
sn: Person
userPassword: 1Mun1qu3

dn: uid=copycat,ou=People,dc=example,dc=com
uid: unique
uid: copycat
givenName: Copycat
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
cn: Copycat Person
sn: Person
userPassword: copycopy
EOF

# ADD operation successful for DN uid=unique,ou=People,dc=example,dc=com

# The LDAP modify request failed: 19 (Constraint Violation)
# Additional Information:  A unique attribute conflict was detected for attribute uid: value unique already
exists in entry uid=unique,ou=People,dc=example,dc=com
```

Use uniqueness with replication

The unique attribute plugin only ensures uniqueness on the replica where the attribute is updated. If client applications write the same attribute value separately at the same time on different replicas, both replicas might use the same "unique" value, especially if the network is down between the replicas:

1. Configure the plugin identically on all replicas.

2. To avoid duplicate values where possible, use DS directory proxy to direct all updates of the unique attribute to the same replica.

Samba password sync

Samba, the Windows interoperability suite for Linux and UNIX, stores accounts because UNIX and Windows password storage management is not interoperable. The default account storage mechanism works well with small numbers of accounts and one domain controller. For larger installations, Samba can use DS replicas to store Samba accounts. See the Samba documentation for your platform for instructions on how to configure LDAP directory servers as Samba `passdb` backends.

The procedures that follow focus on how to keep passwords in sync for Samba account storage.

When you store Samba accounts in a directory server, Samba stores its own attributes as defined in the Samba schema. Samba does not use the LDAP standard `userPassword` attribute to store users' Samba passwords. You can configure Samba to apply changes to Samba passwords to LDAP passwords as well. Yet, if a user modifies their LDAP password directly without updating the Samba password, the LDAP and Samba passwords get out of sync.

The DS Samba Password plugin resolves this problem for you. The plugin intercepts password changes to Samba user profiles, synchronizing Samba password and LDAP password values. For an incoming Password Modify Extended Request or modify request to change the user password, the DS Samba Password plugin detects whether the user's entry is a Samba user profile (entry has object class `sambaSAMAccount`), hashes the incoming password value, and applies the password change to the appropriate password attribute, keeping the password values in sync. The DS Samba Password plugin can perform synchronization as long as new passwords are provided in plaintext in the modification request. If you configure Samba to synchronize LDAP passwords when it changes Samba passwords, the plugin can ignore changes by the Samba user to avoid duplicate synchronization.

Create the Samba administrator

The Samba Administrator updates the LDAP password when a Samba password changes.

In Samba's `smb.conf` configuration file, the value of `ldap admin dn` is set to the DN of this account. When the Samba Administrator changes a user password, the plugin ignores the changes. Choose a distinct account different from the directory superuser and other administrators:

1. Create or choose an account for the Samba Administrator:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=Samba Admin,ou=Special Users,dc=example,dc=com
cn: Samba Administrator
givenName: Samba
mail: samba@example.com
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
sn: Administrator
uid: Samba Admin
userPassword: chngthspwd
EOF
```

2. Let the Samba Administrator reset user passwords:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=Samba Admin,ou=Special Users,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: password-reset

dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=example,dc=com")(targetattr ="*")
 (version 3.0; acl "Samba Admin user rights"; allow(all)
  userdn="ldap:///uid=Samba Admin,ou=Special Users,dc=example,dc=com";)
EOF
```

Enable the Samba password plugin

1. Determine whether the plugin must store passwords hashed like LanManager (`sync-lm-password`) or like Windows NT (`sync-nt-password`), based on the Samba configuration.

2. Enable the plugin:

```
$ dsconfig \
 create-plugin \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --plugin-name "Samba Password Synchronisation" \
 --type samba-password \
 --set enabled:true \
 --set pwd-sync-policy:sync-nt-password \
 --set samba-administrator-dn:"uid=Samba Admin,ou=Special Users,dc=example,dc=com" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The Samba Password plugin is active immediately.

3. When troubleshooting Samba Password plugin issues, turn on debug logging:

```
$ dsconfig \
 create-debug-target \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --target-name org.opends.server.plugins.SambaPasswordPlugin \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ tail -f /path/to/opendj/logs/debug
```

LDAP proxy

Directory Services proxy services let you build a single point of access to a directory service, with a uniform view of the underlying LDAPv3 user data. This hides implementation details from directory client applications, and promotes scalability and availability.

When you set up a directory proxy server, no user data is stored locally. The server acts purely as an LDAP proxy. The only local data is for the server configuration and LDAP schema. In addition, the server is set up to use global access control policies rather than global ACIs. Global access control policies provide coarse-grained access control suitable for use on proxy servers, where the lack of local access to directory data makes ACIs a poor fit.

Proxy services are provided by proxy backends. Proxy backends connect to remote directory servers using a dynamic and configurable discovery mechanism. They route requests to remote directory servers. The way they distribute requests is configurable. The way they handle failures depends on the response from the directory server.

LDAP schema definitions for user data must be aligned on the proxy server and on the remote directory servers.

ACIs are handled by the directory server where the target data is stored. In other words, global access control policies set on a proxy server do not change ACIs on the remote directory server. Set ACIs appropriately on the directory server independent of proxy settings.

Remote LDAP servers

When the target DN of an LDAP request is not in a local backend, an LDAP server can refuse to handle the request, or return a referral. An LDAP proxy can also forward the request to another directory server.

In Directory Services, the LDAP proxy is implemented as a proxy backend. Rather than store user data locally, the proxy backend forwards requests to remote directory servers.

For proxy backends, a *service discovery mechanism* identifies remote directory servers to forward LDAP requests to. A service discovery mechanism's configuration specifies the keys used for secure communications, and how to contact the remote directory servers. It reads remote directory servers' configurations to discover their capabilities, including the naming contexts they serve, and so which target DNs they can handle. It periodically rereads their configurations in case they have been updated since the last service discovery operation.

When preparing to configure a service discovery mechanism, choose one of these alternatives:

Replication service discovery mechanism

This mechanism contacts DS replication servers to discover remote LDAP servers. Each replication server maintains information about the replication topology that lets the proxy server discover directory server replicas.

This mechanism only works with replicated DS 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. The account must have access and privileges to read the 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 remote LDAP servers.

This mechanism is designed to work with all LDAPv3 directory servers that support proxied authorization.

When configuring a service discovery mechanism, make sure all the remote directory servers are replicas of each other, and that they have the same capabilities. A proxy backend expects all remote directory server replicas known to the mechanism to hold the same data. This allows the backend to treat the replicas as equivalent members of a pool. In the configuration, a pool of equivalent replicas is a *shard*.

In deployments where you must distribute data for horizontal write scalability, you can configure multiple service discovery mechanisms. The proxy backend can then distribute write requests across multiple shards.

The following example creates a replication service discovery mechanism that specifies two replication servers:

```
$ dsconfig \
 create-service-discovery-mechanism \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --mechanism-name "Replication Service Discovery Mechanism" \
 --type replication \
 --set bootstrap-replication-server:rs1.example.com:4444 \
 --set bootstrap-replication-server:rs2.example.com:4444 \
 --set ssl-cert-nickname:ssl-key-pair \
 --set key-manager-provider:PKCS12 \
 --set trust-manager-provider:PKCS12 \
 --set use-start-tls:true \
 --set use-sasl-external:true \
 --no-prompt
```

The example above assumes that the servers protect connections using keys generated with a deployment ID and password. If this is not the case, configure the security settings appropriately.

The following example creates a static service discovery mechanism that specifies four remote LDAP servers:

```
$ dsconfig \
 create-service-discovery-mechanism \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --mechanism-name "Static Service Discovery Mechanism" \
 --type static \
 --set primary-server:local1.example.com:636 \
 --set primary-server:local2.example.com:636 \
 --set secondary-server:remote1.example.com:636 \
 --set secondary-server:remote2.example.com:636 \
 --set ssl-cert-nickname:ssl-key-pair \
 --set key-manager-provider:PKCS12 \
 --set trust-manager-provider:"JVM Trust Manager" \
 --set use-ssl:true \
 --set use-sasl-external:true \
 --no-prompt
```

The example above assumes that the remote servers use certificates signed by well-known CAs, and so are recognized using the trust manager provided by the JVM. If this is not the case, configure the security settings appropriately.

If the proxy server must perform SSL mutual authentication when setting up secure connections with the remote servers, configure an appropriate key manager provider and SSL certificate nickname.

Supported service discovery mechanisms define a `discovery-interval` that specifies how often to read the remote server configurations to discover changes. Because the mechanism polls periodically for configuration changes, by default, it can take up to one minute for the mechanism to see the changes. If necessary, you can change this setting in the configuration.

Routing requests

A proxy backend forwards requests according to their target DNs. The proxy backend matches target DNs to base DNs that you specify in the configuration. If you specify multiple shards for data distribution, the proxy also forwards requests to the appropriate shard.

When specifying base DNs, bear in mind the following points:

- A server responds first with local data, such as the server's own configuration or monitoring data. If a request target DN cannot be served from local data, the proxy backend can forward the request to a remote directory server.

  The proxy backend will forward the request if its target DN is under one of the specified base DNs.

- As an alternative to enumerating a list of base DNs to proxy, you can set the proxy backend property `route-all: true`.

  When you activate this property, the proxy backend will attempt to forward all requests that can be served by public naming contexts of remote servers. If the request target DN is not in a naming context supported by any remote directory servers associated with the proxy backend, then the proxy will not forward the request.

- The LDAP proxy capability is intended to proxy user data, not server-specific data.

  A server with a proxy backend still responds directly to requests that target private naming contexts, such as `cn=config`, `cn=tasks`, and `cn=monitor`. Local backends hold configuration and monitoring information that is specific to the server, and these naming contexts are not public.

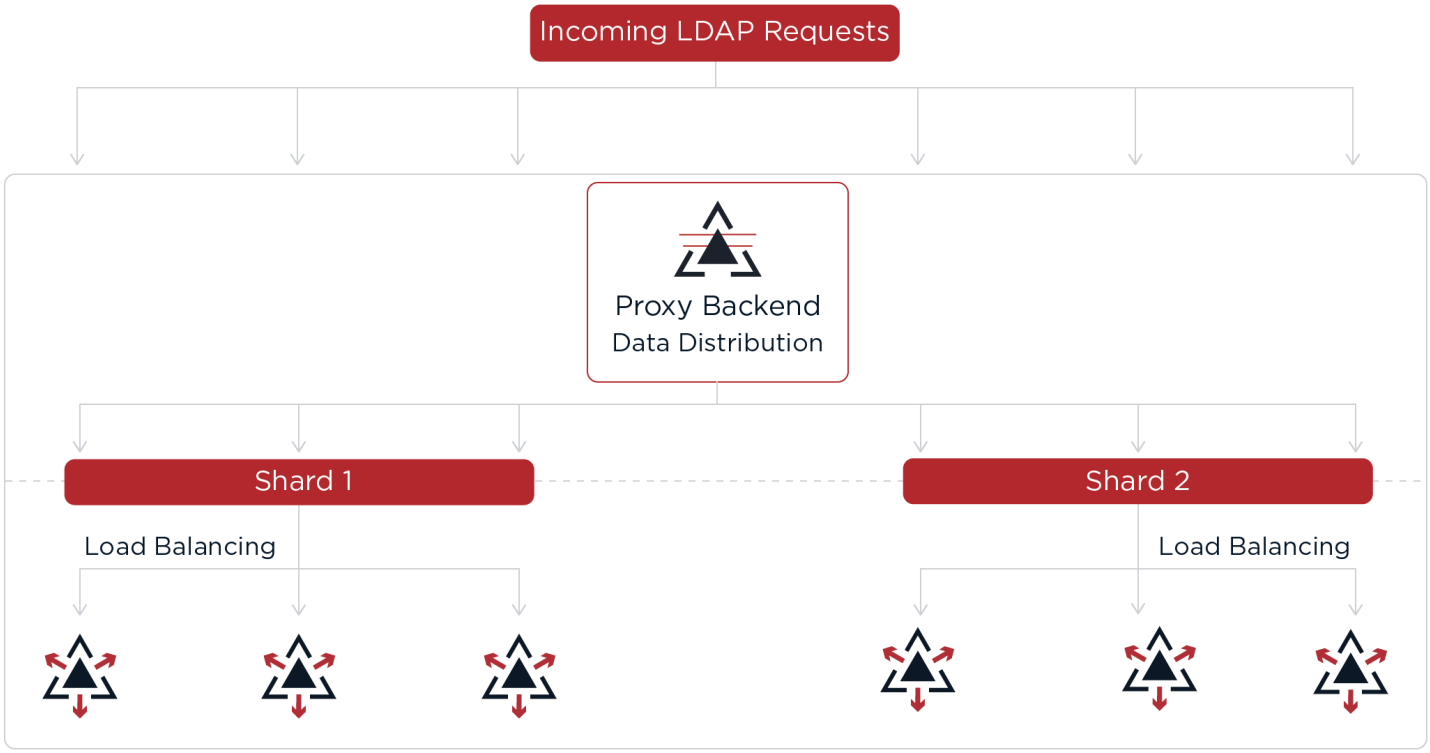
  Make sure client applications for configuration and monitoring access servers directly for the server-specific data they need.

- If you configure multiple proxy backends, each proxy backend must target distinct base DNs.

In deployments where you must distribute data for horizontal write scalability, you can configure multiple service discovery mechanisms for the same base DNs. Each service discovery mechanism targets its own distinct shard of directory data.

To enable write scalability beyond what is possible with a single shard of replicas, each shard must have its own independent replication configuration. Replication replays each update only within the shard.

The proxy backend distributes write requests across the shard, and balances the load within each shard:



Data distribution for horizontal scalability has the following properties:

- The proxy backend always routes requests targeting an entry below the partition base DN to the *same* shard.
- The proxy backend routes read requests targeting the partition base DN entry or above to *any* shard.

  In other words, the proxy backend can route each request to a different shard. When you deploy data distribution, the proxy rejects writes to the partition base DN entry and above through the proxy backend. Instead, you must perform such write operations on a replica in each shard.

  The best practice is therefore to update the partition base DN entry and above prior to deployment.
- The proxy backend routes search requests to *all* shards unless the search scope is below the partition base DN.

  For example, if the partition base DN is `ou=people,dc=example,dc=com`, the proxy backend always routes a search with base DN `uid=bjensen,ou=people,dc=example,dc=com` or `deviceid=12345,uid=bjensen,ou=people,dc=example,dc=com` to the same shard. However, it routes a search with base DN `ou=people,dc=example,dc=com` to all shards.

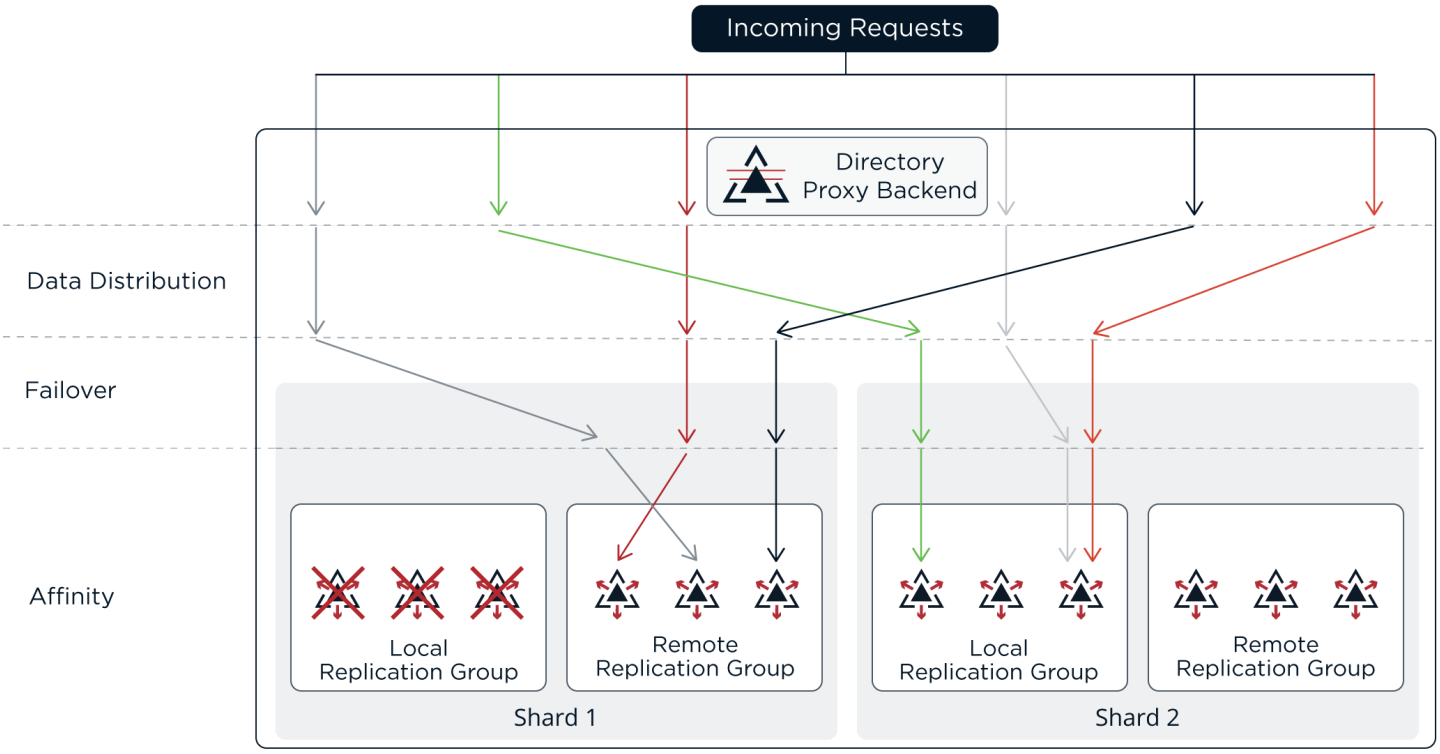
Load balancing

A directory proxy server balances the LDAP request load across the remote LDAP servers.

The proxy performs load balancing for the following purposes:

- **Data distribution**: sharding data to scale out
- **Failover**: routing requests to available directory servers
- **Affinity**: consistently forwarding requests for the same entry to the same server

The proxy applies these load balancing capabilities in hierarchical order to route requests. Follow the paths of requests through the proxy backend to a directory server in this diagram:



Notice in the diagram how the load balancing capabilities work together to route a request to a directory server:

- Data distribution routes to the correct shard.
- Failover routes around directory servers that are down, routing the request to available servers.
- Affinity routes the operation to a specific server.

Feature	Characteristics
Data Distribution	Routes requests with the same target DN below the partition base DN to the same shard. Data distribution helps to scale out the directory service horizontally. *When to Use* Use when you must scale out the directory service, and the deployment fits the constraints. A single DS directory service shard can process thousands of LDAP requests *per second* given the proper configuration and tuning. Furthermore, you can increase the search and read throughput simply by adding replicas. Configure data distribution for deployments when performance testing demonstrates that you cannot achieve write throughput requirements with properly configured and tuned replicas in a single replication topology. For example, use data distribution for a very high scale AM CTS deployment. *Settings* See Data distribution.

Feature	Characteristics
Failover	Routes LDAP requests to LDAP servers that are available and responding to health check probe requests. Failover prevents the proxy server from continuing to forward requests to unavailable servers. See also About failures. *When to Use* Use failover settings to route requests to local servers if they are available, and remote servers only when local servers are not available. *Settings* • Proxy Backend • Replication Service Discovery Mechanism • Static Service Discovery Mechanism
Affinity	Routes LDAP requests with the same target DN to the same server. Affinity load balancing helps applications that update and then reread the same entry in quick succession. This is not a best practice, but is often seen in client applications. With an add or modify request on an entry that is quickly followed by a read of the entry, the requests 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 application obtains the expected result. Affinity routing depends on the values of the proxy backend property, `partition-base-dn`. The proxy consistently routes requests for entries subordinate to these entries to the same server. The values of this property should therefore be the lowest entries in your DIT that are part of the DIT structure and not part of application data. In other words, when using affinity with two main branches, `ou=groups,dc=example,dc=com` and `ou=people,dc=example,dc=com`, set: • `partition-base-dn:ou=groups,dc=example,dc=com` • `partition-base-dn:ou=people,dc=example,dc=com` In terms of the CAP theorem, affinity load balancing provides consistency and availability, but not partition tolerance. As this algorithm lacks partition tolerance, configure it to load balance requests in environments where partitions are unlikely, such as a single data center with all directory servers on the same network. *When to Use* Use whenever possible, in combination with failover. Client application developers may be unaware of LDAP best practices. *Settings* Proxy backend properties: • `partition-base-dn`

LDAP schema

Proxy services are designed to proxy user data rather than server configuration or monitoring data. A proxy server must expose the same LDAP schema for user data as the remote directory servers.

If the schema definitions for user data differ between the proxy server and the remote directory servers, you must update them to bring them into alignment.

For details on DS server schema, see LDAP schema.

If the remote servers are not DS servers, see the schema documentation for the remote servers. Schema formats are not identical across all LDAPv3 servers. You will need to translate the differences into native formats, and apply changes locally on each server.

Proxy backend

Create proxy backends only on servers set up as proxy servers (with `setup --profile ds-proxy-server`).

Important

A directory proxy server connects using an account that must exist in the remote directory service.
The directory proxy server binds with this service account, and then forwards LDAP requests on behalf of other users.
For DS directory servers, use the proxied server setup profile if possible. For details, see Install DS for use with DS proxy.
The service account must have the following on all remote directory servers:

- The same bind credentials.
  If possible, use mutual TLS to authenticate the proxy user with the backend servers.
- The right to perform proxied authorization.
  Make sure the LDAP servers support proxied authorization (control OID: `2.16.840.1.113730.3.4.18`).
  For details, see RFC 4370, *Lightweight Directory Access Protocol (LDAP) Proxied Authorization Control*.
- When using a replication discovery mechanism with remote DS directory servers, the service account requires the `config-read` and `monitor-read` privileges for the service discovery mechanism. It requires the `proxied-auth` privilege and an ACI to perform proxied authorization.

The following listing shows an example service account that you could use with DS replicas. Adapt the account as necessary for your directory service:

```
dn: uid=proxy
objectClass: top
objectClass: account
objectClass: ds-certificate-user
uid: proxy
ds-certificate-subject-dn: CN=DS, O=ForgeRock.com
ds-privilege-name: config-read
ds-privilege-name: monitor-read
ds-privilege-name: proxied-auth
aci: (targetcontrol="ProxiedAuth")
  (version 3.0; acl "Allow proxied authorization";
  allow(read) userdn="ldap:///uid=proxy";)
```

Forward for example.com

The following example creates a proxy backend that forwards LDAP requests when the target DN is in `dc=example,dc=com`:

```
$ dsconfig \
 create-backend \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --backend-name proxyExampleCom \
 --type proxy \
 --set enabled:true \
 --set base-dn:dc=example,dc=com \
 --set route-all:false \
 --set use-sasl-external:true \
 --set ssl-cert-nickname:ssl-key-pair \
 --set key-manager-provider:PKCS12 \
 --set partition-base-dn:ou=groups,dc=example,dc=com \
 --set partition-base-dn:ou=people,dc=example,dc=com \
 --set shard:"Replication Service Discovery Mechanism" \
 --no-prompt
```

This command fails if `dc=example,dc=com` is already served by local data.

The settings for checking remote server availability and keeping connections alive are not shown:

- To check that the remote LDAP server is available, the proxy sends periodic availability-check requests.
- To keep connections from appearing idle and being forcefully closed, the proxy sends periodic keep-alive requests to the remote server.

The request settings are configurable. See the `availability-check-*` and `keep-alive-*` properties in Proxy Backend. About failures describes in more detail how the proxy backend works with these requests.

Forward all requests

The following example creates a proxy backend that forwards LDAP requests targeting user data. It does not specify base DNs explicitly, but uses the `route-all` property:

```
$ dsconfig \
 create-backend \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --backend-name proxyAll \
 --type proxy \
 --set enabled:true \
 --set route-all:true \
 --set use-sasl-external:true \
 --set ssl-cert-nickname:ssl-key-pair \
 --set key-manager-provider:PKCS12 \
 --set shard:"Static Service Discovery Mechanism" \
 --no-prompt
```

The example above assumes that the servers protect connections using keys generated with a deployment ID and password. If this is not the case, configure the security settings appropriately.

Proxy backends define a `discovery-interval` that specifies how often to read the remote server configurations to discover changes. Because the proxy polls periodically for configuration changes, by default, it can take up to one minute to see the changes. If necessary, you can change this setting in the configuration.

About failures

There are many specific ways that an LDAP request can fail. Not all failure result codes indicate a permanent failure, however. The following result codes from a remote directory server indicate a temporary failure:

- 51 (Busy) indicates that the server was too busy to process the request.

  The request can safely be tried on another peer server.

- 52 (Unavailable) indicates that the server was missing at least one resource needed to process the request.

  The request can safely be tried on another peer server.

When a forwarded request finishes with one of these return codes, the proxy backend retries the request on another server. In this case, the client does not receive the result from the remote directory server.

When a forwarded request finishes with a permanent server-side error return code, the proxy backend returns the result to the client application. In this case, the client must handle the error.

Connection failures can prevent the remote directory server from responding at all. The proxy backend handles connection failures differently, depending on whether it is inherently safe to replay the operation:

- For operations that read directory data, including search and compare requests, the proxy backend retries the request if a connection failure occurs.
- For operations that write directory data, including add, delete, modify, and modify DN requests, the proxy backend does not retry the request if a connection failure occurs.

  When the connection fails during a write operation, it is not possible to determine whether the change was applied. It is not safe, therefore, to replay the request on another server.

  The proxy returns an error to the client application.

- Connection failures during bind operations cause the proxy to retry the bind.

  In the unlucky event of repeated connection failures on successive bind attempts combined with a password policy configured to lock an account after a certain number of consecutive failures, it is possible that this behavior could lock an account.

A proxy backend protects against failed and stale connections with feedback from periodic availability-check and keep-alive requests to remote directory servers.

The requests serve these purposes:

- Checks that the remote directory server is still available.

  If the request fails, the proxy closes the unresponsive connection and connects to another remote directory server.

- Determines whether an unresponsive server has recovered.

When the remote directory server responds again to requests, the proxy can begin forwarding client requests to it again.

- Keeps the connection alive, preventing it from appearing idle and being forcefully closed.

If necessary, you can configure how often such requests are sent using the proxy backend configuration properties.

Access control and resource limits

When you deploy a directory proxy server, you limit the mechanisms for access control and resource limits. Such mechanisms cannot rely on ACIs in user data, resource limit settings on user entries, or group membership to determine what the server should allow. They can depend only on the server configuration and on the properties of the incoming request.

Global access control policies provide a mechanism suitable for directory proxy servers. For details, see Access control.

Resource limits set in the server configuration provide a way to prevent directory clients from using an unfair share of system resources. For details, see Resource limits, which covers how to update a server's configuration.

High availability

Directory services are designed for basic availability. Directory data replication makes it possible to read and write directory data when the network is partitioned, where remote servers cannot effectively contact each other. This sort of availability assumes that client applications can handle temporary interruptions.

Some client applications can be configured to connect to a set of directory servers. Some of these clients are able to retry requests when a server is unavailable. Others expect a single point of entry to the directory service, or cannot continue promptly when a directory server is unavailable.

Individual directory servers can become unavailable for various reasons:

- A network problem can make it impossible to reach the server.
- The server can be temporarily down for maintenance (backup, upgrade, and other purposes).
- The server can be removed from a replication topology and replaced with a different server.
- A crash can bring the server down temporarily for a restart or permanently for a replacement.

All of these interruptions must be managed on the client side. Some could require reconfiguration of each client application.

A directory proxy server provides high availability to client applications by hiding these implementation details from client applications. The proxy presents client applications with a uniform view of the remote directory servers. It periodically rereads the remote servers' configurations and checks connections to route requests correctly despite changes in server configurations and availability.

Directory proxy servers are well-suited for applications that need a single entry point to the directory service.

Single point of access

Unlike directory servers with their potentially large sets of volatile user data, directory proxy servers manage only their configuration state. Proxy servers can start faster and require less disk and memory space. Each directory proxy server can effectively be a clone of every other, with a configuration that does not change after server startup. Each clone routes LDAP requests and handles problems in the same way.

When you configure all directory proxy servers as clones of each other, you have a number of identical directory access points. Each point provides the same view of the underlying directory service. The points differ from each other only by their connection coordinates (host, port, and potentially key material to establish secure connections).

To consolidate identical access points to a single access point, configure your network to use a virtual IP address for the proxy servers. You can restart or replace proxy servers at any time, in the worst case losing only the client connections that were established with the individual proxy server.

An alternative to a single, global access point for all applications is to repeat the same approach for each key application. The proxy server configuration is specific to each key application. Clones with that configuration provide a single directory access point for the key application. Other clones do the same other for other key applications. Be sure to provide a more generic access point for additional applications.

Data distribution

Data distribution through the proxy comes with the following constraints:

- Data distribution is not elastic, and does not redistribute data if you add a shard. Once you deploy and use data distribution, you cannot change the number of shards.

  Plan for peak load when using data distribution to scale out your deployment.

  You can, however, add or change individual servers within a shard. For example, you could scale out by deploying many shards, and later upgrade to more powerful, faster underlying systems to scale up each shard.

- You cannot import distributed data from LDIF. The `import-ldif` command does not work with a proxy backend.

  If you must initialize distributed data, add the entries through the proxy server instead. You can use the `ldapmodify` command, for example, setting the `--numConnections` option to perform updates in parallel on multiple LDAP connections. You can also deploy as many proxy servers as necessary to perform the adds in parallel.

- Write requests for entries above the distributed data are not repeated on all shards. Instead, the proxy rejects such requests.

  If you must make a change to entries above the distributed data, make the change behind the proxy to one directory server replica in each shard.

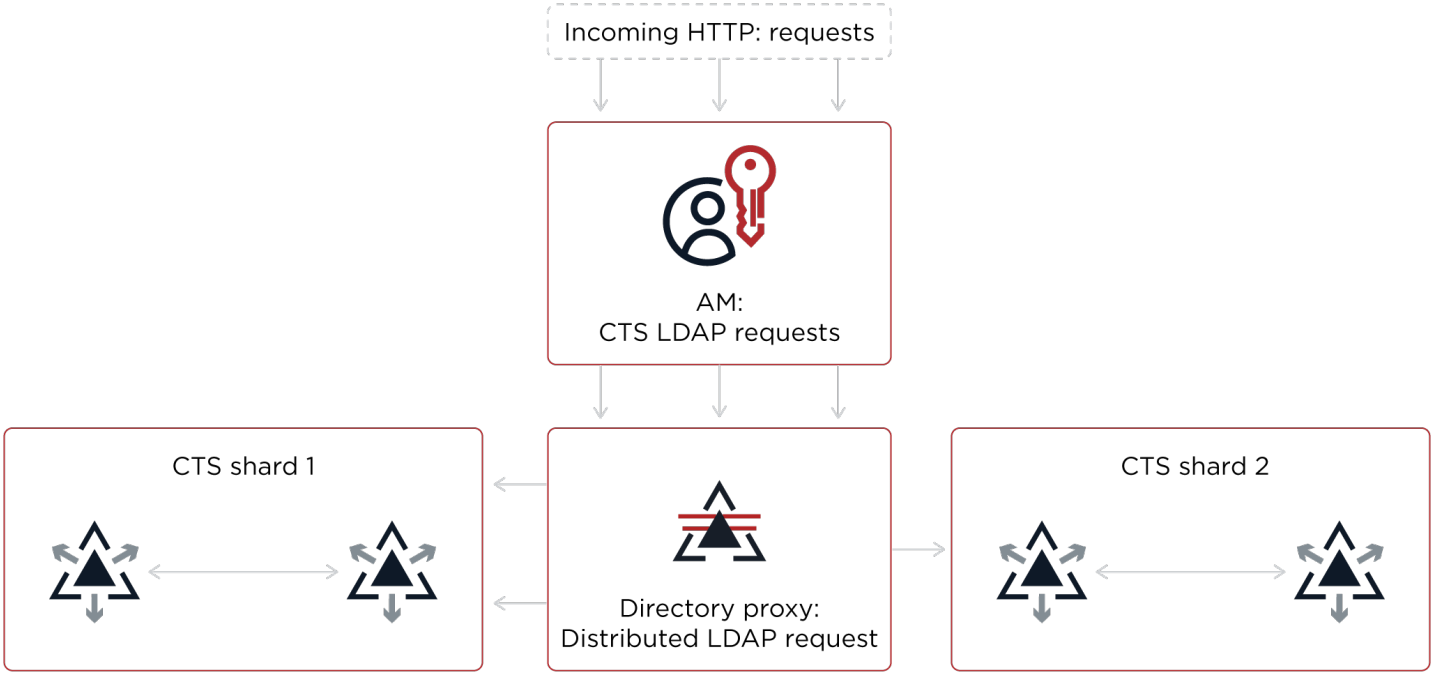
- Subtree and one-level searches can be relatively expensive, as the proxy must potentially forward the search request to every shard to retrieve all search results.

  To optimize searches, use a search base DN that points to a distributed entry.

- Given the present constraints of both data distribution and replication, the best fit for data distribution occurs where the distributed data remains self-contained and logically independent from other data branches.

  The example shown below distributes data for AM CTS tokens. The CTS data model fits DS data distribution well. AM stores all CTS token entries under the same DN. The entries above the distributed CTS tokens do not change during normal operation. CTS token entries do not have DN-value attributes that reference entries in another branch, nor are they members of LDAP groups stored in another branch.

The example that follows is intended for evaluation on a single computer. It involves installing five DS servers and one AM server:



As you follow the example, notice the following characteristics:

- The example is intended for evaluation only.

  In production, deploy each server on a separate system, and use secure connections.

- The AM server connects to the DS proxy for CTS LDAP requests.
- The DS proxy distributes LDAP requests across two shards.
- Each shard holds two DS replicas set up with the AM CTS profile.

  DS servers replicate only *within* shards. Servers in different partitions never replicate to each other. If replication crosses a partition boundary, the data is replicated everywhere. Replicating data everywhere would defeat the purpose of data distribution.

Try the CTS example

1. Make sure you have the Bash shell installed so that you can run the scripts to install and configure DS servers.
2. Download the DS .zip delivery and keep track of where you saved it.

   The scripts use the file name, `~/Downloads/DS-7.2.5.zip`.
3. Stop services that use the following ports. Servers in the example cannot start if these ports are in use:
   - `1636`
   - `4444`
   - `5444`
   - `5636`

- `5989`
- `6444`
- `6636`
- `6989`
- `8080`
- `15444`
- `15636`
- `15989`
- `16444`
- `16636`
- `16989`

4. Set up the DS replicas:
    - Save a local copy of the script: setup-cts-replicas.sh.

```
#!/usr/bin/env bash
#
# Copyright 2018-2020 ForgeRock AS. All Rights Reserved
#
# Use of this code requires a commercial software license with ForgeRock AS.
# or with one of its affiliates. All use shall be exclusively subject
# to such license between the licensee and ForgeRock AS.
#

###
# Set up directory server replicas for CTS in appropriate shards.
# This is intended for evaluation on a single system.
#
# In deployment, each of these replicas would be on a separate host system.
#
# The proxy distributes data across two CTS shards, each with two replicas.
#
# Each shard is served by a pool of separate replicas.
# In other words, each server replicates within one shard only.
#
# All servers have the same uid=Proxy service account.
# The proxy binds with uid=Proxy and uses proxied authorization.
# This script adds an ACI to allow the proxy service account
# to perform proxied authorization under the CTS shards.
#
# All CTS shards have a CTS admin account, uid=openam_cts,ou=tokens.
# To modify the account, for example to change the password,
# you must modify it once for each shard, not through the proxy.
# The proxy would distribute the change to only one shard.
#
# The shards have the following replication configurations,
# with each server's (admin-port ldaps-port):
# cts 1: (5444 5636) <==> (15444 15636)
# cts 2: (6444 6636) <==> (16444 16636)
###

###
# Adjust these variables to fit your circumstances:
ZIP=~/Downloads/opendj-7.2.5-20240524201627-49403efab3a3556b93d8ee62f263ca29a7e752ef.zip
BASE_DIR=/path/to
FQDN=localhost
DEPLOYMENT_ID=ADaMkVIXfryp4tZN3_0V4WoB3BZc9SQ5CBVN1bkVDE6OSY5Kl7pIibg
DEPLOYMENT_PASSWORD=password
###

CURRENT_DIR=$(pwd)

# Install a single DS/RS with the CTS profile from the .zip distribution.
# The AM CTS reaper will manage token expiration and deletion.
# $1: instance number
# $2: bootstrap server base
install() {
  echo "### Installing ${BASE_DIR}/ds-rs-${1} ###"
  unzip -q "${ZIP}"
  mv opendj "ds-rs-${1}"

  "${BASE_DIR}/ds-rs-${1}/setup" \
   --deploymentId "$DEPLOYMENT_ID" \
   --deploymentIdPassword "$DEPLOYMENT_PASSWORD" \
   --serverId "ds-rs-${1}" \
```

```
    --adminConnectorPort "${1}444" \
    --hostname "${FQDN}" \
    --ldapsPort "${1}636" \
    --enableStartTls \
    --replicationPort "${1}989" \
    --bootstrapReplicationServer "${FQDN}:${2}989" \
    --bootstrapReplicationServer "${FQDN}:1${2}989" \
    --rootUserDN uid=admin \
    --rootUserPassword password \
    --profile am-cts \
    --set am-cts/amCtsAdminPassword:password \
    --profile ds-proxied-server \
    --set ds-proxied-server/baseDn:ou=tokens \
    --acceptLicense

  echo "### Starting ds-rs-${1} ###"
  "${BASE_DIR}/ds-rs-${1}/bin/start-ds" --quiet
}

move_cts_admin() {
  echo "### Moving CTS admin account above the distributed data ###"
  "${BASE_DIR}/ds-rs-${1}/bin/ldapmodify" \
   --hostname "${FQDN}" \
   --port "${1}636" \
   --useSsl \
   --usePkcs12TrustStore "${BASE_DIR}/ds-rs-${1}/config/keystore" \
   --trustStorePassword:file "${BASE_DIR}/ds-rs-${1}/config/keystore.pin" \
   --bindDn uid=admin \
   --bindPassword password << EOF
dn: uid=openam_cts,ou=admins,ou=famrecords,ou=openam-session,ou=tokens
changetype: moddn
newrdn: uid=openam_cts
deleteoldrdn: 0
newsuperior: ou=tokens
EOF
}

add_aci() {
  echo "### Adding ACIs for moved CTS admin account on ds-rs-${1} ###"
  "${BASE_DIR}/ds-rs-${1}/bin/ldapmodify" \
   --hostname "${FQDN}" \
   --port "${1}636" \
   --useSsl \
   --usePkcs12TrustStore "${BASE_DIR}/ds-rs-${1}/config/keystore" \
   --trustStorePassword:file "${BASE_DIR}/ds-rs-${1}/config/keystore.pin" \
   --bindDn uid=admin \
   --bindPassword password <<EOF
dn: ou=tokens
changetype: modify
add: aci
aci: (targetattr="*") (version 3.0; acl "Allow read access for debugging";
  allow(read, search, compare) userdn = "ldap:///uid=openam_cts,ou=tokens";)

dn: ou=famrecords,ou=openam-session,ou=tokens
changetype: modify
add: aci
aci: (targetattr="*") (version 3.0; acl "Create, delete, update token entries";
  allow(add, delete, write) userdn = "ldap:///uid=openam_cts,ou=tokens";)
EOF
}

cd "${BASE_DIR}" || exit 1
```

```
for i in 5 6
do
  install ${i} ${i}
  install 1${i} ${i}
done

for i in 5 6
do
  move_cts_admin ${i}
  add_aci ${i}
done

# In this example, all replicas have the same data to start with.
# If the data is not identical on each pair of replicas, initialize replication manually.

cd "${CURRENT_DIR}" || exit
```

- If necessary, edit the script for use on your computer.
- Run the script.

  After the script finishes successfully, four DS replicas in two separate partitions are running on your computer.

5. Set up the DS proxy:
   - Save a local copy of the script: setup-cts-proxy.sh.

```
#!/usr/bin/env bash
#
# Copyright 2018-2022 ForgeRock AS. All Rights Reserved
#
# Use of this code requires a commercial software license with ForgeRock AS.
# or with one of its affiliates. All use shall be exclusively subject
# to such license between the licensee and ForgeRock AS.
#

###
# Set up a directory proxy server listening on the typical ports.
# This is intended for evaluation on a single system.
#
# In deployment, this would be a layer of identical proxy servers
# to balance incoming requests.
#
# This server distributes data across two replication shards for CTS.
# Each shard is served by two replicas.
# Each DS directory server replicates within one shard only.
###

###
# Adjust these variables to fit your circumstances:
ZIP=~/Downloads/opendj-7.2.5-20240524201627-49403efab3a3556b93d8ee62f263ca29a7e752ef.zip
BASE_DIR=/path/to
FQDN=localhost
DEPLOYMENT_ID=ADaMkVIXfryp4tZN3_0V4WoB3BZc9SQ5CBVN1bkVDE6OSY5Kl7pIibg
DEPLOYMENT_PASSWORD=password
###

CURRENT_DIR=$(pwd)

# Install a single proxy from the .zip distribution.
install() {
  echo "### Installing ${BASE_DIR}/proxy ###"
  unzip -q "${ZIP}"
  mv opendj proxy

  # The default proxyRoot is created at setup time, but removed later.
  ./proxy/setup \
   --deploymentId "$DEPLOYMENT_ID" \
   --deploymentIdPassword "$DEPLOYMENT_PASSWORD" \
   --serverId proxy \
   --rootUserDN uid=admin \
   --rootUserPassword password \
   --hostname "${FQDN}" \
   --ldapsPort 1636 \
   --adminConnectorPort 4444 \
   --profile ds-proxy-server \
   --set ds-proxy-server/bootstrapReplicationServer:"${FQDN}:5444" \
   --set ds-proxy-server/rsConnectionSecurity:ssl \
   --set ds-proxy-server/certNickname:ssl-key-pair \
   --set ds-proxy-server/keyManagerProvider:PKCS12 \
   --set ds-proxy-server/trustManagerProvider:PKCS12 \
   --acceptLicense

   # Allow the CTS administrator to read and write directory data.
  ./proxy/bin/dsconfig \
   create-global-access-control-policy \
    --type generic \
```

```
    --policy-name "CTS administrator access" \
    --set allowed-attribute:"*" \
    --set permission:read \
    --set permission:write \
    --set user-dn-equal-to:uid=openam_cts,ou=tokens \
    --set request-target-dn-equal-to:ou=tokens \
    --set request-target-dn-equal-to:**,ou=tokens \
    --offline \
    --no-prompt
}

# Configure distribution to multiple shards.
configure() {
  echo "### Configuring distribution for CTS shards ###"
  ./proxy/bin/dsconfig \
   create-service-discovery-mechanism \
   --mechanism-name "CTS Shard 1" \
   --type replication \
   --set bootstrap-replication-server:"${FQDN}:5444" \
   --set bootstrap-replication-server:"${FQDN}:15444" \
   --set ssl-cert-nickname:ssl-key-pair \
   --set key-manager-provider:PKCS12 \
   --set trust-manager-provider:PKCS12 \
   --set use-ssl:true \
   --set use-sasl-external:true \
   --offline \
   --no-prompt

  ./proxy/bin/dsconfig \
   create-service-discovery-mechanism \
   --mechanism-name "CTS Shard 2" \
   --type replication \
   --set bootstrap-replication-server:"${FQDN}:6444" \
   --set bootstrap-replication-server:"${FQDN}:16444" \
   --set ssl-cert-nickname:ssl-key-pair \
   --set key-manager-provider:PKCS12 \
   --set trust-manager-provider:PKCS12 \
   --set use-ssl:true \
   --set use-sasl-external:true \
   --offline \
   --no-prompt

  ./proxy/bin/dsconfig \
   create-backend \
   --backend-name distributeCts \
   --type proxy \
   --set enabled:true \
   --set partition-base-dn:ou=famrecords,ou=openam-session,ou=tokens \
   --set shard:"CTS Shard 1" \
   --set shard:"CTS Shard 2" \
   --set use-sasl-external:true \
   --set ssl-cert-nickname:ssl-key-pair \
   --set key-manager-provider:PKCS12 \
   --set route-all:true \
   --offline \
   --no-prompt
}

delete_unused_backend() {
  echo "### Removing unused proxyRoot backend ###"
  ./proxy/bin/dsconfig \
   delete-backend \
```

```
    --backend-name proxyRoot \
    --offline \
    --no-prompt
}

cd "${BASE_DIR}" || exit 1
install
configure
delete_unused_backend
echo "### Starting proxy ###"
./proxy/bin/start-ds --quiet
cd "${CURRENT_DIR}" || exit
```

- If necessary, edit the script for use on your computer.
- Run the script.

  After the script finishes successfully, the DS proxy server is configured to distribute requests to the two partitions running on your computer.

6. Install the AM server.

   For details, see the AM Installation Guide.

   You can use the default configuration options during installation.

7. Configure the AM server to use the directory proxy server.

   For details, see Configuring CTS Token Stores.

   This example has the following settings:

Connection string	`localhost:1636` AM must trust the DS `ssl-key-pair` server certificate to use this port.
Root suffix	`ou=famrecords,ou=openam-session,ou=tokens`
Login ID	`uid=openam_cts,ou=tokens`
Password	`password`

   When you restart AM, it uses the distributed CTS store. As you perform operations in AM that use CTS, such as logging in as a regular user, you can see the token entries through the DS proxy server.

   The following example shows two entries written by AM after logging in through the AM console as the `demo` user:

```
$ /path/to/proxy/bin/ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/proxy/config/keystore \
 --trustStorePassword:file /path/to/proxy/config/keystore.pin \
 --bindDn uid=openam_cts,ou=tokens \
 --bindPassword password \
 --baseDn ou=tokens \
 "(coreTokenId=*)" \
 coreTokenId

dn: coreTokenId=id,ou=famrecords,ou=openam-session,ou=tokens
coreTokenId: id
```

As AM performs CTS-related operations, you start to see the CTS tokens distributed across the two DS partitions. To examine the results, use LDAPS port `5636` for CTS partition 1 and `6636` for partition 2.

8. Install a second, identically configured AM server, and then test the system.

   For details, see To Test Session High Availability.

9. After finishing the evaluation, stop the servers and remove the software you installed:

   - Uninstall the AM server(s).
   - Tear down the DS servers:
     - Save a local copy of the script: teardown-cts.sh,

```
#!/usr/bin/env bash
#
# Copyright 2018-2022 ForgeRock AS. All Rights Reserved
#
# Use of this code requires a commercial software license with ForgeRock AS.
# or with one of its affiliates. All use shall be exclusively subject
# to such license between the licensee and ForgeRock AS.
#

###
# Stop and remove the proxy and replica servers.
###

###
# Adjust this path if you changed it in other scripts:
BASE_DIR=/path/to
###

CURRENT_DIR=$(pwd)

cd "${BASE_DIR}" || exit 1
./proxy/bin/stop-ds
rm -rf proxy

for i in 5 15 6 16
do
  ./ds-rs-${i}/bin/stop-ds
done
rm -rf ds-rs-*
cd "${CURRENT_DIR}" || exit
```

- If necessary, edit the script for use on your computer.
- Run the script.

  After the script finishes successfully, the DS software has been removed from your computer.

Proxy Protocol

The Proxy Protocol is an HAProxy Technologies protocol that safely transports connection information, such as a client's IP address, through multiple proxy layers.

DS servers support v1 and v2.

When and why to use the proxy protocol

DS servers have authorization features that rely on information about the client application connection, such as the client IP address and the security strength factor (SSF). DS ACIs, and lists of allowed, restricted, and denied clients use this information, for example.

When running DS behind a software load balancer, such as HAProxy, or AWS Elastic Load Balancing, the load balancer does network address translation. The load balancer connects to the DS server, and the DS does not have access to the client application connection. Cloud deployments often use a load balancer in this way.

Software load balancers that implement the proxy protocol can transfer the original client connection information through the protocol to the DS server. If your deployment uses a software load balancer and any features that rely on client connection information, enable the protocol in the load balancer, and configure proxy protocol support in DS.

Configure proxy protocol support

By default, support for the proxy protocol is disabled. You must enable and configure the feature.

Connections	Configuration
All client application traffic traverses the load balancer	Use the `dsconfig set-global-configuration-prop` command to set these global server configuration properties: • proxy-protocol-enabled • proxy-protocol-allowed-client
Some applications access DS directly	Create a dedicated LDAPS connection handler for the load balancer. Use the create-connection-handler command, and set these properties in the connection handler configuration: • proxy-protocol-enabled • proxy-protocol-allowed-client

Once you set `proxy-protocol-enabled:true`, make sure you include *all* the load balancers in the list of `proxy-protocol-allowed-client` values. DS or the connection handler you configured accepts only connections from these allowed clients, and only if they use the proxy protocol.

Secure connections

- If you configure the load balancer to connect to DS securely, using LDAPS or StartTLS, then you *must* configure the load balancer to listen only for secure client connections as well.

  Otherwise, the deployment becomes vulnerable, as there is no way to prevent the client from starting with an insecure connection to the load balancer.

- To communicate the client SSF from the load balancer to DS, the load balancer must use v2 of the proxy protocol, and you must enable transmission of Type-Length-Value (TLV) vectors for secure connections.

  For example, if you use HAProxy, set send-proxy-v2-ssl.

- For clients that use StartTLS or certificate-based authentication (SASL EXTERNAL), the load balancer must forward the connection to DS. The load balancer must *not* terminate the secure connection.

On load balancers

A load balancer might seem like a natural component for a highly available architecture. Directory services are highly available by design, however. When used with directory services, a load balancer can do more harm than good.

The problem

DS servers rely on data replication for high availability with tolerance for network partitions. The directory service continues to allow both read and write operations when the network is down. As a trade off, replication provides *eventual consistency*, not immediate consistency.

A load balancer configured to distribute connections or requests equitably across multiple servers can therefore *cause an application to get an inconsistent view of the directory data*. This problem arises in particular when a client application uses a pool of connections to access a directory service:

1. The load balancer directs a write request from the client application to a first server.

   The write request results in a change to directory data.

   The first server replicates the change to a second server, but replication is not instantaneous.

2. The load balancer directs a subsequent read request from the client application to a second server.

   The read request arrives before the change has been replicated to the second server.

   As a result, the second server returns the earlier view of the data. *The client application sees data that is different from the data it successfully changed!*

The following sequence diagram illustrates the race condition:

inconsistent

When used in failover mode, also known as active/passive mode, this problem is prevented. However, the load balancer adds network latency while reducing the number of directory servers actively used. This is unfortunate, since the directory server replicas are designed to work together as a pool.

Unlike many load balancers, ForgeRock Identity Platform software has the capability to account for this situation, and to balance the request load appropriately across multiple directory servers.

Recommendations

Apply the following recommendations in your directory service deployments:

Client is a ForgeRock Identity Platform 7.2 component

Do not use a load balancer between ForgeRock Identity Platform components and the DS directory service.

ForgeRock Identity Platform components use the same software as DS directory proxy to balance load appropriately across multiple directory servers.

Examples of platform components include AM and IDM.

Client opens a pool of connections to the directory service

Do not use a load balancer between the client and the DS directory service.

Configure the client application to use multiple directory servers.

Client and server are DS replicas

Never use a load balancer for replication traffic.

Client can only access a single directory server

Consider using DS directory proxy server to provide a single point of entry and balance load. Alternatively, use a load balancer in failover or active/passive mode.

Client only ever opens a single connection to the directory service

Consider using DS directory proxy server to provide a single point of entry and balance load. Alternatively, use a load balancer in failover or active/passive mode.

About request handling

DS servers listen for client requests using *connection handlers* . A connection handler interacts with client applications, accepting connections, reading requests, and sending responses. Most connection handlers expose configurable listen ports with security settings. The security settings point to other configuration objects, so two connection handlers can share the same certificate and private key, for example.

DS servers use different ports for different protocols. For example, a directory server might listen on port `389` for LDAP requests, port `443` for HTTPS requests, and port `4444` for administration requests from server configuration tools. Because DS servers use a different connection handler for each port, DS servers have several connection handlers enabled.

The `setup` command lets you initially configure connection handlers for LDAP or LDAPS, HTTP or HTTPS, and administrative traffic. The `dsconfig` command offers full access to all connection handler configurations.

When a client application opens a secure connection to a server, the JVM has responsibility for transport layer security negotiations. You can configure how connection handlers access keys required during the negotiations. You can also configure which clients on the network are allowed to use the connection handler. For details, see Connection Handler.

Connection handlers receive incoming requests, and pass them along for processing by the core server subsystem.

For example, an LDAP connection handler enqueues requests to the core server, which in turn requests data from the appropriate backend as necessary. For more information about backends, see Data storage. The core server returns the LDAP response.

Figure 1. LDAP Requests

An HTTP connection handler translates each request to LDAP. Internally, the core server subsystem processes the resulting LDAP requests.

Figure 2. HTTP Requests

DS servers support other types of connection handlers. For example, JMX and SNMP connection handlers support monitoring applications. A special LDIF connection handler consumes LDIF files.

When deploying a server, decide which listen ports to expose over which networks. Determine how you want to secure the connections, as described in Secure connections.

Security

This guide helps you to reduce risk and mitigate threats to directory service security.

Threats

Understand security threats.

Authentication

Enforce secure authentication.

Cryptographic Keys

Manage certificates and keys.

Connections

Secure network connections.

Passwords

Store and manage passwords.

Data Encryption

Protect data on disk.

Important

A guide to securing directory services can go wrong for many reasons, including at least the following:

- The author fails to understand or to properly explain the subject.
- The reader fails to understand or to act on what is written.
- Bugs exist in the directory's security-related features.

The authors of this guide aim to understand directory security features and issues before attempting to explain how to manage them.
The reader would do well to gain grounding in securing services and systems, and in applying and designing processes that prevent or mitigate threats, before reading the guide with a critical eye, and a grain of salt. This is not a guide to getting started with security.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

The common REST API provides ForgeRock Identity Platform software common ways to access web resources and collections of resources.

Threats

Review common threats to directory services, which you can mitigate by following the instructions in this guide.

Insecure client applications

Directory services make a good, central, distributed store for identity data and credentials.

Standard access protocols, and delegated access and data management mean you might not know which client applications use your services. Some client applications may behave insecurely:

- Prevent insecure connections.

  Require that applications connect with LDAPS and HTTPS, and restrict the protocol versions and cipher suites for negotiating secure connections to those with no known vulnerabilities. For details, see Secure connections.

- Accept only secure management of sensitive data.

  Nothing in LDAPv3 or HTTP prevents a client application from sending credentials and other sensitive account data insecurely. You can configure the directory to discourage the practice, however.

- Encourage secure authentication.

  For details, see Authentication mechanisms.

- Encourage best practices for client applications, such as scrubbing input to avoid injection vulnerabilities.

  For details, see Client best practices.

Client applications

Client applications can misuse directory services. They may be poorly built or incorrectly configured, and waste server resources. If your organization owns the client, help the owner fix the problem.

Misuse can be intentional, as in a denial-of-service attack. Protect the directory service to limit attacks.

Unreasonable requests from client applications include the following:

- Unindexed searches that would require the directory to evaluate all entries.

  Unindexed searches are not allowed by default for normal accounts, adjustable with the `unindexed-search` privilege.

  The access log records attempts to perform unindexed searches.

- Excessive use of overly broad persistent searches, particularly by clients that do not process the results quickly.

  Review requests before setting ACIs to grant access to use the persistent search control. Alternatively, let client applications read the external change log.

- Extremely large requests, for example, to update directory entries with large values.

  By default, requests larger than 5 MB are refused. The setting is per connection handler, `max-request-size`.

- Requests that make excessive demands on server resources.

  Set resource limits. See Resource limits.

- Requests to read entire large group entries only to check membership.

  Encourage client applications to read the `isMemberOf` attribute on the account instead.

Poor password management

Despite efforts to improve how people manage passwords, users have more passwords than ever before, and many use weak passwords. You can use identity and access management services to avoid password proliferation, and you can ensure the safety of passwords that you cannot eliminate.

As a central source of authentication credentials, directory services provide excellent password management capabilities. DS servers have flexible password policy settings, and a wide range of safe password storage options. Be sure that the passwords stored in your directory service are appropriately strong and securely stored.

For details, see Passwords.

Manage passwords for server administration securely as well. Passwords supplied to directory server tools can be provided in files, interactively, through environment variables, or as system property values. Choose the approach that is most appropriate and secure for your deployment.

Make sure that directory administrators manage their passwords well. To avoid password proliferation for directory administrators, consider assigning administration privileges and granting access to existing accounts for delegated administrative operations. For details, see Administrative roles and Access control.

Misconfiguration

With the power to administer directory services comes the responsibility to make sure the configuration is correct. Misconfiguration can arise from bad or mistaken configuration decisions, and from poor change management.

Bad configuration decisions can result in problems such as the following:

- A particular feature stops working.

  Depending on the configuration applied, features can stop working in obvious or subtle ways.

  For example, suppose a configuration change prevents the server from making LDAPS connections. Many applications will no longer be able to connect, and so the problem will be detected immediately. If the configuration change simply allows insecure TLS protocol versions or cipher suites for LDAPS connections, some applications will negotiate insecure TLS, but they will appear to continue to work properly.

- Access policy is not correctly enforced.

  Incorrect parameters for secure connections and incorrect ACIs can lead to overly permissive access to directory data, and potentially to a security breach.

- The server fails to restart.

  Although failure to start a server is not directly a threat to security, it can affect dependent identity and access management systems.

  Generally a result of editing the server configuration LDIF incorrectly, this problem can usually be avoided by using configuration tools. A server that started correctly saves a copy of the configuration in the `var/config.ldif.startok` file. You can compare this with the `config/config.ldif` file if the server fails to restart.

To guard against bad configuration decisions, implement good change management:

- For all enabled features, document why they are enabled and what your configuration choices mean.

  This implies review of configuration settings, including default settings that you accept.

- Validate configuration decisions with thorough testing.

  For details, see Tests.

- Maintain a record of your configurations, and the changes applied.

  For example, use a filtered directory audit log. Use version control software for any configuration scripts and to record changes to configuration files.

  Make sure you also maintain a record of external changes on the system, such as changes to operating system configuration, and updates to software such as the JVM that introduce security changes.

- Strongly encourage owners of applications that change ACIs, collective attributes, and similar settings in directory data to also follow good change management practices.

Unauthorized access

Data theft can occur when access policies are too permissive, and when the credentials to gain access are too easily cracked. It can also occur when the data is not protected, when administrative roles are too permissive, and when administrative credentials are poorly managed.

To protect against unauthorized access over the network, see the suggestions in Insecure client applications, Poor password management, and Access control.

To protect against unauthorized access by administrators, see the suggestions in Data encryption, and Administrative roles.

Poor risk management

Threats can arise when plans fail to account for outside risks.

Develop appropriate answers to at least the following questions:

- What happens when a server or an entire data center becomes unavailable?
- How do you remedy a serious security issue in the service, either in the directory service software or the underlying systems?
- How do you validate mitigation plans and remedial actions?
- How do client applications work when the directory service is offline?

  If client applications require always-on directory services, how do your operations ensure high availability, even when a server or data center goes offline?

For a critical directory service, you must test both normal, expected operation, and disaster recovery.

Security features

This short introduction provides an overview of DS security features.

Encryption and digests

When DS servers must store sensitive data, and file permissions alone are not sufficient, they use encryption and digests:

- *Encryption* turns source data into a reversible code. Good design makes it extremely hard to recover the data from the code without the decryption key.

  Encryption uses keys and cryptographic algorithms to convert source data into encrypted codes and back again. Given the decryption key and the details of the algorithm, converting an encrypted code back to source data is straightforward, though it can be computationally intensive.

  DS software does not implement its own versions of all encryption algorithms. Instead, it often relies on cryptographic algorithms provided by the underlying JVM. DS servers do manage access to encryption keys, however. An important part of server configuration concerns key management.

  DS servers use encryption to protect data and backup files on disk. They can encrypt password values when you configure a reversible storage scheme. Another important use of encryption is to make network connections secure.

- A *digest* (also called a hash) is a non-reversible code generated from source data using a one-way *hash function*. (A hash function is one that converts input of arbitrary size into output of fixed size.) Good one-way hash design design makes it effectively impossible to retrieve the source data even if you have access to the hash function.

  The hash function makes it simple to test whether a given value matches the original. Convert the value into a digest with the same hash function. If the new digest matches the original digest, then the values are also identical.

  DS server use digests to store hashed passwords, making the original passwords extremely hard to recover. They also use digests for authentication and signing.

In DS software, two types of encryption keys are used:

1. *Symmetric keys* , also called secret keys, because they must be kept secret.

   A single symmetric key is used for both encryption and decryption.

2. *Asymmetric key pairs* , consisting of a sharable public key and a secret private key.

   Either key can be used for encryption and the other for decryption.

Connection management

DS servers manage incoming client connections using *connection handlers*. Each connection handler is responsible for accepting client connections, reading requests, and sending responses. Connection handlers are specific to the protocol and port used. For example, a server uses one connection handler for LDAPS and another for HTTPS.

The connection handler configuration includes optional security settings. When you configure a handler, specify a *key manager provider* and a *trust manager provider*:

- The key manager provider retrieves the server certificate when negotiating a secure connection.

  A key manager provider is backed by a *keystore* , which stores the server's key pairs.

- The trust manager provider retrieves trusted certificates, such as CA certificates, to verify trust for a certificate presented by a peer when setting up a secure connection.

  A trust manager provider is backed by a keystore that contains trusted certificates, referred to as a *truststore* when used in this way.

DS servers support file-based keystores and PKCS#11 security tokens, such as HSMs.

Always use secure connections when allowing access to sensitive information. For details, see Secure connections.

Cryptographic key management

DS servers use cryptographic mechanisms for more than setting up secure connections:

- Encrypted backup files must be decrypted when restored.
- Passwords can be protected by encryption rather than hashing, although this is not recommended.
- Database backends can be encrypted for data confidentiality and integrity.

For all operations where data is stored in encrypted form, all replicas must be trusted to access the secret key.

Trust between servers depends on a public key infrastructure. This type of infrastructure is explained in more detail in Public Key Infrastructure.

Replication requires trust between the servers. Trust enables servers to secure network connections, and to share symmetric keys securely. Servers encrypt symmetric keys with a shared master key, and store them in replicated data. When a server needs the symmetric key for decryption or further encryption, it decrypts its copy with the master key.

The component that provides a common interface for cryptographic functions is called the DS Crypto Manager.

You can configure the following Crypto Manager features with the `dsconfig` command:

- Protection for symmetric keys.
- The alias of the shared master key to use for protecting secret keys.
- Cipher key lengths, algorithms, and other advanced properties.

Authentication

Authentication is the act of confirming the identity of a principal, such as a user, application, or device. The main reason for authentication is that authorization decisions are based on the identity of the principal.

Servers should require authentication before allowing access to any information that is not public.

Authentication mechanisms depend on the access protocol. HTTP has a number of mechanisms, such as HTTP Basic. LDAP has other mechanisms, such as anonymous bind and external SASL. For details on supported mechanisms, see Authentication mechanisms.

Authorization

Authorization is the act of determining whether to grant a principal access to a resource.

DS servers authorize access based on these mechanisms:

- *Access control instructions* (ACI)

  Access control instructions provide fine-grained control over LDAP operations permitted for a principal.

  ACIs can be replicated.

- Administrative *privileges*

  Privileges control access to administrative tasks, such as backup and restore operations, making changes to the configuration, and other tasks.

  Privileges can be replicated.

- *Global access control policies*

  Global access control policies provide coarse-grained access control for proxy servers, where the lack of local access to directory data makes ACIs a poor fit.

For details about ACIs and global access control policies with proxy servers, see Access control.

For details about privileges, see Administrative roles.

Monitoring and logging

You must monitor deployed services for evidence of threats and other problems. Interfaces for monitoring include the following:

- Remote monitoring facilities that clients applications can access over the network.

  These include JMX and SNMP connection handlers, and the monitor backend that is accessible over LDAP and HTTP.

- Alerts to notify administrators of significant problems or notable events over JMX or by email.
- Account status notifications to send users alerts by email, or to log error messages when an account state changes.
- Logging facilities, including local log files for access, debugging, entry change auditing, and errors. ForgeRock Common Audit event handlers support local logging and sending access event messages to a variety of remote logging and reporting systems.

For details, see Monitoring.

Operating systems

When you deploy Directory Services software, secure the host operating system. The suggestions that follow are not exhaustive. Familiarize yourself with the specific recommendations for the host operating systems you use.

System updates

Over the lifetime of a directory services deployment, the operating system might be subject to vulnerabilities. Some vulnerabilities require system upgrades, whereas others require only configuration changes. All updates require proactive planning and careful testing.

For the operating systems used in production, put a plan in place for avoiding and resolving security issues. The plan should answer the following questions:

- How does your organization become aware of system security issues early?

  This could involve following bug reports, mailing lists, forums, and other sources of information.

- How do you test security fixes, including configuration changes, patches, service packs, and system updates?

  Validate the changes first in development, then in one or more test environments, then in production in the same way you would validate other changes to the deployment.

- How do you roll out solutions for security issues?

  In some cases, fixes might involve both changes to the service, and specific actions by those who use the service.

- What must you communicate about security issues?
- How must you respond to security issues?

Software providers often do not communicate what they know about a vulnerability until they have a way to mitigate or fix the problem. Once they do communicate about security issues, the information is likely to become public knowledge quickly. Make sure that you can expedite resolution of security issues.

To resolve security issues quickly, make sure you are ready to validate any changes that must be made. When you validate a change, check the fix resolves the security issue. Validate that the system and DS software continue to function as expected in all the ways they are used.

Disable unused features

By default, operating systems include many features, accounts, and services that DS software does not require. Each optional feature, account, and service on the system brings a risk of additional vulnerabilities. To reduce the surface of attack, enable only required features, system accounts, and services. Disable or remove those that are not needed for the deployment.

The features needed to run and manage DS software securely include the following:

- A Java runtime environment, required to run DS software.
- Software to secure access to service management tools; in particular, when administrators access the system remotely.
- Software to secure access for remote transfer of software updates, backup files, and log files.
- Software to manage system-level authentication, authorization, and accounts.
- Firewall software, intrusion-detection/intrusion-prevention software.
- Software to allow auditing access to the system.
- System update software to allow updates that you have validated previously.
- If required for the deployment, system access management software such as SELinux.
- If the DS server sends email alerts locally, mail services.
- If you use SNMP with DS servers, an SNMP agent.
- Any other software that is clearly indispensable to the deployment.

Consider the minimal installation options for your operating system, and the options to turn off features.

Consider configuration options for system hardening to further limit access even to required services.

For each account used to run a necessary service, limit the access granted to the account to what is required. This reduces the risk that a vulnerability in access to one account affects multiple services across the system.

Make sure that you validate the operating system behavior every time you deploy new or changed software. When preparing the deployment and when testing changes, maintain a full operating system with DS software that is not used for any publicly available services, but only for troubleshooting problems that might stem from the system being *too* minimally configured.

Administrative access

Limit access to the system by protecting network ports and reducing access granted to administrative accounts. This further reduces the attack surface and reduces the advantage to be gained from exploiting a vulnerability.

DS servers listen for protocols listed in the following table. When protecting network ports, you must open some to remote client applications, and for replication. If administrators can connect over SSH, you can restrict access to the administrative port to the localhost only.

Protocols	Ports[1]	Active by default?	Description
LDAP	`389`, `1389`	No	Port for insecure LDAP requests and for StartTLS requests to enable a secure connection. The reserved LDAP port number is `389`. If LDAP is used, leave this port open to client applications.
LDAPS	`636`, `1636`	No	Port for secure LDAPS requests. The standard LDAPS port number is `636`. If LDAPS is used, leave this port open to client applications.
HTTP, HTTPS	`80` / `8080`, `443` / `8443`	No	Port for HTTP client requests, such as RESTful API calls. The standard HTTP port number is `80`. The standard HTTPS port number is `443`. If HTTP or HTTPS is used, leave this port open to client applications. For production deployments, use HTTPS instead of HTTP.
Administration	`4444`	Yes	Port for administrative requests, such as requests from the `dsconfig` command. Initial setup secures access to this port.
Replication	`8989`	No	Port for replication requests, using the DS-specific replication protocol. If replication is used, leave this port open to other replicas. For production deployments, secure access to this port.
JMX	`1689`	No	Port for Java Management Extensions requests (`1689`), and JMX RMI requests. The default setting for the JMX RMI port is `0`, meaning the service chooses a port of its own. This can be configured using the JMX connection handler `rmi-port` setting If used in production deployments, secure access to this port.
SNMP	`161`, `162`	No	Reserved ports are `161` for regular SNMP requests, and `162` for traps. If used in production deployments, secure access to these ports.

[1] You choose actual port numbers at setup time.

When setting up system accounts to manage directory services:

- Set up a separate DS system account for the server.
- Prevent other system accounts from accessing DS files.
- Configure the system to prevent users from logging in as the DS system account user.
- Configure the system to restrict the DS account to server management operations.

System audits

DS logs provide a record of directory service events, but they do not record system-level events. Use the auditing features of the host operating system to record access that is not recorded in DS logs.

System audit logs make it possible to uncover system-level security policy violations, such as unauthorized access to DS files. Such violations are not recorded in DS logs or monitoring information.

Also consider how to prevent or at least detect tampering. A malicious user violating security policy is likely to try to remove evidence of how security was compromised.

Java updates

Security updates are regularly released for the Java runtime environment. Plan to deploy Java security updates to systems where you run DS software:

1. If the DS server relies on any CA certificates that were added to the Java runtime environment truststore, `$JAVA_HOME/Home/lib/security/cacerts`, for the previous update, add them to the truststore for the update Java runtime environment.
2. Edit the `default.java-home` setting in the `config/java.properties` file to use the new path.

   The setting should reflect the updated Java home:

   ```
   default.java-home=/path/to/updated/java/jre
   ```

   When you set up DS servers, the path to the Java runtime environment is saved in the configuration. The server continues to use that Java version until you change the configuration.
3. Restart the DS server to use the updated Java runtime environment:

   ```
   $ stop-ds --restart
   ```

Gateway security

The DS DSML and DS REST to LDAP gateways run as web applications in containers like Apache Tomcat. Security settings depend on the container, and the gateway configuration files.

Container security settings

Security settings are covered in the documentation for supported web application containers. The documentation to use depends on the web application container.

For example, the Apache Tomcat 9 documentation includes the following:

- For instructions on setting up HTTPS, see SSL/TLS Configuration HOW-TO.
- For other security-related settings, see Security Considerations.

DSML settings

Make sure the web application container protects traffic to the gateway with HTTPS.

Review the following settings DSML gateway settings:

`ldap.port`

Use an LDAP port that supports StartTLS or LDAPS.

Using StartTLS or LDAPS is particularly important if the gateway ever sends credentials over LDAP.

`ldap.usessl`

If `ldap.usestarttls` is not used, set this to `true`.

`ldap.usestarttls`

If `ldap.usessl` is not used, set this to `true`.

`ldap.trustall`

Make sure this is set to `false`.

`ldap.truststore.path`

Set this to a truststore with the appropriate certificate(s) for remote LDAP servers.

`ldap.truststore.password`

If `ldap.truststore.path` is set, and the truststore requires a password, set this appropriately.

REST to LDAP settings

Make sure the web application container protects traffic to the gateway with HTTPS.

Review the following settings in the gateway configuration file, `config.json`:

`security/keyManager`

If the LDAP server expects client authentication for TLS, set this to access the gateway's keystore.

`security/trustManager`

Set this to a truststore with the appropriate certificate(s) for remote LDAP servers.

`ldapConnectionFactories/bind/connectionSecurity`

Use `ssl` or `startTLS`.

`ldapConnectionFactories/bind/sslCertAlias`

If the LDAP server expects client authentication for TLS, set this to access the gateway's certificate alias.

`ldapConnectionFactories/primaryLdapServers/port`

Use an LDAP port that supports StartTLS or LDAPS.

Using StartTLS or LDAPS is particularly important if the gateway ever sends credentials over LDAP.

`authorization/resolver`

Check the `endpointUrl` of the resolver to make sure that OAuth 2.0 tokens are sent over HTTPS.

For details on settings, see REST to LDAP reference.

Server security

Secure DS server installations as outlined below.

Server account

Do not run DS servers as the system superuser (root) or Windows Administrator.

Run the server under its own account, and use system capabilities to let the server account:

- Access privileged ports, such as `389`, `443`, and `636`, as required.
- Read and write to server files, and execute server commands.
- Log in to the local system.

Use system capabilities to:

- Allow administrator users to run commands as the server user.
- Allow other user to run commands, such as the `ldapsearch` command.
- Prevent other users from reading configuration files.

  On UNIX, set a umask such as `027` to prevent users in other groups from accessing server files.

  On Windows, use file access control to do the same.

Encryption

By default, only passwords are protected (hashed rather than encrypted). Encrypt other content to protect your data:

Backend files

To encrypt entries and index content, see Data encryption.

Backup files

Backup files are always encrypted. The server uses its Crypto Manager configuration to determine how to encrypt the backup files, and which HMAC algorithm to use to sign and verify backup file integrity. Backup file permissions depend on the UNIX umask or Windows file access control settings.

Changelog files

To encrypt change log files, see Encrypt External Changelog Data.

LDIF exports

When you use the `export-ldif` command, encrypt the LDIF output.

File permissions

Many DS server file permissions depend on the software distribution, not the UNIX file mode creation mask. For example, the server commands are generally executable by all users, but only the server user can read PIN code files.

The following table recommends file permission settings:

Setting	Impact
`umask` of `027`	A UNIX `umask` setting of `027` for the server account prevents members of other groups from reading files, and listing keystore contents. Members of the server user's group can still read the files. Use this setting when other processes read the files to process them independently. For example, other processes might copy backup files to a remote system, or parse the logs to look for particular patterns.
`umask` of `077`	A UNIX `umask` setting of `077` for the server account prevents members of the server user's group from reading files, and listing keystore contents. This setting can be useful when no other processes need access to server files. Other users can still run commands delivered with the server.
`log-file-permissions`	This setting applies to DS-native file-based log publishers on UNIX systems. It does not apply to Common Audit file-based log publishers. Its value is a UNIX mode string. The impact of the setting is independent of the server user's `umask` setting. The default for file-based log publishers is `600`. A value of `640` allows only the user read/write access to the logs.
Windows NTFS ACLs	On Windows systems, set folder ACLs on the NTFS volume where the server files are installed. Apply folder permissions that are inherited by all old and new files. Consider setting ACLs on at least the following folders: • The backup folder, by default `/path/to/opendj/bak`. • The configuration folder, `/path/to/opendj/config`. • The logs folder, by default `/path/to/opendj/logs`.

Disable unused features

Use the `status` command to check which connection handlers are enabled.

Disable any unused connection handlers with the `dsconfig set-connection-handler-prop --set enabled:false` command.

Log settings

By default, DS servers write log messages on error and when the server is accessed. Access logs tend to be much larger than error logs.

The default DS server log levels and rotation and retention policies facilitate troubleshooting while preventing the logs from harming performance or filling up the disk. If you change log settings for more advanced troubleshooting, reset the log configuration to conservative settings when you finish.

Password management

Make sure you keep passwords secret in production.

In configuration files

By default, DS servers keystore passwords in configuration files with `.pin` extensions. These files contain the cleartext, randomly generated passwords. Keep the PIN files readable and writable only by the user running the server.

Alternatively, configure the server to store keystore passwords in environment variables or Java properties. Key Manager Provider and Trust Manager Provider settings let you make this change.

In command arguments

DS commands supply credentials for any operations that are not anonymous. Password credentials can be supplied as arguments, such as the `--bindPassword password` option shown in the documentation.

In production, do not let the password appear in commands. Omit the `--bindPassword` option to provide the password interactively:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --baseDN uid=admin \
 "(&)" \
 userPassword

Password for user 'uid=admin':
dn: uid=admin
userPassword: {PBKDF2}10000:<hash>
```

Notice that the password appears neither in the shell history, nor in the terminal session.

When using scripts where the password cannot be supplied interactively, passwords can be read from files. For example, the `--bindPassword:file file` option takes a file that should be readable only by the user running the command.

In directory data

By default, DS servers hash passwords before storing them. DS servers support many password storage schemes.

Password policies define password storage schemes, and characteristics of valid passwords. Configure your policies to use strong password storage, and to prevent users from using weak passwords or reusing passwords.

Cryptographic keys

DS servers use cryptographic keys for encryption, signing, and securing network connections.

Deployment IDs

A *deployment ID* is a random string generated using the `dskeymgr` command. It is a deployment identifier, not a key, but it is used with a password to generate keys.

A *deployment ID password* is a secret string at least 8 characters long that you choose.

The two are a pair. You must have the deployment ID password to use the deployment ID.

Each deployment requires a *single, unique deployment ID and its password*. DS uses the pair to:

- Protect the keys to encrypt and decrypt backup files and directory data.
- Generate the TLS key pairs to protect secure connections, unless you provide your own.

Store your deployment ID and password in a safe place, and reuse them when configuring other servers in the same deployment.

The DS `setup` and `dskeymgr` commands use the pair to generate the following:

- (Required) A shared master key for the deployment.

  DS replicas share secret keys for data encryption and decryption. DS servers encrypt backend data, backup files, and passwords, and each replica must be able to decrypt data encrypted on another peer replica.

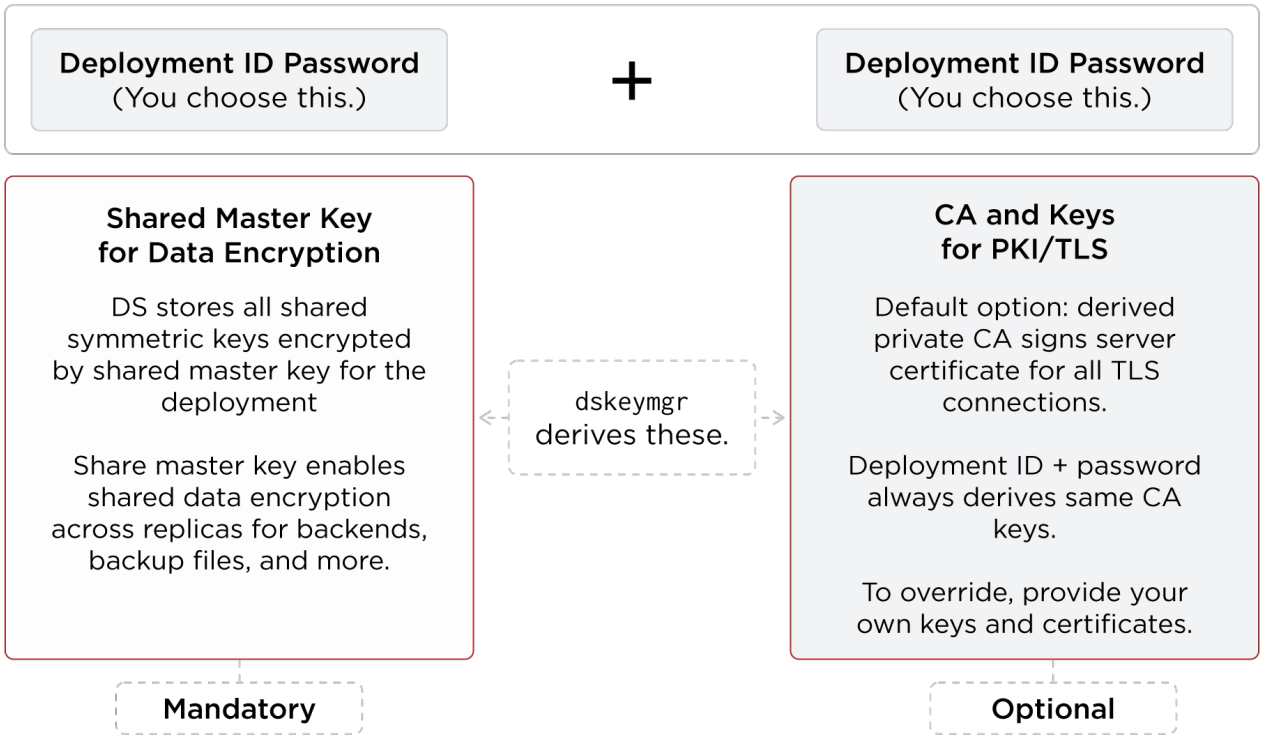
  To avoid exposing secret keys, DS servers encrypt secret keys with a shared master key. DS software uses a deployment ID and its password to derive the master key.

- (Optional) A private PKI for trusted, secure connections.

  A PKI serves to secure network connections from clients and other DS servers. The PKI is a trust network, requiring trust in the CA that signs public key certificates.

  Building a PKI can be complex. You can use self-signed certificates, but you must distribute each certificate to each server and client application. You can pay an existing CA to sign certificates, but that has a cost, and leaves control of trust with a third party. You can set up a CA or certificate management software, but that can be a significant effort and cost. As a shortcut to setting up a private CA, DS software uses deployment IDs and passwords.

  DS software uses the deployment ID and its password to generate key pairs without storing the CA private key.



- Initially, you start without a deployment ID.

  Use the `dskeymgr create-deployment-id` command to create a new deployment ID prior to installation.

  You provide the deployment ID password of your choice when generating the deployment ID, and you use both when setting up DS servers.

  Save the deployment ID wherever is convenient.

  Keep the deployment ID password secret.

- DS software uses the deployment ID and password together to generate a CA key pair.

  Every time you use the same deployment ID and password, you get the same CA key.

  Protect the deployment ID password with the same care you would use to protect the CA private key.

- DS software generates a server key pair used for secure communications in a new PKCS#12 keystore.
- DS software signs the server certificate with the CA key.
- DS software derives the shared master key for protecting secret keys, storing the master key in the PKCS#12 keystore.
- DS software writes the CA certificate to the PKCS#12 keystore.
- DS software discards the CA private key temporarily held in memory.

You can use the `dskeymgr` command with an existing deployment ID and password to add keys to keystores, or to export them in PEM file format.

This private CA alternative, using a deployment ID and password instead, is not appropriate for signing server certificates in some situations:

- External applications you do not control must be able to trust the server certificates.

  In this case, use server certificates signed by a well-known CA.

- Your deployment requires high security around CA private keys.

  If the CA private key needs to be stored in an HSM that it never leaves, you cannot achieve the same level of security with a deployment ID and password. The deployment ID and password must be provided to sign a certificate, and cannot remain secure in an HSM. Furthermore, the CA private key used to sign the certificate is present in memory during the operation.

Secret keys and key pairs

DS software uses two types of cryptographic keys:

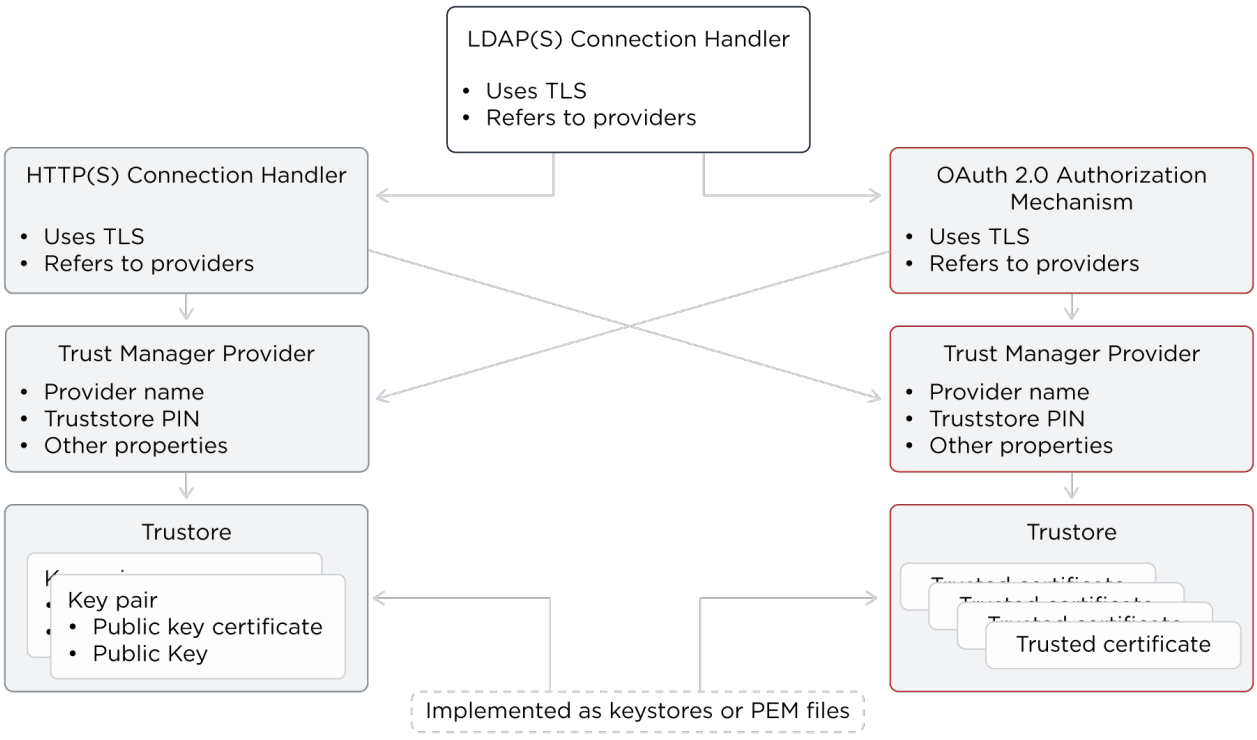
- *Symmetric keys* (also known as secret keys)
- *Asymmetric key pairs* (public/private key pairs)

	Symmetric (Secret) Keys	Asymmetric Key Pairs
Content	Single key, such as a random array of bits.	Pair of keys, one public, the other private.
Encryption	A single key serves to encrypt and to decrypt.	Data encrypted with a public key can only be decrypted with the private key, and vice versa.
Generation	Easier to generate, can be a random array of bits.	Harder to generate a matched pair of keys.
Speed	Faster.	Slower.
Distribution	Must be kept secret. Each party must have a copy. Secure channels must be established to exchange secret keys.	Public key can be shared with any party. Private key must be kept secret by owner. No secure channel is required to distribute public keys. Proving that a public key is valid and belongs to the issuer requires a trust network, such as a public key infrastructure (PKI).

	Symmetric (Secret) Keys	Asymmetric Key Pairs
Uses	Encrypting shared data. DS servers use secret keys for data confidentiality, and for encrypted backup files.	• Public key encryption: Encrypt a message with a public key; only the private key owner can decrypt the message. • Digital signatures: Sign a message with the private key; any party can verify the signature with the public key. DS software uses public/private key pairs to establish secure connections. DS servers can use public keys to authenticate clients.

Asymmetric keys

DS software stores asymmetric key pairs and trusted certificates in keystores or PEM files. In the DS server configuration, key manager providers reference Java keystores or PEM files. (Except for some ForgeRock Common Audit event handlers that manage their own keystores.) Trust manager providers also reference Java keystores or PEM files. Components that access keys reference key manager providers for their certificates and private keys, and trust manager providers for trusted certificates. This enables, but does not require, reuse:



DS servers use keystores or PEM files for server keys and trusted certificates. By default, each server stores keys in a file-based keystore, `/path/to/opendj/config/keystore`. The cleartext password is stored in a `keystore.pin` file next to the keystore file. This password serves as the password for the keystore and each private key.

The password for the keystore and for private keys must be the same. DS servers do not support using different passwords for the keystore and private keys.

By default, the `keystore` file holds these keys based on the deployment ID and password:

- The CA certificate
- The shared master key
- A key pair for secure communications

Symmetric keys

DS servers encrypt data using symmetric keys. Servers store symmetric keys, encrypted with the shared master key, with the data they encrypt. As long as servers have the same shared master key, any server can recover symmetric keys needed to decrypt data.

Symmetric keys for (deprecated) reversible password storage schemes are the exception to this rule. When you configure a reversible password storage scheme, enable the `adminRoot` backend, and configure a replication domain for `cn=admin data`.

Public key infrastructure

A public key infrastructure (PKI) is a set of procedures, roles, and policies required to enable parties to trust each other. A PKI makes it possible to trust that a public key belongs to its owner by enabling the following steps:

1. Each party in the PKI trusts one or more *certificate authorities* (CAs).

   Trusting a CA means trusting that it owns its public key, that it maintains its private key secret, and that it only signs another party's certificate according to standard operating procedures.

   The decision to trust a CA is a prerequisite for other operations, such as negotiating secure connections.

   Trusting a CA equates to storing a trusted copy of the CA's certificate. The trusted copy is used to verify CA signatures.

2. Other trusted parties get their keys certified in one of the following ways:

   - A party wanting to use public key cryptography requests a CA-signed certificate for its key pair:
     - The owner generates a key pair with a *public key* to share and a *private key* to keep secret.

       This key pair is generated for the public key *subject* (or owner).
     - The owner makes a certificate signing request to a trusted CA. The request includes the public key.
     - The CA verifies that the party making the request is indeed the party making the request. If so, the CA signs the certificate request, resulting in the signed public key certificate. The CA responds to the owner with the signed certificate as the response to the request.

       Notice that the certificate is a digital document that certifies ownership of the public key. The certificate includes the public key and other information, such as the validity period, who the subject (owner) is, and the digital signature of the *issuer* CA who signed the certificate.
   - A party registers for an account with a service provider that uses certificate-based authentication.

The service provider, acting as a CA, issues a key pair including a certificate to the account owner over a secure channel. The service provider stores a copy of the certificate with the owner's account.

3. The owner stores the signed certificate, and shares it when necessary with other parties for public key encryption and signature verification.

   It stores the private key in a safe manner and never shares it.

4. Another party wanting to trust the certificate must verify that the certificate is valid.

   Certificate verification involves checking certificate information such as the following:

   - Whether the current time is in the range of the validity dates.
   - Whether the owner's subject identifier in the certificate matches some externally verifiable attribute of the owner, such as the DNS record of the host FQDN or the owner's email address.
   - Whether the certificate has been signed by a trusted party.

     A public CA does not sign certificates with its root certificate directly. Instead, the CA issues signing certificates to itself, and uses them to sign other certificates. Trust is verified for the certificate chain, whereby the root certificate signature on the signing certificate makes it possible to trust the signing certificate. The signing certificate signature on the owner's certificate makes it possible to trust the owner's certificate.
   - Whether the party providing the certificate with the public key can prove that is has the corresponding private key.

     For example, the verifier supplies a nonce for the party to sign with the private key, and verifies the signature with the public key in the certificate.

5. Ultimately, the chain of verification must:

   - End by determining that the issuer's signature is valid and trusted, and that the party providing the certificate is authenticated.
   - Fail at some point, in which case, the signature cannot be trusted.

     This does not mean that the certificate is invalid. It does mean, however, that the party that wants to use the public key cannot be certain that it belongs to the owner, and so, cannot trust the public key.

     This can happen when the party trying to use the public key does not have a means to trust the issuer who signed the certificate. For example, it has no trusted copy of the issuer's certificate.

Trusted certificates

Secure connections between server and client applications are based on TLS. TLS depends on the use of digital certificates. By default, DS servers present their certificates when establishing a secure connection. This process fails if the client cannot trust the server certificate.

By default, DS client tools prompt you if they do not recognize the server certificate. DS servers have no one to prompt, so they refuse to accept a connection with an untrusted certificate. For ease of use, your deployment should enable secure connections without user interaction.

Automating trust is based on configuring applications to trust the CAs who sign the certificates. To achieve this, operating systems, JVMs, and web browsers ship with many trusted public CA certificates. On one hand, this prevents end users from having to understand PKI before using secure connections. On the other hand, it also introduces some risk. When public CAs are installed by default, the user must trust:

- The software distribution mechanism used to obtain the original software and subsequent updates.
- The software distributor to vet each CA and make sure the CA remains worthy of trust.
- The CAs to perform their CA duties correctly.
- The whole process to be safe from serious bugs.

Stronger security requires that you take more control. You can do this by using a private CA to distribute keys used for private communications.

Use	Private CA	Public CA	Rationale
Private connections	✓		You control both parties making the connection.
Public connections		✓	Your service publishes information over HTTPS or LDAPS to unknown end user clients. Your service connects as a client to public HTTPS or LDAPS services.
Mutual TLS	✓		When your private CA signs the client certificate, store certificate information for authentication on the client's entry in the directory.
Replication	✓		Replication messages are private to your service.
Service administration	✓		Service administration is private to your service.

Key management

Update keys

When you update keys, DS servers load them and begin using them for new connections. This section covers common rotation operations, such as renewing and replacing keys.

Renew a TLS certificate

1. Choose the appropriate method to renew an expiring certificate:
    1. If the certificate depends on a deployment ID and password, renew it with the `dskeymgr` command:

    ```
    $ dskeymgr \
     create-tls-key-pair \
     --deploymentId $DEPLOYMENT_ID \
     --deploymentIdPassword password \
     --keyStoreFile /path/to/opendj/config/keystore \
     --keyStorePassword:file /path/to/opendj/config/keystore.pin \
     --hostname localhost \
     --hostname ds.example.com \
     --subjectDn CN=DS,O=ForgeRock
    ```

    For more command options, refer to dskeymgr. The default validity for the certificate is one year.

    2. If you use a CA, renew the CA signature:
        - Create a certificate signing request.
        - Have the request signed by the CA you use.
        - Import the signed certificate from the CA reply.
    3. If you used a self-signed certificate, sign it again, and distribute the new certificate as appropriate.

    Have each server and client application that trusted the old certificate import the renewed certificate.

    For details, see Trust a server certificate.

Replace a TLS key pair

1. Choose the appropriate method to replace or rotate a key pair used for secure communications:
    1. If the certificate depends on a deployment ID and password, use the `dskeymgr` command:

    ```
    $ dskeymgr \
     create-tls-key-pair \
     --deploymentId $DEPLOYMENT_ID \
     --deploymentIdPassword password \
     --keyStoreFile /path/to/opendj/config/keystore \
     --keyStorePassword:file /path/to/opendj/config/keystore.pin \
     --hostname localhost \
     --hostname ds.example.com \
     --subjectDn CN=DS,O=ForgeRock
    ```

    The default validity for the certificate is one year.

    2. If you provided your own keys in the server keystore, do the following:
        - Generate a new key pair in the server keystore with a new alias.

- If you use a CA, get the certificate signed by the CA.
- If you use a self-signed certificate, distribute the certificate for import by all servers and clients that trust your server.
- Once the old key pair is no longer used anywhere, delete the keys using the `keytool -delete` command with the old alias.

Retire secret keys

You do not need to retire secret keys for the following, because encryption is only performed once per key:

- Backup uses a new secret key for each file.
- Confidential replication changelog backends use a new secret key for each file.

You can retire secret keys for confidential database backends:

1. Change the cipher-key-length property for the backend.

   Each time you change the setting, the server generates a new secret key. After you retire the key, DS servers will only use that key for decryption, not for encryption.

Replace deployment IDs

Follow these steps if you must:

- Renew an expiring deployment ID-based CA.

  The default validity period is 10 years.

- Replace a lost or compromised deployment ID password.

1. Generate a new deployment ID with a new password:

   ```
   $ dskeymgr \
    create-deployment-id \
    --outputFile new.deployment.id \
    --deploymentIdPassword password
   ```

   For more command options, refer to dskeymgr. The default validity for the deployment ID is 10 years.

2. On each server, add the new shared master key:

   ```
   $ dskeymgr \
    export-master-key-pair \
    --alias new-master-key \
    --deploymentId "$(<new.deployment.id)" \
    --deploymentIdPassword password \
    --keyStoreFile /path/to/opendj/config/keystore \
    --keyStorePassword:file /path/to/opendj/config/keystore.pin
   ```

Servers continue to use the existing shared master key to decrypt existing symmetric keys. Do not overwrite a shared master key that is already in use.

3. On each server that uses the deployment ID and password for PKI, add the new CA certificate:

```
$ dskeymgr \
 export-ca-cert \
 --alias new-ca-cert \
 --deploymentId "$(<new.deployment.id)" \
 --deploymentIdPassword password \
 --keyStoreFile /path/to/opendj/config/keystore \
 --keyStorePassword:file /path/to/opendj/config/keystore.pin
```

Also distribute the new CA certificate to any client applications that rely on the old CA certificate.

4. On each server that uses the deployment ID and password for PKI, renew the key pair used for secure communications.

Before completing this step, make sure you have added the new CA certificate *on all servers*. Any peer servers missing the new CA certificate will not trust the new keys:

```
$ dskeymgr \
 create-tls-key-pair \
 --deploymentId "$(<new.deployment.id)" \
 --deploymentIdPassword password \
 --keyStoreFile /path/to/opendj/config/keystore \
 --keyStorePassword:file /path/to/opendj/config/keystore.pin \
 --hostname localhost \
 --hostname ds.example.com \
 --subjectDn CN=DS,O=ForgeRock
```

For more command options, refer to dskeymgr. The default validity for the certificate is one year.

Also renew any client application key pairs that were generated using the old deployment ID and password.

5. On each server, update the master key alias to use the new key:

```
$ dsconfig \
 set-crypto-manager-prop \
 --set master-key-alias:new-master-key \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

6. Stop trusting the old CA certificate by removing all references to it.

For example, remove the old CA certificate from all client and server truststores.

7. When installing a new server after replacing deployment IDs:
    - Use the new deployment ID and password to set up the server, and do not start the server at setup time.
    - Copy the keystore and PIN from an existing server, overwriting the existing keystore and PIN.

      This adds the older shared master key to the new server's keystore.
    - Renew the local key pair used for secure communications using the new deployment ID and password.
    - Start the server.

Use new keys

Generate a key pair (wildcard certificate)

A wildcard certificate uses a `*` to replace the leading subdomain. The wildcard hostname appears in the certificate's list of subject alternative names:

1. Generate a key pair with a wildcard certificate:

```
$ dskeymgr \
 create-tls-key-pair \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --hostname localhost \
 --hostname "*.example.com" \
 --subjectDn CN=DS,CN=Example,CN=com \
 --keyStoreFile /path/to/opendj/config/keystore \
 --keyStorePassword:file /path/to/opendj/config/keystore.pin
```

   For more command options, refer to dskeymgr. The default validity for the certificate is one year.

Generate a key pair (CA-signed certificate)

1. Generate a server key pair in the existing server keystore.

   Many client applications will check that the server's DNS name in the certificate matches the server hostname.

   Find the hostname for the server in the `status` command output. When creating the key pair, set it as a `DNSName` in the certificate's `SubjectAlternativeName` list. If the server can respond on multiple FQDNs, use multiple subject alternative names.

2. Create a certificate signing request `.csr` file for the generated certificate.

3. Have the CA sign the request in the `.csr` file.

   See the instructions from your CA on how to provide the request.

   The CA returns the signed certificate, for example, in a `.crt` file.

4. If you have set up your own CA and signed the certificate, or are using a CA whose certificate is not included in the Java runtime environment, import the CA certificate into the DS keystore and truststore so that it can be trusted.

   For an example command, see Trust a CA certificate.

5. Import the signed certificate, such as the `.crt` file, from the CA reply into the keystore where you generated the key pair.
6. If you use a CA certificate that is not known to clients, such as a CA that you set up yourself rather than a well-known CA, import the CA certificate into the client application truststore. For an example command, see Trust a CA certificate.

   Otherwise, the client application cannot trust the signature on the server certificate.

Generate a key pair (self-signed certificate)

1. Generate a server key pair in the existing server keystore.

   The certificate is considered self-signed, because the issuer DN and subject DN are the same.

   Many client applications will check that the server's DNS name in the certificate matches the server hostname.

   Find the hostname for the server in the `status` command output. When creating the key pair, set it as a `DNSName` in the certificate's `SubjectAlternativeName` list.

Use an alternative keystore type

To use an alternative keystore implementation, start with a different keystore type when generating the keypair.

> **Important**
>
> When generating a key pair for TLS with the `keytool` command, set the `-keyalg` option to `EC` or `RSA` for compatibility with TLSv1.3.
> The `-keyalg DSA` option is not compatible with TLSv1.3.

1. Use one of the keystore types supported by the Java runtime environment:

Java Keystore

The basic Java keystore type is `JKS`:

```
$ keytool \
 -genkeypair \
 -keyalg EC \
 -alias new-keys \
 -ext "san=dns:ds.example.com" \
 -dname "CN=ds.example.com,O=Example Corp,C=FR" \
 -keystore /path/to/new-keystore.jks \
 -storetype JKS \
 -storepass:env KEYSTORE_PASSWORD \
 -keypass:env KEYSTORE_PASSWORD
```

This is the keystore type if you do not specify a `-storetype` option.

Java Cryptography Extension Keystore

The `JCEKS` type implements additional Java cryptography extensions and stronger protection for private keys:

```
$ keytool \
 -genkeypair \
 -keyalg EC \
 -alias new-keys \
 -ext "san=dns:ds.example.com" \
 -dname "CN=ds.example.com,O=Example Corp,C=FR" \
 -keystore /path/to/new-keystore.jceks \
 -storetype JCEKS \
 -storepass:env KEYSTORE_PASSWORD \
 -keypass:env KEYSTORE_PASSWORD
```

PKCS#11 device

A PKCS#11 device, such as an HSM, can be used as a keystore.

For details, see PKCS#11 hardware security module.

PKCS#12 Keystore

The `PKCS12` type lets you use a PKCS#12 format file. This is the default for DS servers. It is a standard format and is interoperable with other systems that do not use Java:

```
$ keytool \
 -genkeypair \
 -keyalg EC \
 -alias new-keys \
 -ext "san=dns:ds.example.com" \
 -dname "CN=ds.example.com,O=Example Corp,C=FR" \
 -keystore /path/to/new-keystore \
 -storetype PKCS12 \
 -storepass:env KEYSTORE_PASSWORD \
 -keypass:env KEYSTORE_PASSWORD
```

2. After setting up an alternate keystore type, make sure that you configure:
    - A key manager provider to open the correct keystore with the correct credentials.
    - Any components using the key manager provider to use the correct certificate alias.

Add a new keystore

These steps demonstrate adding a new PKCS#12 keystore with existing server keys. Follow these steps if you have existing keys in a PKCS#12 keystore:

1. Create a Key Manager Provider that references your keystore:

```
$ dsconfig \
 create-key-manager-provider \
 --provider-name MyKeystore \
 --type file-based \
 --set enabled:true \
 --set key-store-file:/path/to/keystore \
 --set key-store-pin:password \
 --set key-store-type:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. For each configuration object that needs to use the keys in the keystore, update the `key-manager-provider` and `ssl-cert-nickname` properties:

```
$ dsconfig \
 set-connection-handler-prop \
 --handler-name LDAPS \
 --set key-manager-provider:MyKeystore \
 --set ssl-cert-nickname:ssl-key-pair \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Many configuration objects use key manager providers. For a full list, see the page of configuration properties that start with K, and review the `key-manager-provider` properties links.

Do not update the key manager provider for the Crypto Manager. The Crypto Manager needs access to the shared master key.

Generate a test deployment ID

1. Use the `dskeymgr` command to generate a deployment ID for testing.

   This example records the deployment ID in a file:

```
$ dskeymgr \
 create-deployment-id \
 --deploymentIdPassword password \
 --outputFile test.deployment.id
```

   If you do not specify the deployment ID password or file containing the password, the tool prompts you for the password interactively.

Use this deployment ID when setting up test servers. The test servers trust each others' certificates.

Trust certificates

Trust a client certificate

These steps let the DS server trust a self-signed client application certificate:

> **Note**
>
> If you control the application, issue the client a certificate from a private CA instead.
> For an example, see Certificate-based authentication.

1. Import the self-signed client certificate.

   The following example imports the client certificate into the default truststore:

   ```
   $ keytool \
    -import \
    -trustcacerts \
    -alias myapp-cert \
    -file myapp-cert.pem \
    -keystore /path/to/opendj/config/keystore \
    -storepass:file /path/to/opendj/config/keystore.pin \
    -storetype PKCS12 \
    -noprompt

   Certificate was added to keystore
   ```

   If the truststore was provided during the setup process, specify the truststore and password.

Trust a server certificate

These steps let a client trust the DS server.

If the server certificate was signed by a well-known CA, the client may not have to configure any further trust.

To trust a certificate signed using a deployment ID and password, get the CA certificate. Either:

- Copy the server's truststore file and PIN.
- Export the CA certificate as a PEM format file, as described in PEM format keys.

For an unknown CA or a self-signed server certificate, follow these steps:

1. Export the CA certificate from its truststore or the server certificate from the server keystore.

   You can export the certificate in PEM format using the `keytool -exportcert -rfc` command.

   Some client applications can use the PEM format directly.

2. If the client uses a Java truststore, import the certificate into the client truststore.

   You can import the server certificate using the `keytool -import -trustcacerts` command.

Trust a CA certificate

These steps let the DS server trust a CA. If the CA's certificate is included with Java, you can let the server use the JVM truststore.

If the CA is not well-known, you can import the trusted CA certificate into a truststore:

1. Import the certificate as a CA certificate.

   The following example imports the CA certificate into the default truststore:

   ```
   $ keytool \
    -import \
    -trustcacerts \
    -alias my-ca-cert \
    -file ca.pem \
    -keystore /path/to/opendj/config/keystore \
    -storepass:file /path/to/opendj/config/keystore.pin \
    -storetype PKCS12 \
    -noprompt

   Certificate was added to keystore
   ```

   If the truststore was provided during the setup process, specify the truststore and password.

2. If no trust manager provider is configured to access the truststore, create one.

   For reference, see create-trust-manager-provider.

   Also, configure connection handlers to use the new trust manager provider.

Add a new truststore

These steps demonstrate adding a new PKCS#12 truststore with existing trusted certificates. Follow these steps if you have existing certificates in a PKCS#12 truststore:

1. Create a Trust Manager Provider that references your truststore:

   ```
   $ dsconfig \
    create-trust-manager-provider \
    --provider-name MyTruststore \
    --type file-based \
    --set enabled:true \
    --set trust-store-file:/path/to/truststore \
    --set trust-store-pin:password \
    --set trust-store-type:PKCS12 \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

2. For each configuration object that needs to trust the certificates in the truststore, update the `trust-manager-provider` property:

```
$ dsconfig \
 set-connection-handler-prop \
 --handler-name LDAPS \
 --set trust-manager-provider:MyTruststore \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

   Many configuration objects use trust manager providers. For a full list, see the page of configuration properties that start with T, and review the `trust-manager-provider` properties links.

Show deployment ID information

The `dskeymgr show-deployment-id` command displays key information about a given deployment ID, such as the expiration date for the derived CA certificate:

```
$ dskeymgr show-deployment-id ADaMkVIXfryp4tZN3_0V4WoB3BZc9SQ5CBVN1bkVDE6OSY5Kl7pIibg
Not before: 2022-03-21T14:55:05Z
Not after: 2032-03-18T14:55:05Z
Version: 0
Serial number: 13A3926392A5EE92226E
Provider name: SunEC
```

If you depend on an expiring deployment ID CA, see Replace deployment IDs.

PEM format keys

DS servers can read keys and trusted certificates from files that contain keys in Privacy-Enhanced Mail (PEM) format:

1. Choose or create a directory for PEM format keys.

   This example uses `/path/to/opendj/pem`:

```
$ mkdir -p /path/to/opendj/pem
```

2. Use the `dskeymgr` command to generate PEM format keys.

   - Add a master key using the deployment ID and password, even if you have your own CA and server keys:

```
$ dskeymgr \
 export-master-key-pair \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --outputFile /path/to/opendj/pem/master-key.pem
```

- Add a trusted CA certificate.

  This example exports a deployment ID CA certificate:

```
$ dskeymgr \
 export-ca-cert \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --outputFile /path/to/opendj/pem/ca-cert.pem
```

- Add server keys.

  This example exports a server key pair based on a deployment ID:

```
$ dskeymgr \
 create-tls-key-pair \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --hostname localhost \
 --hostname ds.example.com \
 --subjectDn CN=DS,O=ForgeRock \
 --outputFile /path/to/opendj/pem/ssl-key-pair.pem
```

  For more command options, refer to dskeymgr. The default validity for the certificate is one year.

3. Allow only the user running the server to read any PEM files that contain private keys.

   The keys are not encrypted, so you must protect the PEM files. For example, if the server runs with user ID `opendj`, restrict access to that user:

```
$ sudo chown opendj /path/to/opendj/pem/*.pem && \
  sudo chmod 600 /path/to/opendj/pem/master-key.pem && \
  sudo chmod 600 /path/to/opendj/pem/ssl-key-pair.pem
```

4. Configure a PEM key manager provider for the master key and server keys:

```
$ dsconfig \
 create-key-manager-provider \
 --provider-name PEM \
 --type pem \
 --set enabled:true \
 --set pem-directory:/path/to/opendj/pem \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

5. Configure a PEM trust manager provider for trusted certificates, such as CA certificates:

```
$ dsconfig \
 create-trust-manager-provider \
 --provider-name PEM \
 --type pem \
 --set enabled:true \
 --set pem-directory:/path/to/opendj/pem \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

6. Configure other components to use the new providers by setting their `key-manager-provider` and `trust-manager-provider` properties.

   When using PEM keys, the alias or `ssl-cert-nickname` property is filename of the key. In this example:

   - The master key alias is `master-key.pem`.
   - The CA certificate alias is `ca-cert.pem`.
   - The TLS keys alias is `ssl-key-pair.pem`.

Notice that the PEM key manager provider, and the PEM trust manager provider share the same `pem-directory`. This works because the key manager provider loads key pairs, and the trust manager provider loads trusted certificates. When the PEM files change, the server regularly reloads the files. For details, see Pem Key Manager Provider and Pem Trust Manager Provider.

PKCS#11 hardware security module

DS servers support key management using a PKCS#11 token store. The PKCS#11 standard defines a cryptographic token interface, a platform-independent API for storing keys in an HSM, for example.

DS servers rely on the PKCS#11 interface and Java APIs to use it. DS servers do not support vendor-specific interfaces.

Important

DS servers use an HSM only to hold asymmetric key pairs and, optionally, CA certificates. (Since the CA certificate holds the CA's *public* key, which is not secret, ForgeRock recommends storing it in a separate, file-based keystore or PEM file, not in the HSM.)
The asymmetric key pairs you can store in the HSM are the server's TLS keys, and the shared master key for the deployment.
DS servers use the shared master key to wrap symmetric (secret) keys. DS servers store secret keys in the data they encrypt. Therefore, DS servers do not use the HSM for secret keys.

Using a PKCS#11 device for storing DS keys involves:

- Storing the keys on the PKCS#11 device.

  How you generate and store keys in your HSM depends on the device. For details, see the documentation for your device.

- Creating the DS PKCS11 key manager provider to access the device.

  The DS server accesses a PKCS#11 device using a PIN. The PIN can be provided in the server configuration, in a separate file, or as the value of an environment variable or Java system property. The PIN must be stored as a cleartext value, so take care to protect it in production environments.

- Configuring DS components to use the key manager provider.

  For example, DS connection handlers and OAuth 2.0 authorization mechanisms requiring mutual TLS can reference the key manager provider in their configurations.

Prepare the HSM

The examples in this page demonstrate using a SoftHSM PKCS#11 software device for evaluation, development, and testing. (Specifically, these examples depend on SoftHSM 2.4.0. The build was prepared with `./configure --with-openssl=/usr/local/opt/openssl --disable-gost --disable-p11-kit`. Your OpenSSL installation location might be different.)

Adapt the examples for use with your HSM.

The examples also use the `sun.security.pkcs11.SunPKCS11` provider implementation with SoftHSM. If you use a different Java implementation, see the documentation for details on how to use PKCS#11 devices with your JVM:

1. Install SoftHSM, including the configuration and the `SOFTHSM2_CONF` environment variable.

   For details, see the SoftHSM documentation, using the following hints:

   - Make sure you can write tokens to SoftHSM:

```
$ cat $SOFTHSM2_CONF
# SoftHSM v2 configuration file

directories.tokendir = /path/to/softhsm/tokens/
objectstore.backend = file

# ERROR, WARNING, INFO, DEBUG
log.level = INFO

# If CKF_REMOVABLE_DEVICE flag should be set
slots.removable = false

# Enable and disable PKCS#11 mechanisms using slots.mechanisms.
slots.mechanisms = ALL

# If the library should reset the state on fork
library.reset_on_fork = false
```

- Keep track of the PINs that you enter when initializing the device:

```
$ softhsm2-util --init-token --slot 0 --label "My token 1" --pin password --so-pin password
The token has been initialized and is reassigned to slot <number>
```

The user PIN is the one the DS server needs to access the device.

The SO PIN is to reinitialize the token.

The <*number*> is the slot number to use in the PKCS#11 configuration.

2. Prepare the PKCS#11 configuration to generate key pairs.

   For example, to use the Java `keytool` command with the device, create a PKCS#11 configuration file that is used by the security provider implementation.

   Replace <*number*> in with the slot number from the previous step. If you lost track of the slot number, run `softhsm2-util --show-slots` to view it again:

```
$ cat /path/to/hsm.conf
name = SoftHSM
library = /usr/local/lib/softhsm/libsofthsm2.so
slot = <number>
attributes(generate, *, *) = {
   CKA_TOKEN = true
}
attributes(generate, CKO_CERTIFICATE, *) = {
   CKA_PRIVATE = false
}
attributes(generate, CKO_PUBLIC_KEY, *) = {
   CKA_PRIVATE = false
}
attributes(*, CKO_SECRET_KEY, *) = {
   CKA_PRIVATE = false
   CKA_EXTRACTABLE = false
}
```

Notes regarding the configuration file:

- The format is described in the Java PKCS#11 Reference Guide.
- The `library` must point to the SoftHSM `libsofthsm2.so` library.
- The `slot` must be one obtained when initializing the device.
- You can find configuration recommendations for production deployments in the platform documentation page, HSMs and ForgeRock software.

3. Verify that Java can access the device using your configuration.

   The following example shows that you can use the Java `keytool` command to list the SoftHSM keys:

```
$ keytool \
 -list \
 -keystore NONE \
 -storetype PKCS11 \
 -storepass password \
 -providerClass sun.security.pkcs11.SunPKCS11 \
 -providerArg /path/to/hsm.conf
Keystore type: PKCS11
Keystore provider: SunPKCS11-SoftHSM

Your keystore contains 0 entries
```

Store keys for securing connections

The examples that follow use SoftHSM to store the TLS keys for securing network connections (LDAPS, HTTPS, and so on). Adapt them to your HSM.

Generate a TLS key pair for securing connections

This is the server key pair for securing TLS connections. It can be specific to one DS server, and does not need to be shared.

DS presents the certificate to client applications, so get it signed by a CA they trust. *This example is for evaluation only*, and uses `localhost` as the hostname and a self-signed certificate:

1. Generate a TLS key pair.

   The following example generates a key pair with the alias `ssl-key-pair`:

   ```
   $ keytool \
    -genkeypair \
    -alias ssl-key-pair \
    -keyalg EC \
    -groupname secp521r1 \
    -ext "san=dns:localhost" \
    -dname "CN=localhost,O=Example Corp,C=FR" \
    -keystore NONE \
    -storetype PKCS11 \
    -storepass password \
    -providerClass sun.security.pkcs11.SunPKCS11 \
    -providerArg /path/to/hsm.conf
   ```

   The keystore password is the user PIN.

2. Get the public key certificate signed.

   The following example self-signs the certificate:

   ```
   $ keytool \
    -selfcert \
    -alias ssl-key-pair \
    -keystore NONE \
    -storetype PKCS11 \
    -storepass password \
    -providerClass sun.security.pkcs11.SunPKCS11 \
    -providerArg /path/to/hsm.conf
   ```

   See your HSM documentation for details on getting the certificate signed by a CA.

Create a PKCS#11 key manager provider

Repeat these steps for each DS server:

1. Make the plaintext PIN available to the DS server.

   With SoftHSM, this is the user PIN set when initializing the slot where you stored the keys, `softhsm2-util --pin password`.

   The following example sets the PIN in an environment variable:

   ```
   $ export SOFTHSM_USER_PIN=password
   ```

   The value must be accessible every time the DS server starts.

2. Prepare the Java environment for the DS server to use the HSM configuration for PKCS#11.

    1. Add the HSM configuration for the security provider in the Java security configuration to use for DS.

        In this example, the `$JAVA_HOME/conf/security/java.security` file for the JVM defines a security provider for PKCS#11, `security.provider.13=SunPKCS11`. You must update or override that value for `security.provider.13` to add the HSM configuration file as a parameter.

        Create a one-line, local `java.security` file the overrides the value:

        ```
        $ cat /path/to/java.security
        security.provider.13=SunPKCS11 /path/to/hsm.conf
        ```

    2. Use the Java security configuration when starting DS.

        For example, add the configuration to the `start-ds.java-args` for the server:

        ```
        # Edit config/java.properties to set java.security.properties on startup:
        $ grep java.security /path/to/opendj/config/java.properties
        start-ds.java-args=-server -Djava.security.properties=/path/to/java.security

        # Restart the server so the changes take effect:
        $ stop-ds --restart
        ```

3. Create the PKCS#11 key manager provider configuration.

    The following example creates a provider for SoftHSM with the PIN from the user environment:

    ```
    $ dsconfig \
     create-key-manager-provider \
     --provider-name SoftHSM \
     --type pkcs11 \
     --set enabled:true \
     --set key-store-pin:"&{softhsm.user.pin}" \
     --hostname localhost \
     --port 4444 \
     --bindDN uid=admin \
     --bindPassword password \
     --usePkcs12TrustStore /path/to/opendj/config/keystore \
     --trustStorePassword:file /path/to/opendj/config/keystore.pin \
     --no-prompt
    ```

    DS key manager providers also support storing the PIN in the configuration, in a file, or in a Java property.

Use the PKCS#11 key manager provider

1. Set a connection handler or authorization mechanism to use the PKCS#11 key manager provider.

    The following example configures the LDAPS connection handler to use the SoftHSM provider:

```
$ dsconfig \
 set-connection-handler-prop \
 --handler-name LDAPS \
 --set listen-port:1636 \
 --set enabled:true \
 --set use-ssl:true \
 --set key-manager-provider:SoftHSM \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Verify that the secure connection negotiation works with the HSM configured:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSSL \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 cn

User DN  : CN=localhost, O=Example Corp, C=FR
Issuer   : CN=localhost, O=Example Corp, C=FR
Validity : <validity-period>

Do you trust this server certificate?

  1) No
  2) Yes, for this session only
  3) Yes, also add it to a truststore
  4) View certificate details

Enter choice: [1]: 2

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
```

Store the shared master key

Before you start

Set up DS servers *without the* `--start` *option*, and *without any data or setup profiles that create base entries or generate data*, as if you were following the instructions in Install DS for custom cases.

Do not start the DS server until you have configured it to use the shared master key in your HSM, as described below.

The `setup` command requires a deployment ID and password. It generates a shared master key that you will discard.

However, if you start DS with any data at all, it protects secret keys for encryption with that shared master key, and you will need to continue to use that shared master key.

This procedure is therefore not intended for deployments where you are upgrading servers with encrypted data.

Important

The shared master key is an asymmetric key pair with a self-signed certificate. Only DS uses the shared master key, and only for encrypting and decrypting secret keys. DS does not check the hostname or the validity period of the shared master key, so neither matters.
When you store the shared master key in an HSM:

- *The HSM must share the identical master key with all DS servers in the deployment*.
  Each server encrypts and decrypts shared secret keys with the shared master key. Without access to the same shared master key, DS servers cannot read each others' encrypted data.
  The (networked) HSM must replicate the same shared master key pair everywhere for each HSM instance that DS servers access.
- *DS servers must not rely on* `cn=admin data`.
  All the secret keys must use the DS and later security model exclusively. The procedures describing how to use an HSM for the shared master key do not apply if you have upgraded from DS 6.5 or earlier, and used encrypted data.

1. Generate a shared master key pair in the HSM, using `RSA` as the key algorithm.

   The following example generates a key pair with the alias `master-key`:

   ```
   $ keytool \
    -genkeypair \
    -alias master-key \
    -keyalg RSA \
    -keysize 3072 \
    -dname "CN=Master key, O=Example.com" \
    -keystore NONE \
    -storetype PKCS11 \
    -storepass password \
    -providerClass sun.security.pkcs11.SunPKCS11 \
    -providerArg /path/to/hsm.conf
   ```

   The keystore password is the user PIN.

2. For each DS server, create the PKCS#11 key manager provider to access the HSM, as described in Create a PKCS#11 key manager provider, but *without starting the server*.

   For the `dsconfig create-key-manager-provider` command, replace the connection parameters with `--offline`.

3. For each DS server, update the Crypto Manager to reference the PKCS#11 key manager provider.

   For example, set the key manager provider to `SoftHSM` using the default alias for the key pair, `master-key`:

```
$ dsconfig \
 set-crypto-manager-prop \
 --set key-manager-provider:SoftHSM \
 --offline \
 --no-prompt
```

4. For each DS server, apply Setup profiles with the `setup-profile` command, and complete any necessary preliminary configuration.

   When you run DS commands, such as `setup-profile` and many others, that involve encrypting or decrypting data, make sure the Crypto Manager can access the configuration for the HSM. For example, add the Java argument `-Djava.security.properties=/path/to/java.security`, as you did for `start-ds`.

5. Start each DS server.

Secure connections

Securing connections depends on PKI and asymmetric, public/private key pairs. For details, see Cryptographic keys.

About secure connections

Incoming connections are from clients to the directory. To match port numbers with protocols, see Administrative Access.

Outgoing connections are from the directory to another service.

Recommendations for incoming connections

Protocol	Recommendations
Administration	DS servers use an Administration Connector for connections from administration tools. Leave the Administration Connector configured to use SSL/TLS unless you are certain the connections are already secured by some other means.
DSML	DSML support is available through the DS DSML gateway. Use HTTPS to protect client connections.
HTTP	HTTP connections that are not protected by SSL/TLS use cleartext messages. When you permit insecure connections, you cannot prevent client applications from sending sensitive data. For example, a client could send unprotected credentials in an HTTP Authorization header. Even if the server were to reject the request, the credentials would already be leaked to any eavesdroppers. HTTP could be allowed instead of HTTPS with anonymous connections if only public information is exposed, and no client applications send credentials or other sensitive information. Configure the HTTP connection handler to use only the default HTTP Anonymous authorization mechanism.

Protocol	Recommendations
HTTPS	Prefer HTTPS for secure connections over HTTP. When using an HTTP connection handler, use HTTPS to protect client connections. Some client applications require a higher level of trust, such as clients with additional privileges or access. Client application deployers might find it easier to manage public keys as credentials than to manage user name/password credentials. Client applications can use SSL client authentication. When using DS REST to LDAP gateway, use HTTPS to protect client connections.
JMX	Secure JMX access with the SSL/TLS-related properties, such as `use-ssl` and others.
LDAP	LDAP connections that are not protected by SSL/TLS use cleartext messages. When you permit insecure connections, you cannot prevent client applications from sending sensitive data. For example, a client could send unprotected credentials in an LDAP simple bind request. Even if the server were to reject the request, the credentials would already be leaked to any eavesdroppers. If all the LDAP applications are under your control, make sure that the only "insecure" requests are anonymous binds, SASL binds, or StartTLS requests.
LDAPS	Prefer LDAPS for secure connections, or make sure that applications use StartTLS after establishing an insecure LDAP connection and before performing other operations. Some client applications require a higher level of trust, such as clients with additional privileges or access. Client application deployers might find it easier to manage public keys as credentials than to manage user name/password credentials. Client applications can use SSL client authentication. See Certificate-based authentication.
Replication	Replication is required in all but a few deployments. If *any of the following are true*, replication is required: • Client applications require highly available access to critical services, such as authentication and updates. • Client applications have specific quality of service requirements. • Client applications use the directory service to share common data. • Directory service downtime, either planned or unplanned, can lead to lost organizational or business opportunities. • Backup operations must be performed while the service is online. • Update and upgrade operations must be performed while the service is online. • Load sometimes exceeds the service that a single server can provide. • Global directory services must be available at more than one location. Configure replication to use secure connections.
SNMP	Secure SNMP access with settings for `security-level` and related properties.

Protocol	Recommendations

SSH	DS administration tools can connect securely.
	If the firewall is configured to prevent remote access to the administration connector port, then use a secure third-party tool to access the system remotely. A recommended choice for UNIX and Linux systems is Secure Shell (SSH).
	The user account for running directory services should not be the same user account for connecting remotely. Instead, connect as a another user who can then assume the role of the directory services account. The following example demonstrates this approach:
	`# Log in to ds.example.com:`
	`me@my-laptop $ ssh user@ds.example.com`
	
	`user@ds.example.com's password:`
	
	`# Logged in to ds.example.com as user.`
	`Last login: ... from ...`
	
	`# Run dsconfig interactively as opendj:`
	`user@ds.example.com $ sudo -i -u opendj dsconfig`
	Secure Copy (SCP) uses SSH to transfer files securely. SCP is an appropriate protocol for copying backup data, for example.

Recommendations for outgoing connections

Client	Recommendations
Common Audit event handlers	Configure ForgeRock Common Audit event handlers to use HTTPS or TLS when connecting to external log services.
DSML gateway	The DS DSML gateway connects to remote LDAP directory servers. Use LDAP and StartTLS or LDAPS to protect the connections.
OAuth 2.0-based HTTP authorization mechanisms	HTTP authorization can be based on OAuth 2.0, where DS servers act as resource servers, and make requests to resolve OAuth 2.0 tokens. Use HTTPS to protect the connections to OAuth 2.0 authorization servers.
Pass-Through Authentication	When DS servers use pass-through authentication, they connect to remote LDAP directory servers for authentication. Use LDAP with StartTLS or LDAPS to protect the connections.
Proxy Requests	A DS directory proxy server can connect to remote directory servers with a bind DN and bind password. Use LDAP with StartTLS or LDAPS to protect requests to remote directory servers.
Replication	Configure replication to use secure connections.
REST to LDAP gateway	The DS REST to LDAP gateway connects to remote LDAP directory servers. Use LDAP with StartTLS or LDAPS to protect the connections.

Client	Recommendations
SMTP account notification and alert handlers	DS servers can send email account notifications and alerts. Use TLS to protect the connections.

Require HTTPS

You can configure an HTTPS port, and no HTTP port, at setup time, or later, as described below. For details on configuring the DS gateway applications to use HTTPS, see your web container documentation.

Set the HTTPS port

At setup time use the `--httpsPort` option.

Later, follow these steps to set up an HTTPS port:

1. Create an HTTPS connection handler.

   The following example sets the port to `8443` and uses the default server certificate:

   ```
   $ dsconfig \
    create-connection-handler \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --handler-name HTTPS \
    --type http \
    --set enabled:true \
    --set listen-port:8443 \
    --set use-ssl:true \
    --set key-manager-provider:PKCS12 \
    --set trust-manager-provider:"JVM Trust Manager" \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

   If the key manager provider has multiple key pairs that DS could use for TLS, where the secret key was generated with the same key algorithm, such as `EC` or `RSA`, you can specify which key pairs to use with the `--set ssl-cert-nickname:server-cert` option. The *server-cert* is the certificate alias of the key pair. This option is not necessary if there is only one server key pair, or if each secret key was generated with a different key algorithm.

2. Enable the HTTP access log.

   1. The following command enables JSON-based HTTP access logging:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

2. The following command enables HTTP access logging:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

3. If the deployment requires SSL client authentication, set the properties `ssl-client-auth-policy` and `trust-manager-provider` appropriately.

4. After you set up an HTTPS port, enable an HTTP endpoint.

   For details, see Configure HTTP user APIs, or Use administrative APIs.

Require LDAPS

You can configure an LDAPS port, and no LDAP port, at setup time, or later, as described below.

Set the LDAPS port

At setup time, use the `--ldapsPort` option.

Later, follow these steps to set up an LDAPS port:

1. Configure the server to activate LDAPS access:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAPS \
 --set enabled:true \
 --set listen-port:1636 \
 --set use-ssl:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. If the deployment requires SSL client authentication, set the `ssl-client-auth-policy` and `trust-manager-provider` properties appropriately.

Disable LDAP

Configure the server to disable insecure LDAP access:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAP \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Message-level security

Server protocols like LDAP, HTTP, JMX, and replication rely on transport layer security to protect connections. Configure connection handlers for TLS or StartTLS, and use access control to enforce secure communications.

- For directory servers, see `ssf` in ACI subjects. Set the security strength factor to achieve a balance; for example, with 128 or 256. If set too low, the server and client can negotiate a connection that is not secure. If set too high, some clients might not be able to connect.
- For directory proxy servers, use a `connection-minimum-ssf` setting that enforces use of transport layer security, such as 128 or 256.

When negotiating connection security, the server and client must use a common security protocol and cipher suite. To update the security protocols and cipher suites, see TLS settings.

Transport layer security

When a client and server set up an HTTPS or LDAPS connection, they use a protocol to establish the security. They then encapsulate HTTP or LDAP messages in the secure protocol. The process depends on digital certificates. The process uses client certificates differently for mutual authentication and for SASL EXTERNAL binds.

A connection that uses TLS, a protocol based on the SSL protocol, is a connection that is private and reliable. Communications on the connection are kept private by being encrypted with a symmetric key for the session that only the client and server know. Communications are reliable because their integrity is checked with a message authentication code (MAC).

The TLS protocol is independent of the application protocol. TLS encapsulates application level protocols like HTTP and LDAP. The client and server negotiate the secure connection before any messages are sent using the application protocol.

When a client and server set up a secure connection, they negotiate a session with the TLS handshake protocol. During the handshake the server and client use asymmetric keys, their public key certificates and associated private keys, to authenticate (prove their identities).

The default configuration for DS servers sends the server certificate during the handshake. The client is not required to send its certificate. This lets the client authenticate the server when negotiating security. It does not let the server authenticate the client at this stage. This avoids requiring clients, such as browsers, to manage keys and certificate signatures.

For client applications that are part of the software infrastructure, rather than end user applications, managing keys and certificate signatures can be a better choice than managing passwords. Such clients present their certificates during the handshake, letting the server authenticate the client. When both the server and client present certificates during the handshake, this is known as *mutual authentication*.

In TLS v1.2, for example, a successful initial handshake has the client and server do the following:

1. Exchange supported algorithms and random values.
2. Exchange cryptographic information to agree on an initial secret.
3. Exchange certificates and cryptographic information to permit authentication.
4. Generate a symmetric key for the sessions with the initial secret and the random values exchanged.
5. Set the security parameters for the session.
6. Verify that each party uses the same security parameters, and that the handshake was not tampered with.

The handshake completes before any HTTP or LDAP messages are sent over the connection. HTTP authentications and LDAP binds happen after the secure connection has been established.

DS support for TLS relies on the Java implementation. DS support for LDAP authentication is part of the DS server. This is an important distinction:

- When a client application presents its certificate for TLS mutual authentication, the JVM checks the certificate independently of the client application's directory entry.

  When securing the transport layer with mutual authentication, the client certificate must be trusted.

- When the client application binds to the DS server with its certificate, the server checks that the certificate presented matches the certificate stored in the client's directory entry. A certificate mapper finds the directory entry based on the client certificate.

Client certificate validation

A first step in establishing a secure connection involves validating the certificate presented by the other party. Part of this is trusting the certificate. The certificate identifies the client or server and the signing certificate. The validating party checks that the other party corresponds to the one identified by the certificate, and checks that the signature can be trusted. If the signature is valid, and the signing certificate is trusted, then the certificate can be trusted.

Certificates can be revoked after they are signed. Therefore, the validation process can involve checking whether the certificate is still valid. This can be done with the Online Certificate Status Protocol (OCSP) or Certificate Revocation Lists (CRLs). OCSP is a newer solution that provides an online service to handle the revocation check for a specific certificate. CRLs are potentially large lists of user certificates that are no longer valid or that are on hold. A CRL is signed by the CA. The validating party obtains the CRL and checks that the certificate being validated is not listed. For a brief comparison, see OCSP: Comparison to CRLs. A certificate can include links to contact the OCSP responder or to the CRL distribution point. The validating party can use these links to check whether the certificate is still valid.

In both cases, the CA who signed the certificate acts as the OCSP responder or publishes the CRLs. When establishing a secure connection with a client application, the server relies on the CA for OCSP and CRLs. This is the case even when the DS server is the repository for the CRLs.

DS directory services are logical repositories for certificates and CRLs. For example, DS servers can store CRLs in a `certificateRevocationList` attribute:

```
dn: cn=My CA,dc=example,dc=com
objectClass: top
objectClass: applicationProcess
objectClass: certificationAuthority
cn: My CA
authorityRevocationList;binary: Base64-encoded ARL
cACertificate;binary:: Base64-encoded CA certificate
certificateRevocationList;binary:: Base64-encoded CRL
```

Replicate the CRL for high availability. (The ARL in the entry is like a CRL, but for CA certificates.)

Despite being a repository for CRLs, the DS server does check client certificates with its CRLs directly. Instead, when negotiating a secure connection, the server depends on the JVM security configuration. The JVM configuration governs whether validation uses OCSP, CRLs, or both. The JVM relies on system properties that define whether to use the CRL distribution points defined in certificates, and how to handle OCSP requests. These system properties can be set system-wide in `$JAVA_HOME/lib/security/java.security`. The JVM handles revocation checking without the DS server's involvement. For details, see *Support for the CRL Distribution Points Extension*, and *Appendix C: On-Line Certificate Status Protocol (OCSP) Support* in the Java PKI Programmer's Guide.

After a connection is negotiated, the client can bind with its certificate using SASL EXTERNAL authentication. For details, see Certificate-based authentication.

OCSP and obtaining CRLs depend on network access to the CA. If DS servers or the DSML or REST to LDAP gateways run on a network where the CA is not accessible, and the deployment requires OSCP or checking CRLs for client application certificates, then you must provide some alternative means to handle OCSP or CRL requests. Configure the JVM to use a locally available OCSP responder, for example. If the solution depends on CRLs, regularly update the CRLs in the directory with downloaded copies of the CA CRLs.

Restrict client access

Use a server's global configuration properties, to restrict how clients access the server. These global configuration settings are per server, and are not replicated:

bind-with-dn-requires-password

Whether the server rejects simple bind requests containing a DN but no password.

Default: `true`

To change this setting use the following command:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set bind-with-dn-requires-password:false \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

max-allowed-client-connections

Restricts the number of concurrent client connections to this server.

Default: 0, meaning no limit is set.

To set a limit of 64K use the following command:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set max-allowed-client-connections:65536 \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

allowed-client

Restrict which clients can connect to the server.

restricted-client

Restrict the number of concurrent connections per client.

unauthenticated-requests-policy

This setting can take the following values:

`reject`

Reject requests (other than bind or StartTLS requests) received from a client:

- Who has not yet authenticated.
- Whose last authentication was unsuccessful.
- Whose last authentication attempt used anonymous authentication.

`allow-discovery`

Like `reject`, but allows unauthenticated base object searches of the root DSE.

This setting supports applications that read the root DSE to discover server capabilities, and applications that target the root DSE for keep-alive heartbeats.

`allow` *(default)*

Allow all unauthenticated requests, subject to privileges and access control.

Although this is the default setting, all setup profiles except `ds-evaluation` use `allow-discovery`.

To allow anonymous binds, use the following command:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set unauthenticated-requests-policy:allow \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

return-bind-error-messages

Does not restrict access, but prevents a server from returning extra information about why a bind failed, as that information could be used by an attacker. Instead, the information is written to the server errors log.

Default: `false`.

To have the server return additional information about why a bind failed, use the following command:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set return-bind-error-messages:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

TLS settings

To negotiate a secure connection, the server and client must agree on a common protocol and cipher suite. Otherwise, they fail to establish a secure connection.

By default, DS servers use only security protocols and cipher suites considered secure at the time of release. DS servers do, however, support all the security protocols and cipher suites provided by the JVM. For details, see the documentation for the JVM, such as the JDK Providers Documentation for the *The SunJSSE Provider*.

Researchers continue to find vulnerabilities in protocols and cipher suites. If a server supports vulnerable protocols or cipher suites, clients can use them. Attackers can then exploit the vulnerabilities. You might therefore need to restrict the list of accepted protocols and cipher suites at any time.

List protocols and cipher suites

1. To list the protocols and cipher suites that DS servers accept, read the root DSE attributes, `supportedTLSProtocols` and `supportedTLSCiphers`:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN "" \
 --searchScope base \
 "(objectclass=*)" \
 supportedTLSCiphers supportedTLSProtocols
```

A `supportedTLSCiphers` name, such as `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384` for use with TLSv1.2, identifies the key attributes of the cipher suite:

`TLS`

Specifies the protocol, in this case TLS.

`ECDHE_RSA`

Specifies the key exchange algorithm used to determine how the client and server authenticate during the handshake phase.

The algorithm in this example uses an elliptic curve variant of the Diffie-Hellman key exchange. In `ECDHE_RSA`, the server signs the ephemeral ECDH public key in the `ServerKeyExchange` message with its RSA private key.

Ideally the server certificate would have a `KeyUsage` extension with only the `digitalSignature` bit set to prevent it being used for encryption.

`WITH_AES_256_GCM`

Specifies the bulk encryption algorithm, including the key size or initialization vectors.

This example specifies the Advanced Encryption Standard (AES) with 256-bit key size and Galois/Counter Mode (GCM) block cipher mode.

SHA384

Specifies the message authentication code algorithm used to create the message digest, which is a cryptographic hash of each block in the message stream.

In this example, the SHA-2 hash function, SHA-384, is used.

A `supportedTLSProtocols` name identifies the protocol and version, such as `TLSv1.2` or `TLSv1.3`.

Cipher suites compatible with TLSv1.3 do not include the key exchange algorithm. In TLSv1.3, the signature algorithm and key exchange are negotiated separately.

Restrict protocols and cipher suites

You can limit the protocols and cipher suites that DS servers accept by setting the properties, `ssl-protocol` and `ssl-cipher-suite` on the appropriate components.

This following settings derive from the server-side TLS recommendations published by the Mozilla Operations Security team at the time of this writing. Recommendations evolve. Make sure you use current recommendations when configuring security settings:

1. For each cipher suite key algorithm to support, create a key pair using the supported key algorithm.

   The following example adds a key pair to the default PKCS#12 keystore:

   ```
   $ keytool \
    -genkeypair \
    -alias ssl-key-pair-ec \
    -keyalg EC \
    -ext "san=dns:ds.example.com" \
    -dname "CN=ds.example.com,O=Example Corp,C=FR" \
    -keystore /path/to/opendj/config/keystore \
    -storetype PKCS12 \
    -storepass:file /path/to/opendj/config/keystore.pin \
    -keypass:file /path/to/opendj/config/keystore.pin
   ```

2. On the components you use, explicitly set the supported protocols and cipher suites.

   The following example adjusts settings for the LDAP and LDAPS connection handlers:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAP \
 --add ssl-protocol:TLSv1.3 \
 --add ssl-cipher-suite:TLS_AES_128_GCM_SHA256 \
 --add ssl-cipher-suite:TLS_AES_256_GCM_SHA384 \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin

$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAPS \
 --add ssl-protocol:TLSv1.3 \
 --add ssl-cipher-suite:TLS_AES_128_GCM_SHA256 \
 --add ssl-cipher-suite:TLS_AES_256_GCM_SHA384 \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

3. Use the appropriate settings for your connection handlers:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAP \
 --set enabled:true \
 --set listen-port:1389 \
 --set allow-start-tls:true \
 --set key-manager-provider:PKCS12 \
 --set trust-manager-provider:PKCS12 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAPS \
 --set listen-port:1636 \
 --set enabled:true \
 --set use-ssl:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Restrict protocols For command-line tools

You can specify which protocol versions to allow when command-line tools negotiate secure connections with LDAP servers.

The command-line tools depend on a system property, `org.opends.ldaps.protocols`. This property takes a comma-separated list of protocols. The default is constructed from the list of all protocols the JVM supports, removing protocol names starting with `SSL`. For example, if support is enabled in the JVM for versions 1.0, 1.1, and 1.2 of the TLS protocol, then the default is `"TLSv1,TLSv1.1,TLSv1.2"`.

1. Restrict the protocols to use by setting the property in one of the following ways:

    1. Set the property by editing the `java-args` for the command in `config/java.properties`.

        For example, to restrict the protocol to TLS v1.2 when the `status` command negotiates a secure administrative connection, edit the corresponding line in `config/java.properties`:

        ```
        status.java-args=-Xms8m -client -Dorg.opends.ldaps.protocols=TLSv1.2
        ```

    2. Set the property at runtime when running the command.

        The following example restricts the protocol to TLS v1.2 when the `status` command negotiates a secure administrative connection:

```
$ export OPENDJ_JAVA_ARGS="-Dorg.opends.ldaps.protocols=TLSv1.2"

$ status \
 --bindDn uid=admin \
 --bindPassword password \
 --hostname localhost \
 --port 4444 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

Authentication mechanisms

Authentication is the process of verifying who is requesting access to a resource. The user or application making the request presents credentials, making it possible to prove that the requester is who they claim to be. The goal is to authorize access to directory resources depending on the confirmed identity of the user or application making the request.

LDAP is a stateful protocol, where the client sets up and maintains a connection with the server, potentially performing many operations or long-lived operations before disconnecting from the LDAP server. One of the LDAP operations, a *bind*, authenticates the client to the server. A bind is the first operation that a client performs after setting up a connection. Clients can bind again on the same connection to reauthenticate.

At the transport layer, DS servers support SSL and TLS protocols with mutual client authentication. This level of authentication is useful to properly secure connections. The authentication at this level is handled by the underlying JVM. The client identity verified at this level is not available to the DS server for the purpose of fine-grained authorization. For fine-grained authorization, you need LDAP authentication.

DS servers support multiple authentication mechanisms for LDAP operations:

Simple bind (name/password) authentication

The client application presents a bind DN/password combination. The server checks that the password matches the password on the entry with the specified bind DN.

This mechanism involves sending the credentials over the network. Always use secure connections at the transport layer when you expect simple LDAP binds. You can configure either or both LDAPS and StartTLS, depending on what the client applications support.

For additional information, see Simple binds.

Anonymous authentication

Simple bind authentication without credentials.

Anonymous authentication lets the server make authorization decisions when the client identity is unknown.

Allow anonymous authentication for publicly readable resources, such as root DSE attributes. Configure access control to let anonymous and other users read them.

For additional information, see Anonymous access.

SASL authentication

Simple Authentication and Security Layer (SASL) is a framework, rather than a single method. DS servers provide handlers for a number of SASL mechanisms, including strong authentication choices like the External SASL mechanism handler for certificate-based authentication, and the GSSAPI SASL mechanism handler for use with Kerberos v5 systems.

Certificate-based authentication is well-suited for applications where distributing keys is easier than protecting passwords. For additional information, see Certificate-based authentication.

GSSAPI-based authentication is useful for interoperation with Kerberos. For additional information, see Authenticate with Kerberos.

Authentication with proxied authorization

The client application binds with its credentials, and uses proxied authorization to perform operations as another user.

Client applications can use another means to authenticate the user before requesting proxied authorization. For details, see Proxied authorization.

Pass-through authentication to another LDAP directory

The client application binds with its credentials, and another LDAP directory service handles the authentication. This is known as *pass-through authentication*.

Pass-through authentication is useful when credentials are stored in a remote directory service, and the DS directory service stores part of the user profile. For details, see Pass-through authentication.

DS servers and DS REST to LDAP gateway support multiple HTTP authentication mechanisms.

The identity from the HTTP request is mapped to an LDAP account for use in authorization decisions, so the mechanisms are known as authorization mechanisms. Their configuration is described in Configure HTTP Authorization. The following authorization mechanisms are available:

HTTP Basic authorization

The client application sends an HTTP request that uses HTTP Basic authentication.

The client application can alternatively send an HTTP request with username and password headers.

You configure DS software to map the HTTP username to an LDAP DN. The result is like a simple name/password bind.

This mechanism involves sending the credentials over the network. Always use secure connections at the transport layer when you allow HTTP Basic.

Anonymous authorization

The client application sends an HTTP request without authenticating.

You configure DS software either to map the HTTP request to anonymous authentication at the LDAP level, or to bind at the LDAP level as a specific user.

OAuth 2.0 authorization

The client application sends an HTTP request bearing an OAuth 2.0 access token that includes at least one scope whose value makes it possible to determine the user identity.

You configure DS software to resolve the access token, and to map the user identity from the scope to an LDAP account.

This mechanism involves sending the credentials over the network. Always use secure connections at the transport layer for OAuth 2.0 authorization.

Anonymous access

In LDAP, an *anonymous bind* is a bind operation using simple authentication with an empty DN and an empty password. DS servers apply access controls (ACIs) to let anonymous clients access only fully public information. Examples include information about the directory server in the root DSE, and LDAP schema definitions.

By default, DS servers disable anonymous access to directory data. ACIs take a user DN subject, `ldap:///anyone`, that matches anonymous and authenticated users.

When a client accesses the directory over HTTP, anonymous operations can be mapped either to a specific user identity or to an anonymous user (default). This is set using the HTTP Anonymous authorization mechanism for the HTTP endpoint.

Simple binds

In LDAP, a *simple bind* is name/password authentication. The client application presents a bind DN/password combination. The DS server checks that the password matches the password on the entry with the specified bind DN.

The LDAP connection transport layer must be secure for a simple bind. Otherwise, eavesdroppers can read the credentials.

DS servers provide two alternatives to secure the connection for a simple bind, both of which depend on certificates and public key infrastructure:

LDAPS

To support LDAP over SSL and TLS, DS servers have a separate connection handler that listens for traffic on a port dedicated to secure connections.

LDAP with StartTLS

DS servers support using the StartTLS extended operation on an insecure LDAP port. With StartTLS, the client initiates the connection on the LDAP port, and then negotiates a secure connection.

Pass-through authentication

Pass-Through Authentication (PTA), a remote LDAP service to determine the response to an authentication request. A typical use case for PTA involves passing authentication through to Active Directory for users coming from Microsoft Windows systems.

About PTA

You use PTA when the credentials for authenticating are stored in a remote directory service. In effect, the DS server redirects the bind operation against a remote LDAP server.

The method a server uses to redirect the bind depends on the mapping from the user entry in the DS server to the corresponding user entry in the remote directory. DS servers provide you several choices to set up the mapping:

- When both the local entry in the DS server and the remote entry in the other server have the same DN, you do not have to set up the mapping. By default, the DS server redirects the bind with the original DN and password from the client application.
- When the local entry in the DS server has an attribute holding the DN of the remote entry, you can specify which attribute holds the DN. The DS server redirects the bind on the remote server using the DN value.
- When you cannot get the remote bind DN directly, you need an attribute and value on the DS entry that corresponds to an identical attribute and value on the remote server. In this case, you also need the bind credentials for a user who can search for the entry on the remote server. The DS server performs a search for the entry using the matching attribute and value, and then redirects the bind with the DN from the remote entry.

You configure PTA as an authentication policy that you associate with a user's entry in the same way that you associate a password policy with a user's entry. Either a user has an authentication policy for PTA, or the user has a local password policy.

Set up PTA

When setting up PTA, you need to know define:

- Which remote server(s) to redirect binds to
- How you map user entries in the DS server to user entries in the remote directory

Secure connections

When performing PTA, you protect communications between the DS server and the authenticating server. When you test secure connections with a CA that is not well-known, make sure the authentication server's certificate is trusted by the DS server.

If the authentication server's CA is well-known or already trusted by the DS server, you can skip these steps:

1. Export the CA or server certificate from the authentication server.

   How you perform this step depends on the authentication directory server.

2. Record the hostname used in the certificate.

   You use the hostname when configuring the secure connection.

3. Import the trusted authentication server certificate into the DS server's keystore.

Configure a PTA policy

Configure authentication policies with the `dsconfig` command. Notice that authentication policies are part of the server configuration, and therefore not replicated:

1. Set up an authentication policy for PTA to the authentication server:

```
$ dsconfig \
 create-password-policy \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "PTA Policy" \
 --type ldap-pass-through \
 --set primary-remote-ldap-server:remote-server.example.com:1636 \
 --set mapped-attribute:uid \
 --set mapped-search-base-dn:"dc=example,dc=com" \
 --set mapping-policy:mapped-search \
 --set use-ssl:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The policy shown here maps identities with this password policy to identities under `dc=example,dc=com` on the authentication server. Users must have the same `uid` values on both servers. This policy uses LDAPS between the DS server and the authentication server.

2. Check that your policy has been added to the list:

```
$ dsconfig \
 list-password-policies \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --property use-ssl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Password Policy         : Type              : use-ssl
------------------------:-------------------:--------
Default Password Policy : password-policy   : -
PTA Policy              : ldap-pass-through : true
Root Password Policy    : password-policy   : -
```

Use PTA To active directory

The steps below demonstrate how to set up PTA to Active Directory. The information that follows will help you make sense of the steps.

Entries on the DS side use `uid` as the naming attribute, and entries also have `cn` attributes. Active Directory entries use `cn` as the naming attribute. User entries on both sides share the same `cn` values. The mapping between entries therefore uses `cn`.

Consider a deployment where the DS account with `cn=LDAP PTA User` and DN `uid=ldapptauser,ou=People,dc=example,dc=com` corresponds to an Active Directory account with DN `CN=LDAP PTA User,CN=Users,DC=internal,DC=forgerock,DC=com`. The steps below enable the user with `cn=LDAP PTA User` on the DS server to authenticate through Active Directory:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery
 --baseDN dc=example,dc=com \
 uid=ldapptauser \
 cn

dn: uid=ldapptauser,ou=People,dc=example,dc=com
cn: LDAP PTA User
$ ldapsearch \
 --hostname ad.example.com \
 --port 636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN "CN=Users,DC=internal,DC=forgerock,DC=com" \
 --bindDN "cn=administrator,cn=Users,DC=internal,DC=forgerock,DC=com" \
 --bindPassword password \
 "(cn=LDAP PTA User)" \
 cn

dn: CN=LDAP PTA User,CN=Users,DC=internal,DC=forgerock,DC=com
cn: LDAP PTA User
```

The DS server must map the `uid=ldapptauser,ou=People,dc=example,dc=com` entry to the Active Directory `CN=LDAP PTA User,CN=Users,DC=internal,DC=forgerock,DC=com` entry. In order to do the mapping, the DS server must search for the user in Active Directory, using the `cn` value that it recovers from its own entry for the user. Active Directory does not allow anonymous searches, so part of the authentication policy configuration consists of the administrator DN and password the DS server uses to bind to Active Directory to search.

Finally, before setting up the PTA policy, make sure the DS server can connect to Active Directory over a secure connection to avoid sending passwords in the clear.

1. Export the certificate from the Windows server.
    - Select start > All Programs > Administrative Tools > Certification Authority, then right-click the CA and select Properties.
    - In the General tab, select the certificate and select View Certificate.
    - In the Certificate dialog, select the Details tab, then select Copy to File.
    - Use the Certificate Export Wizard to export the certificate to a file, such as `windows.cer` .
2. Copy the exported certificate to the system running the DS server.
3. Import the server certificate into the DS keystore:

```
$ keytool \
 -importcert \
 -alias ad-cert \
 -keystore /path/to/opendj/config/keystore \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -storetype PKCS12 \
 -file ~/Downloads/windows.cer \
 -noprompt

Certificate was added to keystore
```

At this point, the DS server can connect securely to Active Directory.

4. Set up an authentication policy for DS users to authenticate to Active Directory:

```
# Create a trust manager provider to access the Active Directory certificate:
$ dsconfig \
 create-trust-manager-provider \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --provider-name PKCS12 \
 --type file-based \
 --set enabled:true \
 --set trust-store-type:PKCS12 \
 --set trust-store-file:config/keystore \
 --set trust-store-pin:"&{file:config/keystore.pin}" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

# Use the trust manager provider in the PTA policy:
$ dsconfig \
 create-password-policy \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --type ldap-pass-through \
 --policy-name "AD PTA Policy" \
 --set primary-remote-ldap-server:ad.example.com:636 \
 --set mapped-attribute:cn \
 --set mapped-search-base-dn:"CN=Users,DC=internal,DC=forgerock,DC=com" \
 --set mapped-search-bind-dn:"cn=administrator,cn=Users,DC=internal,DC=forgerock,DC=com" \
 --set mapped-search-bind-password:password \
 --set mapping-policy:mapped-search \
 --set trust-manager-provider:PKCS12 \
 --set use-ssl:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

5. Assign the authentication policy to a test user:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=ldapptauser,ou=People,dc=example,dc=com
changetype: modify
add: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=AD PTA Policy,cn=Password Policies,cn=config
EOF
```

6. Check that the user can bind using PTA to Active Directory:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN dc=example,dc=com \
 --bindDN uid=ldapptauser,ou=People,dc=example,dc=com \
 --bindPassword password \
 "(cn=LDAP PTA User)" \
 userpassword cn

dn: uid=ldapptauser,ou=People,dc=example,dc=com
cn: LDAP PTA User
```

Notice that to complete the search, the user has authenticated with a password to Active Directory. No `userpassword` value is present in the DS service.

Assign PTA policies

You assign authentication policies in the same way as you assign password policies, by using the `ds-pwp-password-policy-dn` attribute.

> **Note**
>
> Although you assign the PTA policy using the same attribute as for password policy, the authentication policy is not in fact a password policy. Therefore, the user with a PTA policy does not have the operational attribute `pwdPolicySubentry` .

Assign a PTA policy to a user

Users depending on PTA no longer need a local password policy, as they no longer authenticate locally.

Examples in the following procedure work for this user, whose entry is as shown below. Notice that the user has no `userPassword` attribute. The user's password on the authentication server is `password` :

```
dn: uid=ptaUser,ou=People,dc=example,dc=com
uid: ptaUser
objectClass: top
objectClass: person
objectClass: organizationalperson
objectClass: inetorgperson
uid: ptaUser
cn: PTA User
sn: User
```

This user's entry on the authentication server has `uid=ptaUser`. The PTA policy performs the mapping to find the user entry in the authentication server:

1. Give an administrator access to update a user's password policy:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "ds-pwp-password-policy-dn")
 (version 3.0;acl "Allow Kirsten Vaughan to assign password policies";
 allow (all) (userdn = "ldap:///uid=kvaughan,ou=People,dc=example,dc=com");)
EOF
```

2. Update the user's `ds-pwp-password-policy-dn` attribute:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: uid=ptaUser,ou=People,dc=example,dc=com
changetype: modify
replace: ds-pwp-password-policy-dn
ds-pwp-password-policy-dn: cn=PTA Policy,cn=Password Policies,cn=config
EOF
```

3. Check that the user can authenticate through to the authentication server:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN dc=example,dc=com \
 --bindDN uid=ptaUser,ou=People,dc=example,dc=com \
 --bindPassword chngthspwd \
 "(uid=ptaUser)" \
 1.1

dn: uid=ptaUser,ou=People,dc=example,dc=com
```

Assign a PTA policy to a group

Examples in the following steps use the PTA policy defined previously. The administrator's entry is present on the authentication server:

1. Give an administrator the privilege to write subentries, such as those used for password policies:

   ```
   $ ldapmodify \
    --hostname localhost \
    --port 1636 \
    --useSsl \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --bindDN uid=admin \
    --bindPassword password << EOF
   dn: uid=kvaughan,ou=People,dc=example,dc=com
   changetype: modify
   add: ds-privilege-name
   ds-privilege-name: subentry-write
   EOF
   ```

   Notice here that the directory superuser, `uid=admin`, assigns privileges. Any administrator with the `privilege-change` privilege can assign privileges. However, if the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, do not assign the `privilege-change` privilege to normal administrator users.

2. Create a subentry for a collective attribute that sets the `ds-pwp-password-policy-dn` attribute for group members' entries:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=PTA Policy for Dir Admins,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: PTA Policy for Dir Admins
ds-pwp-password-policy-dn;collective: cn=PTA Policy,cn=Password Policies,cn=config
subtreeSpecification: { base "ou=People", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)"}
EOF
```

The `base` entry identifies the branch that holds administrator entries.

3. Check that the DS server has applied the policy.
    - Make sure you can bind as the user on the authentication server:

```
$ ldapsearch \
 --hostname remote-server.example.com \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=People,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN "dc=example,dc=com" \
 "(uid=kvaughan)" \
 1.1

dn: uid=kvaughan,ou=People,dc=example,dc=com
```

    - Check that the user can authenticate through to the authentication server from the DS server:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 1.1

dn: uid=kvaughan,ou=People,dc=example,dc=com
```

Certificate-based authentication

One alternative to simple binds with user name/password combinations consists of storing a digital certificate on the LDAP entry, and using the certificate as credentials during the bind. You can use this mechanism, for example, to let applications bind without using passwords.

By setting up a secure connection with a certificate, the client is in effect authenticating to the server. The server must close the connection if it cannot trust the client certificate. However, the certificate presented when establishing a secure connection does not authenticate the client. The secure connection is established by the JVM at the transport layer, independently of the LDAP protocol.

When binding with a certificate, the client must request the SASL external mechanism. The DS server maps the certificate to the client's entry in the directory. When it finds a matching entry, and the entry contains a certificate, the DS server can check whether the certificate in the entry matches the certificate from the request. It depends on the `certificate-validation-policy` setting of the SASL external handler. On success, the server sets the authorization identity for the connection, and the bind is successful.

For the whole process of authenticating with a certificate to work smoothly, the DS server and the client must trust each others' certificates, and the DS server must be configured to map the certificate to the client entry.

Add certificate attributes to the client entry

Before a client tries to bind to DS servers using a certificate, create a certificate, and add appropriate certificate attributes to the client's entry.

The `ds-evaluation` setup profile includes the entry, `cn=My App,ou=Apps,dc=example,dc=com`, used in these examples. The client key store password is stored in a `MY_KEYSTORE_PIN` environment variable:

1. Create a certificate using the DN of the client entry as the subject DN.

   This example uses the `dskeymgr` command to generate a key pair. The certificate is signed by the private CA based on the deployment ID used when setting up DS servers. Servers set up with the same deployment ID trust the CA, and so can trust the client's certificate:

```
$ dskeymgr \
 create-tls-key-pair \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --alias myapp-cert \
 --subjectDn "cn=My App,ou=Apps,dc=example,dc=com" \
 --keyStoreFile /path/to/opendj/my-keystore \
 --keyStorePassword $MY_KEYSTORE_PIN
```

For more command options, refer to dskeymgr. The default validity for the certificate is one year.

2. Make note of the certificate SHA-256 fingerprint.

   Later in this procedure you update the client application entry with the SHA-256 fingerprint, referred to henceforth as `SHA265_FINGERPRINT`:

   ```
   $ keytool \
    -list \
    -v \
    -alias myapp-cert \
    -keystore /path/to/opendj/my-keystore \
    -storepass $MY_KEYSTORE_PIN | awk '/SHA256:/{print $2}'

   SHA256_FINGERPRINT
   ```

3. Modify the entry to add attributes related to the certificate.

   For example, add the client certificate fingerprint on the `ds-certificate-fingerprint` attribute. This example uses the SHA-256 fingerprint, which is the default for the fingerprint certificate mapper.

   To require that the certificate is issued by a known CA, use the `ds-certificate-issuer-dn` attribute. Use this to verify the certificate issuer whenever multiple CAs are trusted in order to prevent impersonation. Different CAs can issue certificates with the same subject DN, but not with the same issuer DN. You must also specify the issuer attribute in the certificate mapper configuration, as shown below.

   To map the certificate subject DN to an attribute of the entry, use the `ds-certificate-subject-dn` attribute.

   The following entry demonstrates all these attributes. Save the entry in a file `addcert.ldif` in order to edit the fingerprint:

```
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
add: objectclass
objectclass: ds-certificate-user
-
add: ds-certificate-fingerprint
ds-certificate-fingerprint: SHA256_FINGERPRINT
-
add: ds-certificate-issuer-dn
ds-certificate-issuer-dn: CN=Deployment key,O=ForgeRock.com
-
add: ds-certificate-subject-dn
ds-certificate-subject-dn: CN=My App, OU=Apps, DC=example, DC=com
```

Replace the certificate fingerprint with the actual fingerprint before adding the certificate:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 addcert.ldif
```

4. Check your work:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(cn=My App)"

dn: cn=My App,ou=Apps,dc=example,dc=com
objectClass: top
objectClass: applicationProcess
objectClass: simpleSecurityObject
objectClass: ds-certificate-user
cn: My App
ds-certificate-fingerprint: SHA256_FINGERPRINT
ds-certificate-issuer-dn: CN=Deployment key,O=ForgeRock.com
ds-certificate-subject-dn: CN=My App, OU=Apps, DC=example, DC=com
```

Trust a third-party client certificate

> **Tip**
>
> If you control the client application, use your private CA to sign its certificate. This can be done as demonstrated in Add certificate attributes to the client entry. You can then skip this procedure.

To trust the client certificate, a trust manager must be able to trust the signing certificate (issuer). If the client certificate is self-signed or signed by an unknown CA, you must update the server's truststore:

1. Export the self-signed certificate or the CA certificate:

```
$ keytool \
 -export \
 -alias myapp-cert \
 -keystore /path/to/opendj/my-keystore \
 -storepass:env MY_KEYSTORE_PIN \
 -keypass:env MY_KEYSTORE_PIN \
 -file /path/to/opendj/myapp-cert.crt

Certificate stored in file </path/to/opendj/myapp-cert.crt>
```

   The command is similar for a CA certificate.

2. Import the exported, trusted certificate into a server truststore.

   The following examples use the default server keystore and PIN:

   1. The following example imports the self-signed certificate exported in Step 1:

```
$ keytool \
 -import \
 -alias myapp-cert \
 -file /path/to/opendj/myapp-cert.crt \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -noprompt

Certificate was added to keystore
```

   2. The following example imports an exported CA certificate in a file called `ca.crt` :

```
$ keytool \
 -import \
 -alias ca-cert \
 -file ca.crt \
 -keystore /path/to/opendj/config/keystore \
 -storetype PKCS12 \
 -storepass:file /path/to/opendj/config/keystore.pin \
 -noprompt

Certificate was added to keystore
```

Configure certificate mappers

DS servers use certificate mappers during binds to establish a mapping between a client certificate and the entry with the certificate. DS servers have the following certificate mappers:

Fingerprint Certificate Mapper

Looks for the certificate fingerprint in an attribute of the entry (default: `ds-certificate-fingerprint`).

Subject Attribute To User Attribute Mapper

Looks for a match between an attribute of the certificate subject and an attribute of the entry (default: match `cn` in the certificate to `cn` on the entry, or match `emailAddress` in the certificate to `mail` on the entry).

Subject DN to User Attribute Certificate Mapper

Looks for the certificate subject DN in an attribute of the entry (default: `ds-certificate-subject-dn`).

Subject Equals DN Certificate Mapper

Looks for an entry whose DN matches the certificate subject DN.

The following steps demonstrate how to use the Fingerprint Mapper default algorithm of SHA-256:

1. List the certificate mappers to retrieve the correct name:

```
$ dsconfig \
 list-certificate-mappers \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Certificate Mapper                  : Type                                : enabled
------------------------------------:--------------------------------------:--------
Fingerprint Mapper                  : fingerprint                         : true
Subject Attribute to User Attribute : subject-attribute-to-user-attribute : true
Subject DN to User Attribute        : subject-dn-to-user-attribute        : true
Subject Equals DN                   : subject-equals-dn                   : true
```

2. Examine the current configuration:

```
$ dsconfig \
 get-certificate-mapper-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --mapper-name "Fingerprint Mapper" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Property              : Value(s)
----------------------:----------------------------------------------------------
enabled               : true
fingerprint-algorithm : sha256
fingerprint-attribute : ds-certificate-fingerprint
issuer-attribute      : The certificate issuer DN will not be verified.
user-base-dn          : The server performs the search in all public naming
                      : contexts.
```

3. Change the configuration as necessary.

4. Set the External SASL Mechanism Handler to use the appropriate certificate mapper (default: Subject Equals DN):

```
$ dsconfig \
 set-sasl-mechanism-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name External \
 --set certificate-mapper:"Fingerprint Mapper" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Authenticate with the client certificate

Instead of providing a bind DN and password as for simple authentication, use the SASL EXTERNAL authentication mechanism, and provide the certificate. As a test with example data, you can try an anonymous search, then try with certificate-based authentication.

Before you try this example, make sure the DS server is set up to accept StartTLS from clients, and that you have set up the client certificate as described above. The password for the client key store is stored in a `MY_KEYSTORE_PIN` environment variable.

Also, if the DS server uses a certificate for StartTLS that was signed by a private CA, reference a truststore containing the CA certificate. In this example, the DS server uses a keystore with the CA certificate, and the client uses the keystore as its truststore.

Notice that the DS server does not allow an anonymous user to modify its description:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin << EOF
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
replace: description
description: New description

EOF

# The LDAP modify request failed: 50 (Insufficient Access Rights)
# Additional Information:  The entry cn=My App,ou=Apps,dc=example,dc=com cannot be modified due to insufficient
access rights
```

After the client binds successfully, it can modify its description:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --saslOption mech="EXTERNAL" \
 --certNickName myapp-cert \
 --keyStorePath /path/to/opendj/my-keystore \
 --keyStorePassword $MY_KEYSTORE_PIN <<EOF
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
replace: description
description: New description

EOF

# MODIFY operation successful for DN cn=My App,ou=Apps,dc=example,dc=com
```

You can also try the same test with other certificate mappers.

This example uses the fingerprint mapper:

```
$ dsconfig \
 set-sasl-mechanism-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name External \
 --set certificate-mapper:"Fingerprint Mapper" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --saslOption mech="EXTERNAL" \
 --certNickName myapp-cert \
 --keyStorePath /path/to/opendj/my-keystore \
 --keyStorePassword $MY_KEYSTORE_PIN <<EOF
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
replace: description
description: New description

EOF

# MODIFY operation successful for DN cn=My App,ou=Apps,dc=example,dc=com
```

This example uses the subject attribute to user attribute mapper:

```
$ dsconfig \
 set-sasl-mechanism-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name External \
 --set certificate-mapper:"Subject Attribute to User Attribute" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --saslOption mech="EXTERNAL" \
 --certNickName myapp-cert \
 --keyStorePath /path/to/opendj/my-keystore \
 --keyStorePassword $MY_KEYSTORE_PIN <<EOF
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
replace: description
description: New description

EOF

# MODIFY operation successful for DN cn=My App,ou=Apps,dc=example,dc=com
```

This example uses the subject DN to user attribute mapper:

```
$ dsconfig \
 set-sasl-mechanism-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name External \
 --set certificate-mapper:"Subject DN to User Attribute" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --saslOption mech="EXTERNAL" \
 --certNickName myapp-cert \
 --keyStorePath /path/to/opendj/my-keystore \
 --keyStorePassword $MY_KEYSTORE_PIN <<EOF
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
replace: description
description: New description

EOF

# MODIFY operation successful for DN cn=My App,ou=Apps,dc=example,dc=com
```

Authenticate With a Third-Party Certificate

When a client application authenticates with a self-signed certificate or a certificate signed by a public CA, the easiest certificate mapper to use is the fingerprint mapper. When using any other certificate mapper, make sure that the client certificate is in the client's entry, and that the SASL EXTERNAL handler has a `certificate-validation-policy:ifpresent` (default), or `certificate-validation-policy:always`. Any client certificate can be used to secure TLS for the connection. However, to trust the certificate for authentication, the server must ensure a unique match between the client's certificate and the client's entry.

A client application creating its own certificate can set subject DN, issuer DN, and other fields as desired, so these cannot be used to established trust. When obtaining a signature from a public CA, the client might set many fields as desired. The issuer DN guarantees only that the CA signed the certificate.

The following example demonstrates safe use of blind trust and fingerprint mapping. This demonstration is also appropriate when clients use certificates signed by public CAs, in which case, you could use the JVM trust manager, for example.

Enable a blind trust manager:

```
$ dsconfig \
 create-trust-manager-provider \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --type blind \
 --provider-name "Blind Trust" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Use SHA-256 as the fingerprint certificate mapper algorithm, which is the default.

Configure the handler for SASL EXTERNAL binds to use the fingerprint mapper, rather than the default subject DN mapper. As in this example, do not enable a more lenient mapper when using blind trust or public trust:

```
$ dsconfig \
 set-sasl-mechanism-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name External \
 --set certificate-mapper:"Fingerprint Mapper" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

After making these configuration changes, enable the trust manager on the appropriate connection handler. The following command enables blind trust on the LDAP connection handler, where the client will use StartTLS. Notice that only blind trust is enabled. If you want to allow blind trust for some applications and private CA trust for others, use a separate connection handler listening on a separate port:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAP \
 --set trust-manager-provider:"Blind Trust" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Make sure the client application certificate fingerprints use SHA-256. The following command updates the example client entry to change the fingerprint appropriately:

```
$ SHA256_FINGERPRINT=$(keytool \
 -list \
 -v \
 -alias myapp-cert \
 -keystore /path/to/opendj/my-keystore \
 -storepass $MY_KEYSTORE_PIN | awk '/SHA256:/{print $2}')

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
replace: ds-certificate-fingerprint
ds-certificate-fingerprint: $SHA256_FINGERPRINT

EOF
```

When the client binds successfully, it can modify its description. The server JVM checks client certificate validity and proves the client has the private key during the process of setting up TLS. During the SASL EXTERNAL bind, the server verifies that the fingerprint in the client entry matches the certificate:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --saslOption mech="EXTERNAL" \
 --certNickName myapp-cert \
 --keyStorePath /path/to/opendj/my-keystore \
 --keyStorePassword $MY_KEYSTORE_PIN <<EOF
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
replace: description
description: New description

EOF

# MODIFY operation successful for DN cn=My App,ou=Apps,dc=example,dc=com
```

For additional verification during the bind, include the client certificate in the client's entry.

Authenticate with Kerberos

Windows, UNIX, and Linux systems support Kerberos v5 authentication, which can operate safely on an open, unprotected network. In Kerberos authentication, the client application obtains temporary credentials for a service from an authorization server, in the form of tickets and session keys. The service server must be able to handle its part of the Kerberos mutual authentication process.

DS servers can interoperate with Kerberos systems through their GSSAPI SASL authentication mechanism.

Meet the following constraints when working with Kerberos systems:

- The clocks on the host systems where the DS server runs must be kept in sync with other hosts in the system.

  For example, you can use Network Time Protocol (NTP) services to keep the clocks in sync.

- Each DS server needs its own keytab file, the file which holds its pairs of Kerberos principals and keys.
- DS server debug logging can display exceptions about unsupported encryption types when a mismatch occurs. The exceptions are visible only when you activate debug logging.

Configure DS as a Kerberos service server

Follow these steps when setting up DS servers as Kerberos service servers:

1. Make sure the clock on the DS server host system is in sync with the other hosts' clocks in the Kerberos system.
2. Make sure that DNS resolves fully qualified domain names correctly on all systems involved.
3. Make sure the necessary ports are open on the DS server host system.
4. Make sure the encryption strengths required by the Kerberos system are supported by the JVM that the DS server uses.
5. Make sure that Kerberos is operating correctly for other services, including Kerberos client services on the DS server host.

   This step depends on the implementation, but usually includes adding a Kerberos principal for the host.

6. Add a Kerberos principal for the DS server, such as `ldap/ds.example.com`.
7. Create a keytab file for the DS server.

   This step depends on the Kerberos implementation, but generally consists of extracting the key for the DS server Kerberos principal, such as `ldap/ds.example.com` on the host where the DS server runs.

8. Make the keytab file readable only by the DS server, and copy it to the `/path/to/opendj/config/` directory.
9. Configure the DS server to handle GSSAPI SASL authentication:

   ```
   $ dsconfig \
    set-sasl-mechanism-handler-prop \
    --handler-name GSSAPI \
    --set enabled:true \
    --set keytab:/path/to/opendj/config/opendj.keytab \
    --set server-fqdn:ds.example.com \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

10. If your Kerberos principal user identifiers are not of the form `name@realm`, configure an appropriate `identity-mapper` for the GSSAPI SASL mechanism handler.

By default, the DS server uses the regular expression identity mapper, which expects user identifiers to match the pattern `^([^@])@.$`. It maps the string before `@` to the value of the UID attribute. This works well for identifiers like `bjensen@EXAMPLE.COM`. For background information, see Identity mappers.

11. Restart the DS server to ensure the configuration changes are taken into account:

```
$ stop-ds --restart

...GSSAPI SASL mechanism using a server fully qualified domain name of:
 ds.example.com
...GSSAPI mechanism using a principal name of:
 principal="opendj/ds.example.com"
...The GSSAPI SASL mechanism handler initialization was successful
```

12. Test that the mechanism works, by authenticating as a Kerberos user:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN dc=example,dc=com \
 --saslOption mech=GSSAPI \
 --saslOption authid=bjensen@EXAMPLE.COM \
 uid=bjensen \
 cn

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
```

Passwords

DS servers simplify safe, centralized password management. DS servers use password policies to govern passwords.

Which password policy applies

The operational attribute, `pwdPolicySubentry`, identifies an account's password policy. The default global access control instructions prevent this operational attribute from being visible to normal users. The following example grants access to a group of administrators:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "pwdPolicySubentry||ds-pwp-password-policy-dn")
 (version 3.0;acl "Allow Administrators to manage user's password policy";
 allow (all) (groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com");)
EOF

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 pwdPolicySubentry

dn: uid=bjensen,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=Default Password Policy,cn=Password Policies,cn=config
```

Configure password policies

Adjust the default password policy

You can reconfigure the default password policy, for example, to check that passwords do not contain complete attribute values, and to prevent password reuse. The default policy is a per-server password policy.

1. Apply the changes to the default password policy:

   ```
   $ dsconfig \
    set-password-policy-prop \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --policy-name "Default Password Policy" \
    --set password-history-count:7 \
    --set password-validator:Attribute\ Value \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

2. Check your work:

```
$ dsconfig \
 get-password-policy-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Property                                  : Value(s)
------------------------------------------:--------------------------
account-status-notification-handler       : -
allow-expired-password-changes            : false
allow-user-password-changes               : true
default-password-storage-scheme           : PBKDF2-HMAC-SHA256
deprecated-password-storage-scheme        : -
expire-passwords-without-warning          : false
force-change-on-add                       : false
force-change-on-reset                     : false
grace-login-count                         : 0
idle-lockout-interval                     : 0 s
last-login-time-attribute                 : -
last-login-time-format                    : -
lockout-duration                          : 0 s
lockout-failure-count                     : 0
lockout-failure-expiration-interval       : 0 s
max-password-age                          : 0 s
max-password-reset-age                    : 0 s
min-password-age                          : 0 s
password-attribute                        : userPassword
password-change-requires-current-password : false
password-expiration-warning-interval      : 5 d
password-generator                        : Random Password Generator
password-history-count                    : 7
password-history-duration                 : 0 s
password-validator                        : Attribute Value
previous-last-login-time-format           : -
require-change-by-time                    : -
require-secure-authentication             : true
require-secure-password-changes           : true
```

3. Test changes to the default password policy.

   For example, the following tests demonstrate the attribute value password validator. The attribute value password validator rejects a new password when the password is contained in attribute values on the user's entry.

   By default, the attribute value password validator checks all attributes, checks whether portions of the password string match attribute values, where the portions are strings of length 5, and checks the reverse of the password as well:

```
$ dsconfig \
 get-password-validator-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --validator-name Attribute\ Value \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Property               : Value(s)
-----------------------:---------------------------------------------------
check-substrings       : true
enabled                : true
match-attribute        : All attributes in the user entry will be checked.
min-substring-length   : 5
test-reversed-password : true
```

Consider the attributes present on Babs Jensen's entry:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)"

dn: uid=bjensen,ou=People,dc=example,dc=com
objectClass: person
objectClass: cos
objectClass: oauth2TokenObject
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: posixAccount
objectClass: top
classOfService: bronze
cn: Barbara Jensen
cn: Babs Jensen
departmentNumber: 3001
description: Original description
diskQuota: 10 GB
facsimileTelephoneNumber: +1 408 555 1992
gidNumber: 1000
givenName: Barbara
homeDirectory: /home/bjensen
l: San Francisco
mail: bjensen@example.com
mailQuota: 1 GB
manager: uid=trigden, ou=People, dc=example,dc=com
oauth2Token: {"access_token":"123","expires_in":59,"token_type":"Bearer","refresh_token":"456"}
ou: Product Development
ou: People
preferredLanguage: en, ko;q=0.8
roomNumber: 0209
sn: Jensen
street: 201 Mission Street Suite 2900
telephoneNumber: +1 408 555 1862
uid: bjensen
uidNumber: 1076
```

Using the attribute value password validator, passwords like `bjensen12` and `babsjensenspwd` are not valid because substrings of the password match complete attribute values:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --newPassword bjensen12

The LDAP password modify operation failed: 19 (Constraint Violation)
Additional Information:  The provided new password failed the validation
checks defined in the server: The provided password was found in another
attribute in the user entry

$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --newPassword babsjensenspwd

The LDAP password modify operation failed: 19 (Constraint Violation)
Additional Information:  The provided new password failed the validation
checks defined in the server: The provided password was found in another
attribute in the user entry
```

The attribute value password validator does not check, however, whether the password contains substrings of attribute values:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --newPassword babsp4ssw0rd

The LDAP password modify operation was successful

$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword babsp4ssw0rd \
 --newPassword example.com

The LDAP password modify operation was successful
```

To avoid the problem of the latter example, you could use a dictionary password validator where the dictionary includes `example.com`.

Configure a NIST-inspired subentry policy

You can configure a password policy inspired by NIST 800-63 requirements:

- Use a strong password storage scheme.
- Enforce a minimum password length of 8 characters.
- Check for matches in a dictionary of compromised passwords.
- Do not use composition rules for password validation.

  In other words, do not require a mix of special characters, upper and lower case letters, numbers, or other composition rules.
- Do not enforce arbitrary password changes.

  In other words, do not set a maximum password age.

Follow these steps to set up a replicated, NIST-inspired LDAP subentry password policy:

1. Gzip a copy of a text file of common compromised passwords, one word per line.

   This example shows the gzipped text file as `/tmp/10k_most_common.gz`. After successfully updating a subentry password policy with the dictionary data, the input file is no longer required. Lists of common passwords can be found online.
2. Make sure you have enabled a strong storage scheme.

   Creating a password storage scheme requires access to edit the server configuration, which you might not have when creating a subentry password policy. This example therefore uses the `PBKDF2-HMAC-SHA512` storage scheme, which is enabled by default to use 10,000 iterations.

   This scheme is intentionally much slower and more CPU-intensive than the `PBKDF2-HMAC-SHA256` scheme with 10 iterations used by the default password policy when you install DS. Test that you have enough resources to sustain the expected peak rates of impacted operations before using a much stronger password storage scheme in your production deployment.

   Impacted operations include:

   - Adding or importing entries with passwords.
   - Authenticating using a password, such as simple bind.
   - Updating or resetting a password.
3. Make sure password policy administrators have the `subentry-write` privilege, and any required ACIs needed to write password policy subentries in the directory data.

   The following example grants access to password administrators. The administrator accounts are in the data where the password policy is to be stored:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=subentry-write privilege for administrators,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: subentry-write privilege for administrators
ds-privilege-name;collective: subentry-write
subtreeSpecification: {base "ou=people", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)" }

dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=example,dc=com")
 (targetattr = "*||ds-pwp-password-policy-dn||pwdPolicySubentry||subtreeSpecification")
 (version 3.0; acl "Admins can manage entries and password policies"; allow(all)
 groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)
EOF
```

Notice here that the directory superuser, `uid=admin`, assigns privileges. Any administrator with the `privilege-change` privilege can assign privileges. However, if the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, do not assign the `privilege-change` privilege to normal administrator users.

4. Create the password policy as one of the password policy administrators:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=NIST inspired policy,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
objectClass: ds-pwp-validator
objectClass: ds-pwp-length-based-validator
objectClass: ds-pwp-dictionary-validator
cn: NIST inspired policy
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA512
ds-pwp-length-based-min-password-length: 8
ds-pwp-dictionary-data:<file:///tmp/10k_most_common.gz
subtreeSpecification: {base "ou=people", specificationFilter "(objectclass=person)" }
EOF
```

After successfully adding the policy with the dictionary data, you can delete the input file.

5. Check the password policy works appropriately.

   The following example shows a rejected password modification:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --newPassword secret12

The LDAP password modify operation failed: 19 (Constraint Violation)
Additional Information:  The provided new password failed the validation
checks defined in the server: The provided password was found in another
attribute in the user entry
```

   The following example shows an accepted password modification:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --newPassword aET1OjQeVJECSMgxDPs3U6In

The LDAP password modify operation was successful
```

Create a per-server password policy

This example adds a per-server password policy for new users who have not yet used their credentials to bind:

1. Create the new password policy:

   ```
   $ dsconfig \
    create-password-policy \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --policy-name "New Account Password Policy" \
    --set default-password-storage-scheme:PBKDF2-HMAC-SHA256 \
    --set force-change-on-add:true \
    --set password-attribute:userPassword \
    --type password-policy \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

   As per-server password policies are not replicated, repeat this step on all replica directory servers.

2. Check your work:

```
$ dsconfig \
 get-password-policy-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "New Account Password Policy" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Property                                  : Value(s)
------------------------------------------:------------------
account-status-notification-handler       : -
allow-expired-password-changes            : false
allow-user-password-changes               : true
default-password-storage-scheme           : PBKDF2-HMAC-SHA256
deprecated-password-storage-scheme        : -
expire-passwords-without-warning          : false
force-change-on-add                       : true
force-change-on-reset                     : false
grace-login-count                         : 0
idle-lockout-interval                     : 0 s
last-login-time-attribute                 : -
last-login-time-format                    : -
lockout-duration                          : 0 s
lockout-failure-count                     : 0
lockout-failure-expiration-interval       : 0 s
max-password-age                          : 0 s
max-password-reset-age                    : 0 s
min-password-age                          : 0 s
password-attribute                        : userPassword
password-change-requires-current-password : false
password-expiration-warning-interval      : 5 d
password-generator                        : -
password-history-count                    : 0
password-history-duration                 : 0 s
password-validator                        : -
previous-last-login-time-format           : -
require-change-by-time                    : -
require-secure-authentication             : false
require-secure-password-changes           : false
```

3. Change the user's password policy after the password is successfully updated.

   For instructions on assigning a per-server password policy, see Assign a password policy to a user.

Test password policies

Use the LDAP password quality advice control to test passwords, and to validate that a password policy produces the expected failures when a new password does not comply with its requirements.

The feature is available over LDAP, and also over REST for HTTP applications.

List subentry password policies

Per-server password policies are part of the DS server configuration. Use the `dsconfig` command to list, read, and edit them.

Subentry policies are part of the DS directory data. Use the `ldapsearch` command to list and read them.

The following command lists the subentry password policies under `dc=example,dc=com`:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDn dc=example,dc=com \
 "(&(objectClass=subEntry)(objectClass=ds-pwp-password-policy))"

dn: cn=NIST inspired policy,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
objectClass: ds-pwp-validator
objectClass: ds-pwp-length-based-validator
objectClass: ds-pwp-dictionary-validator
cn: NIST inspired policy
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA512
ds-pwp-dictionary-data:: <data>
ds-pwp-length-based-min-password-length: 8
ds-pwp-password-attribute: userPassword
```

Assign password policies

Type	To Assign...
Per-server password policy	Set the ds-pwp-password-policy-dn operational attribute on the user's account.
Subentry password policy	Use one of the following methods: • Set the ds-pwp-password-policy-dn operational attribute on the user's account. • Add the policy to an LDAP subentry whose immediate superior is the root of the subtree containing the accounts. For example, add the subentry password policy under `ou=People,dc=example,dc=com`. It applies to all accounts under `ou=People,dc=example,dc=com`. • Use the capabilities of LDAP subentries. Refine the scope of application by setting the subtreeSpecification attribute on the policy entry.

> **Important**
>
> Do not assign more than one password policy to the same account. Conflicting password policies will yield inconsistent results.
> You can review the password policy assigned to an account by reading the `pwdPolicySubentry` attribute on the entry.

Assign a password policy to a user

1. Make sure the password administrator has access to manage password policies:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=subentry-write privilege for administrators,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: subentry-write privilege for administrators
ds-privilege-name;collective: subentry-write
subtreeSpecification: {base "ou=people", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)" }

dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=example,dc=com")
 (targetattr = "*||ds-pwp-password-policy-dn||pwdPolicySubentry||subtreeSpecification")
 (version 3.0; acl "Admins can manage entries and password policies"; allow(all)
 groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)
EOF
```

   Notice here that the directory superuser, `uid=admin`, assigns privileges. Any administrator with the `privilege-change` privilege can assign privileges. However, if the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, do not assign the `privilege-change` privilege to normal administrator users.

2. Set the user's `ds-pwp-password-policy-dn` attribute as the password administrator:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery << 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: chngthspwd
ds-pwp-password-policy-dn: cn=NIST inspired policy,dc=example,dc=com
EOF
```

3. Check your work:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=newuser)" \
 pwdPolicySubentry

dn: uid=newuser,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=NIST inspired policy,dc=example,dc=com
```

Assign a password policy to a group

You can use a collective attribute to assign a password policy. Collective attributes provide a standard mechanism for defining attributes that appear on all the entries in a subtree. For details, see Collective attributes:

1. Make sure the password administrator has the privilege to write subentries:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: subentry-write
EOF
```

Notice here that the directory superuser, `uid=admin`, assigns privileges. Any administrator with the `privilege-change` privilege can assign privileges. However, if the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, do not assign the `privilege-change` privilege to normal administrator users.

2. Create a subentry defining the collective attribute that sets the `ds-pwp-password-policy-dn` attribute for group members' entries:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=Password Policy for Dir Admins,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: Password Policy for Dir Admins
ds-pwp-password-policy-dn;collective: cn=Root Password Policy,cn=Password Policies,cn=config
subtreeSpecification: { base "ou=People", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)"}
EOF
```

3. Check your work:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 pwdPolicySubentry

dn: uid=kvaughan,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=Root Password Policy,cn=Password Policies,cn=config
```

Assign a password policy to a branch

These steps apply only to subentry password policies:

1. Give an administrator the privilege to write subentries, such as those used for setting password policies:

   ```
   $ ldapmodify \
    --hostname localhost \
    --port 1636 \
    --useSsl \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --bindDN uid=admin \
    --bindPassword password << EOF
   dn: uid=kvaughan,ou=People,dc=example,dc=com
   changetype: modify
   add: ds-privilege-name
   ds-privilege-name: subentry-write
   EOF
   ```

   Notice here that the directory superuser, `uid=admin`, assigns privileges. Any administrator with the `privilege-change` privilege can assign privileges. However, if the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, do not assign the `privilege-change` privilege to normal administrator users.

2. Configure a subentry password policy with a `subtreeSpecification` attribute that defines which accounts are assigned the policy.

   The following example assigns `cn=NIST inspired policy` to accounts under `ou=People,dc=example,dc=com`:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=NIST inspired policy,dc=example,dc=com
changetype: modify
replace: subtreeSpecification
subtreeSpecification: { base "ou=people" }
EOF
```

The subtree specification assigns the policy to the people branch with `{ base "ou=people" }`. You could relax the subtree specification value to `{}` to apply the policy to all entries anywhere underneath `dc=example,dc=com`. You could further restrict the subtree specification by adding a `specificationFilter`. For details, see About subentry scope.

3. Check your work to see that an account under `ou=People` has the policy:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=alutz)" \
 pwdPolicySubentry

dn: uid=alutz,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=NIST inspired policy,dc=example,dc=com
```

About subentry scope

LDAP subentries reside with the user data and so the server replicates them. Subentries hold operational data. They are not visible in search results unless explicitly requested. This section describes how a subentry's `subtreeSpecification` attribute defines the scope of the subtree that the subentry applies to.

An LDAP subentry's subtree specification identifies a subset of entries in a branch of the DIT. The subentry scope is these entries. In other words, these are the entries that the subentry affects.

The attribute value for a `subtreeSpecification` optionally includes the following parameters:

base

Indicates the entry, *relative to the subentry's parent*, at the base of the subtree.

By default, the base is the subentry's parent.

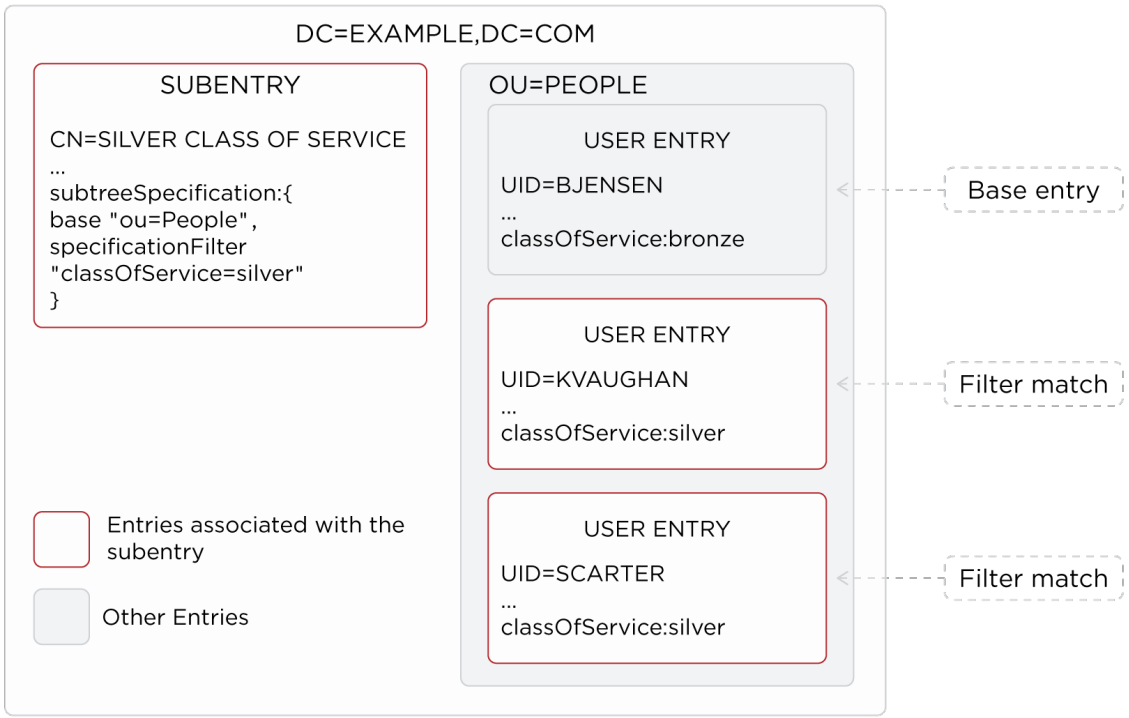
`specificationFilter`

Indicates an LDAP filter. Entries matching the filter are in scope.

DS servers extend the standard implementation to allow any search filter, not just an assertion about the `objectClass` attribute.

By default, all entries under the base entry are in scope.

The following illustration shows this for an example collective attribute subentry:



Notice that the base of `ou=People` on the subentry `cn=Silver Class of Service,dc=example,dc=com` indicates that the base entry is `ou=People,dc=example,dc=com`.

The filter `"(classOfService=silver)"` means that Kirsten Vaughan and Sam Carter's entries are in scope. Babs Jensen's entry, with `classOfService: bronze` does not match and is therefore not in scope. The `ou=People` organizational unit entry does not have a `classOfService` attribute, and so is not in scope, either.

Strong and safe passwords

The difficulty with passwords is that they tend to be relatively easy to guess. Despite decades of advice on how to pick strong passwords, people still routinely pick very weak passwords using common words and phrases or simple variations of them. This makes them extremely easy to guess. Attackers with access to even modest hardware can make billions of guesses per second.

DS servers provide flexible password validation to fit your policies about password content, and to reject weak passwords when users try to save them. It also provides a variety of one-way and reversible password storage schemes. Password strength is a function of both password minimum length, which you can set as part of password policy, and password quality, which requires password validation.

Password validation

When a password is added or updated, a password validator determines whether the server should accept it. Validation does not affect existing passwords.

A user's password policy specifies which password validators apply whenever that user provides a new password.

Subentry password policies can include attributes of password validator object classes. Each object class derives from the abstract `ds-pwp-validator` class:

- ds-pwp-attribute-value-validator attributes
- ds-pwp-character-set-validator attributes
- ds-pwp-dictionary-validator attributes
- ds-pwp-length-based-validator attributes
- ds-pwp-repeated-characters-validator attributes
- ds-pwp-similarity-based-validator attributes
- ds-pwp-unique-characters-validator attributes

The example that follows shows a password policy that requires new passwords to have at least three of the following four character classes:

- English lowercase characters (a through z)
- English uppercase characters (A through Z)
- Base 10 digits (0 through 9)
- Punctuation characters (for example, !, $, #, %)

Notice how the `character-set` values are constructed. The initial `0:` means the set is optional, whereas `1:` means the set is required:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Policy with character set validation,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
objectClass: ds-pwp-validator
objectClass: ds-pwp-character-set-validator
cn: Policy with character set validation
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-character-set-allow-unclassified-characters: true
ds-pwp-character-set-character-set-ranges: 0:a-z
ds-pwp-character-set-character-set-ranges: 0:A-Z
ds-pwp-character-set-character-set-ranges: 0:0-9
ds-pwp-character-set-character-set: 0:!$%^.#
ds-pwp-character-set-min-character-sets: 3
subtreeSpecification: { base "ou=people", specificationFilter "(uid=bjensen)" }
EOF

$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --authzID "u:bjensen" \
 --newPassword '!ABcd$%^'
```

An attempt to set an invalid password fails as shown in the following example:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --authzID "u:bjensen" \
 --newPassword hifalutin

The LDAP password modify operation failed: 19 (Constraint Violation)
Additional Information:  The provided new password failed the validation
checks defined in the server: The provided password did not contain characters
from at least 3 of the following character sets or ranges: '!$%^.#', '0-9',
'A-Z', 'a-z'
```

Per-server password policies use validators that are separate configuration objects. The following example lists the password validators available by default for per-server password policies. By default, no password validators are configured in the default password policy:

```
$ dsconfig \
 list-password-validators \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Password Validator                  : Type                : enabled
------------------------------------:---------------------:--------
At least 8 characters               : length-based        : true
Attribute Value                     : attribute-value     : true
Character Set                       : character-set       : true
Common passwords                    : dictionary          : true
Dictionary                          : dictionary          : false
Length-Based Password Validator     : length-based        : true
Repeated Characters                 : repeated-characters : true
Similarity-Based Password Validator : similarity-based    : true
Unique Characters                   : unique-characters   : true
```

For details, see Password Validator.

For an example showing how to test password quality, see Check Password Quality.

Password storage

Password storage schemes, described in Password Storage Scheme, encode new passwords and store the encoded version. When a client application authenticates with the password, the server encodes the plaintext password using the configured storage scheme, and checks whether the result matches the encoded value stored by the server. If the encoded version is appropriately secure, it is difficult to guess the plaintext password from its encoded value.

DS servers offer a variety of reversible and one-way password storage schemes. With a reversible encryption scheme, an attacker who gains access to the server can recover the plaintext passwords. With a one-way hash storage scheme, the attacker who gains access to the server must still crack the password by brute force, encoding passwords over and over to generate guesses until a match is found. If you have a choice, use a one-way password storage scheme.

Some one-way hash functions are not designed specifically for password storage, but also for use in message authentication and digital signatures. Such functions, like those defined in the Secure Hash Algorithm (SHA-1 and SHA-2) standards, are designed for high performance. Because they are fast, they allow the server to perform authentication at high throughput with low response times. However, high-performance algorithms also help attackers use brute force techniques. One estimate in 2017 is that a single GPU can calculate over one billion SHA-512 hashes per second.

> **Warning**
>
> Some one-way hash functions are designed to be computationally *intensive*. Such functions, like PBKDF2, Argon2, and Bcrypt, are designed to be relatively slow, even on modern hardware. This makes them generally less susceptible to brute force attacks.
> *However*, computationally intensive functions reduce authentication throughput and increase response times. With the default number of iterations, the GPU mentioned above might only calculate 100,000 PBKDF2 hashes per second (or 0.01% of the corresponding hashes calculated with SHA-512). If you use these functions, be aware of the potentially dramatic performance impact and plan your deployment accordingly.

Modern hardware and techniques to pre-compute attempts, such as rainbow tables, make it increasingly easy for attackers to crack passwords by brute force. Password storage schemes that use *salt* make brute force attacks more expensive. In this context, salt is a random value appended to the password before encoding. The salt is then stored with the encoded value and used when comparing an incoming password to the stored password.

Reversible password storage schemes, such as AES and Blowfish, use symmetric keys for encryption.

The following example lists available alternatives, further described in Password Storage Schemes:

```
$ dsconfig \
 list-password-storage-schemes \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Password Storage Scheme : Type               : enabled
------------------------:--------------------:--------
3DES                    : triple-des         : false
AES                     : aes                : false
Argon2                  : argon2             : true
Base64                  : base64             : false
Bcrypt                  : bcrypt             : true
Blowfish                : blowfish           : false
Clear                   : clear              : false
CRYPT                   : crypt              : false
PBKDF2                  : pbkdf2             : false
PBKDF2-HMAC-SHA256      : pbkdf2-hmac-sha256 : true
PBKDF2-HMAC-SHA512      : pbkdf2-hmac-sha512 : true
PKCS5S2                 : pkcs5s2            : false
Salted SHA-1            : salted-sha1        : false
Salted SHA-256          : salted-sha256      : false
Salted SHA-384          : salted-sha384      : false
Salted SHA-512          : salted-sha512      : false
SCRAM-SHA-256           : scram-sha256       : true
SCRAM-SHA-512           : scram-sha512       : true
SHA-1                   : sha1               : false
```

As shown in Adjust the default password policy, the default password storage scheme for users is PBKDF2-HMAC-SHA256. When you add users or import user entries with `userPassword` values in plaintext, the DS server hashes them with the default password storage scheme. The default directory superuser has a different password policy, shown in Assign a password policy to a group. The Root Password Policy uses PBKDF2-HMAC-SHA256 by default.

Tip

The choice of default password storage scheme for normal users can significantly impact server performance. Each time a normal user authenticates using simple bind (username/password) credentials, the directory server encodes the user's password according to the storage scheme in order to compare it with the encoded value in the user's entry.
Schemes such as Salted SHA-512 call for relatively high-performance encoding. Schemes such as PBKDF2-HMAC-SHA256, which are designed to make the encoding process computationally intensive, reduce the bind throughput that can be achieved on equivalent hardware.
Take this performance impact into consideration when sizing your deployment. With a computationally intensive scheme such as PBKDF2-HMAC-SHA256, make sure the directory service has enough compute power to absorb the additional load.

Password Storage Schemes

Name	Type of Algorithm	Notes
3DES[1]	Reversible encryption[2]	Triple DES (Data Encryption Standard) in EDE (Encrypt Decrypt Encrypt) mode. Key size: 168 bits.
AES[1]	Reversible encryption[2]	Advanced Encryption Standard, successor to DES, published by the US National Institute of Standards and Technology (NIST). Key size: 128 bits.
Argon2	One-way hash	Computationally intensive, memory-hard hashing function. For default settings, see Additional Password Storage Scheme Settings.
Base64	Reversible encoding	Transfer encoding for representing binary password values in text. *Not intended as a secure storage scheme.*
Bcrypt	One-way hash	Computationally intensive hashing function, based on the Blowfish cipher. Default cost: 12 (2^12 iterations).
Blowfish[1]	Reversible encryption[2]	Public domain cipher designed by Bruce Schneier as a successor to DES. Key size: 128 bits.
Clear	Cleartext, no encoding	For backwards compatibility and use with certain legacy applications. *Not intended as a secure storage scheme.*
CRYPT	One-way hash	Based on the UNIX Crypt algorithm. For backwards compatibility and use with certain legacy applications. *Not intended as a secure storage scheme.* Default algorithm: `unix`.
MD5	One-way hash	Based on the MD5 algorithm defined in RFC 1321. For backwards compatibility and use with certain legacy applications. *Not intended as a secure storage scheme.*

Name	Type of Algorithm	Notes
PBKDF2	One-way hash	Computationally intensive hashing function, based on PBKDF2 algorithm defined in RFC 8018, 5.2. PBKDF2. Default iterations: 10000. The pseudorandom function for the algorithm corresponds to the HMAC based on SHA-1.
PBKDF2-HMAC-SHA256	One-way hash	Computationally intensive hashing function using PBKDF2. Default iterations: 10000. The pseudorandom function for the algorithm corresponds to the HMAC based on SHA-2, where the hash function is SHA-256.
PBKDF2-HMAC-SHA512	One-way hash	Computationally intensive hashing function using PBKDF2. Default iterations: 10000. The pseudorandom function for the algorithm corresponds to the HMAC based on SHA-2, where the hash function is SHA-512.
PKCS5S2	One-way hash	Computationally intensive hashing function, based on Atlassian's adaptation of the PBKDF2. Number of iterations: 10000.
RC4(1)	Reversible encryption(2)	Based on the Rivest Cipher 4 algorithm. For backwards compatibility and use with certain legacy applications. *Not intended as a secure storage scheme.* Key size: 128 bits.
Salted MD5	One-way hash	Based on MD5, with 64 bits of random salt appended to the plaintext before hashing, and then appended to the hash.
Salted SHA-1	One-way hash	Based on SHA-1, with 64 bits of random salt appended to the plaintext before hashing, and then appended to the hash.
Salted SHA-256	One-way hash	Based on the SHA-256 hash function using 32-bit words and producing 256-bit digests. SHA-256 is defined in the SHA-2 (Secure Hash Algorithm 2) standard developed by the US National Security Agency (NSA) and published by NIST. The salt is applied as for Salted SHA-1.
Salted SHA-384	One-way hash	Based on the SHA-384 hash function that effectively truncates the digest of SHA-512 to 384 bits. SHA-384 is defined in the SHA-2 (Secure Hash Algorithm 2) standard developed by the NSA and published by NIST. The salt is applied as for Salted SHA-1.

Name	Type of Algorithm	Notes
Salted SHA-512	One-way hash	Based on the SHA-512 hash function using 64-bit words and producing 512-bit digests. SHA-512 is defined in the SHA-2 (Secure Hash Algorithm 2) standard developed by the NSA and published by NIST. The salt is applied as for Salted SHA-1.
SCRAM-SHA-256	One-way hash	For use with the standard SASL Salted Challenge Response Authentication Mechanism (SCRAM), named `SCRAM-SHA-256`. A SASL SCRAM mechanism provides a secure alternative to transmitting plaintext passwords during binds. It is an appropriate replacement for DIGEST-MD5 and CRAM-MD5. With a SCRAM SASL bind, the client must demonstrate proof that it has the original plaintext password. During the SASL bind, the client must perform computationally intensive processing to prove that it has the plaintext password. This computation is like what the server performs for PBKDF2, but the password is not communicated during the bind. Once the server has stored the password, the client pays the computational cost to perform the bind. The server only pays a high computational cost when the password is updated, for example, when an entry with a password is added or during a password modify operation. A SASL SCRAM mechanism therefore offers a way to offload the high computational cost of secure password storage to client applications during authentication. Passwords storage using a SCRAM storage scheme is compatible with simple binds and SASL PLAIN binds. When a password is stored using a SCRAM storage scheme, the server pays the computational cost to perform the bind during a simple bind or SASL PLAIN bind. The SCRAM password storage scheme must match the SASL SCRAM mechanism used for authentication. In other words, SASL SCRAM-SHA-256 requires a SCRAM-SHA-256 password storage scheme. SASL SCRAM-SHA-512 requires a SCRAM-SHA-512 password storage scheme. Default iterations: 10000. The pseudorandom function for the algorithm corresponds to the HMAC based on SHA-2, where the hash function is SHA-256.
SCRAM-SHA-512	One-way hash	Like SCRAM-SHA-256, but the hash function is SHA-512. The corresponding SASL mechanism is named `SCRAM-SHA-512`.
SHA-1	One-way hash	SHA-1 (Secure Hash Algorithm 1) standard developed by the NSA and published by NIST. *Not intended as a secure storage scheme.*

(1) Reversible encryption schemes are deprecated.

(2) When you configure a reversible password storage scheme, enable the `adminRoot` backend, and configure a replication domain for `cn=admin data`. These additional steps let the replicas store and replicate the secret keys for password encryption.

Password storage schemes listed in the following table have additional configuration settings.

Additional Password Storage Scheme Settings

Scheme	Setting	Description
Argon2	`argon2-iterations`	The number of iterations to perform. Default: 2.
	`argon2-memory`	The amount of memory to use for a single hash, expressed in kibibytes. Default: 15360.
	`argon2-migration-memory`	The memory requirement expected during an Argon2 password migration. Ignored if less than `argon2-memory`. Default: 0.
	`argon2-parallelism`	The number of threads that work in parallel to compute a hash. Default: 1.
	`argon2-variant`	The variant of Argon2 algorithm to use (I, D, or ID). Default: ID.
	`argon2-length`	The length of the resulting hash. Default: 32.
	`argon2-salt-length`	The length of the salt used when hashing passwords. Default: 16.
	`rehash-policy`	Whether the server should rehash passwords after the cost has been changed. Default: never.
Bcrypt	`bcrypt-cost`	The cost parameter specifies a key expansion iteration count as a power of two. A default value of 12 (2^{12} iterations) is considered in 2016 as a reasonable balance between responsiveness and security for regular users.
	`rehash-policy`	Whether the server should rehash passwords after the cost has been changed. Default: never.
Crypt	`crypt-password-storage-encryption-algorithm`	Specifies the crypt algorithm to use to encrypt new passwords. The following values are supported: **`unix`** The password is encrypted with the weak Unix crypt algorithm. This is the default setting. **`md5`** The password is encrypted with the BSD MD5 algorithm and has a `$1$` prefix. **`sha256`** The password is encrypted with the SHA256 algorithm and has a `$5$` prefix. **`sha512`** The password is encrypted with the SHA512 algorithm and has a `$6$` prefix.

Scheme	Setting	Description
PBKDF2 PBKDF2-HMAC-SHA256 PBKDF2-HMAC-SHA512	`pbkdf2-iterations`	The number of algorithm iterations. The default is 10000.
	`rehash-policy`	Whether the server should rehash passwords after the cost has been changed. Default: never.
SCRAM SCRAM-SHA-256 SCRAM-SHA-512	`scram-iterations`	The number of algorithm iterations. The default is 10000.

You change the default password policy storage scheme for users by changing the applicable password policy:

```
$ dsconfig \
 set-password-policy-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set default-password-storage-scheme:PBKDF2-HMAC-SHA512 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Notice that the change in default password storage scheme does not cause the DS server to update any stored password values. By default, the server only stores a password with the new storage scheme the next time the password is changed.

For subentry password policies, set the `ds-pwp-default-password-storage-scheme` attribute to the common name of an enabled password storage scheme. To list the names of enabled password storage schemes, use the `dsconfig list-password-storage-schemes` command. The name appears in the first column of the output. The third column shows whether the scheme is enabled.

DS servers prefix passwords with the scheme used to encode them, which means it is straightforward to see which password storage scheme is used. After the default password storage scheme is changed to PBKDF2-HMAC-SHA512, old user passwords remain encoded with PBKDF2-HMAC-SHA256:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 userPassword

dn: uid=bjensen,ou=People,dc=example,dc=com
userPassword: {PBKDF2-HMAC-SHA256}10:<hash>
```

When the password is changed, the new default password storage scheme takes effect:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --authzID "u:bjensen" \
 --newPassword changeit

The LDAP password modify operation was successful

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword changeit \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 userPassword

dn: uid=bjensen,ou=People,dc=example,dc=com
userPassword: {PBKDF2-HMAC-SHA512}10000:<hash>
```

When you change the password storage scheme for users, realize that the user passwords must change in order for the DS server to encode them with the chosen storage scheme. If you are changing the storage scheme because the old scheme was too weak, then you no doubt want users to change their passwords anyway.

If, however, the storage scheme change is not related to vulnerability, use the `deprecated-password-storage-scheme` property in per-server password policies, or the `ds-pwp-deprecated-password-storage-scheme` attribute in subentry password policies. This setting causes the DS server to store the password in the new format after successful authentication. This makes it possible to do password migration for active users as users gradually change their passwords:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 userPassword

dn: uid=kvaughan,ou=People,dc=example,dc=com
userPassword: {PBKDF2-HMAC-SHA256}10:<hash>

$ dsconfig \
 set-password-policy-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set deprecated-password-storage-scheme:PBKDF2-HMAC-SHA256 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 userPassword

dn: uid=kvaughan,ou=People,dc=example,dc=com
userPassword: {PBKDF2-HMAC-SHA512}10000:<hash>
```

Notice that with `deprecated-password-storage-scheme` set appropriately, Kirsten Vaughan's password was hashed again after she authenticated successfully.

Password generation

DS servers use password generators when responding with a generated password for the LDAP Password Modify extended operation. A directory administrator resetting a user's password has the server generate the new password, and the server sends the new password in the response:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --authzID "u:bjensen"

The LDAP password modify operation was successful
Generated Password:  <random>
```

The default password policy uses the Random Password Generator, described in Random Password Generator:

```
$ dsconfig \
 get-password-policy-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --property password-generator \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Property           : Value(s)
-------------------:-------------------------
password-generator : Random Password Generator

$ dsconfig \
 get-password-generator-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --generator-name "Random Password Generator" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Property               : Value(s)
-----------------------:-------------------------------------------------------
enabled                : true
password-character-set : alphanum:abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRS
                       : TUVWXYZ0123456789
password-format        : alphanum:10
```

Notice that the default configuration for the Random Password Generator sets the `password-character-set` property, and references the settings in the `password-format` property. Generated passwords have eight characters: three from the `alpha` set, followed by two from the `numeric` set, followed by three from the `alpha` set. The `password-character-set` name must be ASCII.

Subentry password policies configure `ds-pwp-random-generator` object class attributes. The following example creates a password with password generation, and demonstrates its use:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Policy with random password generation,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
objectClass: ds-pwp-validator
objectClass: ds-pwp-length-based-validator
objectClass: ds-pwp-random-generator
cn: Policy with random password generation
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-random-password-character-set: alpha:ABCDEFGHIJKLMNOPQRSTUVWabcdefghijklmnopqrstuvwxyz
ds-pwp-random-password-character-set: punct:,.!&+=-_
ds-pwp-random-password-character-set: numeric:0123456789
ds-pwp-random-password-format: alpha:3,punct:1,numeric:2,punct:2,numeric:3,alpha:3,punct:2
ds-pwp-length-based-min-password-length: 8
subtreeSpecification: { base "ou=people" }
EOF

$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --authzID "u:bjensen"

The LDAP password modify operation was successful
Generated Password:  <random>
```

For details, see ds-pwp-random-generator attributes.

When configuring both password validators and password generators, make sure the generated passwords are acceptable to the validator. In this case, the minimum length is less than the generated password length, for example.

Sample password policies

Lock accounts after repeated bind failures

To help you prevent brute-force attacks, where an attacker tries many passwords in the hope of eventually guessing correctly, DS password policies support configurable account lockout. This feature is an important part of a secure password policy.

> **Note**
>
> When you configure account lockout as part of password policy, DS servers lock an account after the specified number of consecutive authentication failures. *Account lockout is not transactional across all replicas in a deployment.* Global account lockout occurs as soon as the authentication failure times have been replicated.

The following commands demonstrate a subentry password policy that locks accounts for five minutes after three consecutive bind failures. With this policy, the directory server records failure times, and slowly discards them. As a result, a brute-force attack is hopefully too slow to be effective, but no administrative action is needed when a user temporarily forgets or mistypes their password.

Once an account is locked, binds continue to fail for the lockout period, even if the credentials are correct. An account administrator can use the `manage-account` command to view the account status, and to change it if necessary:

```
# Set the password policy:
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Lock After Repeated Bind Failures,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
cn: Lock After Repeated Bind Failures
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-lockout-duration: 5 m
ds-pwp-lockout-failure-count: 3
ds-pwp-lockout-failure-expiration-interval: 2 m
subtreeSpecification: { base "ou=people", specificationFilter "(objectClass=posixAccount)" }
EOF

# Attempt to bind three times using the wrong password:
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword wrongPassword \
 --baseDn dc=example,dc=com \
 "(uid=bjensen)"

The LDAP bind request failed: 49 (Invalid Credentials)

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword wrongPassword \
 --baseDn dc=example,dc=com \
 "(uid=bjensen)"

The LDAP bind request failed: 49 (Invalid Credentials)

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword wrongPassword \
 --baseDn dc=example,dc=com \
 "(uid=bjensen)"
```

```
The LDAP bind request failed: 49 (Invalid Credentials)

# Observe the results:
$ manage-account \
 get-all \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --targetDN uid=bjensen,ou=people,dc=example,dc=com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin

Password Policy DN:  cn=Lock After Repeated Bind Failures,dc=example,dc=com
Seconds Until Authentication Failure Unlock:  <seconds>
```

Enforce regular password changes

The following commands configure a subentry password policy that sets age limits on passwords, requiring that users change their passwords at least every 13 weeks, but not more often than every 4 weeks. The policy also sets the number of passwords to keep in the password history of the entry, preventing users from reusing the same password on consecutive changes:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Enforce Regular Password Changes,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
cn: Enforce Regular Password Changes
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-max-password-age: 13 w
ds-pwp-min-password-age: 4 w
ds-pwp-password-history-count: 7
subtreeSpecification: { base "ou=people" }
EOF
```

Track last login time

The following command configures a subentry password policy that keeps track of the last successful login:

1. Create the password policy to write the timestamp to the attribute on successful login:

```
$ dsconfig \
 create-password-policy \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Track Last Login Time" \
 --type password-policy \
 --set default-password-storage-scheme:PBKDF2-HMAC-SHA256 \
 --set password-attribute:userPassword \
 --set last-login-time-attribute:ds-last-login-time \
 --set last-login-time-format:"yyyyMMddHH'Z'" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Deprecate a password storage scheme

The following commands configure a subentry password policy for deprecating a password storage scheme. This policy uses elements from Enforce regular password changes. The DS server applies the new password storage scheme to re-encode passwords:

- When they change.
- When the user successfully binds with the correct password, and the password is currently hashed with a deprecated scheme.

```
$ dsconfig \
 set-password-storage-scheme-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --scheme-name "Salted SHA-512" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Deprecate a Password Storage Scheme,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
cn: Deprecate a Password Storage Scheme
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-deprecated-password-storage-scheme: Salted SHA-512
ds-pwp-max-password-age: 13 w
ds-pwp-min-password-age: 4 w
ds-pwp-password-history-count: 7
subtreeSpecification: { base "ou=people" }
EOF
```

Lock idle accounts

The following commands configure a subentry password policy that locks accounts idle for more than 13 weeks. This policy extends the example from Track last login time. The DS server must track last successful login time to calculate how long the account has been idle:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Lock Idle Accounts,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
cn: Lock Idle Accounts
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-idle-lockout-interval: 13 w
ds-pwp-last-login-time-attribute: ds-last-login-time
ds-pwp-last-login-time-format: yyyyMMddHH'Z'
subtreeSpecification: { base "ou=people" }
EOF
```

Allow log in to change an expired password

The following commands configure a subentry password policy that lets users log in twice with an expired password to set a new password:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Allow Grace Login,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
cn: Allow Grace Login
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-grace-login-count: 2
subtreeSpecification: { base "ou=people" }
EOF
```

Require password change on add or reset

The following commands configure a subentry password policy that requires new users to change their password after logging in for the first time. This policy also requires users to change their password after it is reset:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Require Password Change on Add or Reset,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
cn: Require Password Change on Add or Reset
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-force-change-on-add: true
ds-pwp-force-change-on-reset: true
subtreeSpecification: { base "ou=people" }
EOF
```

About password policies

DS password policies govern passwords, account lockout, and account status notification.

DS servers support per-server password policies stored in the configuration, and subentry password policies stored in the (replicated) directory data:

Type	Notes
Per-server password policies	• Use for default policies, and policies for top-level administrative accounts. • You must manually apply policy updates to each replica server configuration. • Updates require write access to the server configuration.
DS subentry password policies	• Use for all user accounts stored in application data. • Replication applies each policy update to all replicas. • Updates require the `subentry-write` privilege, and ACIs to write the policy.

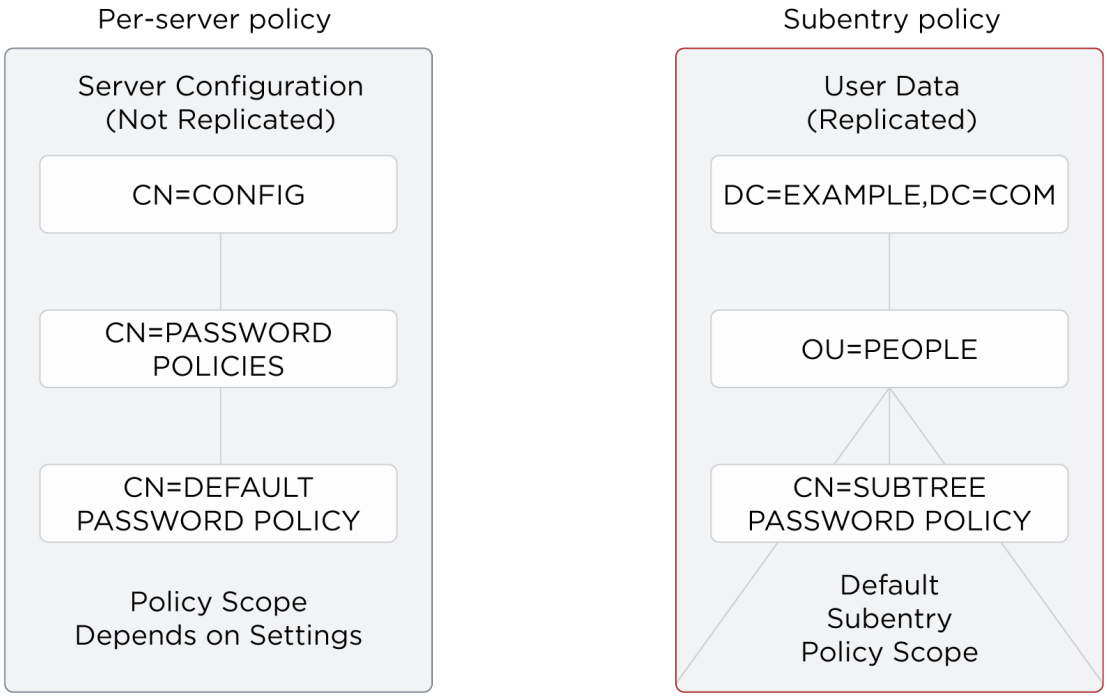


Figure 1. Per-Server and Subentry Password Policies

Per-server password policies

You manage per-server password policies with the `dsconfig` command. When changing a per-server policy, you must update each replica in your deployment.

By default, there are two per-server password policies:

- The `Default Password Policy` for users.
- The `Root Password Policy` for the directory superuser, `uid=admin`.

The following example displays the default per-server password policy for users:

```
$ dsconfig \
 get-password-policy-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --advanced \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Property                                  : Value(s)
------------------------------------------:-----------------------------------
account-status-notification-handler       : -
allow-expired-password-changes            : false
allow-multiple-password-values            : false
allow-pre-encoded-passwords               : false
allow-user-password-changes               : true
default-password-storage-scheme           : PBKDF2-HMAC-SHA256
deprecated-password-storage-scheme        : -
expire-passwords-without-warning          : false
force-change-on-add                       : false
force-change-on-reset                     : false
grace-login-count                         : 0
idle-lockout-interval                     : 0 s
java-class                                : org.opends.server.core.PasswordPoli
                                          : cyFactory
last-login-time-attribute                 : -
last-login-time-format                    : -
lockout-duration                          : 0 s
lockout-failure-count                     : 0
lockout-failure-expiration-interval       : 0 s
max-password-age                          : 0 s
max-password-reset-age                    : 0 s
min-password-age                          : 0 s
password-attribute                        : userPassword
password-change-requires-current-password : false
password-expiration-warning-interval      : 5 d
password-generator                        : Random Password Generator
password-history-count                    : 0
password-history-duration                 : 0 s
password-validator                        : At least 8 characters, Common
                                          : passwords
previous-last-login-time-format           : -
require-change-by-time                    : -
require-secure-authentication             : true
require-secure-password-changes           : true
skip-validation-for-administrators        : false
state-update-failure-policy               : reactive
```

For detailed descriptions of each property, see Password Policy.

These settings are configured by default:

- When granted access, users can change their passwords.
- DS servers use the standard `userPassword` attribute to store passwords.

DS servers also support the alternative standard `authPassword` attribute.

- When you import LDIF with `userPassword` values, DS servers apply a one-way hash to the passwords before storing them.

  When a user provides a password value during a bind, for example, the server hashes the incoming password, and compares it with the stored value. This mechanism helps prevent even the directory superuser from recovering the plain text password:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 userpassword

dn: uid=bjensen,ou=People,dc=example,dc=com
userpassword: {PBKDF2-HMAC-SHA256}10:<hash>
```

- The server can set a random password when a password administrator resets a user's password.

Many capabilities are not set by default:

- No lockout.
- No password expiration.
- No password validator to check that passwords contain the appropriate mix of characters.

If the directory service enforces password policy, configure at least the default password policy accordingly.

DS subentry password policies

You manage password policies as LDAP subentries in the application data. Replication applies updates to subentry password policies to all other replicas. Password policy administrators do not need access to the server configuration.

The DS subentry password policy entries have the object classes:

- `ds-pwp-password-policy` for most password policy features.
- A set of password validator object classes for specific validators that derive from the abstract `ds-pwp-validator` class for password validation configuration.
- `ds-pwp-random-generator` for password generation on reset.

The following tables describe password policy attributes per object class:

ds-pwp-password-policy Attributes

Attribute	Description
`ds-pwp-password-attribute` (required)	The attribute type used to hold user passwords.
`ds-pwp-default-password-storage-scheme` (required)	Names of enabled password storage schemes used to encode plaintext passwords. Default: PBKDF2-HMAC-SHA256.
`cn`	Name of the password policy
`ds-pwp-allow-user-password-changes`	Whether users can change their passwords, assuming access control allows it. Default: true.
`ds-pwp-account-status-notification-handler`	Names of enabled account status notification handlers to use with this policy. Use the `dsconfig list-account-status-notification-handlers` command. The first column of the output shows the names. The third column shows whether the handler is enabled.
`ds-pwp-allow-expired-password-changes`	Whether the user can change an expired password with the password modify extended operation. Default: false.
`ds-pwp-allow-multiple-password-values`	Whether user entries can have multiple distinct passwords. Any password is sufficient to authenticate. Default: false.
`ds-pwp-allow-pre-encoded-passwords`	Whether users can change their passwords by providing a pre-encoded value. Default: false.
`ds-pwp-deprecated-password-storage-scheme`	Names of deprecated password storage schemes for this policy. On successful authentication, encode the password with the default.
`ds-pwp-expire-passwords-without-warning`	Whether to allow a user's password to expire even if that user has never seen an expiration warning notification. Default: false.
`ds-pwp-force-change-on-add`	Whether users are forced to change their passwords upon first authentication after their accounts are added. Default: false.

Attribute	Description
`ds-pwp-force-change-on-reset`	Whether users are forced to change their passwords after password reset by an administrator. For this purpose, anyone with permission to change a given user's password other than that user is an administrator. Default: false.
`ds-pwp-grace-login-count`	Number of grace logins that a user is allowed after the account has expired so the user can update their password. Default: 0 (disabled).
`ds-pwp-idle-lockout-interval`	Maximum number of seconds that an account may remain idle (the associated user does not authenticate to the server) before that user is locked out. Requires maintaining a last login time attribute. Default: 0 seconds (inactive).
`ds-pwp-last-login-time-attribute`	Name or OID of the attribute type that is used to hold the last login time for users. Default: The `last-login-time-attribute` setting from the default password policy. By default, `last-login-time-attribute` is not set.
`ds-pwp-last-login-time-format`	Format string that is used to generate the last login time value for users. The format string must match the syntax of the `ds-pwp-last-login-time-attribute` attribute, and must be a valid format string for the `java.text.SimpleDateFormat` class. Default: `yyyyMMddHHmmss'Z'`.
`ds-pwp-lockout-duration`	Duration that an account is locked after too many authentication failures. Default: 0 seconds (account remains locked until the administrator resets the password).
`ds-pwp-lockout-failure-count`	Maximum number of authentication failures that a user is allowed before the account is locked out. Default: 0 (disabled).
`ds-pwp-lockout-failure-expiration-interval`	Duration before an authentication failure is no longer counted against a user for the purposes of account lockout. Default: 0 seconds (never expire).
`ds-pwp-max-password-age`	Duration that a user can continue using the same password before it must be changed (the password expiration interval). Default: 0 seconds (passwords never expire).

Attribute	Description
`ds-pwp-max-password-reset-age`	Maximum number of seconds that users have to change passwords after they have been reset by an administrator before they become locked. Default: 0 seconds.
`ds-pwp-min-password-age`	Minimum duration after a password change before the user is allowed to change the password again. Default: 0 seconds.
`ds-pwp-password-change-requires-current-password`	Whether user password changes must include the user's current password before the change is allowed. This can be done with either the password modify extended operation, or a modify operation using delete and add. Default: false.
`ds-pwp-password-expiration-warning-interval`	Duration before a user's password actually expires that the server begins to include warning notifications in bind responses for that user. Default: 5 days.
`ds-pwp-password-history-count`	Maximum number of former passwords to maintain in the password history. A value of zero indicates that either no password history is to be maintained if the password history duration has a value of zero seconds, or that there is no maximum number of passwords to maintain in the history if the password history duration has a value greater than zero seconds. Default: 0.
`ds-pwp-password-history-duration`	Maximum number of seconds that passwords remain in the password history. Default: 0 seconds (inactive).
`ds-pwp-previous-last-login-time-format`	Format string(s) that might have been used with the last login time at any point in the past for users associated with the password policy. Default: `yyyyMMddHHmmss'Z'`.
`ds-pwp-require-change-by-time`	Time by which all users with the associated password policy must change their passwords. Specified in generalized time form.
`ds-pwp-require-secure-authentication`	Whether users with the associated password policy are required to authenticate in a secure manner. Default: false.

Attribute	Description
`ds-pwp-require-secure-password-changes`	Whether users with the associated password policy are required to change their password in a secure manner that does not expose the credentials. Default: false.
`ds-pwp-skip-validation-for-administrators`	Whether passwords set by administrators are allowed to bypass the password validation process. Default: false.
`ds-pwp-state-update-failure-policy`	How the server deals with the inability to update password policy state information during an authentication attempt. One of the following: • `ignore` : If a bind attempt would otherwise be successful, then do not reject it if a problem occurs while attempting to update the password policy state information for the user. • `proactive` : Proactively reject any bind attempt if it is known ahead of time that it would not be possible to update the user's password policy state information. • `reactive` (default): Even if a bind attempt would otherwise be successful, reject it if a problem occurs while attempting to update the password policy state information for the user.

ds-pwp-attribute-value-validator Attributes

Attribute	Description
`ds-pwp-attribute-value-test-reversed-password`	Whether this password validator should test the reversed value of the provided password as well as the order in which it was given. Default: false.
`ds-pwp-attribute-value-match-attribute`	Name(s) of the attribute(s) whose values should be checked to determine whether they match the provided password. If no values are provided, then the server checks if the proposed password matches the value of any user attribute in the user's entry. The server does not check values of operational attributes.

`ds-pwp-attribute-value-check-substrings`

Whether this password validator is to match portions of the password string against attribute values.
When false, the server checks whether the entire password matches any user attribute values. When true, the server checks whether the password contains any user attribute values.
Consider the case of Babs Jensen (`uid: bjensen`) changing her password. The following table describes the effects of the settings:

Setting	New Password	Password Modification Result
`ds-pwp-attribute-value-check-substrings: false`	`bjense`	Success
`ds-pwp-attribute-value-check-substrings: false`	`bjensen`	Failure: 19 (Constraint Violation)
`ds-pwp-attribute-value-check-substrings: false`	`bjensens`	Success
`ds-pwp-attribute-value-check-substrings: true`	`bjense`	Success
`ds-pwp-attribute-value-check-substrings: true`	`bjensen`	Failure: 19 (Constraint Violation)
`ds-pwp-attribute-value-check-substrings: true`	`bjensens`	Failure: 19 (Constraint Violation)

Attribute	Description
	In summary: • `bjense` is allowed in both cases because the password does not contain any of the attribute values in Babs's entry. • `bjensen` is rejected in both cases because the password exactly matches and contains Babs's user ID. • `bjensens` is allowed when the setting is false because the password does not exactly match, and rejected when the setting is true because the password contains Babs's user ID. Default: false.
`ds-pwp-attribute-value-min-substring-length`	The minimal length of the substring within the password when substring checking is enabled. Default: 0.

ds-pwp-character-set-validator Attributes

Attribute	Description
`ds-pwp-character-set-allow-unclassified-characters`	Whether this password validator allows passwords to contain characters outside of any of the user-defined character sets and ranges. Default: false.
`ds-pwp-character-set-min-character-sets`	Minimum number of character sets and ranges that a password must contain. Use in conjunction with optional character sets and ranges (those requiring zero characters). The value must include any mandatory character sets and ranges (those requiring greater than zero characters). This is useful in situations where a password must contain characters from mandatory character sets and ranges, and characters from at least N optional character sets and ranges. For example, it is quite common to require that a password contains at least one non-alphanumeric character as well as characters from two alphanumeric character sets (lower-case, upper-case, digits). In this case, this property should be set to 3.

Attribute	Description
`ds-pwp-character-set-character-set`	A character set containing characters that a password may contain, and a value indicating the minimum number of characters required from that set. Each value must be an integer (indicating the minimum required characters from the set which may be zero, indicating that the character set is optional) followed by a colon and the characters to include in that set. For example, `3:abcdefghijklmnopqrstuvwxyz` indicates that a user password must contain at least three characters from the set of lowercase ASCII letters. Multiple character sets can be defined in separate values, although no character can appear in more than one character set.
`ds-pwp-character-set-character-set-ranges`	A character range containing characters that a password may contain, and a value indicating the minimum number of characters required from that range. Each value must be an integer (indicating the minimum required characters from the range which may be zero, indicating that the character range is optional) followed by a colon and one or more range specifications. A range specification is 3 characters: the first character allowed, a minus, and the last character allowed. For example, `3:A-Za-z0-9`. The ranges in each value should not overlap, and the characters in each range specification should be ordered.

ds-pwp-dictionary-validator Attributes

Attribute	Description
`ds-pwp-dictionary-data` (required)	A gzipped password dictionary, one word per line. This is a single-valued attribute.
`ds-pwp-dictionary-case-sensitive-validation`	Whether this password validator should treat password characters in a case-sensitive manner. Default: false.
`ds-pwp-dictionary-check-substrings`	Whether this password validator is to match portions of the password string against dictionary words. Default: false (match only the entire password against dictionary words).
`ds-pwp-dictionary-min-substring-length`	The minimal length of the substring within the password in case substring checking is enabled. Default: 0.

Attribute	Description
`ds-pwp-dictionary-test-reversed-password`	Whether this password validator should test the reversed value of the provided password as well as the order in which it was given. Default: false.

ds-pwp-length-based-validator Attributes

Attribute	Description
`ds-pwp-length-based-max-password-length`	Minimum plaintext password length. Default: 0 (undefined).
`ds-pwp-length-based-min-password-length`	Minimum plaintext password length. Default: 6.

ds-pwp-repeated-characters-validator Attributes

Attribute	Description
`ds-pwp-repeated-characters-max-consecutive-length`	The maximum number of times that any character can appear consecutively in a password value. Default: 0 (no maximum limit is enforced).
`ds-pwp-repeated-characters-case-sensitive-validation`	Whether this password validator should treat password characters in a case-sensitive manner. Default: false.

ds-pwp-similarity-based-validator Attributes

Attribute	Description
`ds-pwp-similarity-based-min-password-difference`	The minimum difference the new and old password. The implementation uses the Levenshtein Distance algorithm to determine the minimum number of changes (where a change may be inserting, deleting, or replacing a character) to transform one string into the other. It can prevent users from making only minor changes to their current password when setting a new password. Note that for this password validator to be effective, it must have access to the user's current password. Therefore, if this password validator is to be enabled, also set `ds-pwp-password-change-requires-current-password: true`. Default: 0 (no difference between passwords is acceptable).

ds-pwp-unique-characters-validator Attributes

Attribute	Description
`ds-pwp-unique-characters-case-sensitive-validation`	Whether this password validator should treat password characters in a case-sensitive manner. Default: false.
`ds-pwp-unique-characters-min-unique-characters`	The minimum number of unique characters that a password will be allowed to contain. Default: 0 (no minimum value is enforced).

ds-pwp-random-generator Attributes

Attribute	Description
`ds-pwp-random-password-character-set` (required)	Named character sets. The format of the character set is the name of the set followed by a colon and the characters that are in that set. For example, the value `alpha:abcdefghijklmnopqrstuvwxyz` defines a character set named `alpha` containing all of the lower-case ASCII alphabetic characters.
`ds-pwp-random-password-format` (required)	The format to use for the generated password. The value is a comma-delimited list of elements in which each of those elements is comprised of the name of a character set defined in the password-character-set property, a colon, and the number of characters to include from that set. For example, a value of `alpha:3,numeric:2,alpha:3` generates an 8-character password in which the first three characters are from the `alpha` set, the next two are from the `numeric` set, and the final three are from the `alpha` set.

Interoperable subentry password policies

DS servers support the Internet-Draft, Password Policy for LDAP Directories (version 09). The password policies are expressed as LDAP subentries with `objectClass: pwdPolicy`. An Internet-Draft password policy effectively overrides settings in the default per-server password policy for users, inheriting settings that it does not support or does not include from the per-server password policy.

The following table describes Internet-Draft policy attributes:

pwdPolicy Attributes

Attribute	Description
`pwdAttribute` (required)	The attribute type used to hold user passwords.

Attribute	Description
`pwdAllowUserChange`	Whether users can change their passwords. Default: true.
`pwdExpireWarning`	Maximum number of seconds before a user's password actually expires that the server begins to include warning notifications in bind responses for that user. Default: 432000 seconds.
`pwdFailureCountInterval`	Length of time before an authentication failure is no longer counted against a user for the purposes of account lockout. Default: 0 seconds (never expire).
`pwdGraceAuthNLimit`	Number of grace logins that a user is allowed after the account has expired so the user can update their password. Default: 0 (disabled).
`pwdInHistory`	Maximum number of former passwords to maintain in the password history. Default: 0 (disabled).
`pwdLockoutDuration`	Number of seconds that an account is locked after too many authentication failures. Default: 0 seconds (account remains locked indefinitely).
`pwdMaxAge`	Maximum number of seconds that a user can continue using the same password before it must be changed (the password expiration interval). Default: 0 seconds (disabled).
`pwdMaxFailure`	Maximum number of authentication failures that a user is allowed before the account is locked out. Default: 0.
`pwdMinAge`	Minimum number of seconds after a password change before the user is allowed to change the password again. Default: 0 seconds (disabled).
`pwdMustChange`	Whether users are forced to change their passwords after password reset by an administrator. Default: false.
`pwdSafeModify`	Whether user password changes must use the password modify extended operation, and must include the user's current password before the change is allowed. Default: false.

The following table lists Internet-Draft policy attributes that override the per-server policy properties:

Internet-Draft Policy Attribute	Overrides This Server Policy Property
`pwdAllowUserChange`	`allow-user-password-changes`
`pwdMustChange`	`force-change-on-reset`
`pwdGraceAuthNLimit`	`grace-login-count`
`pwdLockoutDuration`	`lockout-duration`
`pwdMaxFailure`	`lockout-failure-count`
`pwdFailureCountInterval`	`lockout-failure-expiration-interval`
`pwdMaxAge`	`max-password-age`
`pwdMinAge`	`min-password-age`
`pwdAttribute`	`password-attribute`
`pwdSafeModify`	`password-change-requires-current-password`
`pwdExpireWarning`	`password-expiration-warning-interval`
`pwdInHistory`	`password-history-count`

DS servers *ignore* the following Internet-Draft password policy attributes:

- `pwdCheckQuality`, because DS servers have password validators.
- `pwdMinLength`, because you can use a length-based password validator instead.
- `pwdLockout`, because DS servers use other lockout-related password policy attributes.

Internet-Draft based password policies inherit these settings from the default per-server policy for users:

- `account-status-notification-handlers`
- `allow-expired-password-changes`
- `allow-multiple-password-values`
- `allow-pre-encoded-passwords`
- `default-password-storage-schemes`
- `deprecated-password-storage-schemes`
- `expire-passwords-without-warning`
- `force-change-on-add`
- `idle-lockout-interval`

- `last-login-time-attribute`
- `last-login-time-format`
- `max-password-reset-age`
- `password-generator`
- `password-history-duration`
- `password-validators`
- `previous-last-login-time-formats`
- `require-change-by-time`
- `require-secure-authentication`
- `require-secure-password-changes`
- `skip-validation-for-administrators`
- `state-update-failure-policy`

Administrative roles

The server setup process creates one directory superuser account. The directory superuser has unrestricted access to manage the directory service and data.

For any directory service with more than one administrator, one account is not enough. Instead, grant appropriate access to each administrator, based on their duties.

Administrative access

Only the directory superuser should have all the default access and privileges of that account. Directory service administrators should have limited access, as outlined in the following table:

Tasks	Required Access(1) and Privileges(2)
Install and upgrade servers	Access to file system. Access to run server commands.
Delegate administration	Access to write administration-related attributes on others' entries. DS server privileges: `config-read`, `config-write`, `modify-acl`, `privilege-change`.
Manage server processes (start, restart, stop)	Access to run the `start-ds` and `stop-ds` commands. DS server privileges: `server-restart`, `server-shutdown`.

Tasks	Required Access[1] and Privileges[2]
Manage changes to server configuration, including global and default settings	Access to read and write to `cn=config`, `cn=schema`, and potentially other administrative DNs, such as `cn=tasks`. DS server privileges: `config-read`, `config-write`, `modify-acl` (for global ACIs), `update-schema`.
Manage containers for user data, including backends and indexes	File system access for backup data and exported LDIF. Access to create entries under `cn=tasks`. DS server privileges: `backend-backup`, `backend-restore`, `config-read`, `config-write`, `ldif-export`, `ldif-import`, `modify-acl` (for ACIs in user data), `subentry-write`.
Manage changes to server schemas	Access to write to `cn=schema`. File system access to add schema files. DS server privilege: `update-schema`.
Manage directory server data replication	File system access to read `logs/replication`. Access to read and write to `cn=admin data`, if any password policies configure a reversible password storage scheme. Access to run the `dsrepl` command. DS server privileges: `changelog-read`, `config-read`, `data-sync`.
Monitor the directory service	Access to read `cn=monitor`. DS server privileges: `monitor-read`, `jmx-notify`, `jmx-read`, `jmx-write` (the last three being useful when using JMX for monitoring).
Back up and restore directory data	File system access for backup data and exported LDIF. Access to create entries under `cn=tasks`. DS server privileges: `backend-backup`, `backend-restore`, `ldif-export`, `ldif-import`.
Troubleshoot problems with the directory service	File system access to read log messages. Write access to create entries under `cn=tasks`. Access to read `cn=monitor`. DS server privileges: `bypass-lockdown`, `cancel-request`, `config-read`, `config-write` (to enable debug logging, for example), `disconnect-client`, `monitor-read`, `server-lockdown`.

[1] Access control is covered in Access control.

[2] Privileges are covered in Administrative privileges.

Directory data administrators should have limited access, as outlined in the following table:

Tasks	Required Access[1] and Privileges[2]
Manage changes to users, groups, and other accounts for their organizations	Access to read and write to others' entries.

Tasks	Required Access[1] and Privileges[2]
Delegate administration within their organizations	Access to write administration-related attributes on others' entries. DS server privileges: `modify-acl`, `privilege-change`.
Update administrative user data, such as subentry password policies and access controls	Access to write administration-related attributes on others' entries. DS server privileges: `modify-acl`, `subentry-write`.
Help users who are locked out, or have forgotten or lost their password	Access to use the `manage-account` command. Access to request a password modify extended operation. Access to update passwords on user entries. DS server privilege: `password-reset`.
Assist users and application developers who access the directory service	Access to read (and potentially write to) others' entries. If performing operations on behalf of other users, access to request proxied authorization. DS server privilege: `proxied-auth`.

[1] Access control is covered in Access control.

[2] Privileges are covered in Administrative privileges.

Administrative accounts

> **Tip**
>
> Limit use of the `uid=admin` superuser account.
> To bootstrap the system, the default directory superuser account is not subject to access control. It has privileges to perform almost every administrative operation, including increasing its own privileges.
> Treat this account as you would the UNIX `root` account, or the Windows `Administrator` account. Use it only when you must.

Use a non-default superuser account

The following steps replace the directory superuser account:

1. Create an administrator account that duplicates the default directory superuser account.

   The default administrator account is `uid=admin`. It is stored in its own backend, `rootUser`. The `rootUser` LDIF backend holds the file `db/rootUser.ldif`.

   The following example shows the LDIF for an alternative directory superuser:

```
dn: uid=altadmin
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Alternative Superuser
givenName: Alternative
sn: Superuser
ds-rlim-size-limit: 0
ds-rlim-time-limit: 0
ds-rlim-idle-time-limit: 0
ds-rlim-max-candidate-set-size: 100000
ds-pwp-password-policy-dn: cn=Root Password Policy,cn=Password Policies,cn=config
ds-privilege-name: bypass-lockdown
ds-privilege-name: bypass-acl
ds-privilege-name: modify-acl
ds-privilege-name: config-read
ds-privilege-name: config-write
ds-privilege-name: ldif-import
ds-privilege-name: ldif-export
ds-privilege-name: backend-backup
ds-privilege-name: backend-restore
ds-privilege-name: server-lockdown
ds-privilege-name: server-shutdown
ds-privilege-name: server-restart
ds-privilege-name: disconnect-client
ds-privilege-name: cancel-request
ds-privilege-name: password-reset
ds-privilege-name: update-schema
ds-privilege-name: privilege-change
ds-privilege-name: unindexed-search
ds-privilege-name: subentry-write
ds-privilege-name: changelog-read
ds-privilege-name: monitor-read
uid: altadmin
userPassword: password
```

Do not use `altadmin`, since it shows up here in the documentation.

2. Create a private backend to store the new directory superuser account.

   The following example creates an LDIF backend to store the entry. Before creating the backend, create a separate directory to hold the backend files:

```
$ mkdir /path/to/opendj/db/altRootUser

$ dsconfig \
 create-backend \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name altRootUser \
 --type ldif \
 --set enabled:true \
 --set base-dn:uid=altadmin \
 --set ldif-file:db/altRootUser/altRootUser.ldif \
 --set is-private-backend:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. Import the new directory superuser entry into the new backend:

```
$ import-ldif \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backendID altRootUser \
 --ldifFile /tmp/alt-root.ldif \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

4. Remove the backend database and the unused LDIF files for the default superuser account:

```
$ dsconfig \
 delete-backend \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name rootUser \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

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

   In this example, `uid=altadmin` now has the same rights as the original superuser.

5. Repeat these steps for other DS servers.

   The account is not replicated by default.

Make users administrators

You can assign access and privileges to any directory account:

1. Adjust the resource limits to as needed.

   For details, see Resource limits.

2. Assign any necessary administrative privileges.

   For details, see Administrative privileges.

3. Add any necessary ACIs.

   For details, see Access control.

Administrative privileges

Privileges provide access control for server administration independently from ACIs. The default directory superuser, `uid=admin`, is granted the privileges marked with an asterisk (*):

Privilege	Description
`backend-backup` *	Request a task to back up data, or to purge backup files.
`backend-restore` *	Request a task to restore data from backup.
`bypass-acl` *	Perform operations without regard to ACIs.
`bypass-lockdown` *	Perform operations without regard to lockdown mode.
`cancel-request` *	Cancel any client request.
`changelog-read` *	Read the changelog (under `cn=changelog`).
`config-read` *	Read the server configuration.
`config-write` *	Change the server configuration.
`data-sync`	Perform data synchronization.
`disconnect-client` *	Close any client connection.
`jmx-notify`	Subscribe to JMX notifications.
`jmx-read`	Read JMX attribute values.
`jmx-write`	Write JMX attribute values.
`ldif-export` *	Export data to LDIF.

Privilege	Description
`ldif-import` *	Import data from LDIF.
`modify-acl` *	Change ACIs.
`monitor-read` *	Read metrics under `cn=monitor`, `/metrics/api`, `/metrics/prometheus`, and over JMX.
`password-reset` *	Reset other users' passwords.
`privilege-change` *	Change the privileges assigned to users, including their own privileges.
`proxied-auth`	Use the LDAP proxied authorization control.
`server-lockdown` *	Put the server into and take the server out of lockdown mode.
`server-restart` *	Request a task to restart the server.
`server-shutdown` *	Request a task to stop the server.
`subentry-write` *	Perform LDAP subentry write operations.
`unindexed-search` *	Search using a filter with no corresponding index.
`update-schema` *	Change LDAP schema definitions.

Assign individual account privileges

Specify privileges as values of the `ds-privilege-name` operational attribute:

1. Determine the privileges to add.

   Kirsten Vaughan has access to modify user entries. Kirsten lacks privileges to read the server configuration, and reset user passwords:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN cn=config \
 "(objectclass=*)"

# The LDAP search request failed: 50 (Insufficient Access Rights)
# Additional Information:  You do not have sufficient privileges to perform search operations in the Directory
Server configuration

$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --authzID "dn:uid=scarter,ou=People,dc=example,dc=com" \
 --newPassword chngthspwd

The LDAP password modify operation failed: 50 (Insufficient Access Rights)
Additional Information:  You do not have sufficient privileges to perform
password reset operations
```

2. Apply the change as a user with the `privilege-change` privilege, and give Kirsten access to read `cn=config`:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: config-read
ds-privilege-name: password-reset
EOF

$ dsconfig \
 set-access-control-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --add "global-aci:(target=\"ldap:///cn=config\")(targetattr=\"*||+\")\
(version 3.0; acl \"Config read for Kirsten Vaughan\"; allow (read,search,compare)\
userdn=\"ldap:///uid=kvaughan,ou=People,dc=example,dc=com\";)" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Kirsten can perform the operations now:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN cn=config \
 "(objectclass=*)"

dn: cn=config

$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --authzID "dn:uid=scarter,ou=People,dc=example,dc=com" \
 --newPassword chngthspwd

The LDAP password modify operation was successful
```

Assign group privileges

Use a collective attribute subentry to assign privileges to a group:

1. Create an LDAP subentry that specifies the collective attributes:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Administrator Privileges,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: Administrator Privileges
ds-privilege-name;collective: config-read
ds-privilege-name;collective: config-write
ds-privilege-name;collective: ldif-export
ds-privilege-name;collective: modify-acl
ds-privilege-name;collective: password-reset
ds-privilege-name;collective: proxied-auth
subtreeSpecification: {base "ou=people", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)" }
EOF
```

   For details, see Collective attributes, and About Subentry Scope.

2. The change takes effect immediately:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=hmiller,ou=people,dc=example,dc=com" \
 --bindPassword hillock \
 --authzID "dn:uid=scarter,ou=People,dc=example,dc=com"

The LDAP password modify operation was successful
Generated Password:  <password>
```

Limit inherited privileges

Privileges assigned by collective attributes are inherited by every target account. To limit effective privileges, override the privilege in the account by preceding the privilege attribute value with a `-`.

This examples shows how to prevent Kirsten Vaughan from using the privilege to reset passwords:

1. Check the privilege settings for the account:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(uid=kvaughan)" \
 ds-privilege-name

dn: uid=kvaughan,ou=People,dc=example,dc=com
ds-privilege-name: config-read
ds-privilege-name: password-reset
```

2. Use the override to deny the privilege:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=kvaughan,ou=people,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: -password-reset
EOF
```

3. The change takes effect immediately:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --authzID "dn:uid=scarter,ou=People,dc=example,dc=com"

The LDAP password modify operation failed: 50 (Insufficient Access Rights)
Additional Information:  You do not have sufficient privileges to perform
password reset operations
```

Identity management

DS software lets you manage user and administrator accounts individually or in groups. If the deployment calls for provisioning and workflow capabilities, or custom tools, then consider using identity management software, such as ForgeRock Identity Management. For details, see the IDM documentation.

Simple or local deployments might require a GUI for generic or system-specific administrative operations. DS software's support for standards like LDAP v3 makes it interoperable with many third-party tools.

Access control

The DS evaluation setup profile leaves access more open, especially to sample Example.com data. This makes it easy to demonstrate and learn features before you fully understand access control. When deploying DS servers in production, grant only the necessary access.

Access control mechanisms

DS servers support two access control mechanisms, ACIs for directory servers, and global access control policies for proxy servers.

Characteristic	Access Control Instructions (ACIs)	Global Access Control Policies
Default for	Directory Servers.	Directory Proxy Servers.
Where	Operational `aci` attributes (replicated). `global-aci` properties (not replicated).	`global-access-control-policy` entries (not replicated).
Default access	No access unless explicitly granted.(1)	No access unless explicitly granted.
Level of control	Very fine-grained control that can depend on the directory data.	Overall control that does not require access to directory data.
Interaction(2)	When configured, global policies have no effect.	When configured, ACIs have no effect.
Reference	Directory server ACIs. DSEE Compatible Access Control Handler.	Global Access Control Policy.

(1) The `bypass-acl` privilege grants users access regardless of ACIs.

(2) In the rare event that you choose to change the type of server and the type of its access control handler, you must stop the server and make the change with the `dsconfig --offline` command.

Some operations require administrative privileges *and* access control. By combining access control and privileges, you effectively restrict the scope of the privileges. Privileges are described in Administrative roles.

Directory server ACIs

- ACIs set scoped permissions which depend on what operation is requested, who requested the operation, and how the client connected to the server.
- To let other users change ACIs, grant them the `modify-acl` privilege and permission to edit `aci` attributes.

  For examples, see Learn access control.

ACI syntax

```
targets (version 3.0; acl "name"; permissions subjects;)
```

targets

The ACI applies to the target entries, attributes, controls, and extended operations.

To define multiple *targets*, put each target in parentheses, `()`. All targets must match for the ACI to apply (`AND`).

name

Human-readable description of what the ACI does.

permissions

Actions to allow, and which to deny.

Paired with subjects.

subjects

Clients the permissions apply to, and the conditions under which they apply.

Paired with permissions.

Separate multiple permissions-subjects pairs with semicolons, `;`. At least one must match for the ACI to apply (`OR`).

ACI targets

Most target expressions let you use either `=` (target must match), or `!=` (target must not match):

`(target [!]= "ldap:///DN")`

The ACI scope is the entry with distinguished name DN, and subordinates.

Use an asterisk, `\*`, to replace attribute types, attribute values, and entire DN components. The following example targets `uid=bjensen,ou=People,dc=example,dc=com` and `cn=My App,ou=Apps,dc=example,dc=com`:

```
(target = "ldap:///*=*,*,dc=example,dc=com")
```

The DN must be in the subtree of the entry where the ACI is defined.

If you omit `target`, the ACI applies to its entry.

If you omit `targetscope` as well, the ACI applies to its entry and all subordinates.

`(targetattr [!]= "attr-list")`

The ACI targets the specified attributes.

In the attr-list, separate attribute names with `||`.

This ACI affects its entry, or the entries specified by other targets in the ACI.

For best performance, explicitly list attributes. Use an asterisk, `\*`, to specify all user attributes. Use a plus sign, `\+`, to specify all operational attributes.

A negated attr-list of operational attributes matches only other operational attributes, never any user attributes, and vice-versa.

If you omit `targetattr`, by default this ACI does not affect attributes.

`(targetfilter [!]= "ldap-filter")`

This ACI is scoped to match the ldap-filter dynamically, as in an LDAP search. The ldap-filter can be any valid LDAP filter.

`(targattrfilters = "expression")`

Use this target specification when managing changes made to particular attributes.

The expression takes one of the following forms. Separate expressions with commas (`,`):

```
op=attr1:filter1[&& attr2:filter2 ...][,op=attr3:filter3[&& attr4:filter4 ...] ...]
```

The op can be either `add` for operations creating attributes, or `del` for operations removing them.

Replace attr with an attribute type. Replace filter with an LDAP filter that corresponds to the attr attribute type.

`(targetscope = "base|onelevel|subtree|subordinate")`

- `base` refers to the entry with the ACI.
- `onelevel` refers to immediate children.
- `subtree` refers to the base entry and all children.
- `subordinate` refers to all children only. If you omit `targetscope`, the default is `subtree`.

`(targetcontrol [!]= "alias-or-OID")`

The ACI targets the LDAP control with the specified alias or object identifier alias-or-OID. Separate multiple aliases or OIDs with `||`.

DS servers support the following control aliases for ACIs:

- `AccountUsable`, `AccountUsability` (`1.3.6.1.4.1.42.2.27.9.5.8`)
- `ActiveDirectoryChangeNotification`, `AdChangeNotification` (`1.2.840.113556.1.4.528`)
- `Affinity` (`1.3.6.1.4.1.36733.2.1.5.2`)
- `Assertion`, `LdapAssertion` (`1.3.6.1.1.12`)

- `AuthorizationIdentity`, `AuthzId` (`2.16.840.1.113730.3.4.16`)
- `Csn`, `ChangeNumber`, `ChangeSequenceNumber` (`1.3.6.1.4.1.42.2.27.9.5.9`)
- `Ecl`, `EclCookie`, `ExternalChangelogCookie` (`1.3.6.1.4.1.26027.1.5.4`)
- `EffectiveRights`, `GetEffectiveRights` (`1.3.6.1.4.1.42.2.27.9.5.2`)
- `ManageDsaIt` (`2.16.840.1.113730.3.4.2`)
- `MatchedValues` (`1.2.826.0.1.3344810.2.3`)
- `NoOp` (`1.3.6.1.4.1.4203.1.10.2`)
- `PasswordPolicy`, `PwdPolicy`, `PwpPolicy` (`1.3.6.1.4.1.42.2.27.8.5.1`)
- `PasswordQualityAdvice` (`1.3.6.1.4.1.36733.2.1.5.5`)
- `PermissiveModify` (`1.2.840.113556.1.4.1413`)
- `PersistentSearch`, `PSearch` (`2.16.840.1.113730.3.4.3`)
- `PostRead` (`1.3.6.1.1.13.2`)
- `PreRead` (`1.3.6.1.1.13.1`)
- `ProxiedAuthV1` (`2.16.840.1.113730.3.4.12`)
- `ProxiedAuthV2`, `ProxiedAuth` (`2.16.840.1.113730.3.4.18`)
- `RealAttrsOnly`, `RealAttributesOnly` (`2.16.840.1.113730.3.4.17`)
- `RelaxRules` (`1.3.6.1.4.1.4203.666.5.12`)
- `ReplicationRepair` (`1.3.6.1.4.1.26027.1.5.2`)
- `ServerSideSort`, `Sort` (`1.2.840.113556.1.4.473`)
- `SimplePagedResults`, `PagedResults` (`1.2.840.113556.1.4.319`)
- `SubEntries` (`1.3.6.1.4.1.4203.1.10.1`)
- `SubtreeDelete`, `TreeDelete` (`1.2.840.113556.1.4.805`)
- `TransactionId`, `TxnId` (`1.3.6.1.4.1.36733.2.1.5.1`)
- `VirtualAttrsOnly`, `VirtualAttributesOnly` (`2.16.840.1.113730.3.4.19`)
- `Vlv`, `VirtualListView` (`2.16.840.1.113730.3.4.9`)

To use an LDAP control, the bind DN user must have `allow(read)` permissions. This target cannot be restricted to a specific subtree.

`(extop [!]= "alias-or-OID")`

This ACI targets the LDAP extended operation with the specified alias or object identifier alias-or-OID. Separate multiple aliases or OIDs with `||`.

DS servers support the following extended operation aliases for ACIs:

- `Cancel` (`1.3.6.1.1.8`)
- `GetConnectionId`, `ConnectionId` (`1.3.6.1.4.1.26027.1.6.2`)
- `GetSymmetricKey`, `SymmetricKey` (`1.3.6.1.4.1.26027.1.6.3`)
- `PasswordModify` (`1.3.6.1.4.1.4203.1.11.1`)
- `PasswordPolicyState` (`1.3.6.1.4.1.26027.1.6.1`)
- `StartTls` (`1.3.6.1.4.1.1466.20037`)
- `WhoAmI` (`1.3.6.1.4.1.4203.1.11.3`)

To use an LDAP extended operation, the bind DN user must have `allow(read)` permissions. This target cannot be restricted to a specific subtree.

ACI permissions

ACI permission definitions take one of the following forms:

```
allow(action[, action ...])
deny(action[, action ...])
```

> **Tip**
>
> Avoid using `deny`.
> Instead, explicitly `allow` access only as needed. What looks harmless and simple in tests and examples can grow complicated quickly with nested ACIs.

The action is one of the following:

`add`

Entry creation, as for an LDAP add operation.

`all`

All permissions, except `export`, `import`, `proxy`.

`compare`

Attribute value comparison, as for an LDAP compare operation.

`delete`

Entry deletion, as for an LDAP delete operation.

`export`

Entry export during a modify DN operation.

Despite the name, this action is unrelated to LDIF export operations.

`import`

Entry import during a modify DN operation.

Despite the name, this action is unrelated to LDIF import operations.

`proxy`

Access the ACI target using the rights of another user.

`read`

Read entries and attributes, or use an LDAP control or extended operation.

`search`

Search the ACI targets.

Combine with `read` to read the search results.

`selfwrite`

Add or delete own DN from a group.

`write`

Modify attributes on ACI target entries.

ACI subjects

Subjects restrict whether the ACI applies depending on who connected, and when, where, and how they connected.

Most target expressions allow you to use either `=` (condition must match), or `!=` (condition must not match):

`authmethod [!]= "none|simple|ssl|sasl mech"`

- `none`: ignore the authentication method.
- `simple`: simple authentication.
- `ssl`: certificate-based authentication over LDAPS.
- `sasl mech`: SASL authentication, where *mech* is the SASL mechanism, such as `EXTERNAL`, or `GSSAPI`.

`dayofweek [!]= "day[, day…]"`

Valid days:

- `sun` (Sunday)
- `mon` (Monday)
- `tue` (Tuesday)
- `wed` (Wednesday)
- `thu` (Thursday)

- `fri` (Friday)
- `sat` (Saturday)

`dns [!]= "hostname"`

Use an asterisk, `*`, to replace name components, as in `dns = "*.example.com"`.

`groupdn [!]= "ldap:///DN [|| ldap:///DN …]"`

The subjects are the members of the group with the specified DN.

`ip [!]= "addresses"`

Valid IP addresses:

- Individual IPv4 or IPv6 addresses.

  Put IPv6 addresses in brackets, as in `ldap://[address]/subnet-prefix`, where `/subnet-prefix` is optional.

- Addresses with asterisk (`*`) for a subnet or host number.
- CIDR notation.
- Forms such as `192.168.0.*+255.255.255.0` to specify subnet masks.

`ssf = "strength"`

The security strength factor (`ssf`) reflects the cipher key strength for a secure connection.

The `ssf` takes an integer in the range 0-1024:

- `ssf = 0`: send plain text with no connection security.
- `ssf = 1`: configure TLS without a cipher. The server verifies integrity using packet checksums, but all content is sent in cleartext.
- `ssf >= "256"`: require a cipher strength of at least 256 bits.

The `ssf` setting can help to neutralize STRIPTLS attacks. A TLS stripping attack is a man-in-the-middle attack. It takes advantage of the fact that the initial TLS handshake starts on an unencrypted connection. An attacker who has control of the network makes it appear during the handshake that TLS is not available. Client applications may then fall back to using the connection without TLS encryption. In this case, ACIs with `ssf` settings greater than `1` require encryption to grant access. Use an appropriately high `ssf` setting in your ACIs, such as `ssf >= "256"` to ensure secure encryption.

`timeofday = "hhmm"`

Express times, *hhmm*, as on a 24-hour clock.

For example, 1:15 PM is written `1315`.

`userattr [!]= "attr#value"`

The `userattr` subject specifies an attribute that must match on the bind entry and the ACI target entry:

- Use `userattr [!]= "attr#value"` when the bind entry and target entry have the same attribute. The attr is a user attribute. The value is the attribute value.

The server does an internal search to get the attributes of the bind entry. Therefore, this ACI subject does not work with operational attributes.

- Use `userattr [!]= ldap-url#LDAPURL"` when the target entry is identified by the LDAP URL, and the bind entry is in the subtree scope of the DN in the LDAP URL.
- Use `userattr [!]= "[parent[child-level].]attr#GROUPDN"` when the bind DN is a member of the group identified by the attr of the target entry.
- Use `userattr [!]= "[parent[child-level].]attr#USERDN"` when the bind DN is referenced by the attr of the target entry.

The optional inheritance specification, `parent[child-level].`, defines how many levels below the target entry inherit the ACI. The child-level is a number from 0 to 9, with 0 indicating the target entry only. Separate multiple child-level digits with commas (`,`).

`userdn [!]= "ldap-url+_[|| _ldap-url+ …]"`

This subject matches either a valid LDAP URL, or a special LDAP URL-like keyword from the following list:

`ldap:///all`

Match authenticated users.

`ldap:///anyone`

Match anonymous and authenticated users.

`ldap:///parent`

Match when the bind DN is a parent of the ACI target.

`ldap:///self`

Match when the bind DN entry corresponds to ACI target.

ACI evaluation

The rules the server follows are simple:

1. To determine whether an operation is allowed or denied, DS servers look in the directory for the target of the operation. The server collects any ACI values from that entry, and then walks up the directory tree to the base DN, collecting all ACI values en route. It then collects global ACI values.
2. The server separates the ACI values into two lists. One list contains all the ACI values that match the target and deny the required access. The other list contains all the ACI values that match the target and allow the required access.
3. If the deny list contains any ACI values after this procedure, access is immediately denied.
4. If the deny list is empty, the server processes the allow list. If the allow list contains any ACI values, access is allowed.
5. If both lists are empty, access is denied.

> **Note**
>
> Some operations require multiple permissions and involve multiple targets. Evaluation therefore takes place multiple times.
> For example, a search operation requires the `search` permission for each attribute in the search filter. If applicable ACIs allow all search permissions, the server uses `read` permissions to decide which attributes and values to return.

ACI by operation

Add

The ACI must allow the `add` permission to entries in the target. This implicitly lets users set attributes and values.

Use `targattrfilters` to explicitly deny access to any values if required.

For example, this ACI lets `uid=bjensen,ou=People,dc=example,dc=com` add an entry:

```
aci: (version 3.0;acl "Add entry"; allow (add)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Bind

Because a bind establishes the user's identity and derived authorizations, ACI is irrelevant for this operation and is not checked.

To prevent authentication, disable the account instead.

Compare

The ACI must allow the `compare` permission to the attribute in the target entry.

For example, this ACI lets `uid=bjensen,ou=People,dc=example,dc=com` compare values against the `sn` attribute:

```
aci: (targetattr = "sn")(version 3.0;acl "Compare surname"; allow (compare)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Delete

The ACI must allow the `delete` permission to the target entry. This implicitly lets users delete attributes and values in the target.

Use `targattrfilters` to explicitly deny access to the values if required.

For example, this ACI lets `uid=bjensen,ou=People,dc=example,dc=com` delete an entry:

```
aci: (version 3.0;acl "Delete entry"; allow (delete)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Modify

The ACI must allow the `write` permission to attributes in the target entries. This implicitly lets users modify all values of the target attribute.

Use `targattrfilters` to explicitly deny access to specific values if required.

For example, this ACI lets `uid=bjensen,ou=People,dc=example,dc=com` modify the `description` attribute in an entry:

```
aci: (targetattr = "description")(version 3.0; acl "Modify description";
 allow (write) (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

ModifyDN

If the entry is being moved to a `newSuperior`, the `export` permission must be allowed on the target, and the `import` permission must be allowed on the `newSuperior` entry.

The ACI must allow `write` permission to the attributes in the old RDN and the new RDN. This implicitly lets users write all values of the old RDN and new RDN.

Use `targattrfilters` to explicitly deny access to values used if required.

For example, this ACI lets `uid=bjensen,ou=People,dc=example,dc=com` rename entries named with the `uid` attribute to new locations:

```
aci: (targetattr = "uid")(version 3.0;acl "Rename uid= entries";
 allow (write, import, export)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Search

ACI is required to process the search filter, and to determine which attributes and values the server returns. The `search` permission allows particular attributes in the search filter. The `read` permission allows particular attributes to be returned.

If `read` permission is allowed to any attribute, the server automatically allows reads of the `objectClass` attribute.

For example, this ACI lets `uid=bjensen,ou=People,dc=example,dc=com` search for `uid` attributes, and read that attribute in matching entries:

```
aci: (targetattr = "uid")(version 3.0;acl "Search and read uid";
 allow (search, read) (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Use Control or Extended Operation

The ACI must allow the `read` permission to the `targetcontrol` or `extop` OIDs.

For example, this ACI lets `uid=bjensen,ou=People,dc=example,dc=com` use the Persistent Search request control with OID `2.16.840.1.113730.3.4.3`:

```
aci: (targetcontrol = "PSearch")
 (version 3.0;acl "Request Persistent Search"; allow (read)
 (userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)
```

Default global ACIs

Modifying and removing global ACIs can have deleterious effects. Modifications to global ACIs fall into the following categories:

- Modification or removal is permitted.

  You must test client applications when deleting the specified ACI.

- Modification or removal may affect applications.

  You must test client applications when modifying or deleting the specified ACI.

- Modification or removal may affect applications, but is not recommended.

  You must test client applications when modifying or deleting the specified ACI.

- Do not modify or delete.

Name	Description	ACI definition
Anonymous extended operation access	Anonymous and authenticated users can request the LDAP extended operations that are specified by OID or alias. Modification or removal may affect applications.	`(extop="Cancel||GetSymmetricKey|| PasswordModify||StartTls||WhoAmI") (version 3.0; acl "Anonymous extended operation access"; allow(read) userdn="ldap:/// anyone";)`
Anonymous extended operation access	Anonymous and authenticated users can request the LDAP extended operations that are specified by OID or alias. Modification or removal may affect applications.	`(targetcontrol="Assertion|| AuthorizationIdentity|| MatchedValues||NoOp|| PasswordPolicy|| PasswordQualityAdvice|| PermissiveModify||PostRead|| PreRead||RealAttrsOnly|| SimplePagedResults|| TransactionId||VirtualAttrsOnly|| Vlv") (version 3.0; acl "Anonymous extended operation access"; allow(read) userdn="ldap:/// anyone";)`

Name	Description	ACI definition
Authenticated users extended operation access	Authenticated users can request the LDAP extended operations that are specified by OID or alias. Modification or removal may affect applications.	`(targetcontrol="ManageDsaIt||RelaxRules||ServerSideSort||SubEntries||SubtreeDelete") (version 3.0; acl "Authenticated users extended operation access"; allow(read) userdn="ldap:///all";)`
Authenticated users extended operation access	Authenticated users can request the LDAP extended operations that are specified by OID or alias. Modification or removal may affect applications.	`(extop="PasswordPolicyState") (version 3.0; acl "Authenticated users extended operation access"; allow(read) userdn="ldap:///all";)`
User-Visible Monitor Attributes	Authenticated users can read monitoring information if they have the monitor read privilege. Modification or removal may affect applications.	`(target="ldap:///cn=monitor") (targetattr="*||+") (version 3.0; acl "User-Visible Monitor Attributes"; allow (read,search,compare) userdn="ldap:///all";)`
User-Visible Root DSE Operational Attributes	Anonymous and authenticated users can read attributes that describe what the server supports. Modification or removal may affect applications.	`(target="ldap:///") (targetscope="base") (targetattr="objectClass||namingContexts||subSchemaSubEntry||supportedAuthPasswordSchemes||supportedControl||supportedExtension||supportedFeatures||supportedLDAPVersion||supportedSASLMechanisms||supportedTLSCiphers||supportedTLSProtocols||vendorName||vendorVersion||fullVendorVersion||alive||healthy")(version 3.0; acl "User-Visible Root DSE Operational Attributes"; allow (read,search,compare) userdn="ldap:///anyone";)`

Name	Description	ACI definition
User-Visible Schema Operational Attributes	Authenticated users can read LDAP schema definitions. Modification or removal may affect applications.	`(target="ldap:///cn=schema") (targetscope="base") (targetattr="objectClass|| attributeTypes||dITContentRules|| dITStructureRules ||ldapSyntaxes|| matchingRules||matchingRuleUse|| nameForms||objectClasses||etag|| modifiersName||modifyTimestamp") (version 3.0; acl "User-Visible Schema Operational Attributes"; allow (read,search,compare) userdn="ldap:///all";)`

Effective rights

As the number of ACIs increases, it can be difficult to understand what rights a user actually has. The Get Effective Rights control (OID `1.3.6.1.4.1.42.2.27.9.5.2`) lets you see the rights as evaluated by the server.

By default, only users who can bypass ACIs can use the Get Effective Rights control, and the related operational attributes, `aclRights` and `aclRightsInfo`. The following command grant access to My App:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --add global-aci:\(targetcontrol=\"EffectiveRights\"\)\ \(version\ 3.0\;acl\ \"Allow\ My\ App\ to\ get\ effective\
rights\"\;\ allow\(read\)\ userdn=\"ldap:///cn=My\ App,ou=Apps,dc=example,dc=com\"\;\) \
 --add global-aci:\(targetattr=\"aclRights\|\|aclRightsInfo\"\)\(version\ 3.0\;\ acl\ \"Allow\ My\ App\ to\ read\
effective\ rights\ attributes\"\;\ allow\ \(read,search,compare\)\ userdn=\"ldap:///cn=My\
App,ou=Apps,dc=example,dc=com\"\;\) \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

In this example, Babs Jensen owns the LDAP group that includes people who are willing to carpool:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --baseDN "ou=Self Service,ou=Groups,dc=example,dc=com" \
 "(cn=Carpoolers)"

dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
description: People who are willing to carpool
cn: Carpoolers
owner: uid=bjensen,ou=People,dc=example,dc=com
member: uid=bjensen,ou=People,dc=example,dc=com
```

When My App does the same search with the get effective rights control, and requests the `aclRights` attribute, it sees the rights it has on the entry:

```
$ ldapsearch \
 --control effectiverights \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=Groups,dc=example,dc=com" \
 "(cn=Carpoolers)" \
 aclRights

dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
aclRights;entryLevel: add:1,delete:1,read:1,write:1,proxy:1
```

When My App requests the `aclRightsInfo` attribute, the server shows the ACIs that apply:

```
$ ldapsearch \
 --control effectiverights \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=Groups,dc=example,dc=com" \
 "(cn=Carpoolers)" \
 aclRights \
 aclRightsInfo

dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
aclRights;entryLevel: add:1,delete:1,read:1,write:1,proxy:1
aclRightsInfo;logs;entryLevel;add: acl_summary(main): access allowed(add) on entry/attr(cn=Carpoolers,ou=Self
Service,ou=Groups,dc=example,dc=com, NULL) to (cn=My App,ou=Apps,dc=example,dc=com) (not proxied) ( reason: evaluated
allow , deciding_aci: Proxied authorization for apps)
aclRightsInfo;logs;entryLevel;delete: acl_summary(main): access allowed(delete) on entry/attr(cn=Carpoolers,ou=Self
Service,ou=Groups,dc=example,dc=com, NULL) to (cn=My App,ou=Apps,dc=example,dc=com) (not proxied) ( reason: evaluated
allow , deciding_aci: Proxied authorization for apps)
aclRightsInfo;logs;entryLevel;proxy: acl_summary(main): access allowed(proxy) on entry/attr(cn=Carpoolers,ou=Self
Service,ou=Groups,dc=example,dc=com, NULL) to (cn=My App,ou=Apps,dc=example,dc=com) (not proxied) ( reason: evaluated
allow , deciding_aci: Proxied authorization for apps)
aclRightsInfo;logs;entryLevel;read: acl_summary(main): access allowed(read) on entry/attr(cn=Carpoolers,ou=Self
Service,ou=Groups,dc=example,dc=com, objectClass) to (cn=My App,ou=Apps,dc=example,dc=com) (not proxied) ( reason:
evaluated allow , deciding_aci: Anonymous read and search access)
aclRightsInfo;logs;entryLevel;write: acl_summary(main): access allowed(write) on entry/attr(cn=Carpoolers,ou=Self
Service,ou=Groups,dc=example,dc=com, NULL) to (cn=My App,ou=Apps,dc=example,dc=com) (not proxied) ( reason: evaluated
allow , deciding_aci: Proxied authorization for apps)
```

To request effective rights for another user, use the `--getEffectiveRightsAuthzid` option. This option takes the authorization identity of the user as an argument. The following example shows My App checking Babs's rights to the same entry:

```
$ ldapsearch \
 --getEffectiveRightsAuthzid "dn:uid=bjensen,ou=People,dc=example,dc=com" \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=groups,dc=example,dc=com" \
 "(cn=Carpoolers)" \
 aclRights

dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
aclRights;entryLevel: add:0,delete:1,read:1,write:0,proxy:0
```

The following example checks anonymous user rights to the same entry. Notice that the authorization identity for an anonymous user is expressed as the empty DN:

```
$ ldapsearch \
 --getEffectiveRightsAuthzid "dn:" \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=groups,dc=example,dc=com" \
 "(cn=Carpoolers)" \
 aclRights

dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
aclRights;entryLevel: add:0,delete:0,read:1,write:0,proxy:0
```

To check access to a potentially nonexistent attribute, use the `--getEffectiveRightsAttribute` option. This option takes a comma-separated attribute list as an argument. The following example checks Andy Hall's access to the member attribute for the Carpooler's group entry:

```
$ ldapsearch \
 --getEffectiveRightsAuthzid "dn:uid=ahall,ou=People,dc=example,dc=com" \
 --getEffectiveRightsAttribute member \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --baseDN "ou=Self Service,ou=groups,dc=example,dc=com" \
 "cn=Carpoolers" \
 aclRights

dn: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
aclRights;entryLevel: add:0,delete:0,read:1,write:0,proxy:0
aclRights;attributeLevel;member: search:1,read:1,compare:1,write:0,selfwrite-add:1,selfwrite-delete:1,proxy:0
```

Proxy global policies

By default, a policy matches all entries, all types of connection, and all users. You set the properties of the policy to restrict its scope of application.

Global access control policies can allow the following:

- Requests for specified LDAP controls and extended operations.
- Access to specific attributes, with support for wildcards, @objectclass notation, and exceptions to simplify settings.
- Read access (for read, search, and compare operations).
- Write access (for add, delete, modify, and modify DN operations).
- Requiring authentication before other requests.

- Requests targeting a particular scope, with wildcards to simplify settings.
- Requests originating or not from specific client addresses or domains.
- Requests using a specified protocol.
- Requests using a specified port.
- Requests using a minimum security strength factor.
- Requests from a user whose DN does or does not match a DN pattern.

Default global policies

Access control rules are defined using individual access control policy entries. A user's access is defined as the union of all access control rules that apply to that user. In other words, an individual access control rule can only grant additional access and can not remove rights granted by another rule. This approach results in an access control policy which is easier to understand and audit, since all rules can be understood in isolation.

Policy	Settings
Anonymous extended operation and control access	**authentication-required** • `false` **allowed-extended-operation** • `Cancel` • `GetSymmetricKey` • `PasswordModify` • `StartTLS` • `WhoAmI` **allowed-control** • `Assertion` • `MatchedValues` • `NoOp` • `PasswordQualityAdvice` • `PermissiveModify` • `PostRead` • `PreRead` • `RealAttrsOnly` • `SimplePagedResults` • `VirtualAttrsOnly` • `AuthorizationIdentity` • `PasswordPolicy` • `TransactionId` • `Vlv`

Policy	Settings
Authenticated extended operation and control access	**authentication-required** • `true` **allowed-extended-operation** • `PasswordPolicyState` **allowed-control** • `ManageDsaIt` • `SubEntries` • `RelaxRules` • `SubtreeDelete` • `ServerSideSort`
Schema access	**authentication-required** • `true` **request-target-dn-equal-to** • `cn=schema` **permission** • `read` **allowed-attribute** • `objectClass` • `@subschema` • `etag` • `ldapSyntaxes` • `modifiersName` • `modifyTimestamp`

Policy	Settings
Root DSE access	**authentication-required** • `false` **request-target-dn-equal-to** • `""` **permission** • `read` **allowed-attribute** • `objectClass` • `namingContexts` • `subSchemaSubEntry` • `supportedAuthPasswordSchemes` • `supportedControl` • `supportedExtension` • `supportedFeatures` • `supportedLDAPVersion` • `supportedSASLMechanisms` • `supportedTLSCiphers` • `supportedTLSProtocols` • `vendorName` • `vendorVersion` • `fullVendorVersion` • `alive` • `healthy`
Monitor access	**authentication-required** • `true` **request-target-dn-equal-to** • `cn=monitor` **permission** • `read` **allowed-attribute** • `*` • `+`

Reject unauthenticated requests

The following example uses a single broad policy for authenticated access, and another narrow policy for anonymous extended operation access:

```
$ dsconfig \
 create-global-access-control-policy \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Authenticated access all entries" \
 --set authentication-required:true \
 --set request-target-dn-not-equal-to:"**,cn=changelog" \
 --set permission:read \
 --set allowed-attribute:"*" \
 --set allowed-attribute:createTimestamp \
 --set allowed-attribute:creatorsName \
 --set allowed-attribute:entryDN \
 --set allowed-attribute:entryUUID \
 --set allowed-attribute:etag \
 --set allowed-attribute:governingStructureRule \
 --set allowed-attribute:hasSubordinates \
 --set allowed-attribute:isMemberOf \
 --set allowed-attribute:modifiersName \
 --set allowed-attribute:modifyTimestamp \
 --set allowed-attribute:numSubordinates \
 --set allowed-attribute:structuralObjectClass \
 --set allowed-attribute:subschemaSubentry \
 --set allowed-attribute-exception:authPassword \
 --set allowed-attribute-exception:userPassword \
 --set allowed-attribute-exception:debugSearchIndex \
 --set allowed-attribute-exception:@changeLogEntry \
 --set allowed-control:Assertion \
 --set allowed-control:AuthorizationIdentity \
 --set allowed-control:Csn \
 --set allowed-control:ManageDsaIt \
 --set allowed-control:MatchedValues \
 --set allowed-control:Noop \
 --set allowed-control:PasswordPolicy \
 --set allowed-control:PermissiveModify \
 --set allowed-control:PostRead \
 --set allowed-control:PreRead \
 --set allowed-control:ProxiedAuthV2 \
 --set allowed-control:RealAttributesOnly \
 --set allowed-control:ServerSideSort \
 --set allowed-control:SimplePagedResults \
 --set allowed-control:TransactionId \
 --set allowed-control:VirtualAttributesOnly \
 --set allowed-control:Vlv \
 --set allowed-extended-operation:GetSymmetricKey \
 --set allowed-extended-operation:PasswordModify \
 --set allowed-extended-operation:PasswordPolicyState \
 --set allowed-extended-operation:StartTls \
 --set allowed-extended-operation:WhoAmI \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 create-global-access-control-policy \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
```

```
 --policy-name "Anonymous extended operation access" \
 --set authentication-required:false \
 --set allowed-extended-operation:GetSymmetricKey \
 --set allowed-extended-operation:StartTls \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Require secure connections

The following example creates a policy with a minimum security strength factor of 128. This effectively permits only secure connections for requests targeting data in `dc=example,dc=com`. The security strength factor defines the key strength for GSSAPI, SSL, and TLS:

```
$ dsconfig \
 create-global-access-control-policy \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Require secure connections for example.com data" \
 --set request-target-dn-equal-to:"**,dc=example,dc=com" \
 --set request-target-dn-equal-to:dc=example,dc=com \
 --set connection-minimum-ssf:128 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Anonymous requests from specific network

The following example allows anonymous requests from clients in the `example.com` domain:

```
$ dsconfig \
 set-global-access-control-policy-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Anonymous extended operation access" \
 --set connection-client-address-equal-to:.example.com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-global-access-control-policy-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Root DSE access" \
 --set connection-client-address-equal-to:.example.com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

You can also use the `connection-client-address-not-equal-to` property to reject requests from a particular host, domain, address, or address mask.

For additional details, see Global Access Control Policy.

Data encryption

DS directory servers can encrypt directory data in backend files on disk. This keeps the data confidential until it is accessed by a directory client.

> **Important**
>
> Encrypting stored directory data does not prevent it from being sent over the network in the clear. Use secure connections to protect data sent over the network.

Advantages	Trade-offs
Confidentiality and integrity Encrypted directory data is confidential, remaining private until decrypted with a proper key. Encryption ensures data integrity at the moment it is accessed. The DS directory service cannot decrypt corrupted data.	**Equality indexes limited to equality matching** When an equality index is configured without confidentiality, the server maintains values in sorted order. It can then use the cleartext equality index can for searches that require ordering or match initial substrings. An example of a search that requires ordering is a search with a filter `"(cn<=App)"`. The filter matches entries with `commonName` up to those starting with `App` (case-insensitive) in alphabetical order. An example of a search that matches an initial substring is a search with a filter `"(cn=A*)"`. The filter matches entries having a `commonName` that starts with `a` (case-insensitive). In an equality index with confidentiality enabled, the server can no longer sort the values. As a result, you must either add indexes or accept that ordering and initial substring searches are unindexed.
Protection on shared infrastructure When you share the infrastructure, you relinquish full and sole control of directory data. For example, if the DS directory server runs in the cloud, or in a data center with shared disks, the file system and disk management are not under your control. With encrypted data, other users cannot read the files unless they also have the decryption keys.	**Performance impact** Encryption and decryption requires more processing than handling plaintext values. Encrypted values also take up more space than plaintext values.

To check what a system user with read access to backend database files can see, use the `backendstat dump-raw-db` command. The `backendstat` subcommands `list-raw-dbs` and `dump-raw-db` let you list and view the low-level databases within a backend. Unlike the output of other subcommands, the output of the `dump-raw-db` subcommand is neither decrypted nor formatted for readability. Instead, you can see values as they are stored in the backend file.

In a backend database, the `id2entry` index holds LDIF for directory entries. For a database that is not encrypted, the corresponding low-level database shows the cleartext strings:

```
$ stop-ds

$ backendstat list-raw-dbs --backendId dsEvaluation
...
/dc=com,dc=example/id2entry

$ backendstat \
 dump-raw-db \
 --backendId dsEvaluation \
 --dbName /dc=com,dc=example/id2entry
...
Key (len 8):
 00 00 00 00 00 00 00 1E                          ........
 Value (len 437):
 02 00 81 B1 03 01 06 27 75 69 64 3D 62 6A 65 6E  .......'uid=bjen
 73 65 6E 2C 6F 75 3D 50 65 6F 70 6C 65 2C 64 63  sen,ou=People,dc
 3D 65 78 61 6D 70 6C 65 2C 64 63 3D 63 6F 6D 01  =example,dc=com.
 06 11 01 08 01 13 62 6A 65 6E 73 65 6E 40 65 78  ......bjensen@ex
 61 6D 70 6C 65 2E 63 6F 6D 01 09 01 04 30 32 30  ample.com....020
 39 01 16 01 0C 65 6E 2C 20 6B 6F 3B 71 3D 30 2E  9....en, ko;q=0.
 38 01 10 01 29 75 69 64 3D 74 72 69 67 64 65 6E  8...)uid=trigden
 2C 20 6F 75 3D 50 65 6F 70 6C 65 2C 20 64 63 3D  , ou=People, dc=
 65 78 61 6D 70 6C 65 2C 64 63 3D 63 6F 6D 01 04  example,dc=com..
 02 13 50 72 6F 64 75 63 74 20 44 65 76 65 6C 6F  ..Product Develo
 70 6D 65 6E 74 06 50 65 6F 70 6C 65 01 0B 01 07  pment.People....
 42 61 72 62 61 72 61 01 0C 01 0F 2B 31 20 34 30  Barbara....+1 40
 38 20 35 35 35 20 31 38 36 32 01 0D 01 06 4A 65  8 555 1862....Je
 6E 73 65 6E 01 07 02 0E 42 61 72 62 61 72 61 20  nsen....Barbara
 4A 65 6E 73 65 6E 0B 42 61 62 73 20 4A 65 6E 73  Jensen.Babs Jens
 65 6E 01 0E 01 0D 2F 68 6F 6D 65 2F 62 6A 65 6E  en..../home/bjen
 73 65 6E 01 0F 01 0F 2B 31 20 34 30 38 20 35 35  sen....+1 408 55
 35 20 31 39 39 32 01 11 01 04 31 30 30 30 01 12  5 1992....1000..
 01 2E 7B 53 53 48 41 7D 33 45 66 54 62 33 70 37  ..{SSHA}3EfTb3p7
 71 75 6F 75 73 4B 35 34 2B 41 4F 34 71 44 57 6C  quousK54+AO4qDWl
 56 33 4F 39 54 58 48 57 49 4A 49 32 4E 41 3D 3D  V3O9TXHWIJI2NA==
 01 13 01 04 31 30 37 36 01 05 01 14 4F 72 69 67  ....1076....Orig
 69 6E 61 6C 20 64 65 73 63 72 69 70 74 69 6F 6E  inal description
 01 14 01 07 62 6A 65 6E 73 65 6E 01 15 01 0D 53  ....bjensen....S
 61 6E 20 46 72 61 6E 63 69 73 63 6F 01 01 02 01  an Francisco....
 24 38 38 37 37 33 32 65 38 2D 33 64 62 32 2D 33  $887732e8-3db2-3
 31 62 62 2D 62 33 32 39 2D 32 30 63 64 36 66 63  1bb-b329-20cd6fc
 65 63 63 30 35                                   ecc05
```

Enable confidentiality to encrypt database backends. When confidentiality is enabled, the server encrypts entries before storing them in the backend. The following command enables backend confidentiality with the default encryption settings:

```
$ dsconfig \
 set-backend-prop \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --set confidentiality-enabled:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

After confidentiality is enabled, the server encrypts entries *only when it next writes them*. The server does not automatically rewrite all entries in encrypted form. Instead, it encrypts each entry the next time it is updated, or when you re-import data from LDIF.

To see the impact of backend encryption, import the data and view the results, as in the following example:

```
$ stop-ds

$ export-ldif \
 --offline \
 --backendId dsEvaluation \
 --includeBranch dc=example,dc=com \
 --ldifFile backup.ldif

$ import-ldif \
 --offline \
 --backendId dsEvaluation \
 --includeBranch dc=example,dc=com \
 --ldifFile backup.ldif

$ backendstat \
 dump-raw-db \
 --backendId dsEvaluation \
 --dbName /dc=com,dc=example/id2entry
...
Key (len 8):
 00 00 00 00 00 00 00 C2                          ........
 Value (len 437):
 02 02 81 82 01 C4 95 87 5B A5 2E 47 97 80 23 F4  ........[..G..#.
 CE 5D 93 25 97 D4 13 F9 0A A3 A8 31 9A D9 7A 70  .].%.......1..zp
 FE 3E AC 9D 64 41 EB 7B D5 7F 7E B8 B7 74 52 B8  .>..dA.{.^?~..tR.
 C7 7F C8 79 19 46 7D C5 5D 5B 83 9C 5B 9F 85 28  .^?.y.F}.][..[..(
 83 A2 5F A0 C1 B1 09 FC 2F E3 D8 82 4A AA 8B D9  .._...../...J...
 78 43 34 50 AE A1 52 88 5B 70 97 D2 E1 EA 87 CA  xC4P..R.[p......
 3B 4D 07 DC F9 F8 30 BB D2 76 51 C8 75 FF FA 80  ;M....0..vQ.u...
 77 E1 6A 8B 5B 8F DA A4 F4 0B B5 20 56 B3 19 19  w.j.[...... V...
 22 D8 9D 38 04 E3 4D 94 A7 99 4B 81 16 AD 88 46  "..8..M...K....F
 FC 3F 7E 78 66 B8 D1 E9 86 A0 F3 AC B6 68 0D A9  .?~xf........h..
 9A A7 3C 30 40 37 97 4E 90 DD 63 16 8E 11 0F 5E  ..<0@7.N..c....^
 9D 5B 86 90 AF 4E E2 1F 9E 70 73 14 0A 11 5C DB  .[...N...ps...\.
 B7 BC B8 A9 31 3F 74 8D 0A 9F F4 6C E1 B0 36 78  ....1?t....l..6x
 F0 5A 5E CD 7C B3 A2 36 66 8E 88 86 A0 8B 9A 77  .Z^.|..6f......w
 D5 CD 7E 9C 4E 62 20 0E D0 DB AD E7 7E 99 46 4F  ..~.Nb .....~.FO
 67 C7 A6 7E 2C 24 82 50 51 9F A7 B2 02 44 5B 30  g..~,$.PQ....D[0
 74 41 99 D9 83 69 EF AE 2E C0 FF C4 E6 4F F2 2F  tA...i.......O./
 95 FB 93 65 30 2A 2D 8D 20 88 83 B5 DE 35 B6 20  ...e0*-. ....5.
 47 17 30 25 60 FD E3 43 B9 D6 A4 F7 47 B6 6C 9F  G.0%`..C....G.l.
 47 FD 63 8E 7F A5 00 CE 6C 3E BC 95 23 69 ED D0  G.c.^?...l>..#i..
 69 4F BE 61 BD 30 C2 40 66 F6 F9 C3 3E C1 D7 8C  iO.a.0.@f...>...
 B0 C8 4A 2E 27 BE 13 6C 40 88 B0 13 A3 12 F4 50  ..J.'..l@......P
 CA 92 D8 EB 4A E5 3F E2 64 A3 76 C7 5C 2B D8 89  ....J.?.d.v.\+..
 A3 6E C1 F7 0A C2 37 7A BD AF 14 4B 52 04 6B F2  .n....7z...KR.k.
 8F 4F C3 F8 00 90 BA 0F EC 6D B1 2D A8 18 0C A6  .O.......m.-....
 29 96 82 3B 5C BC D0 F4 2B BE 9C C5 8B 18 7A DE  )..;\...+.....z.
 C7 B5 10 2D 45 50 4F 77 ED F7 23 34 95 AF C3 2E  ...-EPOw..#4....
 B0 9B FA E9 DF                                   .....
```

The settings for data confidentiality depend on the encryption capabilities of the JVM. See the explanations in javax.crypto.Cipher. You can accept the default settings, or choose to specify the following:

- The cipher algorithm defining how the plaintext is encrypted and decrypted.
- The cipher mode of operation defining how a block cipher algorithm transforms data larger than a single block.

- The cipher padding defining how to pad the plaintext to reach appropriate size for the algorithm.
- The cipher key length, where longer key lengths strengthen encryption at the cost of lower performance.

The default settings for confidentiality are `cipher-transformation: AES/GCM/NoPadding`, and `cipher-key-length: 128`. This means the algorithm is the Advanced Encryption Standard (AES), and the cipher mode is Galois/Counter Mode (GCM). The syntax for the `cipher-transformation` is `algorithm/mode/padding`. You must specify the *algorithm*, *mode*, and *padding*. When the algorithm does not require a mode, use `NONE`. When the algorithm does not require padding, use `NoPadding`.

DS servers encrypt data using symmetric keys. Servers store symmetric keys, encrypted with the shared master key, with the data they encrypt. As long as servers have the same shared master key, any server can recover symmetric keys needed to decrypt data.

Symmetric keys for (deprecated) reversible password storage schemes are the exception to this rule. When you configure a reversible password storage scheme, enable the `adminRoot` backend, and configure a replication domain for `cn=admin data`.

You can enable confidentiality by backend index, as long as confidentiality is enabled for the backend itself. Confidentiality hashes keys for equality type indexes using SHA-1. It encrypts the list of entries matching a substring key for substring indexes. The following example shows how to enable confidentiality for the `mail` index:

```
$ dsconfig \
 set-backend-index-prop \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name mail \
 --set confidentiality-enabled:true \
 --no-prompt \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

After changing the index configuration, rebuild the index to enforce confidentiality immediately.

Confidentiality cannot be enabled for VLV indexes. Avoid using sensitive attributes in VLV indexes.

When reading files for an encrypted backend database, be aware that although user data is kept confidential, the following are *not encrypted* on disk:

- Existing backend database files.

  The server encrypts backend database content *only when it is next written*. If you must make sure that all relevant data are encrypted, export the data to LDIF and then import the data again.

- The `dn2id` index and its keys.
- Substring index keys.

  Substring index values are encrypted.

Encrypting and decrypting data require cryptographic processing that reduces throughput, and extra space for larger encrypted values. Tests with default settings show that the cost of enabling confidentiality can be quite modest. Your results can vary based on the host system hardware, the JVM, and the settings for `cipher-transformation` and `cipher-key-length`. Make sure you test your deployment to qualify the impact of confidentiality before changing settings in production.

Client best practices

Encourage best practices for directory clients that you control and influence.

Handle input securely

When taking input directly from a user or another program, handle the input securely by using appropriate methods to sanitize the data. Failure to sanitize the input data can leave your client vulnerable to injection attacks.

For example, the DS Java APIs have `Filter.format()` and `DN.format()` methods. Like the Java `String.format()` methods, these escape input objects when formatting output.

When writing command-line or HTTP clients, make sure you sanitize the input.

Use secure connections

Use secure connections except when reading public information anonymously. Always use secure connections when sending credentials for authentication, and when reading or writing any data that is not public.

For LDAP clients, either connect to the directory server's LDAPS port, or begin each session with the StartTLS extended operation on the insecure LDAP port.

For HTTP clients, use HTTPS.

Authenticate appropriately

Unless your client only reads public information, authenticate to the directory server.

Use an account that is specific to your client when authenticating. This helps avoid risks involved in sharing credentials between accounts. Furthermore, it makes debugging easier because your client's actions are associated with its account.

Avoid username/password credentials for clients, by certificate-based authentication. For details, see Certificate-based authentication.

Consider OAuth 2.0

DS servers support OAuth 2.0 for HTTP authorization. This lets the HTTP client access directory data without having a directory account. The directory acts as an OAuth 2.0 resource server, as described in Configure HTTP authorization.

An OAuth 2.0 client gets authorization from the resource owner, such as the user, device, or thing whose account it needs to access, and presents the OAuth 2.0 bearer access token to get access to the account. Access tokens give the bearer access, regardless of the bearer's identity.

Send access tokens only over secure HTTPS connections to prevent eavesdroppers from stealing the token.

Consider proxied authorization

DS servers support LDAP proxied authorization control. With proxied authorization, an LDAP client binds to the directory using its own account, and sends requests with the user authorization ID in the control. For details, see Proxied authorization.

When the user is already safely authenticated by other means, proxied authorization makes it easy to reuse a connection that is dedicated and bound to the client.

Apply resource limits

LDAP clients can set time limits and size limits on search requests. Setting limits is appropriate when searches are partially or fully determined by user input.

You can use the DS Java APIs methods `SearchRequest.setSizeLimit()` and `SearchRequest.setTimeLimit()` for this purpose.

The Directory Services `ldapsearch` command has `--sizeLimit` and `--timeLimit` options.

Tests

In this context, *validation* means checking that you are building the right thing, whereas *verification* means checking that you are building it in the right way.

Requirement validation

Before you begin to write specific tests, step back to review the big picture. Do the security requirements make sense for the directory service, and for the users of the service?

For the directory service, aim to prevent problems caused by security threats. The directory service must expose sensitive information only in secure ways. It must require strong authentication credentials, and limit access.

For the users, the service must permit access from any device or application using standard protocols and tools. The directory should protect your sensitive information while making it easy to connect.

In refining the security requirements for the directory service, make sure there is an appropriate balance between security and user needs. An ultimately secure directory service is one that denies all user access. An ultimately flexible directory service is one that relinquishes all control. The appropriate balance is somewhere in the middle.

Be aware that directory service security is only part of an overall strategy, one that aims to help users and application developers make appropriately secure choices. In validating the requirements, understand how the directory service fits into the overall identity and access management strategy.

Functional tests

Long before you roll out the directory service, when you start to prepare server configurations in a lab environment, begin by testing the directory's functional capabilities. As you understand your users' requirements, reproduce what their client applications will do in your tests. In most cases, it is not feasible to exhaustively reproduce everything that every directory client will do. Instead, choose a representative sample of actions. Test your expectations, both for normal and for insecure client application and user behavior.

The goals for your functional testing are to verify compliance, to uncover problems, and to begin automating tests early in the project. Test automation should drive you to use version control software, and continuous integration software as well. Be ready to roll back any change you make if a test fails, and make sure that every change is reviewed and tested before it is pushed further along in the process.

Aim to keep the automated tests both representative and short. As you build out the deployment and complexity grows, automated tests let you build the service with confidence, by repeatedly iterating with small changes to fix problems, and to better match users' expectations.

Integration tests

Before you apply a change to a production environment, verify the impact under conditions as close as possible to those of the production environment.

In the test environment, the directory service should mirror production for configuration, client application configuration and load, and directory data, including access policies. This is the environment where end-to-end testing first takes place.

Although you might not test at a scale that is identical to production, the test environment must remain representative. For example, when using replicas in production, also use replicas in the pre-production test environment. When using secure connections everywhere in production, also use them in the test environment. If you expect many client applications accessing the production directory, and particular client usage patterns, also simulate those in the test environment.

Automate your testing in the pre-production environment as well. Each change for the production environment should first pass the pre-production tests. You will need to iterate through the tests for each change.

In deployments where updates to new servers would not harm production data, consider using canary servers. You deploy a small group of canary servers in production that have the change. You then test and monitor these servers to compare with unchanged servers. If the change looks good after enough testing and monitoring, you roll out more servers with the change. If something goes wrong, you have only exposed a small proportion of production clients to the change. Only use this when you are sure that replication from a canary server would not corrupt any production data.

Continuous verification

Directory services integrate with many monitoring solutions. Access monitoring information over HTTP, LDAP, SNMP, and JMX, and by sending Common Audit events to local log files and remote systems for further processing.

Adapt some of your tests to verify operation of the service in production. For example, a few end-to-end tests in production can alert you to problems early before they impact many users.

Maintenance

This guide covers recurring administrative operations.

Tools

Run DS command-line tools.

Server Process

Start, stop, restart DS.

Backup/Restore

Backup and restore data.

Resource Limits

Set limits for user and application.

Tuning

Tune server performance.

Troubleshooting

Solve common problems.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

Maintenance tools

Server commands

- Add DS server command-line tools to your PATH:

Bash

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

PowerShell

```
PS C:\path\to> $env:PATH += ";C:\path\to\opendj\bat"
```

- For reference information, use the `--help` option with any DS tool.
- All commands call Java programs. This means every command starts a JVM, so it takes longer to start than a native binary.
- The DS `bash-completion` command generates a completion script for the Bash shell that makes it easier to write other DS commands.

  The completion script depends on support for `bash-completion`, which is not included by default on macOS.

  To set up Bash completion for DS commands, source the output of the script:

Bash 4

```
source <(/path/to/opendj/bin/bash-completion)
```

Bash 3.2 macOS

```
# First, install bash-completion support.
# Next:
eval "$( /path/to/opendj/bin/bash-completion )"
```

You can make completion available in any new interactive shell by adding it to your `~/.bash_profile` file, or `~/.bashrc` file if it is loaded by the new shell.

DS running on…	DS installed from…	Default path to tools…
Linux distributions	.zip	`/path/to/opendj/bin`
Linux distributions	.deb, .rpm	`/opt/opendj/bin`
Microsoft Windows	.zip	`C:\path\to\opendj\bat`

The installation and upgrade tools, `setup`, and `upgrade`, are found in the parent directory of the other tools. These tools are not used for everyday administration.

Commands	Constraints

`dsbackup`
`dsconfig`
`export-ldif`
`import-ldif`
`rebuild-index`
`setup`
`setup-profile`
`start-ds`
	When the server is offline, or when running commands in offline mode, these commands can modify server files. They must, therefore, access server files as a user who has the same filesystem permissions as the user who installs and runs the server.
	For most systems, the simplest way to achieve this is to run the command as the same user who installs and runs the server. When following best practices for auditing and separation of duty, provision administrative and server user accounts with compatible group or access control list permissions.

`backendstat`
`create-rc-script`
`encode-password`
`setup`
`setup-profile`
`start-ds`
`supportextract`
`upgrade`
`windows-service`
	These commands must be used with the local DS server in the same installation as the tools. These commands are not useful with non-DS servers.

Commands	Constraints

`dsbackup`
`changelogstat`
`dsconfig`
`dsrepl`
`encode-password`
`export-ldif`
`import-ldif`
`manage-account`
`manage-tasks`
`rebuild-index`
`status`
`stop-ds`
`verify-index`
	These commands must be used with DS servers having the same version as the command.
	These commands are not useful with non-DS servers.

`makeldif`	This command depends on template files. The template files can make use of configuration files installed with DS servers under `config/MakeLDIF/`. The LDIF output can be used with any directory server.

`base64`
`ldapcompare`
`ldapdelete`
`ldapmodify`
`ldappasswordmodify`
`ldapsearch`
`ldifdiff`
`ldifmodify`
`ldifsearch`
	These commands can be used independently of DS servers, and are not tied to a specific version.

Command[1]	Description
`addrate`	Measure add and delete throughput and response time.
`authrate`	Measure bind throughput and response time.
`backendstat`	Debug databases for pluggable backends.
`base64`	Encode and decode data in base64 format. Base64-encoding represents binary data in ASCII, and can be used to encode character strings in LDIF, for example.
`bash-completion`	Generate a completion script for use with Bash shell. Requires `bash-completion` support.
`changelogstat`	Debug file-based changelog databases.

Command[1]	Description
`create-rc-script` (UNIX)	Generate a script you can use to start, stop, and restart the server, either directly, or at system boot and shutdown. Use `create-rc-script -f script-file`. This lets you register and manage DS servers as services on UNIX and Linux systems.
`dsbackup`	Back up or restore directory data.
`dskeymgr`	Generate a deployment ID, a shared master key, a private CA certificate based on a deployment ID and password, or a key pair with the certificate signed by the private CA.
`dsconfig`	The `dsconfig` command is the primary command-line tool for viewing and editing DS server configurations. When started without arguments, `dsconfig` prompts you for administration connection information. Once connected to a running server, it presents you with a menu-driven interface to the server configuration. To edit the configuration when the server is not running, use the `--offline` command. Some advanced properties are not visible by default when you run the `dsconfig` command interactively. Use the `--advanced` option to access advanced properties. When you pass connection information, subcommands, and additional options to `dsconfig`, the command runs in script mode, so it is not interactive. You can prepare `dsconfig` batch scripts with the `--commandFilePath` option in interactive mode, then read from the batch file with the `--batchFilePath` option in script mode. Batch files can be useful when you have many `dsconfig` commands to run, and want to avoid starting the JVM for each command. Alternatively, you can read commands from standard input with the `--batch` option.
`dsrepl`	Manage data replication between directory servers to keep their contents in sync.
`encode-password`	Encode a plaintext password according to one of the available storage schemes.
`export-ldif`	Export directory data to LDIF, the standard, portable, text-based representation of directory content.
`import-ldif`	Load LDIF content into the directory, which overwrites existing data. It cannot be used to append data to the backend database.
`ldapcompare`	Compare the attribute values you specify with those stored on entries in the directory.
`ldapdelete`	Delete one entry or an entire branch of subordinate entries in the directory.
`ldapmodify`	Modify the specified attribute values for the specified entries.
`ldappasswordmodify`	Modify user passwords.
`ldapsearch`	Search a branch of directory data for entries that match the LDAP filter you specify.
`ldifdiff`	Display differences between two LDIF files. The output is LDIF.
`ldifmodify`	Similar to the `ldapmodify` command, modify specified attribute values for specified entries in an LDIF file.

Command[1]	Description
`ldifsearch`	Similar to the `ldapsearch` command, search a branch of data in LDIF for entries matching the LDAP filter you specify.
`makeldif`	Generate directory data in LDIF based on templates that define how the data should appear. The `makeldif` command generates test data that mimics data expected in production, and does not compromise real, potentially private information.
`manage-account`	Lock and unlock user accounts, and view and manipulate password policy state information.
`manage-tasks`	View information about tasks scheduled to run in the server, and cancel specified tasks.
`modrate`	Measure modification throughput and response time.
`rebuild-index`	Rebuild an index stored in an indexed backend.
`searchrate`	Measure search throughput and response time.
`setup-profile`	Configure a setup profile after initial installation.
`start-ds`	Start one DS server.
`status`	Display information about the server.
`stop-ds`	Stop one DS server.
`supportextract`	Collect troubleshooting information for technical support purposes.
`verify-index`	Verify that an index stored in an indexed backend is not corrupt.
`windows-service` (Windows)	Register and manage one DS server as a Windows service.

[1] UNIX names for the commands. Equivalent Windows commands have .bat extensions.

Trusted certificates

When a client tool initiates a secure connection to a server, the server presents its digital certificate.

The tool must determine whether it trusts the server certificate and continues to negotiate a secure connection, or does not trust the server certificate and drops the connection. To trust the server certificate, the tool's truststore must contain the trusted certificate. The trusted certificate is a CA certificate, or the self-signed server certificate.

The following table explains how the tools locate the truststore.

Truststore Option	Truststore Used
None	The default truststore, `user.home/.opendj/keystore`, where *user.home* is the Java system property. *user.home* is `$HOME` on Linux and UNIX, and `%USERPROFILE%` on Windows. The keystore password is `OpenDJ`. Neither the file name, nor the password can be changed. • In interactive mode, DS command-line tools prompt for approval to trust an unrecognized certificate, and whether to store it in the default truststore for future use. • In silent mode, the tools rely on the default truststore.
`--use<Type>TrustStore {trustStorePath}`	Only the specified truststore is used. The <*Type*> in the option name reflects the trust store type. The tool fails with an error if it cannot trust the server certificate.

Default settings

You can set defaults in the `~/.opendj/tools.properties` file, as in the following example:

```
hostname=localhost
port=4444
bindDN=uid=admin
useSsl=true
trustAll=true
```

The file location on Windows is `%UserProfile%\.opendj\tools.properties`.

Server processes

Start a server

- Start the server in the background:

  ```
  $ start-ds
  ```

  Alternatively, specify the `--no-detach` option to start the server in the foreground.

- (Linux) If the DS server was installed from a .deb or .rpm package, then service management scripts were created at setup time:

```
centos# service opendj start

Starting opendj (via systemctl):                           [  OK  ]
```

```
ubuntu$ sudo service opendj start

$Starting opendj: > SUCCESS.
```

- (UNIX) Create an RC script, and use the script to start the server.

  Unless you run DS servers on Linux as root, use the `--userName userName` option to specify the user who installed the server:

  ```
  $ sudo create-rc-script --outputFile /etc/init.d/opendj --userName opendj

  $ sudo /etc/init.d/opendj start
  ```

  For example, if you run the DS server on Linux as root, you can use the RC script to start the server at system boot, and to stop the server at system shutdown:

  ```
  $ sudo update-rc.d opendj defaults

  update-rc.d: warning: /etc/init.d/opendj missing LSB information
  update-rc.d: see <http://wiki.debian.org/LSBInitScripts>
   Adding system startup for /etc/init.d/opendj ...
     /etc/rc0.d/K20opendj -> ../init.d/opendj
     /etc/rc1.d/K20opendj -> ../init.d/opendj
     /etc/rc6.d/K20opendj -> ../init.d/opendj
     /etc/rc2.d/S20opendj -> ../init.d/opendj
     /etc/rc3.d/S20opendj -> ../init.d/opendj
     /etc/rc4.d/S20opendj -> ../init.d/opendj
     /etc/rc5.d/S20opendj -> ../init.d/opendj
  ```

  Alternatively, generate a service file with the `--systemdService` option, and use `systemd` to manage the service.

- (Windows) Register the DS server as a Windows service:

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

  Manage the service with Windows-native administration tools.

Stop a server

Although DS servers are designed to recover from failure and disorderly shutdown, it is safer to shut the server down cleanly, because a clean shutdown reduces startup delays. During startup, the server attempts to recover database backend state. Clean shutdown prevents situations where the server cannot recover automatically.

Clean server retirement

1. Before shutting down the system where the server is running, and before detaching any storage used for directory data, cleanly stop the server using one of the following techniques:
    - Use the `stop-ds` command:

      ```
      $ stop-ds
      ```

    - (Linux) If the DS server was installed from a .deb or .rpm package, then service management scripts were created at setup time:

      ```
      centos# service opendj stop

      Stopping opendj (via systemctl):                           [  OK  ]
      ```

      ```
      ubuntu$ sudo service opendj stop

      $Stopping opendj: ... > SUCCESS.
      ```

    - (UNIX) Create an RC script, and then use the script to stop the server:

      ```
      $ sudo create-rc-script --outputFile /etc/init.d/opendj --userName opendj

      $ sudo /etc/init.d/opendj stop
      ```

    - (Windows) Register the DS server once as a Windows service:

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

      Manage the service with Windows-native administration tools.

   Do not intentionally kill the DS server process unless the server is completely unresponsive.

   + When stopping cleanly, the server writes state information to database backends, and releases locks that it holds on database files.

Restart a server

- Use the `stop-ds` command:

  ```
  $ stop-ds --restart
  ```

- (Linux) If the DS server was installed from a .deb or .rpm package, then service management scripts were created at setup time:

```
centos# service opendj restart

Restarting opendj (via systemctl):                          [  OK  ]
```

```
ubuntu$ sudo service opendj restart

$Stopping opendj: ... > SUCCESS.

$Starting opendj: > SUCCESS.
```

- (UNIX) Create an RC script, and then use the script to stop the server:

```
$ sudo create-rc-script --outputFile /etc/init.d/opendj --userName opendj

$ sudo /etc/init.d/opendj restart
```

- (Windows) Register the DS server once as a Windows service:

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

Manage the service with Windows-native administration tools.

Server tasks

The following server administration commands can be run in online and offline mode. They invoke data-intensive operations, and so potentially take a long time to complete. The links below are to the reference documentation for each command:

- dsbackup
- export-ldif
- import-ldif
- rebuild-index

When you run these commands in online mode, they run as *tasks* on the server. Server tasks are scheduled operations that can run one or more times as long as the server is up. For example, you can schedule the `dsbackup` and `export-ldif` commands to run recurrently in order to back up server data on a regular basis.

You schedule a task as a directory administrator, sending the request to the administration port. You can therefore schedule a task on a remote server if you choose. When you schedule a task on a server, the command returns immediately, yet the task can start later, and might run for a long time before it completes. You can access tasks by using the manage-tasks command.

Although you can schedule a server task on a remote server, *the data for the task must be accessible to the server locally*. For example, when you schedule a backup task on a remote server, that server writes backup files to a file system on the remote server. Similarly, when you schedule a restore task on a remote server, that server restores backup files from a file system on the remote server.

The reference documentation describes the available options for each command:

- Configure email notification for success and failure
- Define alternatives on failure
- Start tasks immediately (`--start 0`)
- Schedule tasks to start at any time in the future

Server recovery

DS servers can restart after a crash or after the server process is killed abruptly. After disorderly shutdown, the DS server must recover its database backends. Generally, DS servers return to service quickly.

Database recovery messages are found in the database log file, such as `/path/to/opendj/db/userData/dj.log`.

The following example shows two example messages from the recovery log. The first message is written at the beginning of the recovery process. The second message is written at the end of the process:

```
[/path/to/opendj/db/userData]Recovery underway, found end of log
...
[/path/to/opendj/db/userData]Recovery finished: Recovery Info ...
```

The JVM's heap-based database cache is lost when the server stops or crashes. The cache must therefore be reconstructed from the directory database files. Database files might still be in the filesystem cache on restart, but rebuilding the JVM's heap-based database cache takes time. DS servers start accepting client requests before this process is complete.

Backup and restore

Important

- Backup archives are *not guaranteed to be compatible* across major and minor server releases. *Restore backups only on directory servers of the same major or minor version.*
  To share data between servers of different versions, either use replication, or use LDIF.
- DS servers use cryptographic keys to sign and verify the integrity of backup files, and to encrypt data. Servers protect these keys by encrypting them with the shared master key for a deployment. For portability, servers store the encrypted keys in the backup files.
  Any server can therefore restore a backup taken with the same server version, *as long as it holds a copy of the shared master key used to encrypt the keys*.

Back up

When you set up a directory server, the process creates a `/path/to/opendj/bak/` directory. You can use this for backups if you have enough local disk space, and when developing or testing backup processes. In deployment, store backups remotely to avoid losing your data and backups in the same crash.

Back up data (server task)

When you schedule a backup as a server task, the DS server manages task completion. The server must be running when you schedule the task, and when the task runs:

1. Schedule the task on a running server, binding as a user with the `backend-backup` administrative privilege.

   The following example schedules an immediate backup task for the `dsEvaluation` backend:

   ```
   $ dsbackup \
    create \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --backupLocation bak \
    --backendName dsEvaluation
   ```

   To back up all backends, omit the `--backendName` option.

   To back up more than one backend, specify the `--backendName` option multiple times.

   For details, see dsbackup.

Back up data (scheduled task)

When you schedule a backup as a server task, the DS server manages task completion. The server must be running when you schedule the task, and when the task runs:

1. Schedule backups using the `crontab` format with the `--recurringTask` option.

   The following example schedules nightly online backup of all user data at 2 AM, notifying `diradmin@example.com` when finished, or on error:

```
$ dsbackup \
 create \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --backupLocation bak \
 --recurringTask "00 02 * * *" \
 --description "Nightly backup at 2 AM" \
 --taskId NightlyBackup \
 --completionNotify diradmin@example.com \
 --errorNotify diradmin@example.com
```

For details, see dsbackup.

Back up data (external command)

When you back up data without contacting the server, the `dsbackup create` command runs as an external command, independent of the server process. It backs up the data whether the server is running or not.

> **Note**
>
> When you back up LDIF-based backends with this method, the command does not lock the files. To avoid corrupting the backup files, do not run the `dsbackup create --offline` command on an LDIF backend simultaneously with any changes to the backend.
> This applies to LDIF backends, schema files, and the task backend, for example.

Use this method to schedule backup with a third-party tool, such as the `cron` command:

1. Back up data without contacting the server process, and use the `--offline` option.

   The following example backs up the `dsEvaluation` backend immediately:

   ```
   $ dsbackup \
    create \
    --offline \
    --backupLocation bak \
    --backendName dsEvaluation
   ```

   To back up all backends, omit the `--backendName` option.

   To back up more than one backend, specify the `--backendName` option multiple times.

   For details, see dsbackup.

Back up configuration files

When you back up directory data using the `dsbackup` command, you do not back up server configuration files. The server stores configuration files under the `/path/to/opendj/config/` directory.

The server records snapshots of its configuration under the `/path/to/opendj/var/` directory. You can use snapshots to recover from misconfiguration performed with the `dsconfig` command. *Snapshots only reflect the main configuration file,* `config.ldif`.

1. Stop the server:

```
$ stop-ds
```

2. Back up the configuration files:

```
$ tar -zcvf backup-config-$(date +%s).tar.gz config
```

   By default, this backup includes the server keystore, so store it securely.

3. Start the server:

```
$ start-ds
```

Back up using snapshots

ForgeRock recommends using the `dsbackup` command when possible for backup and restore operations. You can use snapshot technology as an alternative to the `dsbackup` command, but you must be careful how you use it.

While DS directory servers are running, database backend cleanup operations write data even when there are no pending client or replication operations. An ongoing file system backup operation may record database log files that are not in sync with each other.

Successful recovery after restore is only guaranteed under certain conditions.

The snapshots must:

- Be *atomic*, capturing the state of all files at exactly the same time.

  If you are not sure that the snapshot technology is atomic, do not use it. Use the `dsbackup` *command instead.*

- Capture the state of all data (`db/`) and (`changelogDb/`) changelog files together.

  When using a file system-level snapshot feature, for example, keep at least all data and changelog files on the same file system. This is the case in a default server setup.

- Be paired with a specific server configuration.

  A snapshot of all files includes configuration files that may be specific to one DS server, and cannot be restored safely on another DS server with a different configuration. If you restore all system files, this principle applies to system configuration as well.

  For details on making DS configuration files as generic as possible, see Property value substitution.

If snapshots in your deployment do not meet these criteria, *you must stop the DS server before taking the snapshot*. You must also take care not to restore incompatible configuration files.

Restore

> **Important**
>
> After you restore a replicated backend, replication brings it up to date with changes newer than the backup.
> Replication uses internal change log records to determine which changes to apply. This process happens *even if you only have a single server* that you configured for replication at setup time (by setting the replication port with the `--replicationPort port` option). To prevent replication from replaying changes newer than the backup you restore, refer to Disaster recovery.
> Replication purges internal change log records, however, to prevent the change log from growing indefinitely.
> Replication can only bring the backend up to date if the change log still includes the last change backed up.
> For this reason, when you restore a replicated backend from backup, *the backup must be newer than the last purge of the replication change log (default: 3 days).*
> If no backups are newer than the replication purge delay, do not restore from a backup. Initialize the replica instead, without using a backup. For details, see Manual initialization.

Restore data (server task)

1. Verify the backup you intend to restore.

   The following example verifies the most recent backup of the `dsEvaluation` backend:

   ```
   $ dsbackup \
    list \
    --backupLocation bak \
    --backendName dsEvaluation \
    --last \
    --verify
   ```

2. Schedule the restore operation as a task, binding as a user with the `backend-restore` administrative privilege.

   The following example schedules an immediate restore task for the `dsEvaluation` backend:

   ```
   $ dsbackup \
    restore \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --backupLocation bak \
    --backendName dsEvaluation
   ```

   To restore the latest backups of more than one backend, specify the `--backendName` option multiple times.

   To restore a specific backup, specify the `--backupId` option. To restore multiple specific backups of different backends, specify the `--backupId` option multiple times.

   To list backup information without performing verification, use the `dsbackup list` command without the `--verify` option. The output includes backup IDs for use with the `--backupId` option.

For details, see dsbackup.

Restore data (external command)

1. Stop the server if it is running:

```
$ stop-ds --quiet
```

2. Verify the backup you intend to restore.

   The following example verifies the most recent backup of the `dsEvaluation` backend:

```
$ dsbackup \
 list \
 --backupLocation bak \
 --backendName dsEvaluation \
 --last \
 --verify
```

3. Restore using the `--offline` option.

   The following example restores the `dsEvaluation` backend:

```
$ dsbackup \
 restore \
 --offline \
 --backupLocation bak \
 --backendName dsEvaluation
```

   To restore the latest backups of more than one backend, specify the `--backendName` option multiple times.

   To restore a specific backup, specify the `--backupId` option. To restore multiple specific backups of different backends, specify the `--backupId` option multiple times.

   To list backup information without performing verification, use the `dsbackup list` command without the `--verify` option. The output includes backup IDs for use with the `--backupId` option.

   For details, see dsbackup.

4. Start the server:

```
$ start-ds --quiet
```

Restore configuration files

1. Stop the server:

```
$ stop-ds --quiet
```

2. Restore the configuration files from the backup, overwriting existing files:

```
$ tar -zxvf backup-config-<date>.tar.gz
```

3. Start the server:

```
$ start-ds --quiet
```

Restore from a snapshot

ForgeRock recommends using the `dsbackup` command when possible for backup and restore operations.

You can use snapshot technology as an alternative to the `dsbackup` command, but you must be careful how you use it. For details, see Back up using snapshots.

Take the following points into account before restoring a snapshot:

- When you restore files for a replicated backend, *the snapshot must be newer than the last purge of the replication change log (default: 3 days).*
- Stop the DS server before you restore the files.
- The DS configuration files in the snapshot must match the configuration where you restore the snapshot.

  If the configuration uses expressions, define their values for the current server before starting DS.
- When using snapshot files to initialize replication, only restore the data (`db/`) files for the target backend.

  Depending on the snapshot technology, you might need to restore the files separately, and then move only the target backend files from the restored snapshot.
- When using snapshot files to restore replicated data to a known state, stop all affected servers before you restore.

Purge old files

Periodically purge old backup files with the `dsbackup purge` command. The following example removes all backup files older than the default replication purge delay:

```
$ dsbackup \
 purge \
 --offline \
 --backupLocation bak \
 --olderThan 3d
```

This example runs the external command without contacting the server process. You can also purge backups by ID, or by backend name, and you can specify the number of backups to keep. For details, see dsbackup.

To purge files as a server task, use the task options, such as `--recurringTask`. The user must have the `backend-backup` administrative privilege to schedule a purge task.

Cloud storage

You can stream backup files to cloud storage, and restore them directly from cloud storage.

The implementation supports these providers:

- Amazon AWS S3
- Azure Cloud Storage
- Google Cloud Storage

Follow these steps to store backup files in the cloud:

1. If you upgraded in place from DS 6.5 or earlier, activate cloud storage for backup.
2. Get a storage account and space from the cloud provider where the server can store backup files.

   This storage space is referred to below as *cloud-bak*.
3. Get credentials from the cloud provider.

   The DS server backing up files must have read, write, and delete access. For information about granting access, see the access control documentation for your provider.

   If you are not yet familiar with cloud storage, see the documentation from your provider for help. The following table provides links to the documentation for supported providers:

Provider	Hints
Amazon AWS S3	For details on setting up S3 and working with S3 buckets, see the Amazon Web Services documentation on Getting started with Amazon Simple Storage Service.
Azure Cloud Storage	DS authenticates to Azure with an Azure storage account. For details, see the Microsoft documentation on how to Create an Azure Storage account, or to Create a BlockBlobStorage account.
Google Cloud Storage	DS authenticates to Google Cloud with a service account. For details, see the Google documentation on Getting Started with Authentication. For details about creating and managing storage buckets, see the Google How-To documentation on Creating buckets, and Working with buckets.

4. Set environment variables for the credentials:

Provider	Environment Variable(s)

Amazon AWS S3	`export AWS_ACCESS_KEY_ID=aws-access-key`
	`export AWS_SECRET_ACCESS_KEY=aws-secret-key`
	When using temporary credentials, also export the session token:
	`export AWS_SESSION_TOKEN=aws-session-token`

Azure Cloud Storage	`export AZURE_ACCOUNT_NAME=azure-account-name` `export AZURE_ACCOUNT_KEY=azure-account-key`
Google Cloud Storage	`export GOOGLE_CREDENTIALS=/path/to/gcp-credentials.json` (optional)

5. Restart the DS server so that it reads the environment variables you set:

```
$ stop-ds --restart
```

6. Run `dsbackup` commands with all required provider-specific options.

   The options in the following table use the providers' default storage endpoints:

Provider	Required Options

Amazon AWS S3	`--storageProperty s3.keyId.env.var:AWS_ACCESS_KEY_ID \`
	`--storageProperty s3.secret.env.var:AWS_SECRET_ACCESS_KEY \`
	`--backupLocation s3://cloud-bak`
	
	`# When using temporary credentials, also use the session token:`
	`--storageProperty s3.keyId.env.var:AWS_ACCESS_KEY_ID \`
	`--storageProperty s3.secret.env.var:AWS_SECRET_ACCESS_KEY \`
	`--storageProperty s3.sessionToken.env.var:AWS_SESSION_TOKEN \`
	`--backupLocation s3://cloud-bak`

Azure Cloud Storage	`--storageProperty az.accountName.env.var:AZURE_ACCOUNT_NAME \`
	`--storageProperty az.accountKey.env.var:AZURE_ACCOUNT_KEY \`
	`--backupLocation az://cloud-bak`

Provider	Required Options

Google Cloud Storage	`--storageProperty gs.credentials.path:/path/to/gcp-credentials.json \`
	`--backupLocation gs://cloud-bak`
	or
	`--storageProperty gs.credentials.env.var:GOOGLE_CREDENTIALS \`
	`--backupLocation gs://cloud-bak`

If your cloud storage *does not use the default endpoint*, add one of the following options:

`--storage-property endpoint:endpoint-url`

`--storage-property endpoint.env.var:environment-variable-for-endpoint-url`

For Azure cloud storage, the *endpoint-url* starts with the account name. Examples include `https://azure-account-name.blob.core.windows.net`, `https://${AZURE_ACCOUNT_NAME}.blob.core.windows.net`, and `https://${AZURE_ACCOUNT_NAME}.some.private.azure.endpoint`.

Cloud storage requires working space in the local system temporary directory. Some cloud storage providers require sending the content length with each file.

To send the correct content length, the `dsbackup` command writes each prepared backup file to the system temporary directory before upload. It deletes each file after successful upload.

Cloud storage samples

Click the samples for your storage provider to expand the section and see the commands:

```
#
# API keys created through the AWS API gateway console:
#
export AWS_ACCESS_KEY_ID=aws-access-key-id
export AWS_SECRET_ACCESS_KEY=aws-secret-key
# When using temporary credentials:
# export AWS_SESSION_TOKEN=aws-session-token

# These samples use the following S3 bucket, and a non-default endpoint:
# S3 bucket: s3://ds-test-backup
# S3 endpoint: https://s3.us-east-1.amazonaws.com
#
# When using temporary credentials, also add
# the AWS session token storage property option to each of the commands:
# --storageProperty s3.sessionToken.env.var:AWS_SESSION_TOKEN

# Back up the dsEvaluation backend offline:
dsbackup create --backendName dsEvaluation --offline \
 --backupLocation s3://ds-test-backup \
 --storageProperty s3.keyId.env.var:AWS_ACCESS_KEY_ID \
 --storageProperty s3.secret.env.var:AWS_SECRET_ACCESS_KEY \
 --storageProperty endpoint:https://s3.us-east-1.amazonaws.com

# List and verify the latest backup files for each backend at this location:
dsbackup list --verify --last \
 --backupLocation s3://ds-test-backup \
 --storageProperty s3.keyId.env.var:AWS_ACCESS_KEY_ID \
 --storageProperty s3.secret.env.var:AWS_SECRET_ACCESS_KEY \
 --storageProperty endpoint:https://s3.us-east-1.amazonaws.com

# Restore dsEvaluation from backup offline:
dsbackup restore --backendName dsEvaluation --offline \
 --backupLocation s3://ds-test-backup \
 --storageProperty s3.keyId.env.var:AWS_ACCESS_KEY_ID \
 --storageProperty s3.secret.env.var:AWS_SECRET_ACCESS_KEY \
 --storageProperty endpoint:https://s3.us-east-1.amazonaws.com

# Purge all dsEvaluation backup files:
dsbackup purge --backendName dsEvaluation --keepCount 0 --offline \
 --backupLocation s3://ds-test-backup \
 --storageProperty s3.keyId.env.var:AWS_ACCESS_KEY_ID \
 --storageProperty s3.secret.env.var:AWS_SECRET_ACCESS_KEY \
 --storageProperty endpoint:https://s3.us-east-1.amazonaws.com
```

```
#
# Credentials for Azure storage, where the Azure account is found in key1 in the Azure console:
#
export AZURE_ACCOUNT_NAME=azure-account-name
export AZURE_ACCOUNT_KEY=azure-account-key

# These samples use the following Azure storage, and a non-default endpoint:
# Azure storage: az://ds-test-backup/test1
# Azure endpoint: https://${AZURE_ACCOUNT_NAME}.blob.core.windows.net

# Back up the dsEvaluation backend offline:
dsbackup create --backendName dsEvaluation --offline \
 --backupLocation az://ds-test-backup/test1 \
 --storageProperty az.accountName.env.var:AZURE_ACCOUNT_NAME \
 --storageProperty az.accountKey.env.var:AZURE_ACCOUNT_KEY \
 --storageProperty "endpoint:https://${AZURE_ACCOUNT_NAME}.blob.core.windows.net"

# List and verify the latest backup files for each backend at this location:
dsbackup list --verify --last \
 --backupLocation az://ds-test-backup/test1 \
 --storageProperty az.accountName.env.var:AZURE_ACCOUNT_NAME \
 --storageProperty az.accountKey.env.var:AZURE_ACCOUNT_KEY \
 --storageProperty "endpoint:https://${AZURE_ACCOUNT_NAME}.blob.core.windows.net"

# Restore dsEvaluation from backup offline:
dsbackup restore --backendName dsEvaluation --offline \
 --backupLocation az://ds-test-backup/test1 \
 --storageProperty az.accountName.env.var:AZURE_ACCOUNT_NAME \
 --storageProperty az.accountKey.env.var:AZURE_ACCOUNT_KEY \
 --storageProperty "endpoint:https://${AZURE_ACCOUNT_NAME}.blob.core.windows.net"

# Purge all dsEvaluation backup files:
dsbackup purge --backendName dsEvaluation --keepCount 0 --offline \
 --backupLocation az://ds-test-backup/test1 \
 --storageProperty az.accountName.env.var:AZURE_ACCOUNT_NAME \
 --storageProperty az.accountKey.env.var:AZURE_ACCOUNT_KEY \
 --storageProperty "endpoint:https://${AZURE_ACCOUNT_NAME}.blob.core.windows.net"
```

```
#
# Credentials generated with and download from the Google cloud console:
#
export GOOGLE_CREDENTIALS=/path/to/gcp-credentials.json

# These samples use the following cloud storage, and endpoint:
# Google storage: gs://ds-test-backup/test1
# Google endpoint: https://www.googleapis.com

# Back up the dsEvaluation backend offline:
dsbackup create --backendName dsEvaluation --offline \
 --backupLocation gs://ds-test-backup/test1 \
 --storageProperty gs.credentials.env.var:GOOGLE_CREDENTIALS \
 --storageProperty endpoint:https://www.googleapis.com

# List and verify the latest backup files for each backend at this location:
dsbackup list --verify --last \
 --backupLocation gs://ds-test-backup/test1 \
 --storageProperty gs.credentials.env.var:GOOGLE_CREDENTIALS \
 --storageProperty endpoint:https://www.googleapis.com

# Restore dsEvaluation from backup offline:
dsbackup restore --backendName dsEvaluation --offline \
 --backupLocation gs://ds-test-backup/test1 \
 --storageProperty gs.credentials.env.var:GOOGLE_CREDENTIALS \
 --storageProperty endpoint:https://www.googleapis.com

# Purge all dsEvaluation backup files:
dsbackup purge --backendName dsEvaluation --keepCount 0 --offline \
 --backupLocation gs://ds-test-backup/test1 \
 --storageProperty gs.credentials.env.var:GOOGLE_CREDENTIALS \
 --storageProperty endpoint:https://www.googleapis.com
```

Disaster recovery

Directory services are critical to authentication, session management, authorization, and more. When directory services are broken, quick recovery is a must.

In DS directory services, a *disaster* is a serious data problem affecting the entire replication topology. Replication can't help you recover from a disaster because it replays data changes everywhere.

Disaster recovery comes with a service interruption, the loss of recent changes, and a reset for replication. It is rational in the event of a real disaster. It's unnecessary to follow the disaster recovery procedure for a hardware failure or a server that's been offline too long and needs reinitialization. Even if you lose most of your DS servers, you can still rebuild the service without interruption or data loss.

> **Important**
>
> For disaster recovery to be quick, you must prepare in advance.
> Don't go to production until you have successfully tested your disaster recovery procedures.

The following example helps prepare to recover from a disaster. It shows the following tasks:

- Back up a DS directory service.

- Restore the service to a known state.
- Validate the procedure.

Tasks

The following tasks demonstrate a disaster recovery procedure on a single computer two replicated DS servers set up for evaluation.

In deployment, the procedure involves multiple computers, but the order and content of the tasks remain the same. Before you perform the procedure in production, make sure you have copies of the following:

- The deployment description, documentation, plans, runbooks, and scripts.
- The system configuration and software, including the Java installation.
- The DS software and any customizations, plugins, or extensions.
- A recent backup of any external secrets required, such as an HSM or a CA key.
- A recent backup of each server's configuration files, matching the production configuration.
- The deployment ID and password.

> **Important**
>
> This procedure applies to DS versions providing the `dsrepl disaster-recovery` command.
> For deployments with any earlier DS servers that don't provide the command, you can't use this procedure. Instead, refer to How do I perform disaster recovery steps in DS?

Disaster recovery has these characteristics:

- You perform disaster recovery on a stopped server, one server at a time.
- Disaster recovery is per base DN, like replication.
- On each server you recover, you use the same *disaster recovery ID*, a unique identifier for this recovery.

To minimize the service interruption, this example recovers the servers one by one. It is also possible to perform disaster recovery in parallel by stopping and starting all servers together.

Task 1: Back up directory data

Back up data while the directory service is running smoothly. For additional details, refer to Backup and restore.

1. Back up the directory data.

   The following command backs up directory data created for evaluation:

```
$ /path/to/opendj/bin/dsbackup \
 create \
 --start 0 \
 --backupLocation /path/to/opendj/bak \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

The command returns, and the DS server runs the backup task in the background.

When adapting the recovery process for deployment, schedule a backup task to run regularly for each database backend.

2. Check the backup task finishes successfully:

```
$ /path/to/opendj/bin/manage-tasks \
 --summary \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The status of the backup task is "Completed successfully" when it is done.

Recovery from disaster means stopping the directory service and losing the latest changes. The more recent the backup, the fewer changes you lose during recovery. Backup operations are cumulative, so you can schedule them regularly without using too much disk space as long as you purge outdated backup files. As you script your disaster recovery procedures for deployment, schedule a recurring backup task to have safe, current, and complete backup files for each backend.

Task 2: Recover from a disaster

This task restores the directory data from backup files created before the disaster. Adapt this procedure as necessary if you have multiple directory backends to recover.

> **Important**
>
> All changes since the last backup operation are lost.

Subtasks:

- Prepare for recovery
- Recover the first directory server
- Recover remaining servers

Prepare for recovery

1. If you have lost DS servers, replace them with servers configured as before the disaster.

   In this example, no servers were lost. Reuse the existing servers.

2. On each replica, prevent applications from making changes to the backend for the affected base DN. Changes made during recovery would be lost or could not be replicated:

```
$ /path/to/opendj/bin/dsconfig \
 set-backend-prop \
 --backend-name dsEvaluation \
 --set writability-mode:internal-only \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ /path/to/replica/bin/dsconfig \
 set-backend-prop \
 --backend-name dsEvaluation \
 --set writability-mode:internal-only \
 --hostname localhost \
 --port 14444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

   In this example, the first server's administrative port is `4444`. The second server's administrative port is `14444`.

Recover the first directory server

Important

DS uses the *disaster recovery ID* to set the *generation ID*, an internal, shorthand form of the initial replication state. Replication only works when the data for the base DN share the same generation ID on each server.
There are two approaches to using the `dsrepl disaster-recovery` command. Use one or the other:

- (*Recommended*) Let DS generate the disaster recovery ID on a first replica. Use the generated ID on all other servers you recover.
  When you use the generated ID, the `dsrepl disaster-recovery` command verifies each server you recover has the same initial replication state as the first server.
- Use the recovery ID of your choice on all servers.
  Don't use this approach if the replication topology includes one or more standalone replication servers. It won't work.
  This approach works when you can't define a "first" replica, for example, because you've automated the recovery process in an environment where the order of recovery is not deterministic.
  When you choose the recovery ID, the `dsrepl disaster-recovery` command *doesn't* verify the data match. The command uses your ID as the random seed when calculating the new generation ID. For the new generation IDs to match, your process must have restored the same data on each server. Otherwise, replication won't work between servers whose data does not match.
  If you opt for this approach, skip these steps. Instead, proceed to Recover remaining servers.

Don't mix the two approaches in the same disaster recovery procedure. Use the generated recovery ID or the recovery ID of your choice, but do not use both.

This process generates the disaster recovery ID to use when recovering the other servers.

1. Stop the directory server you use to start the recovery process:

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

2. Restore the affected data on this directory server:

```
$ /path/to/opendj/bin/dsbackup \
 restore \
 --offline \
 --backendName dsEvaluation \
 --backupLocation /path/to/opendj/bak
```

Changes to the affected data that happened after the backup are lost. Use the most recent backup files prior to the disaster.

Tip

This approach to restoring data works in deployments with the same DS server version. When all DS servers share the same DS version, you can restore all the DS directory servers from the same backup data.
Backup archives are *not guaranteed to be compatible* across major and minor server releases. *Restore backups only on directory servers of the same major or minor version.*

3. Run the command to begin the disaster recovery process.

   When this command completes successfully, it displays the disaster recovery ID:

   ```
   $ /path/to/opendj/bin/dsrepl \
    disaster-recovery \
    --baseDn dc=example,dc=com \
    --generate-recovery-id \
    --no-prompt
   Disaster recovery id: <generatedId>
   ```

   Record the <generatedId>. You will use it to recover all other servers.

4. Start the recovered server:

   ```
   $ /path/to/opendj/bin/start-ds
   ```

5. Test the data you restored is what you expect.

6. Start backing up the recovered directory data.

   As explained in New backup after recovery, you can no longer rely on pre-recovery backup data after disaster recovery. Unless the new backup is stored in a different location than the backup used for recovery, the operation won't take a long time, as it takes advantage of the cumulative backup feature.

7. Allow external applications to make changes to directory data again:

   ```
   $ /path/to/opendj/bin/dsconfig \
    set-backend-prop \
    --backend-name dsEvaluation \
    --set writability-mode:enabled \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

You have recovered this replica and begun to bring the service back online. To enable replication with other servers to resume, recover the remaining servers.

Recover remaining servers

> **Important**
>
> Make sure you have a disaster recovery ID. Use the same ID for all DS servers in this recovery procedure:
>
> - (*Recommended*) If you generated the ID as described in Recover the first directory server, use it.
> - If not, use a unique ID of your choosing for this recovery procedure.
>   For example, you could use the date at the time you begin the procedure.

You can perform this procedure in parallel on all remaining servers or on one server at a time. For each server:

1. Stop the server:

   ```
   $ /path/to/replica/bin/stop-ds
   ```

2. Unless the server is a standalone replication server, restore the affected data from the same backup files you used for the first server:

   ```
   $ /path/to/replica/bin/dsbackup \
    restore \
    --offline \
    --backendName dsEvaluation \
    --backupLocation /path/to/opendj/bak
   ```

3. Run the recovery command.

   The following command uses a generated ID. It verifies this server's data matches the first server you recovered:

   ```
   $ export DR_ID=<generatedId>
   $ /path/to/replica/bin/dsrepl \
    disaster-recovery \
    --baseDn dc=example,dc=com \
    --generated-id ${DR_ID} \
    --no-prompt
   ```

   If the recovery ID is a unique ID of your choosing, use `dsrepl disaster-recovery --baseDn <base-dn> --user-generated-id <recoveryId>` instead. This alternative doesn't verify the data on each replica match and won't work if the replication topology includes one or more standalone replication servers.

4. Start the recovered server:

   ```
   $ /path/to/replica/bin/start-ds
   ```

5. If this is a directory server, test the data you restored is what you expect.

6. If this is a directory server, allow external applications to make changes to directory data again:

```
$ /path/to/replica/bin/dsconfig \
 set-backend-prop \
 --backend-name dsEvaluation \
 --set writability-mode:enabled \
 --hostname localhost \
 --port 14444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

After completing these steps for all servers, you have restored the directory service and recovered from the disaster.

Validation

After recovering from the disaster, validate replication works as expected. Use the following steps as a simple guide.

1. Modify an entry on one replica.

   The following command updates Babs Jensen's description to `Post recovery`:

   ```
   $ /path/to/opendj/bin/ldapmodify \
    --hostname localhost \
    --port 1636 \
    --useSsl \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --bindDn uid=bjensen,ou=People,dc=example,dc=com \
    --bindPassword hifalutin <<EOF
   dn: uid=bjensen,ou=People,dc=example,dc=com
   changetype: modify
   replace: description
   description: Post recovery
   EOF
   # MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com
   ```

2. Read the modified entry on another replica:

   ```
   $ /path/to/replica/bin/ldapsearch \
    --hostname localhost \
    --port 11636 \
    --useSsl \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --bindDN uid=bjensen,ou=People,dc=example,dc=com \
    --bindPassword hifalutin \
    --baseDn dc=example,dc=com \
    "(cn=Babs Jensen)" \
    description
   dn: uid=bjensen,ou=People,dc=example,dc=com
   description: Post recovery
   ```

You have shown the recovery procedure succeeded.

Before deployment

When planning to deploy disaster recovery procedures, take these topics into account.

Recover before the purge delay

When recovering from backup, you must complete the recovery procedure while the backup is newer than the replication delay.

If this is not possible for all servers, recreate the remaining servers from scratch after recovering as many servers as possible and taking a new backup.

New backup after recovery

Disaster recovery resets the replication generation ID to a different format than you get when importing new directory data.

After disaster recovery, you can no longer use backups created before the recovery procedure started for the recovered base DN. Directory servers can only replicate data under a base DN with directory servers having the same generation ID. The old backups no longer have the right generation IDs.

Instead, immediately after recovery, back up data from the recovered base DN and use the new backups going forward when you restore servers after the disaster recovery has completed.

You can purge older backup files to prevent someone accidentally restoring from a backup with an outdated generation ID.

Change notifications reset

Disaster recovery clears the changelog for the recovered base DN.

If you use change number indexing for the recovered base DN, disaster recovery resets the change number.

Standalone servers

If you have standalone replication servers and directory servers, you might not want to recover them all at once.

Instead, in each region, alternate between recovering a standalone directory server then a standalone replication server to reduce the time to recovery.

Reference material

Reference	Description
About replication	In-depth introduction to replication concepts
Backup and restore	The basics, plus backing up to the cloud and using filesystem snapshots
Cryptographic keys	About keys, including those for encrypting and decrypting backup files
Data storage	Details about exporting and importing LDIF, common data stores

Accounts

Account lockout

Account lockout settings are part of password policy. The server locks an account after the specified number of consecutive authentication failures. For example, users are allowed three consecutive failures before being locked out for five minutes. Failures themselves expire after five minutes.

The aim of account lockout is not to punish users who mistype their passwords. It protects the directory when an attacker attempts to guess a user password with repeated attempts to bind.

> **Note**
>
> Account lockout is not transactional across a replication topology. Under normal circumstances, replication propagates lockout quickly. If replication is ever delayed, an attacker with direct access to multiple replicas could try to authenticate up to the specified number of times on each replica before being locked out on all replicas.

The following command adds a replicated password policy to activate lockout:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Lock after three failures,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
cn: Lock after three failures
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-lockout-failure-expiration-interval: 5 m
ds-pwp-lockout-duration: 5 m
ds-pwp-lockout-failure-count: 3
subtreeSpecification: { base "ou=people" }
EOF
```

Users with this policy are locked out after three failed attempts in succession:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 uid=bjensen \
 mail

dn: uid=bjensen,ou=People,dc=example,dc=com
mail: bjensen@example.com

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword fatfngrs \
 --baseDN dc=example,dc=com \
 uid=bjensen \
 mail

The LDAP bind request failed: 49 (Invalid Credentials)

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword fatfngrs \
 --baseDN dc=example,dc=com \
 uid=bjensen \
 mail

The LDAP bind request failed: 49 (Invalid Credentials)

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
 --bindPassword fatfngrs \
 --baseDN dc=example,dc=com \
 uid=bjensen \
 mail

The LDAP bind request failed: 49 (Invalid Credentials)

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
```

```
  --useSsl \
  --usePkcs12TrustStore /path/to/opendj/config/keystore \
  --trustStorePassword:file /path/to/opendj/config/keystore.pin \
  --bindDN "uid=bjensen,ou=people,dc=example,dc=com" \
  --bindPassword hifalutin \
  --baseDN dc=example,dc=com \
  uid=bjensen \
  mail

The LDAP bind request failed: 49 (Invalid Credentials)
```

Account management

Disable an account

1. Make sure the user running the `manage-account` command has access to perform the appropriate operations.

   Kirsten Vaughan is a member of the Directory Administrators group. For this example, she must have the `password-reset` privilege, and access to edit user attributes and operational attributes:

   ```
   $ ldapmodify \
    --hostname localhost \
    --port 1636 \
    --useSsl \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --bindDN uid=admin \
    --bindPassword password << EOF
   dn: uid=kvaughan,ou=People,dc=example,dc=com
   changetype: modify
   add: ds-privilege-name
   ds-privilege-name: password-reset

   dn: ou=People,dc=example,dc=com
   changetype: modify
   add: aci
   aci: (target="ldap:///ou=People,dc=example,dc=com")(targetattr ="*||+")
    (version 3.0;acl "Admins can run amok"; allow(all)
     groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)
   EOF
   ```

   Notice here that the directory superuser, `uid=admin`, assigns privileges. Any administrator with the `privilege-change` privilege can assign privileges. However, if the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, do not assign the `privilege-change` privilege to normal administrator users.

2. Set the account status to disabled:

```
$ manage-account \
 set-account-is-disabled \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --operationValue true \
 --targetDN uid=bjensen,ou=people,dc=example,dc=com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin

Account Is Disabled:  true
```

Activate a disabled account

1. Clear the disabled status:

```
$ manage-account \
 set-account-is-disabled \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --operationValue false \
 --targetDN uid=bjensen,ou=people,dc=example,dc=com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin

Account Is Disabled:  false
```

Account status notifications

DS servers can send mail about account status changes. The DS server needs an SMTP server to send messages, and needs templates for the mail it sends. By default, message templates are in English, and found in the `/path/to/opendj/config/messages/` directory.

DS servers generate notifications only when the server writes to an entry or evaluates a user entry for authentication. A server generates account enabled and account disabled notifications when the user account is enabled or disabled with the `manage-account` command. A server generates password expiration notifications when a user tries to bind.

For example, if you configure a notification for password expiration, that notification gets triggered when the user authenticates during the password expiration warning interval. The server does not automatically scan entries to send password expiry notifications.

DS servers implement controls that you can pass in an LDAP search to determine whether a user's password is about to expire. See Supported LDAP controls for a list. Your script or client application can send notifications based on the results of the search.

Send account status mail

1. Configure an SMTP server to use when sending messages:

```
$ dsconfig \
 create-mail-server \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --server-name "SMTP server" \
 --set enabled:true \
 --set auth-username:mail.user \
 --set auth-password:password \
 --set smtp-server:smtp.example.com:587 \
 --set trust-manager-provider:"JVM Trust Manager" \
 --set use-start-tls:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Prepare the DS server to mail users about account status.

   The following example configures the server to send text-format mail messages:

```
$ dsconfig \
 set-account-status-notification-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name "SMTP Handler" \
 --set enabled:true \
 --set email-address-attribute-type:mail \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

   Notice that the server finds the user's mail address on the attribute on the user's entry, specified by `email-address-attribute-type`. You can also configure the `message-subject` and `message-template-file` properties. Use interactive mode to make the changes.

   You find templates for messages by default under the `config/messages` directory. Edit the templates as necessary.

   If you edit the templates to send HTML rather than text messages, then set the advanced property, `send-email-as-html`:

```
$ dsconfig \
 set-account-status-notification-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name "SMTP Handler" \
 --set enabled:true \
 --set send-email-as-html:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. Adjust applicable password policies to use the account status notification handler you configured:

```
$ dsconfig \
 set-password-policy-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --policy-name "Default Password Policy" \
 --set account-status-notification-handler:"SMTP Handler" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

When configuring a subentry password policy, set the `ds-pwp-account-status-notification-handler` attribute, an attribute of the `ds-pwp-password-policy` object class.

Message templates

When editing the `config/messages` templates, use the following tokens, which the server replaces with text:

`%%notification-type%%`

The name of the notification type.

`%%notification-message%%`

The message for the notification.

`%%notification-user-dn%%`

The string representation of the user DN that is the target of the notification.

`%%notification-user-attr:attrname%%`

The value of the attribute specified by attrname from the user's entry.

If the specified attribute has multiple values, then this is the first value encountered. If the specified attribute does not have any values, then this is an empty string.

%%notification-property:propname%%

The value of the specified property.

If the specified property has multiple values, then this is the first value encountered. If the specified property does not have any values, then this is an empty string.

Valid propname values include the following:

- `account-unlock-time`
- `new-password`
- `old-password`
- `password-expiration-time`
- `password-policy-dn`
- `seconds-until-expiration`
- `seconds-until-unlock`
- `time-until-expiration`
- `time-until-unlock`

Resource limits

Search limits

You can set limits on search operations:

- The *size limit* sets the maximum number of entries returned for a search.

  The default size limit of 1000 is set by the global server property `size-limit`.

  You can override the limit per user with the operational attribute, `ds-rlim-size-limit`.

  Search requests can include a size limit setting. The `ldapsearch` command has a `--sizeLimit` option.

- The *time limit* defines the maximum processing time for a search operation.

  The default time limit of 1 minute is set by the global server property `time-limit`.

  You can override the limit on a per user basis with the operational attribute, `ds-rlim-time-limit`. Times for `ds-rlim-time-limit` are expressed in seconds.

  In addition, search requests themselves can include a time limit setting. The `ldapsearch` command has an `--timeLimit` option.

- The *idle time limit* defines how long an idle connection remains open.

  No default idle time limit is set. You can set an idle time limit by using the global server property `idle-time-limit`.

You can override the limit on a per user basis with the operational attribute, `ds-rlim-idle-time-limit`. Times for `ds-rlim-idle-time-limit` are expressed in seconds.

- The maximum number of persistent searches is set by the global server property `max-psearches`.

Set limits for a user

1. Give an administrator access to update the operational attributes related to search limits:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "ds-rlim-time-limit||ds-rlim-size-limit")
 (version 3.0;acl "Allow Kirsten Vaughan to manage search limits";
 allow (all) (userdn = "ldap:///uid=kvaughan,ou=People,dc=example,dc=com");)
EOF
```

2. Change the user entry to set the limits to override:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
add: ds-rlim-size-limit
ds-rlim-size-limit: 10
EOF
```

When Babs Jensen performs an indexed search returning more than 10 entries, she sees the following message:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(sn=jensen)"

# The LDAP search request failed: 4 (Size Limit Exceeded)
# Additional Information:  This search operation has sent the maximum of 10 entries to the client
```

Set limits for users in a group

1. Give an administrator the privilege to write subentries:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: subentry-write
EOF
```

   Notice here that the directory superuser, `uid=admin`, assigns privileges. Any administrator with the `privilege-change` privilege can assign privileges. However, if the administrator can update administrator privileges, they can assign themselves the `bypass-acl` privilege. Then they are no longer bound by access control instructions, including both user data ACIs and global ACIs. For this reason, do not assign the `privilege-change` privilege to normal administrator users.

2. Create an LDAP subentry to specify the limits using collective attributes:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: cn=Remove Administrator Search Limits,dc=example,dc=com
objectClass: collectiveAttributeSubentry
objectClass: extensibleObject
objectClass: subentry
objectClass: top
cn: Remove Administrator Search Limits
ds-rlim-size-limit;collective: 0
ds-rlim-time-limit;collective: 0
subtreeSpecification: {base "ou=people", specificationFilter
  "(isMemberOf=cn=Directory Administrators,ou=Groups,dc=example,dc=com)" }
EOF
```

The `base` entry identifies the branch that holds administrator entries. For details on how subentries apply, see About subentry scope.

3. Check the results:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN uid=kvaughan,ou=people,dc=example,dc=com \
 --searchScope base \
 "(&)" \
 ds-rlim-time-limit ds-rlim-size-limit

dn: uid=kvaughan,ou=People,dc=example,dc=com
ds-rlim-size-limit: 0
ds-rlim-time-limit: 0
```

Limit persistent searches

An LDAP persistent search maintains an open a connection that may be idle for long periods of time. Whenever a modification changes data in the search scope, the server returns a search result. The more concurrent persistent searches, the more work the server has to do for each modification:

1. Set the global property `max-psearches` to limit total concurrent persistent searches.

   The following example limits the maximum number of persistent searchees to 30:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set max-psearches:30 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Connection limits

Limit total connections

Each connection uses memory. On UNIX and Linux systems, each connection uses an available file descriptor.

To limit the total number of concurrent client connections that the server accepts, use the global setting `max-allowed-client-connections`. The following example sets the limit to 64K. 64K is the minimum number of file descriptors that should be available to the DS server:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set max-allowed-client-connections:65536 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Restrict who can connect

To restrict which clients can connect to the server, use the global setting `allowed-client`, or `denied-client`. The following example restricts access to clients from the `example.com` domain:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set allowed-client:example.com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Set these properties per `Connection Handler`. The settings on a connection handler override the global settings.

Limit connections per client

To limit the number of concurrent connections from a client, use the global settings `restricted-client`, and `restricted-client-connection-limit`. The following example sets the limit for all clients on the `10.0.0.*` network to 1000 concurrent connections:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set restricted-client:"10.0.0.*" \
 --set restricted-client-connection-limit:1000 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Set these properties per `Connection Handler`. The settings on a connection handler override the global settings.

The server applies the properties in this order:

1. If the `denied-client` property is set, the server denies connections from any client matching the settings.
2. If the `restricted-client` property is set, the server checks the number of connections from any client matching the settings.

   If a matching client exceeds `restricted-client-connection-limit` connections, the server refuses additional connections.
3. If the `allowed-client` property is set, the server allows connections from any client matching the settings.
4. If none of the properties are set, the server allows connections from any client.

Idle time limits

If client applications leave connections idle for long periods, you can drop their connections by setting the global configuration property idle-time-limit. By default, no idle time limit is set.

If your network is configured to drop connections that have been idle for some time, set the DS idle time limit to a lower value than the idle time limit for the network. This helps to ensure that idle connections are shut down in orderly fashion. Setting the DS limit lower than the network limit is particularly useful with networks that drop idle connections without cleanly closing the connection and notifying the client and server.

> **Note**
>
> DS servers do not enforce idle timeout for persistent searches.

The following example sets the idle-time-limit to 24 hours:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set idle-time-limit:24h \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Request size limits

The default maximum request size is 5 MB. This is sufficient for most deployments. In cases where clients add groups with large numbers of members, requests can exceed the 5 MB limit.

The following example increases the limit to 20 MB for the LDAP connection handler:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name LDAP \
 --set max-request-size:20mb \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

This setting affects only the size of requests, not responses.

Move a server

The following procedure moves a server to the new host `new-server.example.com`. The steps skip creation of system accounts, startup scripts, and registration as a Windows service:

1. Stop the server:

   ```
   $ stop-ds
   ```

2. Renew the server certificate to account for the new hostname.

   Skip this step if the server certificate is a wildcard certificate that is already valid for the new hostname.

   The following command renews the server certificate generated with a deployment ID and password:

```
$ dskeymgr \
 create-tls-key-pair \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --keyStoreFile /path/to/opendj/config/keystore \
 --keyStorePassword:file /path/to/opendj/config/keystore.pin \
 --hostname localhost \
 --hostname new-server.example.com \
 --subjectDn CN=DS,O=ForgeRock
```

For more command options, refer to dskeymgr. The default validity for the certificate is one year.

3. Find and replace the old hostname with the new hostname in the server's configuration file, `config/config.ldif`.

   The following list includes configuration settings that may specify the server hostname:

   - `ds-cfg-advertised-listen-address`
   - `ds-cfg-bootstrap-replication-server`
   - `ds-cfg-listen-address`
   - `ds-cfg-server-fqdn`
   - `ds-cfg-source-address`

4. Move all files in the `/path/to/opendj` directory to the new server.

5. Start the server:

   ```
   $ start-ds
   ```

6. If the server you moved is referenced by others as a replication bootstrap server, update the replication bootstrap server configuration on those servers.

Performance tuning

Performance requirements

Your key performance requirement is to satisfy your users or customers with the resources available to you. Before you can solve potential performance problems, define what those users or customers expect. Determine which resources you will have to satisfy their expectations.

Service level objectives

A service level objective (SLO) is a target for a directory service level that you can measure quantitatively. If possible, base SLOs on what your key users expect from the service in terms of performance.

Define SLOs for at least the following areas:

- Directory service *response times*

  Directory service response times range from less than a millisecond on average, across a low latency connection on the same network, to however long it takes your network to deliver the response.

  More important than average or best response times is the response time distribution, because applications set timeouts based on worst case scenarios.

  An example response time performance requirement is, *Directory response times must average less than 10 milliseconds for all operations except searches returning more than 10 entries, with 99.9% of response times under 40 milliseconds.*

- Directory service *throughput*

  Directories can serve many thousands of operations per second. In fact there is no upper limit for read operations such as searches, because only write operations must be replicated. To increase read throughput, simply add additional replicas.

  More important than average throughput is peak throughput. You might have peak write throughput in the middle of the night when batch jobs update entries in bulk, and peak binds for a special event or first thing Monday morning.

  An example throughput performance requirement is, *The directory service must sustain a mix of 5,000 operations per second made up of 70% reads, 25% modifies, 3% adds, and 2% deletes.*

  Ideally, you mimic the behavior of key operations during performance testing, so that you understand the patterns of operations in the throughput you need to provide.

- Directory service *availability*

  DS software is designed to let you build directory services that are basically available, including during maintenance and even upgrade of individual servers.

  To reach very high levels of availability, you must also ensure that your operations execute in a way that preserves availability.

  Availability requirements can be as lax as a best effort, or as stringent as 99.999% or more uptime.

  Replication is the DS feature that allows you to build a highly available directory service.

- Directory service administrative support

  Be sure to understand how you support your users when they run into trouble.

  While directory services can help you turn password management into a self-service visit to a web site, some users still need to know what they can expect if they need your help.

Creating an SLO, even if your first version consists of guesses, helps you reduce performance tuning from an open-ended project to a clear set of measurable goals for a manageable project with a definite outcome.

Resource constraints

With your SLOs in hand, inventory the server, networks, storage, people, and other resources at your disposal. Now is the time to estimate whether it is possible to meet the requirements at all.

If, for example, you are expected to serve more throughput than the network can transfer, maintain high-availability with only one physical machine, store 100 GB of backups on a 50 GB partition, or provide 24/7 support all alone, no amount of tuning will fix the problem.

When checking that the resources you have at least theoretically suffice to meet your requirements, do not forget that high availability in particular requires at least two of everything to avoid single points of failure. Be sure to list the resources you expect to have, when and how long you expect to have them, and why you need them. Make note of what is missing and why.

Server hardware

DS servers are pure Java applications, making them very portable. DS servers tend to perform best on single-board, x86 systems due to low memory latency.

Storage

High-performance storage is essential for handling high-write throughput. When the database stays fully cached in memory, directory read operations do not result in disk I/O. Only writes result in disk I/O. You can further improve write performance by using solid-state disks for storage or file system cache.

> **Warning**
>
> DS directory servers are designed to work with *local storage* for database backends. *Do not use network file systems, such as NFS, where there is no guarantee that a single process has access to files.*
> Storage area networks (SANs) and attached storage are fine for use with DS directory servers.

Regarding database size on disk, sustained write traffic can cause the database to grow to more than twice its initial size on disk. This is normal behavior. The size on disk does not impact the DB cache size requirements.

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 set the journaling mode for data, see the options for your file system in the `mount` command manual page.

Performance tests

Even if you do not need high availability, you still need two of everything, because your test environment needs to mimic your production environment as closely as possible.

In your test environment, set up DS servers just as you do in production. Conduct experiments to determine how to best meet your SLOs.

The following command-line tools help with basic performance testing:

- The makeldif command generates sample data with great flexibility.
- The `addrate` command measures add and delete throughput and response time.
- The `authrate` command measures bind throughput and response time.
- The `modrate` command measures modification throughput and response time.
- The `searchrate` command measures search throughput and response time.

All `*rate` commands display response time distributions measurements, and support testing at specified levels of throughput.

For additional precision when evaluating response times, use the global configuration setting etime-resolution. To change elapsed processing time resolution from milliseconds (default) to nanoseconds:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set etime-resolution:nanoseconds \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The `etime`, recorded in the server access log, indicates the elapsed time to process the request. The `etime` starts when the decoded operation is available to be processed by a worker thread.

Test performance with your production-ready configuration. If, however, you simply want to demonstrate top performance, take the following points into account:

- Incorrect JVM tuning slows down server and tool performance. Make sure the JVM is tuned for best performance.

  For example, set the following environment variable, then restart the server and run the performance tools again to take the change into account:

  ```
  export OPENDJ_JAVA_ARGS="-XX:+UseParallelGC -XX:MaxTenuringThreshold=1"
  ```

  If the server heap is very large, see the details in Java settings.

- Unfiltered access logs record messages for each client request. Turn off full access logging.

  For example, set `enabled:false` for the `Json File-Based Access Logger` log publisher, and any other unfiltered log publishers that are enabled.

- Secure connections are recommended, and they can be costly.

  Set `require-secure-authentication:false` in the password policies governing the bind entries, and bind using insecure connections.

Performance settings

Use the following suggestions when your tests show that DS performance is lacking, even though you have the right underlying network, hardware, storage, and system resources in place.

Maximum open files

DS servers must open many file descriptors when handling thousands of client connections.

Linux systems often set a limit of 1024 per user. That setting is too low to accept thousands of client connections.

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, 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. Check the Linux system overall maximum as follows:

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

Linux page caching

Default Linux virtual memory settings cause significant buildup of dirty data pages before flushing them. When the kernel finally flushes the pages to disk, the operation can exhaust the disk I/O for up to several seconds. Application operations waiting on the file system to synchronize to disk are blocked.

The default virtual memory settings can therefore cause DS server operations to block for seconds at a time. Symptoms included high outlier etimes, even for very low average etimes. For sustained high loads, such as import operations, the server has to maintain thousands of open file descriptors.

To avoid these problems, tune Linux page caching. As a starting point for testing and tuning, set `vm.dirty_background_bytes` to one quarter of the disk I/O per second, and `vm.dirty_expire_centisecs` to 1000 (10 seconds) using the `sysctl` command. This causes the kernel to flush more often, and limits the pauses to a maximum of 250 milliseconds.

For example, if the disk I/O is 80 MB/second for writes, the following example shows an appropriate starting point. It updates the `/etc/sysctl.conf` file to change the setting permanently, and uses the `sysctl -p` command to reload the settings:

```
$ echo vm.dirty_background_bytes=20971520 | sudo tee -a /etc/sysctl.conf
[sudo] password for admin:

$ echo vm.dirty_expire_centisecs=1000 | sudo tee -a /etc/sysctl.conf

$ sudo sysctl -p
vm.dirty_background_bytes = 20971520
vm.dirty_expire_centisecs = 1000
```

Be sure to test and adjust the settings for your deployment.

For additional details, see the Oracle documentation on Linux Page Cache Tuning, and the Linux `sysctl` command virtual memory kernel reference.

Java settings

Default Java settings let you evaluate DS servers using limited system resources. For high performance production systems, test and run with a tuned JVM.

> **Tip**
>
> To apply JVM settings for a server, edit `config/java.properties`, and restart the server.

Availability of the following `java` options depends on the JVM:

`-Xmx`

If you observe any critical evictions, add more RAM to the system. If adding RAM is not an option, increase the maximum heap size to optimize RAM allocation. For details, see Cache internal nodes.

Use at least a 2 GB heap unless your data set is small.

`-XX:+DisableExplicitGC`

When using JMX, add this option to the list of `start-ds.java-args` arguments to avoid periodic full GC events.

JMX is based on RMI, which uses references to objects. By default, the JMX client and server perform a full GC periodically to clean up stale references. As a result, the default settings cause JMX to cause a full GC every hour.

Avoid using this argument with `import-ldif.offline.java-args` or when using the `import-ldif` command. The import process uses garbage collection to manage memory and references to memory-mapped files.

`-XX:MaxTenuringThreshold=1`

This sets the maximum number of GC cycles an object stays in survivor spaces before it is promoted into the old generation space.

Setting this option as suggested reduces the new generation GC frequency and duration. The JVM quickly promotes long-lived objects to the old generation space, rather than letting them accumulate in new generation survivor spaces, copying them for each GC cycle.

`-Xlog:gc=level:file`

Log garbage collection messages when diagnosing JVM tuning problems. You can turn the option off when everything is running smoothly.

Always specify the output file for the garbage collection log. Otherwise, the JVM logs the messages to the `opendj/logs/server.out` file, mixing them with other messages, such as stack traces from the `supportextract` command.

For example, `-Xlog:gc=info:file=/path/to/gc.log` logs informational messages about garbage collection to the file, `/path/to/gc.log`.

For details, use the `java -Xlog:help` command.

`-XX:TieredStopAtLevel=1`

Short-lived client tools, such as the `ldapsearch` command, start up faster when this option is set to `1` as shown.

`-XX:+UseG1GC -XX:MaxGCPauseMillis=100`

Java 11

Use G1 GC (the default) when the heap size is 8 GB or more.

Java 17

Use G1 GC.

`-XX:+UseParallelGC`

Java 11

Use parallel GC when the heap size is less than 8 GB.

Java 17

Use G1 GC instead.

Data storage settings

By default, DS servers compress attribute descriptions and object class sets to reduce data size. This is called compact encoding.

By default, DS servers do not compress entries stored in its backend database. If your entries hold values that compress well, such as text, you can gain space. Set the backend property `entries-compressed:true`, and reimport the data from LDIF. The DS server compresses entries before writing them to the database:

```
$ dsconfig \
 set-backend-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --set entries-compressed:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ import-ldif \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --ldifFile backup.ldif \
 --backendID dsEvaluation \
 --includeBranch dc=example,dc=com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

DS directory servers do not proactively rewrite all entries after you change the settings. To force the DS server to compress all entries, you must import the data from LDIF.

LDIF import settings

By default, the temporary directory used for scratch files is `opendj/import-tmp`. Use the `import-ldif --tmpDirectory` option to set this directory to a `tmpfs` file system, such as `/tmp`.

If you are certain your LDIF contains only valid entries with correct syntax, you can skip schema validation. Use the `import-ldif --skipSchemaValidation` option.

Database cache settings

> **Important**
>
> By default, DS directory servers:
>
> - Use shared cache for all JE database backends.
>   The recommended setting is to leave the global property, `je-backend-shared-cache-enabled`, set to `true`.
>   If you have more than one JE database backend, *before* you change this setting to `false`, you must set either `db-cache-percent` or `db-cache-size` appropriately for each JE backend. By default, `db-cache-percent` is 50% for each backend. If you have multiple backends, including backends created with setup profiles, the default settings can prevent the server from starting if you first disable the shared cache.
> - Cache JE database internal and leaf notes to achieve best performance.
>   The recommended setting is to leave this advanced property, `db-cache-mode`, set to `cache-ln`.
>   In very large directory deployments, monitor the server and minimize critical evictions. For details, see Cache internal nodes.

If you require fine-grained control over JE backend cache settings, you can configure the amount of memory requested for database cache per database backend:

1. Configure `db-cache-percent` or `db-cache-size` for each JE backend.

   `db-cache-percent`

   Percentage of JVM memory to allocate to the database cache for the backend.

   If the directory server has multiple database backends, the total percent of JVM heap used must remain less than 100 (percent), and must leave space for other uses.

   Default: 50 (percent)

   `db-cache-size`

   JVM memory to allocate to the database cache.

   This is an alternative to `db-cache-percent`. If you set its value larger than 0, then it takes precedence over `db-cache-percent`.

   Default: 0 MB

2. Set the global property `je-backend-shared-cache-enabled:false`.

3. Restart the server for the changes to take effect.

Cache internal nodes

A JE backend has a B-tree data structure. A B-tree consists of nodes that can have children. Nodes with children are *internal nodes*. Nodes without children are *leaf nodes*.

The directory stores data in key-value pairs. Internal nodes hold the keys and can hold small values. Leaf nodes hold the values. One internal node usually holds keys to values in many leaf nodes. A B-tree has many more leaf nodes than internal nodes.

To read a value by its key, the backend traverses all internal nodes on the branch from the B-tree root to the leaf node holding the value. The closer a node is to the B-tree root, the more likely the backend must access it to get to the value. In other words, the backend accesses internal nodes more often than leaf nodes.

When a backend accesses a node, it loads the node into the DB cache. Loading a node because it wasn't in cache is a *cache miss*. When you first start DS, all requests result in cache misses until the server loads active nodes.

As the DB cache fills, the backend makes space to load nodes by evicting nodes from the cache. The backend evicts leaf nodes, then least recently used internal nodes. As a last resort, the backend evicts even recently used internal nodes with changes not yet synced to storage.

The next time the backend accesses an evicted node, it must load the node from storage. Storage may mean the file system cache, or it may mean a disk. Reading from memory can be orders of magnitude faster than reading from disk. For the best DB performance, cache the nodes the DB accesses most often, which are the internal nodes.

Once DS has run for some time and active nodes are in cache, watch the cache misses for internal nodes. DS has "warmed up" and the active nodes are in the cache. The number of evicted internal nodes should remain constant. When the cache size is right, and no sudden changes occur in access patterns, the number of cache misses for internal nodes should stop growing:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN cn=monitor \
 "(objectClass=ds-monitor-backend-db)" \
 ds-mon-db-cache-evict-internal-nodes-count \
 ds-mon-db-cache-misses-internal-nodes
dn: ds-cfg-backend-id=dsEvaluation,cn=backends,cn=monitor
ds-mon-db-cache-evict-internal-nodes-count: <number>
ds-mon-db-cache-misses-internal-nodes: <number>
```

If `ds-mon-db-cache-evict-internal-nodes-count` is greater than `0` and growing, or `ds-mon-db-cache-misses-internal-nodes` continues to grow even after DS has warmed up, DS is evicting internal nodes from the DB cache.

If you can rule out big changes in access cache patterns like large unindexed searches, DS does not have enough space for the DB cache. Increase the DB cache size and add more RAM to your system if necessary. If adding RAM isn't an option, increase the maximum heap size (`-Xmx`) to optimize RAM allocation.

Estimate minimum DB cache size

This section explains how to estimate the minimum DB cache size.

The examples below use a directory server with a 10 million entry `dsEvaluation` backend. The backend holds entries generated using the `--set ds-evaluation/generatedUsers:10,000,000` setup option.

> **Important**
>
> Before estimating DB cache size for your deployment:
>
> - Configure the servers with production replication and indexing settings.
> - Import realistic data.
>   If you can't find test data matching production data, generate realistic data.
>   Immediately after import, a JE backend has the minimum number of internal nodes the data requires.
> - Simulate realistic traffic to your service.
>   Even better, learn about real loads from analysis of production access logs, and build custom test clients to match the access patterns of your applications.
>   The backend appends updates to its database log and cleans the database log in the background. Over time, as more updates occur, the number of internal nodes grows, tracking backend growth.

After simulating realistic traffic for some time, stop the server. Use the output from the `backendstat` command to estimate the required DB cache size JE `DbCacheSize` tool together to estimate the required DB cache size:

```
# Stop the server before using backendstat:
$ stop-ds

$ backendstat list-raw-dbs --backendId dsEvaluation
Raw DB Name                                                                                        Total Keys
Keys Size  Values Size  Total Size
------------------------------------------------------------------------------------------------------------------------
/compressed_schema/compressed_attributes
54                54          837         891
/compressed_schema/compressed_object_classes
18                18          938         956
/dc=com,dc=example/aci.presence
1                  1            3           4
/dc=com,dc=example/cn.caseIgnoreMatch                                                              10000165
139242471      47887210    187129681
/dc=com,dc=example/cn.caseIgnoreSubstringsMatch:6                                                  858658
5106085     204936279    210042364
/dc=com,dc=example/counter.objectClass.big.objectIdentifierMatch
2                 34            2          36
/dc=com,dc=example/dn2id                                                                           10000181
268892913      80001448    348894361
/dc=com,dc=example/ds-certificate-fingerprint.caseIgnoreMatch
0                  0            0           0
/dc=com,dc=example/ds-certificate-subject-dn.distinguishedNameMatch
1                 18            3          21
/dc=com,dc=example/ds-sync-conflict.distinguishedNameMatch
0                  0            0           0
/dc=com,dc=example/ds-sync-hist.changeSequenceNumberOrderingMatch
0                  0            0           0
/dc=com,dc=example/entryUUID.uuidMatch                                                             9988518
39954072      47871653     87825725
/dc=com,dc=example/givenName.caseIgnoreMatch
8614            51691      20017387     20069078
/dc=com,dc=example/givenName.caseIgnoreSubstringsMatch:6
19652           97670      48312528     48410198
/dc=com,dc=example/id2childrencount
8                 26           14          40
```

```
/dc=com,dc=example/id2entry                                                                    10000181
80001448    5379599451   5459600899
/dc=com,dc=example/json.caseIgnoreJsonQueryMatch
4                   56            8          64
/dc=com,dc=example/jsonToken.extensibleJsonEqualityMatch:caseIgnoreStrings:ignoreWhiteSpace:/id
2                   34            4          38
/dc=com,dc=example/mail.caseIgnoreIA5Match                                                     10000152
238891751     47887168   286778919
/dc=com,dc=example/mail.caseIgnoreIA5SubstringsMatch:6                                         1222798
7336758    112365097   119701855
/dc=com,dc=example/member.distinguishedNameMatch
1                   40            2          42
/dc=com,dc=example/oauth2Token.caseIgnoreOAuth2TokenQueryMatch
4                   74           10          84
/dc=com,dc=example/objectClass.big.objectIdentifierMatch
6                  156            0         156
/dc=com,dc=example/objectClass.objectIdentifierMatch
24                 396          395         791
/dc=com,dc=example/referral
0                    0            0           0
/dc=com,dc=example/sn.caseIgnoreMatch
13457           92943     20027045    20119988
/dc=com,dc=example/sn.caseIgnoreSubstringsMatch:6                                              41585
219522     73713958    73933480
/dc=com,dc=example/state
25                1271           24        1295
/dc=com,dc=example/telephoneNumber.telephoneNumberMatch                                        9989952
109889472     47873522   157762994
/dc=com,dc=example/telephoneNumber.telephoneNumberSubstringsMatch:6                            1111110
6543210    221281590   227824800
/dc=com,dc=example/uid.caseIgnoreMatch                                                         10000152
118889928     47887168   166777096
/dc=com,dc=example/uniqueMember.uniqueMemberMatch
10                 406           21         427

Total: 32

# Calculate the sum of total keys, the average key size, and the average value size.
# Sum of total keys: 73255335
# Average key size: sum of key sizes/sum of total keys = 1015212518 / 73255335 ~= 13.86
# Average value size: sum of values sizes/sum of total keys = 6399663765 / 73255335 ~= 87.36

# Use the results rounded to the nearest integer as arguments to the DbCacheSize tool:
$ java -cp editable:dsDockerBase[/path/to/opendj]/lib/opendj.jar com.sleepycat.je.util.DbCacheSize \
 -records 73255335 -key 14 -data 87

=== Environment Cache Overhead ===

3,158,773 minimum bytes

To account for JE daemon operation, record locks, HA network connections, etc,
a larger amount is needed in practice.

=== Database Cache Size ===
```

```
   Number of Bytes  Description
   ---------------  -----------
     2,953,929,424  Internal nodes only
    12,778,379,408  Internal nodes and leaf nodes

For further information see the DbCacheSize javadoc.
```

The resulting recommendation for caching `Internal nodes only` is 2,953,929,424 bytes (~ 3 GB) in this example. This setting for DB cache includes space for all internal nodes, including those with keys and data. To cache all DB data, `Internal nodes and leaf nodes`, would require 12,778,379,408 (~13 GB).

Round up when configuring backend settings for `db-cache-percent` or `db-cache-size`. If the system in this example has 8 GB available memory, use the default setting of `db-cache-percent: 50`. (50% of 8 GB is 4 GB, which is larger than the minimum estimate.)

Database log file settings

With default settings, if the database has more than 200 files on disk, then the JE backend must start closing one log file in order to open another. This has serious impact on performance when the file cache starts to thrash.

Having the JE backend open and close log files from time to time is okay. Changing the settings is only necessary if the JE backend has to open and close the files very frequently.

A JE backend stores data on disk in append-only log files. The maximum size of each log file is configurable. A JE backend keeps a configurable maximum number of log files open, caching file handles to the log files. The relevant JE backend settings are the following:

`db-log-file-max`

Maximum size of a database log file.

Default: 1 GB

`db-log-filecache-size`

File handle cache size for database log files.

Default: 200

With these defaults, if the size of the database reaches 200 GB on disk (1 GB x 200 files), the JE backend must close one log file to open another. To avoid this situation, increase `db-log-filecache-size` until the JE backend can cache file handles to all its log files. When changing the settings, make sure the maximum number of open files is sufficient.

Cache for large groups

DS servers implement an entry cache designed for a few large entries that are regularly updated or accessed, such as large static groups. An entry cache is used to keep such groups in memory in a format that avoids the need to constantly read and deserialize the large entries.

When configuring an entry cache, take care to include only the entries that need to be cached. The memory devoted to the entry cache is not available for other purposes. Use the configuration properties `include-filter` and `exclude-filter` for this.

The following example adds a Soft Reference entry cache to hold entries that match the filter `(ou=Large Static Groups)`. A Soft Reference entry cache releases entries when the JVM runs low on memory. It does not have a maximum size setting. The number of entries cached is limited only by the `include-filter` and `exclude-filter` settings:

```
$ dsconfig \
 create-entry-cache \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --cache-name "Large Group Entry Cache" \
 --type soft-reference \
 --set cache-level:1 \
 --set include-filter:"(ou=Large Static Groups)" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The entry cache configuration takes effect when the entry cache is enabled.

Log settings

Debug logs trace the internal workings of DS servers, and should be used sparingly. Be particularly careful when activating debug logging in high-performance deployments.

In general, leave other logs active for production environments to help troubleshoot any issues that arise.

For servers handling 100,000 operations per second or more, the access log can be a performance bottleneck. Each client request results in at least one access log message. Test whether disabling the access log improves performance in such cases.

The following command disables the JSON-based LDAP access logger:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The following command disables the HTTP access logger:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Changelog settings

By default, a replication server indexes change numbers for replicated user data. This allows legacy applications to get update notifications by change number, as described in Align draft change numbers. Indexing change numbers requires additional CPU, disk accesses and storage, so it should not be used unless change number-based browsing is required.

Disable change number indexing if it is not needed. For details, see Disable change number indexing.

Troubleshooting

Define the problem

To solve your problem, save time by clearly defining it first. A problem statement *compares the difference between observed behavior and expected behavior*:

- What exactly is the problem?

  What is the behavior you expected?

  What is the behavior you observed?

- How do you reproduce the problem?
- When did the problem begin?

  Under similar circumstances, when does the problem not occur?

- Is the problem permanent?

  Intermittent?

  Is it getting worse? Getting better? Staying the same?

Performance

Before troubleshooting performance, make sure:

- The system meets the DS installation requirements.
- The performance expectations are reasonable.

For example, a deployment can use password policies with cost-based, resource-intensive password storage schemes such as Argon2, Bcrypt, or PBKDF2. This protects passwords at the cost of slow LDAP simple binds or HTTP username/password authentications and lower throughput.

When directory operations take too long, meaning request latency is high, fix the problem first in your test or staging environment. Perform these steps in order and stop when you find a fix:

1. Check for unindexed searches and prevent them when possible.

   Unindexed searches are expensive operations, particularly for large directories. When unindexed searches consume the server's resources, performance suffers for concurrent operations and for later operations if an unindexed search causes widespread changes to database and file system caches.

2. Check performance settings for the server including JVM heap size and DB cache size.

   Try adding more RAM if memory seems low.

3. Read the request queue monitoring statistics over LDAP or over HTTP.

   If many requests are in the queue, the troubleshooting steps are different for read and write operations. Read and review the request statistics available over LDAP or over HTTP.

   If you persistently have many:

   - Pending read requests, such as unindexed searches or big searches, try adding CPUs.
   - Pending write requests, try adding IOPS, such as faster or higher throughput disks.

Installation problems

Use the logs

Installation and upgrade procedures result in a log file tracing the operation. Look for this in the command output:

```
See file for a detailed log of this operation.
```

Antivirus interference

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

Before using DS 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 DS file access, particularly write access.

Antivirus and intrusion detection software have incorrectly marked DS files as suspect to infection, because they misinterpret normal DS processing.

Prevent antivirus and intrusion detection systems from scanning DS files, except these folders:

`/path/to/opendj/bat/`

Windows command-line tools

`/path/to/opendj/bin/`

UNIX/Linux command-line tools

`/path/to/opendj/extlib/`

Optional additional `.jar` files used by custom plugins

`/path/to/opendj/lib/`

Scripts and libraries shipped with DS servers

Port blocking

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

Make sure that your software does not block the ports that DS software uses. For details, see Administrative access.

Negative performance impact

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

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

JE initialization

When starting a directory server on a Linux system, make sure the server user can watch enough files. If the server user cannot watch enough files, you might see an error message in the server log such as this:

```
InitializationException: The database environment could not be opened:
com.sleepycat.je.EnvironmentFailureException: (JE version) /path/to/opendj/db/userData
or its sub-directories to WatchService.
UNEXPECTED_EXCEPTION: Unexpected internal Exception, may have side effects.
Environment is invalid and must be closed.
```

File notification

A directory server backend database monitors file events. On Linux systems, backend databases use the inotify API for this purpose. The kernel tunable `fs.inotify.max_user_watches` indicates the maximum number of files a user can watch with the inotify API.

Make sure this tunable is set to at least 512K:

```
$ sysctl fs.inotify.max_user_watches

fs.inotify.max_user_watches = 524288
```

If this tunable is set lower than that, update the `/etc/sysctl.conf` file to change the setting permanently, and use the `sysctl -p` command to reload the settings:

```
$ echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf
[sudo] password for admin:

$ sudo sysctl -p
fs.inotify.max_user_watches = 524288
```

Forgotten superuser password

By default, DS servers store the entry for the directory superuser in an LDIF backend. Edit the file to reset the password:

1. Generate the encoded version of the new password:

   ```
   $ encode-password --storageScheme PBKDF2-HMAC-SHA256 --clearPassword password

   {PBKDF2-HMAC-SHA256}10<hash>
   ```

2. Stop the server while you edit the LDIF file for the backend:

   ```
   $ stop-ds
   ```

3. Replace the existing password with the encoded version.

   In the `db/rootUser/rootUser.ldif` file, carefully replace the `userPassword` value with the new, encoded password:

   ```
   dn: uid=admin
   ...
   uid: admin
   userPassword: <encoded-password>
   ```

   Trailing whitespace is significant in LDIF. *Take care not to add any trailing whitespace at the end of the line.*

4. Restart the server:

   ```
   $ start-ds
   ```

5. Verify that you can use the directory superuser account with the new password:

```
$ status \
 --bindDn uid=admin \
 --bindPassword password \
 --hostname localhost \
 --port 4444 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --script-friendly
...
"isRunning" : true,
```

Debug logging

> **Caution**
>
> DS debug logging can generate a high volume of debug messages. Use debug logging very sparingly on production systems.

1. Create one or more debug targets.

   No debug targets are enabled by default:

   ```
   $ dsconfig \
    list-debug-targets \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --publisher-name "File-Based Debug Logger" \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt

   Debug Target : enabled : debug-exceptions-only
   -------------:---------:----------------------
   ```

   A debug target specifies a fully qualified DS Java package, class, or method:

   ```
   $ dsconfig \
    create-debug-target \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --publisher-name "File-Based Debug Logger" \
    --type generic \
    --target-name org.opends.server.api \
    --set enabled:true \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

2. Enable the debug log, `opendj/logs/debug` :

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

   The server immediately begins to write debug messages to the log file.

3. Read messages in the debug log file:

```
$ tail -f /path/to/opendj/logs/debug
```

4. Disable the debug log as soon as it is no longer required.

Lockdown mode

Misconfiguration can put the DS server in a state where you must prevent users and applications from accessing the directory until you have fixed the problem.

DS servers support *lockdown mode* . Lockdown mode permits connections only on the loopback address, and permits only operations requested by superusers, such as `uid=admin` .

To put the DS server into lockdown mode, the server must be running. You cause the server to enter lockdown mode by starting a task. Notice that the modify operation is performed over the loopback address (accessing the DS server on the local host):

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: ds-task-id=Enter Lockdown Mode,cn=Scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
ds-task-id: Enter Lockdown Mode
ds-task-class-name: org.opends.server.tasks.EnterLockdownModeTask
EOF
```

The DS server logs a notice message in `logs/errors` when lockdown mode takes effect:

```
...msg=Lockdown task Enter Lockdown Mode finished execution
```

Client applications that request operations get a message concerning lockdown mode:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN "" \
 --searchScope base \
 "(objectclass=*)" \
 +

# The LDAP search request failed: 53 (Unwilling to Perform)
# Additional Information:  Rejecting the requested operation because the server is in lockdown mode and will only
accept requests from root users over loopback connections
```

Leave lockdown mode by starting a task:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: ds-task-id=Leave Lockdown Mode,cn=Scheduled Tasks,cn=tasks
objectClass: top
objectClass: ds-task
ds-task-id: Leave Lockdown Mode
ds-task-class-name: org.opends.server.tasks.LeaveLockdownModeTask
EOF
```

The DS server logs a notice message when leaving lockdown mode:

```
...msg=Leave Lockdown task Leave Lockdown Mode finished execution
```

LDIF import

- By default, DS directory servers check that entries you import match the LDAP schema.

  You can temporarily bypass this check with the `import-ldif --skipSchemaValidation` option.

- By default, DS servers ensure that entries have only one structural object class.

  You can relax this behavior with the advanced global configuration property, `single-structural-objectclass-behavior`.

  This can be useful when importing data exported from Sun Directory Server.

  For example, warn when entries have more than one structural object class, rather than rejecting them:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set single-structural-objectclass-behavior:warn \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

- By default, DS servers check syntax for several attribute types. Relax this behavior using the advanced global configuration property, `invalid-attribute-syntax-behavior`.
- Use the `import-ldif -R rejectFile --countRejects` options to log rejected entries and to return the number of rejected entries as the command's exit code.

Once you resolve the issues, reinstate the default behavior to avoid importing bad data.

Security problems

Incompatible Java versions

Due to a change in Java APIs, the same DS deployment ID generates different CA key pairs with Java 11 and Java 17 and later. When running the `dskeymgr` and `setup` commands, use the same Java environment everywhere in the deployment.

Using different Java versions is a problem if you use deployment ID-based CA certificates. Replication breaks, for example, when you use the `setup` command for a new server with a more recent version of Java than was used to set up existing servers. The error log includes a message such as the following:

```
...category=SYNC severity=ERROR msgID=119 msg=Directory server DS(server_id)
encountered an unexpected error while connecting to replication server host:port for domain "base_dn":
ValidatorException: PKIX path validation failed: java.security.cert.CertPathValidatorException:
signature check failed
```

To work around the issue, follow these steps:

1. Update all DS servers to use the same Java version.

   Make sure you have a required Java environment installed on the system.

   If your default Java environment is not appropriate, use one of the following solutions:

   - Edit the `default.java-home` setting in the `opendj/config/java.properties` file.
   - Set `OPENDJ_JAVA_HOME` to the path to the correct Java environment.
   - Set `OPENDJ_JAVA_BIN` to the absolute path of the `java` command.

2. Export CA certificates generated with the different Java versions.

   1. Export the CA certificate from an old server:

```
$ keytool \
 -exportcert \
 -alias ca-cert \
 -keystore /path/to/old-server/config/keystore \
 -storepass:file /path/to/old-server/config/keystore.pin \
 -file java11-ca-cert.pem
```

2. Export the CA certificate from a new server:

```
$ keytool \
 -exportcert \
 -alias ca-cert \
 -keystore /path/to/new-server/config/keystore \
 -storepass:file /path/to/new-server/config/keystore.pin \
 -file java17-ca-cert.pem
```

3. On *all* existing DS servers, import the *new* CA certificate:

```
$ keytool \
 -importcert \
 -trustcacerts \
 -alias alt-ca-cert \
 -keystore /path/to/old-server/config/keystore \
 -storepass:file /path/to/old-server/config/keystore.pin \
 -file java17-ca-cert.pem \
 -noprompt
```

4. On *all* new DS servers, import the *old* CA certificate:

```
$ keytool \
 -importcert \
 -trustcacerts \
 -alias alt-ca-cert \
 -keystore /path/to/new-server/config/keystore \
 -storepass:file /path/to/new-server/config/keystore.pin \
 -file java11-ca-cert.pem \
 -noprompt
```

The servers reload their keystores dynamically and replication works as expected.

Certificate-based authentication

Replication uses TLS to protect directory data on the network. Misconfiguration can cause replicas to fail to connect due to handshake errors. This leads to repeated error log messages in the `replication` log file such as the following:

```
...msg=Replication server accepted a connection from address
 to local address address but the SSL handshake failed.
 This is probably benign, but may indicate a transient network outage
 or a misconfigured client application connecting to this replication server.
 The error was: Received fatal alert: certificate_unknown
```

You can collect debug trace messages to help determine the problem. To see the TLS debug messages, start the server with `javax.net.debug` set:

```
$ OPENDJ_JAVA_ARGS="-Djavax.net.debug=all" start-ds
```

The debug trace settings result in many, many messages. To resolve the problem, review the output of starting the server, looking in particular for handshake errors.

If the chain of trust for your PKI is broken somehow, consider renewing or replacing keys, as described in Key management. Make sure that trusted CA certificates are configured as expected.

FIPS and key wrapping

DS servers use shared asymmetric keys to protect shared symmetric secret keys for data encryption.

By default, DS uses direct encryption to protect the secret keys.

When using a FIPS-compliant security provider that doesn't allow direct encryption, such as Bouncy Castle, change the Crypto Manager configuration to set the advanced property, `key-wrapping-mode: WRAP`. With this setting, DS uses wrap mode to protect the secret keys in a compliant way.

Compromised keys

How you handle the problem depends on which key was compromised:

- For keys generated by the server, or with a deployment ID and password, see Retire secret keys.
- For a private key whose certificate was signed by a CA, contact the CA for help. The CA might choose to publish a certificate revocation list (CRL) that identifies the certificate of the compromised key.

  Replace the key pair that has the compromised private key.
- For a private key whose certificate was self-signed, replace the key pair that has the compromised private key.

  Make sure the clients remove the compromised certificate from their truststores. They must replace the certificate of the compromised key with the new certificate.

Client problems

Use the logs

By default, DS servers record messages for LDAP client operations in the `logs/ldap-access.audit.json` log file.

```
[
{
  "eventName": "DJ-LDAP",
  "client": {
    "ip": "<clientIp>",
    "port": 12345
  },
  "server": {
    "ip": "<clientIp>",
    "port": 1389
  },
  "request": {
    "protocol": "LDAP",
    "operation": "CONNECT",
    "connId": 0
  },
  "transactionId": "0",
  "response": {
    "status": "SUCCESSFUL",
    "statusCode": "0",
    "elapsedTime": 0,
    "elapsedTimeUnits": "MILLISECONDS"
  },
  "timestamp": "<timestamp>",
  "_id": "<uuid>"
},
{
  "eventName": "DJ-LDAP",
  "client": {
    "ip": "<clientIp>",
    "port": 12345
  },
  "server": {
    "ip": "<clientIp>",
    "port": 1389
  },
  "request": {
    "protocol": "LDAP",
    "operation": "SEARCH",
    "connId": 0,
    "msgId": 1,
    "dn": "dc=example,dc=com",
    "scope": "sub",
    "filter": "(uid=bjensen)",
    "attrs": ["ALL"]
  },
  "transactionId": "0",
  "response": {
    "status": "SUCCESSFUL",
    "statusCode": "0",
    "elapsedTime": 9,
    "elapsedTimeUnits": "MILLISECONDS",
    "nentries": 1
  },
  "timestamp": "<timestamp>",
  "_id": "<uuid>"
},
{
  "eventName": "DJ-LDAP",
  "client": {
```

```
      "ip": "<clientIp>",
      "port": 12345
    },
    "server": {
      "ip": "<clientIp>",
      "port": 1389
    },
    "request": {
      "protocol": "LDAP",
      "operation": "UNBIND",
      "connId": 0,
      "msgId": 2
    },
    "transactionId": "0",
    "timestamp": "<timestamp>",
    "_id": "<uuid>"
  },
  {
    "eventName": "DJ-LDAP",
    "client": {
      "ip": "<clientIp>",
      "port": 12345
    },
    "server": {
      "ip": "<clientIp>",
      "port": 1389
    },
    "request": {
      "protocol": "LDAP",
      "operation": "DISCONNECT",
      "connId": 0
    },
    "transactionId": "0",
    "response": {
      "status": "SUCCESSFUL",
      "statusCode": "0",
      "elapsedTime": 0,
      "elapsedTimeUnits": "MILLISECONDS",
      "reason": "Client Unbind"
    },
    "timestamp": "<timestamp>",
    "_id": "<uuid>"
  }
  ]
```


Each message specifies the operation performed, the client that requested the operation, and when it completed.

By default, the server does not log internal LDAP operations corresponding to HTTP requests. To match HTTP client operations to internal LDAP operations:

1. Prevent the server from suppressing log messages for internal operations.

   Set `suppress-internal-operations:false` on the LDAP access log publisher.

2. Match the `request/connId` field in the HTTP access log with the same field in the LDAP access log.

Client access

To help diagnose client errors due to access permissions, see Effective rights.

Simple paged results

For some versions of Linux, you see a message in the DS access logs such as the following:

```
The request control with Object Identifier (OID) "1.2.840.113556.1.4.319"
cannot be used due to insufficient access rights
```

This message means clients are trying to use the simple paged results control without authenticating. By default, a global ACI allows only authenticated users to use the control.

To grant anonymous (unauthenticated) user access to the control, add a global ACI for anonymous use of the simple paged results control:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword "password" \
 --add global-aci:"(targetcontrol=\"SimplePagedResults\") \
 (version 3.0; acl \"Anonymous simple paged results access\"; allow(read) \
 userdn=\"ldap:///anyone\";)" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Replication problems

Replicas do not connect

If you set up servers with different deployment IDs, they cannot share encrypted data. By default, they also cannot trust each other's secure connections. You may see messages like the following in the `logs/replication` log file:

```
msg=Replication server accepted a connection from /address:port
to local address /address:port but the SSL handshake failed.
```

Unless the servers use your own CA, make sure their keys are generated with the same deployment ID/password. Either set up the servers again with the same deployment ID, or see Replace deployment IDs.

Temporary delays

Replication can generally recover from conflicts and transient issues. Temporary delays are normal and expected while replicas converge, especially when the write load is heavy. This is a feature of eventual convergence, not a bug.

For more information, see Replication delay (LDAP).

Use the logs

Replication uses its own error log file, `logs/replication`. Error messages in the log file have `category=SYNC`.

The messages have the following form. The following example message is folded for readability:

```
...msg=Replication server accepted a connection from 10.10.0.10/10.10.0.10:52859
 to local address 0.0.0.0/0.0.0.0:8989 but the SSL handshake failed.
 This is probably benign, but may indicate a transient network outage
 or a misconfigured client application connecting to this replication server.
 The error was: Remote host closed connection during handshake
```

Stale data

DS servers maintain historical information to bring replicas up to date, and to resolve conflicts. To prevent historical information from growing without limit, servers purge historical information after a configurable delay (replication-purge-delay, default: 3 days). A replica can become irrevocably out of sync if you restore it from a backup that is older than the purge delay, or if you stop it for longer than the purge delay. If this happens, reinitialize the replica from a recent backup or from a server that is up to date.

Incorrect configuration

When replication is configured incorrectly, fixing the problem can involve adjustments on multiple servers. For example, adding or removing a bootstrap replication server means updating the `bootstrap-replication-server` settings in the synchronization provider configuration of other servers. (The settings can be hard-coded in the configuration, or read from the environment at startup time, as described in Property value substitution. In either case, changing them involves at least restarting the other servers.)

For details, see sections in Replication.

Support

Sometimes you cannot resolve a problem yourself, and must ask for help or technical support. In such cases, identify the problem and how you reproduce it, and the version where you see the problem:

```
$ status --offline --version

ForgeRock Directory Services 7.2.5-20240524201627-49403efab3a3556b93d8ee62f263ca29a7e752ef
Build <datestamp>
```

Be prepared to provide the following additional information:

- The Java home set in `config/java.properties`.
- Access and error logs showing what the server was doing when the problem started occurring.
- A copy of the server configuration file, `config/config.ldif`, in use when the problem started occurring.
- Other relevant logs or output, such as those from client applications experiencing the problem.
- A description of the environment where the server is running, including system characteristics, hostnames, IP addresses, Java versions, storage characteristics, and network characteristics. This helps to understand the logs, and other information.
- The `.zip` file generated using the `supportextract` command.

  For an example showing how to use the command, see supportextract.

Logging

This guide covers DS server logs and logging options.

Logs

Understand server logs.

HTTP

Configure HTTP logs.

LDAP

Configure LDAP logs.

Log to Service

Log to local or remote services.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

About logs

Type	Description
Access	Messages about clients accessing the server. Each message includes a datestamp, information about the connection, and information about the operation. DS servers implement access logs for HTTP and LDAP. It is possible to configure multiple access logs at the same time. Do not enable multiple *unfiltered* file-based access loggers for the same protocol, however. This can put significant write load on the disk subsystem for access log files, because every client request results in at least one new log message.

Type	Description
Audit	Records changes to directory data in LDIF. DS servers implement an audit log as a special type of file-based access log. By default, the server writes messages to `opendj/logs/audit`. For an example, see Enable an audit log.
Debug	Messages tracing internal server events, for troubleshooting. By default, this is a file-based log, written to `opendj/logs/debug`. Debug logs can grow large quickly, and therefore no debug logs are enabled by default. For debug logging, you must set a *debug target* to control what gets logged. For details, see Debug logging.
Error	Messages tracing server events, error conditions, and warnings, categorized and identified by severity. By default, this is a file-based log, written to `opendj/logs/errors`. Messages have the following format: `[datestamp] category=category severity=severity msgID=ID number msg=message string` For lists of severe and fatal error messages by category, see Log message reference.
Replication repair	Messages to help repair problems in data replication. This is a file-based log, written to `opendj/logs/replication`. Messages have the following format: `[datestamp] category=SYNC severity=severity msgID=ID number msg=message string` The replication log does not trace replication operations. Use the external changelog instead to get notifications about changes to directory data. For details, see Changelog for notifications.
Server	Messages about server events since startup. This is a file-based log, written to `opendj/logs/server.out`. A `opendj/logs/server.pid` process ID file is also available when the server is running. Messages in this file have the same format as error log messages.

You configure logging using *log publishers*. Log publishers determine which messages to publish, where to publish them, and what output format to use.

DS server logging supports extensibility through the ForgeRock Common Audit event framework. Common Audit deals with any event you can audit, not only the data updates recorded in a directory audit log. The ForgeRock Common Audit event framework provides log handlers for publishing to local files or to remote systems.

Common ForgeRock access logs

DS servers support the ForgeRock Common Audit event framework. The log message formats are compatible for all products using the framework. The framework uses transaction IDs to correlate requests as they traverse the platform. This makes it easier to monitor activity and to enrich reports:

- The ForgeRock Common Audit event framework is built on *audit event handlers*. Audit event handlers can encapsulate their own configurations. Audit event handlers are the same in each product in the ForgeRock platform. You can plug in custom handlers that comply with the framework without having to upgrade the server.

- The ForgeRock Common Audit event framework includes handlers for logging to local files and to external services.

  Although the ForgeRock Common Audit event framework supports multiple topics, DS software currently supports handling only access events. DS software divides access events into `ldap-access` events and `http-access` events.

- Common Audit transaction IDs are not recorded by default. To record transaction IDs in the access logs, configure the DS server to trust them.

Common Audit LDAP events have the following format:

```
{
  "eventName": "DJ-LDAP",
  "client": {
    "ip": string,                          // Client IP address
    "port": number                         // Client port number
  },
  "server": {
    "ip": string,                          // Server IP address
    "port": number                         // Server port number
  },
  "request": {                             // LDAP request
    "attrs": [ string ],                   // Requested attributes
    "authType": string,                    // Bind type such as "SIMPLE"
    "connId": number,                      // Connection ID
    "controls": string,                    // Request controls
    "deleteOldRDN": boolean,               // For a modify DN request
    "dn": string,                          // Bind DN
    "filter": string,                      // Search filter
    "idToAbandon": number,                 // ID to use to abandon operation
    "message": string,                     // Localized request message
    "msgId": number,                       // Message ID
    "name": string,                        // Operation name
    "newRDN": string,                      // For a modify DN request
    "newSup": string,                      // For a modify DN request
    "oid": string,                         // Operation name or OID
    "operation": string,                   // Examples: "CONNECT", "BIND", "SEARCH"
    "opType": "sync",                      // Replication operation
    "protocol": "LDAP",
    "runAs": string,                       // Authorization ID
    "scope": string,                       // Search scope such as "sub"
    "version": string                      // Version "2", "3"
  },
  "response": {
    "additionalItems": string              // Additional information
    "controls": string,                    // Response controls
    "elapsedTime": number,                 // Number of time units
    "elapsedTimeUnits": string,            // Time unit such as "MILLISECONDS"
    "entrySize": number,                   // Entry size in bytes
    "failureReason": string,               // Human-readable information
    "maskedMessage": string,               // Real, masked result message
    "maskedResult": string,                // Real, masked result code
    "nentries": number,                    // Number of entries returned
    "reason": string,                      // Reason for disconnect
    "status": string,                      // "SUCCESSFUL", "FAILED"
    "statusCode": string                   // For example, "0" for success
  },
  "timestamp": string,                     // UTC date
  "transactionId": string,                 // Unique ID for the transaction
  "userId": string,                        // User who requested the operation
  "_id": string                            // Unique ID for the operation
}
```

Common Audit HTTP events have the following format:

```
{
  "eventName": "DJ-HTTP",
  "client": {
    "ip": string,                          // Client IP address
    "port": number                         // Client port number
  },
  "server": {
    "ip": string,                          // Server IP address
    "port": number                         // Server port number
  },
  "http": {                                // HTTP request and response
    "request": {
      "secure": boolean,                   // HTTP: false; HTTPS: true
      "method": string,                    // Examples: "GET", "POST", "PUT"
      "path": string,                      // URL
      "queryParameters": map,              // map: { key-string: [ value-string ] }
      "cookies": map                       // map: { key-string: [ value-string ] }
    },
    "response": {
      "headers": map                       // map: { key-string: [ value-string ] }
    }
  },
  "response": {
    "detail": string,                      // Human-readable information
    "elapsedTime": number,                 // Number of time units
    "elapsedTimeUnits": string,            // Time unit such as "MILLISECONDS"
    "status": string,                      // "SUCCESSFUL", "FAILED"
    "statusCode": string                   // For example, "0" for success
  },
  "timestamp": string,                     // UTC date
  "transactionId": string,                 // Unique ID for the transaction
  "trackingIds": [ string ],               // Unique IDs from the transaction context
  "userId": string,                        // User who requested the operation
  "_id": string                            // Unique ID for the operation
}
```

Access log filtering

With the default access log configuration (no filtering), for every client application request, the server writes at least one message to its access log. This volume of logging gives you the information to analyze overall access patterns, or to audit access when you do not know in advance what you are looking for.

When you do know what you are looking for, log filtering lets you throttle logging to focus on what you want to see. You specify the criteria for a filtering policy, and apply the policy to a log publisher.

Log filtering policies use the following criteria:

- Client IP address, bind DN, group membership
- Operation type (abandon, add, bind, compare, connect, delete, disconnect, extended operation, modify, rename, search, and unbind)
- Minimum entry size
- Port number

- Protocol used
- Response time
- Result codes (only log error results, for example)
- Search response criteria (number of entries returned, unindexed search, and others)
- Target DN
- User DN and group membership

A log publisher's filtering policy determines whether to include or exclude log messages that match the criteria.

For examples, see Filter out administrative messages and Audit configuration changes.

Log HTTP access to files

JSON format

When you install DS using procedures from Installation, the default JSON-based HTTP access log file is `logs/http-access.audit.json`. The name of the access log publisher in the configuration is `Json File-Based HTTP Access Logger`.

The sample DS Docker image logs to standard output instead of files. This makes it easy to read log messages with the `docker logs` command, and is a pattern you should follow when creating your own DS Docker images. The name of the LDAP access log publisher configuration in the sample image is `Console HTTP Access Logger`:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs correlate audit events, tracing the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal DS request control. They are sent over HTTP in an HTTP header.

   By default, DS servers do not trust transaction IDs sent with client application requests.

   When a server trusts transaction IDs from client application requests, outgoing requests reuse the incoming ID. For each outgoing request in the transaction, the request's transaction ID has the form `original-transaction-id/sequence-number`, where sequence-number reflects the position of the request in the series of requests for this transaction. For example, if the original-transaction-id is `abc123`, the first outgoing request has the transaction ID `abc123/0`, the second `abc123/1`, the third `abc123/2`, and so on. This lets you distinguish specific requests within a transaction when correlating audit events from multiple services.

   To trust transactions, set the advanced global server property, `trust-transaction-ids:true`:

```
$ dsconfig \
 set-global-configuration-prop \
 --advanced \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set trust-transaction-ids:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Edit the default HTTP access log publisher as necessary.

   The following example enables the default log publisher for DS installed locally, not in a Docker image:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based HTTP Access Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

CSV format

A CSV handler sends messages to a comma-separated variable (CSV) file.

> **Important**
>
> The CSV handler does not sanitize messages when writing to CSV log files.
> Do not open CSV logs in spreadsheets and other applications that treat data as code.

The default CSV HTTP access log file is `logs/http-access.csv`:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs correlate audit events, tracing the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal DS request control. They are sent over HTTP in an HTTP header.

   By default, DS servers do not trust transaction IDs sent with client application requests.

When a server trusts transaction IDs from client application requests, outgoing requests reuse the incoming ID. For each outgoing request in the transaction, the request's transaction ID has the form `original-transaction-id/sequence-number`, where sequence-number reflects the position of the request in the series of requests for this transaction. For example, if the original-transaction-id is `abc123`, the first outgoing request has the transaction ID `abc123/0`, the second `abc123/1`, the third `abc123/2`, and so on. This lets you distinguish specific requests within a transaction when correlating audit events from multiple services.

To trust transactions, set the advanced global server property, `trust-transaction-ids:true`:

```
$ dsconfig \
 set-global-configuration-prop \
 --advanced \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set trust-transaction-ids:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Create an enabled CSV file HTTP access logger with optional rotation and retention policies:

```
$ dsconfig \
 create-log-publisher \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Common Audit Csv File HTTP Access Logger" \
 --type csv-file-http-access \
 --set enabled:true \
 --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
 --set "rotation-policy:Size Limit Rotation Policy" \
 --set "retention-policy:File Count Retention Policy" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. For tamper-evident logs, follow these steps.

> **Important**
>
> Tamper-evident logging relies on digital signatures and regularly flushing messages to the log system. In high-volume directory deployments with heavy access patterns, signing log messages has a severe negative impact on server performance, reducing throughput by orders of magnitude.
> Be certain to test the performance impact with realistic access patterns for your deployment before enabling the feature in production.

   1. Prepare a keystore.

      For details, see Make tampering evident.

2. Enable the tamper-evident capability:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Common Audit Csv File HTTP Access Logger" \
 --set tamper-evident:true \
 --set key-store-file:config/audit-keystore \
 --set key-store-pin:"&{audit.keystore.pin}" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

In this example, `AUDIT_KEYSTORE_PIN` is an environment variable containing the keystore PIN.

Standard HTTP format

For HTTP requests, you can configure an access logger that uses the Extended Log File Format, a W3C working draft. The default log file is `logs/http-access`:

1. Enable the standard format HTTP access logger:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based HTTP Access Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The following example shows an excerpt of an HTTP access log with space reformatted:

```
- <client-ip> bjensen  <datestamp> GET  /users/bjensen HTTP/1.1 200 <user-agent> 3  40
- <client-ip> bjensen  <datestamp> GET  /users/scarter HTTP/1.1 200 <user-agent> 4   9
- <client-ip> -        <datestamp> GET  /users/missing HTTP/1.1 401 <user-agent> 5   0
- <client-ip> kvaughan <datestamp> POST /users         HTTP/1.1 200 <user-agent> 6 120
```

Missing values are replaced with `-`. Tabs separate the fields, and if a field contains a tab character, then the field is surrounded with double quotes. DS software repeats double quotes in the field to escape them.

Configure the `log-format` property to set the fields. The default fields are shown here in the order they occur in the log file:

Field	Description
`cs-host`	Client hostname.
`c-ip`	Client IP address.
`cs-username`	Username used to authenticate.
`x-datetime`	Completion timestamp for the HTTP request. Configure with the `log-record-time-format` property.
`cs-method`	HTTP method requested by the client.
`cs-uri`	URI requested by the client.
`cs-uri-stem`	URL-encoded path requested by the client.
`cs-uri-query`	URL-encoded query parameter string requested by the client.
`cs-version`	HTTP version requested by the client.
`sc-status`	HTTP status code for the operation.
`cs(User-Agent)`	User-Agent identifier.
`x-connection-id`	Connection ID used for DS internal operations. When using this field to match HTTP requests with internal operations in the LDAP access log, set the access log advanced property, `suppress-internal-operations:false`. By default, internal operations do not appear in the LDAP access log.
`x-etime`	Execution time in milliseconds needed by DS to service the HTTP request.
`x-transaction-id`	ForgeRock Common Audit event framework transaction ID for the request. This defaults to `0`, unless you configure the server to trust transaction IDs.

The following additional fields are supported:

Field	Description
`c-port`	Client port number.
`s-computername`	Server name writing the access log.

Field	Description
`s-ip`	Server IP address.
`s-port`	Server port number.

Log LDAP access to files

JSON format

When you install DS using procedures from Installation, the primary JSON-based LDAP access log file is `logs/ldap-access.audit.json`. The name of the access log publisher in the configuration is `Json File-Based Access Logger`.

The sample DS Docker image logs to standard output instead of files. This makes it easy to read log messages with the `docker logs` command, and is a pattern you should follow when creating your own DS Docker images. The name of the LDAP access log publisher configuration in the sample image is `Console LDAP Access Logger`.

Primary access logs include messages for each LDAP operation. They can grow quickly, but are particularly useful for analyzing overall client behavior:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs correlate audit events, tracing the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal DS request control. They are sent over HTTP in an HTTP header.

   By default, DS servers do not trust transaction IDs sent with client application requests.

   When a server trusts transaction IDs from client application requests, outgoing requests reuse the incoming ID. For each outgoing request in the transaction, the request's transaction ID has the form `original-transaction-id/sequence-number`, where sequence-number reflects the position of the request in the series of requests for this transaction. For example, if the original-transaction-id is `abc123`, the first outgoing request has the transaction ID `abc123/0`, the second `abc123/1`, the third `abc123/2`, and so on. This lets you distinguish specific requests within a transaction when correlating audit events from multiple services.

   To trust transactions, set the advanced global server property, `trust-transaction-ids:true`:

   ```
   $ dsconfig \
    set-global-configuration-prop \
    --advanced \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --set trust-transaction-ids:true \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

2. Edit the default access log publisher as necessary.

   The following example applies the default settings for DS installed locally, not in a Docker image:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --set enabled:true \
 --add "rotation-policy:24 Hours Time Limit Rotation Policy" \
 --add "rotation-policy:Size Limit Rotation Policy" \
 --set "retention-policy:File Count Retention Policy" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Filtered JSON format

DS servers write messages to a filtered access log file, `logs/filtered-ldap-access.audit.json`. This log grows more slowly than the primary access log. It includes only messages about the following:

- Administrative requests related to backing up and restoring data, scheduling tasks, and reading and writing configuration settings
- Authentication failures
- Requests from client applications that are misbehaving
- Requests that take longer than one second for the server to process
- Search requests that return more than 1000 entries
- Unindexed searches

Follow these steps to change the configuration:

1. Edit the filtered access log publisher as necessary.

   The following example updates the configuration to include control OIDs in log records:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger" \
 --set log-control-oids:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Edit the filtering criteria as necessary.

   The following commands list the relevant default filtering criteria settings for the filtered access log:

```
$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger" \
 --criteria-name "Administrative Requests" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                       : add, bind, compare, delete, extended,
                                      : modify, rename, search
request-target-dn-equal-to            : "**,cn=config", "**,cn=tasks",
                                      : cn=config, cn=tasks

$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger" \
 --criteria-name "Auth Failures" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                       : add, bind, compare, delete, extended,
                                      : modify, rename, search
response-result-code-equal-to         : 7, 8, 13, 48, 49, 50, 123

$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger" \
 --criteria-name "Long Requests" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                       : add, bind, compare, delete, extended,
                                      : modify, rename, search
response-etime-greater-than           : 1000

$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger" \
 --criteria-name "Misbehaving Clients" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

```
log-record-type                       : add, bind, compare, delete, extended,
                                      : modify, rename, search
response-result-code-equal-to         : 1, 2, 17, 18, 19, 21, 34, 60, 61, 64,
                                      : 65, 66, 67, 69

$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger" \
 --criteria-name "Searches Returning 1000+ Entries" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                       : search
search-response-nentries-greater-than  : 1000

$ dsconfig \
 get-access-log-filtering-criteria-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Filtered Json File-Based Access Logger" \
 --criteria-name "Unindexed Searches" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

log-record-type                       : search
search-response-is-indexed            : false
```

For details about the LDAP result codes listed in the criteria, see LDAP result codes.

For details about how filtering works, see Access Log Filtering.

CSV format

A CSV handler sends messages to a comma-separated variable (CSV) file.

> **Important**
>
> The CSV handler does not sanitize messages when writing to CSV log files.
> Do not open CSV logs in spreadsheets and other applications that treat data as code.

The default CSV LDAP access log file is `logs/ldap-access.csv`:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs correlate audit events, tracing the request through multiple applications.

Transaction IDs are sent over LDAP using an internal DS request control. They are sent over HTTP in an HTTP header.

By default, DS servers do not trust transaction IDs sent with client application requests.

When a server trusts transaction IDs from client application requests, outgoing requests reuse the incoming ID. For each outgoing request in the transaction, the request's transaction ID has the form `original-transaction-id/sequence-number`, where sequence-number reflects the position of the request in the series of requests for this transaction. For example, if the original-transaction-id is `abc123`, the first outgoing request has the transaction ID `abc123/0`, the second `abc123/1`, the third `abc123/2`, and so on. This lets you distinguish specific requests within a transaction when correlating audit events from multiple services.

To trust transactions, set the advanced global server property, `trust-transaction-ids:true`:

```
$ dsconfig \
 set-global-configuration-prop \
 --advanced \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set trust-transaction-ids:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Create an enabled CSV file access logger with optional rotation and retention policies:

```
$ dsconfig \
 create-log-publisher \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Common Audit Csv File Access Logger" \
 --type csv-file-access \
 --set enabled:true \
 --set "rotation-policy:24 Hours Time Limit Rotation Policy" \
 --set "rotation-policy:Size Limit Rotation Policy" \
 --set "retention-policy:File Count Retention Policy" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. For tamper-evident logs, follow these steps.

> **Important**
>
> Tamper-evident logging relies on digital signatures and regularly flushing messages to the log system. In high-volume directory deployments with heavy access patterns, signing log messages has a severe negative impact on server performance, reducing throughput by orders of magnitude.
> Be certain to test the performance impact with realistic access patterns for your deployment before enabling the feature in production.

1. Prepare a keystore.

   For details, see Make tampering evident.

2. Enable the tamper-evident capability:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Common Audit Csv File Access Logger" \
 --set tamper-evident:true \
 --set key-store-file:config/audit-keystore \
 --set key-store-pin:"&{audit.keystore.pin}" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

   In this example, `AUDIT_KEYSTORE_PIN` is an environment variable containing the PIN.

Backwards-compatible format

This access log format was the default for older DS servers. Use this log format if you already have software configured to consume that format. The default log file is `logs/access`:

1. Enable the LDAP access logger:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Access Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

By default, this access log contains a message for each request, and a message for each response. It also includes messages for connection and disconnection.

Write messages only on responses by setting the `log-format:combined` property. The setting is useful when filtering messages based on response criteria. It causes the server to log one message per operation, rather than one for each request and response.

Log to a service

The Common Audit framework supports logging access events to an external service:

JDBC

A JDBC handler sends messages to an appropriately configured relational database table.

Before you enable the JDBC handler, create the necessary schema and tables in the target database. See the following example files:

- `/path/to/opendj/config/audit-handlers/mysql_tables-example.sql`
- `/path/to/opendj/config/audit-handlers/oracle_tables-example.sql`
- `/path/to/opendj/config/audit-handlers/postgres_tables-example.sql`

The JDBC handler depends on the JDBC driver for the database, and on HirakiCP. Copy the JDBC driver .jar file for your database, the HirakiCP .jar file for your Java version, and any other dependent libraries required to the `opendj/extlib/` directory.

To enable the JDBC handler, see Configure a custom access log. The JSON configuration file for the JDBC handler has the following format:

```
{
    "class": "org.forgerock.audit.handlers.jdbc.JdbcAuditEventHandler",
    "config": {
        "name": string,                     // Handler name, such as "jdbc".
        "topics": array,                    // LDAP: "ldap-access"; HTTP: "http-access".
        "databaseType": string,             // Supported by default: "h2", "mysql",
                                            // "oracle", "postgres".
        "enabled": boolean,                 // Is the handler enabled?
        "buffering": {                      // (Optional) Default: write each message separately,
                                            // no buffering.
            "enabled": boolean,             // Buffer messages to be sent? Default: false.
            "writeInterval": duration,      // Duration; must be > 0 if buffering is enabled.
            "autoFlush": boolean,           // Flush messages automatically? Default: true.
            "maxBatchedEvents": number,     // Maximum messages in prepared statement. Default: 100.
            "maxSize": number,              // Maximum number of buffered messages. Default: 5000.
            "writerThreads": number         // Threads to write buffered messages: Default: 1.
        },
        "connectionPool": {
            "dataSourceClassName": string, // Either set this to the class name of the data source...
            "jdbcUrl": string,              // ...or set this to the JDBC URL to
                                            // connect to the database.
            "username": string,             // Username to connect to the database.
            "password": string,             // Password to connect to the database.
            "autoCommit": boolean,          // (Optional) Commit transactions automatically?
                                            // Default: true.
            "connectionTimeout": number,    // (Optional) Milliseconds to wait before timing out.
                                            // Default: 30,000.
            "idleTimeout": number,          // (Optional) Milliseconds to wait before timing out.
                                            // Default: 600,000.
            "maxLifetime": number,          // (Optional) Milliseconds thread remains in pool.
                                            // Default: 1,800,000.
            "minIdle": number,              // (Optional) Minimum connections in pool.
                                            // Default: 10.
            "maxPoolSize": number,          // (Optional) Maximum number of connections in pool.
                                            // Default: 10.
            "poolName": string,             // (Optional) Name of connection pool.
                                            // Default: audit.
            "driverClassName": string       // (Optional) Class name of database driver.
                                            // Default: null.
        },
        "tableMappings": [                  // Correspondence of message fields to database columns.
            {
                "event": string,            // LDAP: "ldap-access"; HTTP: "http-access".
                "table": string,            // LDAP: "ldapaccess"; HTTP: "httpaccess".
                "fieldToColumn": {          // Map of field names to database column names.
                    "event-field": "database-column" // Event-field takes JSON pointer.
                }
            }
        ]
    }
}
```

For a sample configuration, see `opendj/config/audit-handlers/jdbc-config.json-example`.

The `writeInterval` takes a duration, which is a lapse of time expressed in English, such as `23 hours 59 minutes and 59 seconds`. Durations are not case sensitive. Negative durations are not supported. Durations use these units:

- `indefinite`, `infinity`, `undefined`, `unlimited`: unlimited duration

- `zero`, `disabled`: zero-length duration
- `days`, `day`, `d`: days
- `hours`, `hour`, `h`: hours
- `minutes`, `minute`, `min`, `m`: minutes
- `seconds`, `second`, `sec`, `s`: seconds
- `milliseconds`, `millisecond`, `millisec`, `millis`, `milli`, `ms`: milliseconds
- `microseconds`, `microsecond`, `microsec`, `micros`, `micro`, `us`: microseconds
- `nanoseconds`, `nanosecond`, `nanosec`, `nanos`, `nano`, `ns`: nanoseconds

JMS

A JMS handler is a JMS producer that publishes messages to an appropriately configured Java Message Service.

To enable the JMS handler, see Configure a custom access log. The JSON configuration file for the JMS handler has the following format:

```
{
    "class": "org.forgerock.audit.handlers.jms.JmsAuditEventHandler",
    "config": {
        "name": string,                     // Handler name, such as "jms".
        "enabled": boolean,                 // Is the handler enabled?
        "topics": array,                    // LDAP: "ldap-access"; HTTP: "http-access".
        "deliveryMode": string,             // One of "NON_PERSISTENT", "PERSISTENT".
        "sessionMode": string,              // One of "AUTO", "CLIENT", "DUPS_OK".
        "batch": {                          // (Optional) Default: Use default settings.
            "capacity": number,             // Maximum capacity of publishing queue. Default: 1.
            "maxBatchedEvents": number,     // Maximum events to deliver in single publishing call.
                                            // Default: 1.
            "writeInterval": string         // Interval between transmissions to JMS.
                                            // Default: "10 millis".
        },
        "jndi": {                           // (Optional) Default: Use default settings.
            "connectionFactoryName": string, // JNDI name for JMS connection factory.
                                            // Default: "ConnectionFactory".
            "topicName": string             // (Optional) Match the value in the context.
                                            // Default: "audit".
            "contextProperties": {          // JNDI InitialContext properties.
                // These depend on the JNDI provider. See the provider documentation for details.
            }
        }
    }
}
```

For a sample configuration, see `opendj/config/audit-handlers/jms-config.json-example`.

Syslog

A Syslog handler sends messages to the UNIX system log as governed by RFC 5424, The Syslog Protocol.

> **Note**
>
> The implementation currently only supports writing *access* messages, not error messages. As a result, this feature is of limited use in most deployments.

To enable a Syslog handler, see Configure a custom access log. The JSON configuration file for the Syslog handler has the following format:

```
{
    "class": "org.forgerock.audit.handlers.syslog.SyslogAuditEventHandler",
    "config": {
        "name": string,            // Handler name, such as "syslog".
        "enabled": boolean,        // Default: false.
        "topics": array,           // LDAP: "ldap-access"; HTTP: "http-access".
        "protocol": string,        // "TCP" or "UDP".
        "host": string,            // Syslog daemon host, such as localhost;
                                   // must resolve to IP address.
        "port": number,            // Syslog daemon port number, such as 514; range: 0 to 65535.
        "connectTimeout": number,  // If using TCP, milliseconds to wait before timing out.
        "facility": string,        // Syslog facility to use for event messages.
        "buffering": {             // (Optional) Default: write each message separately, no buffering.
            "enabled": boolean,    // Buffer messages to be sent? Default: false.
            "maxSize": number      // Maximum number of buffered messages. Default: 5000.
        }
    }
}
```

For a sample configuration, see `opendj/config/audit-handlers/syslog-config.json-example`.

For additional details, see Syslog facility values.

Syslog Facility Values

Value	Description
`kern`	Kernel messages.
`user`	User-level messages.
`mail`	Mail system.
`daemon`	System daemons.
`auth`	Security/authorization messages.
`syslog`	Messages generated internally by `syslogd`.
`lpr`	Line printer subsystem.
`news`	Network news subsystem.
`uucp`	UUCP subsystem.

Value	Description
`cron`	Clock daemon.
`authpriv`	Security/authorization messages.
`ftp`	FTP daemon.
`ntp`	NTP subsystem.
`logaudit`	Log audit.
`logalert`	Log alert.
`clockd`	Clock daemon.
`local0`	Local use 0.
`local1`	Local use 1.
`local2`	Local use 2.
`local3`	Local use 3.
`local4`	Local use 4.
`local5`	Local use 5.
`local6`	Local use 6.
`local7`	Local use 7.

Manage logs

Configure a custom access log

This procedure applies only to Common Audit logs.

An access logger with a JSON configuration lets you use any Common Audit event handler, including customer handlers. The content of the configuration file depends on the audit event handler:

1. Decide whether to trust transaction IDs sent by client applications, used to correlate requests as they traverse multiple servers.

   Client applications using the ForgeRock Common Audit event framework send transaction IDs with their requests. The transaction IDs correlate audit events, tracing the request through multiple applications.

   Transaction IDs are sent over LDAP using an internal DS request control. They are sent over HTTP in an HTTP header.

By default, DS servers do not trust transaction IDs sent with client application requests.

When a server trusts transaction IDs from client application requests, outgoing requests reuse the incoming ID. For each outgoing request in the transaction, the request's transaction ID has the form `original-transaction-id/sequence-number`, where sequence-number reflects the position of the request in the series of requests for this transaction. For example, if the original-transaction-id is `abc123`, the first outgoing request has the transaction ID `abc123/0`, the second `abc123/1`, the third `abc123/2`, and so on. This lets you distinguish specific requests within a transaction when correlating audit events from multiple services.

To trust transactions, set the advanced global server property, `trust-transaction-ids:true`:

```
$ dsconfig \
 set-global-configuration-prop \
 --advanced \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set trust-transaction-ids:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Create the external JSON configuration file for the handler.

   Base your work on the appropriate template in the `config/audit-handlers` directory.

3. If this is a custom access logger provided separately, copy the custom handler .jar file to `opendj/lib/extensions`.

4. Create a log publisher configuration for the access log.

   The `type` defines whether the log contains messages about LDAP or HTTP requests:

   1. For LDAP access logging, create an external access log publisher:

      ```
      $ dsconfig \
       create-log-publisher \
       --publisher-name "Custom LDAP Access Logger" \
       --type external-access \
       --set enabled:true \
       --set config-file:config/audit-handlers/handler-conf.json \
       --hostname localhost \
       --port 4444 \
       --bindDN uid=admin \
       --bindPassword password \
       --usePkcs12TrustStore /path/to/opendj/config/keystore \
       --trustStorePassword:file /path/to/opendj/config/keystore.pin \
       --no-prompt
      ```

   2. For HTTP access logging, create an external HTTP access log publisher:

```
$ dsconfig \
 create-log-publisher \
 --publisher-name "Custom HTTP Access Logger" \
 --type external-http-access \
 --set enabled:true \
 --set config-file:config/audit-handlers/handler-conf.json \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Log access to standard output

This procedure applies only to Common Audit file-based logs, and when you install DS using procedures from Installation.

The sample DS Docker image creates these console access loggers by default:

- `Console LDAP Access Logger`
- `Console HTTP Access Logger`

A JSON stdout handler sends messages to standard output.

> **Important**
>
> Only use this logger when running the server with `start-ds --noDetach`.
> When running as a daemon without the `--noDetach` option, the server also logs the messages to the file, `/path/to/logs/server.out`. The server has no mechanism for rotating or removing the `server.out` log file, which is only cleared when the server starts.
> As a result, using the JSON stdout handler when running the server without the `--noDetach` option can cause the server to eventually run out of disk space.

1. Enable the JSON stdout handler.

   For details, see Configure a Custom Access Log.

   The JSON configuration file for the JSON stdout handler has the following format:

   ```
   {
     "class": "org.forgerock.audit.handlers.json.stdout.JsonStdoutAuditEventHandler",
     "config": {
       "enabled": boolean,                   // Is the handler enabled?
       "name": string,                       // Handler name, such as "json.stdout".
       "elasticsearchCompatible" : boolean,  // If true, the message ID field is named _eventId.
                                             // (Default: _id)
       "topics": array,                      // LDAP: "ldap-access"; HTTP: "http-access".
     }
   }
   ```

   For a sample configuration, see `opendj/config/audit-handlers/json-stdout-config.json-example`.

Log errors to standard output

A `console-error` logger sends messages to standard output.

This procedure applies only when you install DS using procedures from Installation. The sample DS Docker image creates a `Console Error Logger` by default.

> **Important**
>
> Only use this logger when running the server with `start-ds --noDetach`.
> When running as a daemon without the `--noDetach` option, the server also logs the messages to the file, `/path/to/logs/server.out`. The server has no mechanism for rotating or removing the `server.out` log file, which is only cleared when the server starts.
> As a result, using the JSON stdout handler when running the server without the `--noDetach` option can cause the server to eventually run out of disk space.

1. Switch to a `console-error` logger while the server is offline:

```
$ stop-ds

$ dsconfig \
 delete-log-publisher \
 --publisher-name "File-Based Error Logger" \
 --offline \
 --configFile /path/to/opendj/config/config.ldif \
 --no-prompt

$ dsconfig \
 create-log-publisher \
 --type console-error \
 --publisher-name "Console Error Logger" \
 --set enabled:true \
 --set default-severity:error \
 --set default-severity:warning \
 --set default-severity:notice \
 --offline \
 --configFile /path/to/opendj/config/config.ldif \
 --no-prompt

$ start-ds --noDetach
```

Rotate and retain logs

Each file-based log has a *rotation policy*, and a *retention policy*.

The rotation policy specifies when to rotate a log file based on a time, log file age, or log file size. Rotated logs have a rotation timestamp appended to their name.

The retention policy specifies whether to retain logs based on the number of logs, their size, or how much free space should be left on the disk.

1. List log rotation policies:

```
$ dsconfig \
 list-log-rotation-policies \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Log Rotation Policy                 : Type       : file-size-limit : rotation-interval : time-of-day
------------------------------------:------------:-----------------:--------------------:------------
24 Hours Time Limit Rotation Policy : time-limit : -               : 1 d                : -
7 Days Time Limit Rotation Policy   : time-limit : -               : 1 w                : -
Fixed Time Rotation Policy          : fixed-time : -               : -                  : 2359
Size Limit Rotation Policy          : size-limit : 100 mb          : -                  : -
```

2. List log retention policies:

```
$ dsconfig \
 list-log-retention-policies \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Log Retention Policy             : Type            : disk-space-used : free-disk-space : number-of-files
---------------------------------:-----------------:-----------------:-----------------:----------------
File Count Retention Policy      : file-count      : -               : -               : 10
Free Disk Space Retention Policy : free-disk-space : -               : 500 mb          : -
Size Limit Retention Policy      : size-limit      : 500 mb          : -               : -
```

3. View the policies that apply to a given log with the `dsconfig get-log-publisher-prop` command.

   The following example shows that the server keeps 10 access log files, rotating either each day or when the log size reaches 100 MB:

```
$ dsconfig \
 get-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --property retention-policy \
 --property rotation-policy \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

Property         : Value(s)
-----------------:---------------------------------------------------------
retention-policy : File Count Retention Policy
rotation-policy  : 24 Hours Time Limit Rotation Policy, Size Limit Rotation
                 : Policy
```

4. Use the `dsconfig` command to create, update, delete, and assign log rotation and retention policies. Set the policy that applies to a logger with the `dsconfig set-log-publisher-prop` command.

> **Note**
>
> When using access logs based on the ForgeRock Common Audit event framework, you can only configure one of each type of retention or rotation policy.
> This means you can configure only one file count, free disk space, and size limit log retention policy. You can configure only one fixed time, size limit, and time limit log rotation policy.

Enable an audit log

1. Enable a file-based audit logger:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Audit Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Wait for, or make a change to directory data.

   The following example changes a description:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=bjensen,ou=People,dc=example,dc=com" \
 --bindPassword hifalutin << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: New description
EOF
```

The audit log records the changes as shown in the following excerpt:

```
# <datestamp>; conn=<number>; op=<number>
dn: cn=File-Based Audit Logger,cn=Loggers,cn=config
changetype: modify
replace: ds-cfg-enabled
ds-cfg-enabled: true
-

# <datestamp>; conn=<number>; op=<number>
dn: uid=bjensen,ou=people,dc=example,dc=com
changetype: modify
add: description
description: New description
-
```

Audit logs record changes in LDIF format. This means that when an LDAP entry is deleted, the audit log records only its DN.

Filter out administrative messages

A common development troubleshooting technique consists of sending client requests while tailing the access log:

```
$ tail -f /path/to/opendj/logs/ldap-access.audit.json
```

When the `dsconfig` command accesses the configuration, the access log records this. Such messages can prevent you from seeing the messages of interest from client applications.

You can filter access log messages for administrative connections to the administration port:

1. Configure access log filtering criteria:

```
$ dsconfig \
 create-access-log-filtering-criteria \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --criteria-name "Exclude LDAPS on 4444" \
 --type generic \
 --set connection-port-equal-to:4444 \
 --set connection-protocol-equal-to:ldaps \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Activate filtering to exclude administrative messages:

```
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "Json File-Based Access Logger" \
 --set filtering-policy:exclusive \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The publisher filters messages about administrative requests to the administration port.

Audit configuration changes

This example demonstrates how to set up an audit log file to track changes to the server configuration.

Audit log change records have timestamped comments with connection and operation IDs. You can use these to correlate the changes with messages in access logs:

1. Create an audit log publisher:

```
$ dsconfig \
 create-log-publisher \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Server Configuration Audit Log" \
 --type file-based-audit \
 --set enabled:true \
 --set filtering-policy:inclusive \
 --set log-file:logs/config-audit \
 --set rotation-policy:"24 Hours Time Limit Rotation Policy" \
 --set rotation-policy:"Size Limit Rotation Policy" \
 --set retention-policy:"File Count Retention Policy" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Create log filtering criteria for the logger that matches operations targeting `cn=config`:

```
$ dsconfig \
 create-access-log-filtering-criteria \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Server Configuration Audit Log" \
 --criteria-name "Record changes to cn=config" \
 --set request-target-dn-equal-to:"**,cn=config" \
 --set request-target-dn-equal-to:"cn=config" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The server now writes to the audit log file, `/path/to/opendj/logs/config-audit`, whenever an administrator changes the server configuration. The following example output shows the resulting LDIF that defines the log filtering criteria:

```
# <datestamp>; conn=<id>; op=<id>
dn: cn=Record changes to cn=config,cn=Filtering Criteria,cn=File-Based Server Configuration Audit
Log,cn=Loggers,cn=config
changetype: add
objectClass: top
objectClass: ds-cfg-access-log-filtering-criteria
cn: Record changes to cn=config
ds-cfg-request-target-dn-equal-to: **,cn=config
ds-cfg-request-target-dn-equal-to: cn=config
createTimestamp: <timestamp>
creatorsName: uid=admin
entryUUID: <uuid>
```

Allow log message fields

1. When an object is passed in a Common Audit event, it might contain information that should not be logged. By default, the Common Audit implementation uses a whitelist to specify which fields of the event appear:

    1. For Common Audit HTTP access log publishers, edit the `log-field-whitelist` property.

        The following fields appear by default, with each field listed by its JSON path. You cannot change the default whitelist.

        If a whitelisted field contains an object, then listing the field means the whole object is whitelisted:

        - `/_id`
        - `/timestamp`
        - `/eventName`
        - `/transactionId`
        - `/trackingIds`
        - `/userId`
        - `/client`
        - `/server`
        - `/http/request/secure`
        - `/http/request/method`
        - `/http/request/path`
        - `/http/request/headers/accept`
        - `/http/request/headers/accept-api-version`
        - `/http/request/headers/content-type`
        - `/http/request/headers/host`
        - `/http/request/headers/user-agent`
        - `/http/request/headers/x-forwarded-for`
        - `/http/request/headers/x-forwarded-host`
        - `/http/request/headers/x-forwarded-port`
        - `/http/request/headers/x-forwarded-proto`
        - `/http/request/headers/x-original-uri`
        - `/http/request/headers/x-real-ip`
        - `/http/request/headers/x-request-id`

- `/http/request/headers/x-requested-with`
- `/http/request/headers/x-scheme`
- `/request`
- `/response`

For CSV logs, the values map to the column headers. The terms are separated by dots (`.`) rather than by slashes (`/`).

1. LDAP access loggers do not support whitelisting.

   By default, all fields are whitelisted.

Deny log message fields

When an object is passed in a Common Audit event, it might contain information that should not be logged. Loggers allow all fields that are safe to log by default. The whitelist is processed before the blacklist, so blacklist settings overwrite the whitelist defaults:

1. Blacklist individual fields in common audit access logs to prevent the fields from appearing in messages.

   The following example prevents all request headers from appearing in JSON HTTP access logs:

   ```
   $ dsconfig \
    set-log-publisher-prop \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --publisher-name "Json File-Based HTTP Access Logger" \
    --set log-field-blacklist:/http/response/headers \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

   The blacklist values are JSON paths to the fields in log messages.

   For CSV logs, the blacklist values map to the column headers. The terms are separated by dots (`.`) rather than by slashes (`/`).

Make tampering evident

This procedure applies only to Common Audit-based logs.

Tamper-evident logging depends on a public key/private key pair and a secret key. The Common Audit framework accesses the keys in a JCEKS-type keystore. Follow these steps to prepare the keystore:

1. Create a password for the keystore.

   The examples below use an `AUDIT_KEYSTORE_PIN` environment variable that contains the password.

2. Generate a key pair in the keystore.

The keystore holds a signing key with the alias `Signature`. Generate the key with the `RSA` key algorithm, and the `SHA256withRSA` signature algorithm.

The following example uses the default file name:

```
$ keytool \
 -genkeypair \
 -keyalg RSA \
 -sigalg SHA256withRSA \
 -alias "Signature" \
 -dname "CN=ds.example.com,O=Example Corp,C=FR" \
 -keystore /path/to/opendj/config/audit-keystore \
 -storetype JCEKS \
 -storepass:env AUDIT_KEYSTORE_PIN \
 -keypass:env AUDIT_KEYSTORE_PIN
```

You can configure the file name with the log publisher `key-store-file` property.

3. Generate a secret key in the keystore.

   The keystore holds a symmetric key with the alias `Password`. Generate the key with the `HmacSHA256` key algorithm, and 256-bit key size.

   The following example uses the default file name:

   ```
   $ keytool \
    -genseckey \
    -keyalg HmacSHA256 \
    -keysize 256 \
    -alias "Password" \
    -keystore /path/to/opendj/config/audit-keystore \
    -storetype JCEKS \
    -storepass:env AUDIT_KEYSTORE_PIN \
    -keypass:env AUDIT_KEYSTORE_PIN
   ```

   You can configure the file name with the log publisher `key-store-file` property.

4. Verify that the keystore contains signature and password keys:

   ```
   $ keytool \
    -list \
    -keystore /path/to/opendj/config/audit-keystore \
    -storetype JCEKS \
    -storepass:env AUDIT_KEYSTORE_PIN

   password, <date>, SecretKeyEntry,
   signature, <date>, PrivateKeyEntry,
   Certificate fingerprint (SHA-256): <fingerprint>
   ```

Monitoring

This guide covers monitoring and alerts.

What to Monitor

Things to key an eye on.

HTTP

Monitor DS over HTTP.

LDAP

Monitor DS over LDAP.

Status/Tasks

About status and tasks.

Alerts

Manage alerts.

Metrics

Reference for DS metrics.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

What to monitor

Monitor the directory service for the following reasons:

- Noticing availability problems as they occur.

  If a server becomes unresponsive, goes offline, or crashes, you discover the problem quickly, and take corrective action.

- Identifying how client applications use the directory service.

  You can parse directory access logs to determine what client applications do. This information helps you understand what is most important, and make decisions about indexing, for example.

  Access log messages can also provide evidence of security threats, and traces of insecure client application behavior.

- Spotting performance problems, where the directory service does not meet habitual, expected, or formally defined functional, throughput, or response time characteristics.

  For example, if it suddenly becomes impossible to perform updates, the directory service has a performance problem. Alternatively, if a search that regularly completes in 500 milliseconds now takes 15 seconds, the directory service has a performance problem.

  A performance problem could also be evidence of a security threat.

Monitoring directory security is thus part of an overall monitoring strategy. Aim to answer at least the following questions when monitoring specifically for security problems:

- What insecure client behaviors do you observe?

  Examples:

  - Attempts to send simple bind credentials over insecure connections
  - Attempts to change passwords over insecure connections
  - Attempts to change configuration over insecure connections

- What unusual or unexpected usage patterns do you observe?

  Examples:

  - Search requests that perform unindexed searches
  - Requests that hit resource limits
  - Unusually large numbers of bind requests that fail
  - Unusual large numbers of password change requests that fail
  - Unusual large numbers of account lockout events

- Are you observing any sudden or hard-to-explain performance problems?

  Examples:

  - Unusual increases in throughput

- Unusual increases in response times for typical requests
- Servers suddenly starved for system resources

Keep in mind when you see evidence of what looks like a security problem that it might be explained by a mistake made by an administrator or an application developer. Whether the problem is due to malice or user error, you can nevertheless use monitoring information to guide corrective actions.

HTTP-based monitoring

DS servers publish monitoring information at these HTTP endpoints:

`/alive`

Whether the server is currently *alive*, meaning that its internal checks have not found any errors that would require administrative action.

`/healthy`

Whether the server is currently *healthy*, meaning that it is alive, that the replication server is accepting connections on the configured port, and that any replication delays are below the configured threshold.

`/metrics/api`

Read-only access through Forgerock Common REST to a JSON-based view of `cn=monitor` and the monitoring backend.

The endpoint represents a collection where each LDAP entry maps to a resource under `/metrics/api`. Use a REST Query with a `_queryFilter` parameter to access this endpoint. To return all resources, use `/metrics/api?_queryFilter=true`.

`/metrics/prometheus`

Monitoring information for Prometheus monitoring software.

For details, see Prometheus metrics reference.

The following example command accesses the Prometheus endpoint:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus
```

To give a regular user privileges to read monitoring data, see Monitor privilege.

Server is alive (HTTP)

The following example reads the `/alive` endpoint anonymously. If the DS server's internal tests do not find errors that require administrative action, then it returns HTTP 200 OK:

```
$ curl --cacert ca-cert.pem --head https://localhost:8443/alive

HTTP/1.1 200 OK
...
```

If the server finds that it is subject to errors requiring administrative action, it returns HTTP 503 Service Unavailable.

If there are errors, anonymous users receive only the 503 error status. Error strings for diagnosis are returned as an array of `"alive-errors"` in the response body, but the response body is only returned to a user with the `monitor-read` privilege.

When a server returns `"alive-errors"`, diagnose and fix the problem, and then either restart or replace the server.

Server health (HTTP)

The following example reads the `/healthy` endpoint anonymously. If the DS server is alive, as described in Server is alive (HTTP), any replication listener threads are functioning normally, and any replication delay is below the threshold configured as `max-replication-delay-health-check` (default: 5 seconds), then it returns HTTP 200 OK:

```
$ curl --cacert ca-cert.pem --head https://localhost:8443/healthy

HTTP/1.1 200 OK
...
```

If the server is subject to a replication delay above the threshold, then it returns HTTP 503 Service Unavailable. This result only indicates a problem if the replication delay is steadily high and increasing for the long term.

If there are errors, anonymous users receive only the 503 error status. Error strings for diagnosis are returned as an array of `"ready-errors"` in the response body, but the response body is only returned to a user with the `monitor-read` privilege.

When a server returns `"ready-errors"`, route traffic to another server until the current server is ready again.

Server health (Prometheus)

In addition to the examples above, you can monitor whether a server is alive and able to handle requests as Prometheus metrics:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
health_status

# HELP ds_health_status_alive Indicates whether the server is alive
# TYPE ds_health_status_alive gauge
ds_health_status_alive 1.0
# HELP ds_health_status_healthy Indicates whether the server is able to handle requests
# TYPE ds_health_status_healthy gauge
ds_health_status_healthy 1.0
```

Replication delay (Prometheus)

The following example reads a metric to check the delay in replication:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
receive_delay

# HELP ds_replication_replica_remote_replicas_receive_delay_seconds Current local delay in receiving replicated
operations
# TYPE ds_replication_replica_remote_replicas_receive_delay_seconds gauge
ds_replication_replica_remote_replicas_receive_delay_seconds{<labels>} <delay>
```

DS replicas measure replication delay as the local delay when receiving and replaying changes. A replica calculates these local delays based on changes received from other replicas. Therefore, a replica can only calculate delays based on changes it has received. Network outages cause inaccuracy in delay metrics.

A replica calculates delay metrics based on times reflecting the following events:

- t_0: the remote replica records the change in its data
- t_1: the remote replica sends the change to a replica server
- t_2: the local replica receives the change from a replica server
- t_3: the local replica applies the change to its data

This figure illustrates when these events occur:

repl-delay

Replication keeps track of changes using change sequence numbers (CSNs), opaque and unique identifiers for each change that indicate when and where each change first occurred. The t_n values are CSNs.

When the CSNs for the last change received and the last change replayed are identical, the replica has applied all the changes it has received. In this case, there is no known delay. The receive and replay delay metrics are set to 0 (zero).

When the last received and last replayed CSNs differ:

- Receive delay is set to the time t_2 - t_0 for the last change received.

  Another name for receive delay is current delay.

- Replay delay is approximately t_3 - t_2 for the last change replayed. In other words, it is an approximation of how long it took the last change to be replayed.

As long as replication delay tends toward zero regularly and over the long term, temporary spikes and increases in delay measurements are normal. When all replicas remain connected and yet replication delay remains high and increases over the long term, the high replication delay indicates a problem. Steadily high and increasing replication delay shows that replication is not converging, and the service is failing to achieve eventual consistency.

For a current snapshot of replication delays, you can also use the `dsrepl status` command. For details, see replication status.

Disk space (Prometheus)

The following example shows monitoring metrics you can use to check whether the server is running out of disk space:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep disk

# HELP ds_disk_free_space_bytes The amount of free disk space (in bytes)
# TYPE ds_disk_free_space_bytes gauge
ds_disk_free_space_bytes{disk="<partition>",} <bytes>
# HELP ds_disk_free_space_full_threshold_bytes The effective full disk space threshold (in bytes)
# TYPE ds_disk_free_space_full_threshold_bytes gauge
ds_disk_free_space_full_threshold_bytes{disk="<partition>",} <bytes>
# HELP ds_disk_free_space_low_threshold_bytes The effective low disk space threshold (in bytes)
# TYPE ds_disk_free_space_low_threshold_bytes gauge
ds_disk_free_space_low_threshold_bytes{disk="<partition>",} <bytes>
```

In your monitoring software, compare free space with the disk low and disk full thresholds. For database backends, these thresholds are set using the configuration properties: disk-low-threshold and disk-full-threshold.

When you read from `cn=monitor` instead ,as described in LDAP-based monitoring, the relevant data are exposed on child entries of `cn=disk space monitor,cn=monitor`.

Certificate expiration (Prometheus)

The following example shows how you can use monitoring metrics to check whether the server certificate is due to expire soon:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep cert

# HELP ds_certificates_certificate_expires_at_seconds Certificate expiration date and time
# TYPE ds_certificates_certificate_expires_at_seconds gauge
ds_certificates_certificate_expires_at_seconds{alias="ssl-key-pair",key_manager="PKCS12",} <sec_since_epoch>
```

In your monitoring software, compare the expiration date with the current date.

When you read from `cn=monitor` instead, as described in LDAP-based monitoring, the relevant data are exposed on child entries of `cn=certificates,cn=monitor`.

Request statistics (Prometheus)

DS server connection handlers respond to client requests. The following example uses the default monitor user account to read statistics about client operations on each of the available connection handlers:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
connection_handlers
```

Work queue (Prometheus)

DS servers have a work queue to track request processing by worker threads, and whether the server has rejected any requests due to a full queue. If enough worker threads are available, then no requests are rejected. The following example uses the default monitor user account to read statistics about the work queue:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
work_queue
```

To adjust the number of worker threads, see the settings for Traditional Work Queue.

Database size (Prometheus)

DS servers maintain counts of the number of entries in each backend. The following example uses the default monitor user account to read the counts:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
backend_entry_count
```

Active users (Prometheus)

DS server connection handlers respond to client requests. The following example uses the default monitor user account to read active connections on each connection handler:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
"active_[cp]"
```

Index use (Prometheus)

DS maintains metrics about index use. The metrics indicate how often an index was accessed since the DS server started.

The following example demonstrates how to read the metrics for all monitored indexes:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
index_uses
```

Filtering results (Prometheus)

By default, DS servers return all Prometheus metrics. To limit what the server returns, set one of these HTTP endpoint properties for the `/metrics/prometheus`:

- `excluded-metric-pattern`
- `included-metric-pattern`

Set these properties to valid Java regular expression patterns.

The following configuration change causes the server to return only `ds_connection_handlers_ldap_requests_*` metrics. As mentioned in the reference documentation, "*The metric name prefix must not be included in the filter.*" Notice that the example uses `connection_handlers_ldap_requests`, not including the leading `ds_`:

```
$ dsconfig \
 set-http-endpoint-prop \
 --endpoint-name /metrics/prometheus \
 --set included-metric-pattern:'connection_handlers_ldap_requests' \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

The following configuration change causes the server to exclude metrics whose names start with `ds_jvm_`. Notice that the example uses the regular expression `jvm_.*`:

```
$ dsconfig \
 set-http-endpoint-prop \
 --endpoint-name /metrics/prometheus \
 --set excluded-metric-pattern:'jvm_.*' \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

LDAP-based monitoring

DS servers publish whether the server is alive and able to handle requests in the root DSE. They publish monitoring information over LDAP under the entry `cn=monitor`.

The following example reads all available monitoring entries:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN cn=monitor \
 "(&)"
```

The monitoring entries under `cn=monitor` reflect activity since the server started.

Many different types of metrics are exposed. For details, see LDAP metrics reference.

Monitor privilege

The following example assigns the required privilege to Kirsten Vaughan's entry to read monitoring data, and shows monitoring information for the backend holding Example.com data:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: monitor-read
EOF

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN cn=monitor \
 "(ds-cfg-backend-id=dsEvaluation)"

dn: ds-cfg-backend-id=dsEvaluation,cn=backends,cn=monitor
objectClass: top
objectClass: ds-monitor
objectClass: ds-monitor-backend
objectClass: ds-monitor-backend-pluggable
objectClass: ds-monitor-backend-db
ds-cfg-backend-id: dsEvaluation
ds-mon-backend-degraded-index-count: <number>
ds-mon-backend-entry-count: <number>
ds-mon-backend-entry-size-read: <json>
ds-mon-backend-entry-size-written: <json>
ds-mon-backend-filter-use-indexed: <number>
ds-mon-backend-filter-use-start-time: <timestamp>
ds-mon-backend-filter-use-unindexed: <number>
ds-mon-backend-is-private: <boolean>
ds-mon-backend-ttl-entries-deleted: <json>
ds-mon-backend-ttl-is-running: <boolean>
ds-mon-backend-ttl-last-run-time: <timestamp>
ds-mon-backend-ttl-queue-size: <number>
ds-mon-backend-ttl-thread-count: <number>
ds-mon-backend-writability-mode: enabled
ds-mon-db-cache-evict-internal-nodes-count: <number>
ds-mon-db-cache-evict-leaf-nodes-count: <number>
ds-mon-db-cache-leaf-nodes: <boolean>
ds-mon-db-cache-misses-internal-nodes: <number>
ds-mon-db-cache-misses-leaf-nodes: <number>
ds-mon-db-cache-size-active: <number>
ds-mon-db-cache-size-total: <number>
ds-mon-db-cache-total-tries-internal-nodes: <number>
ds-mon-db-cache-total-tries-leaf-nodes: <number>
ds-mon-db-checkpoint-count: <number>
ds-mon-db-log-cleaner-file-deletion-count: <number>
ds-mon-db-log-files-open: <number>
ds-mon-db-log-files-opened: <number>
```

```
ds-mon-db-log-size-active: 146529047
ds-mon-db-log-size-total: 143655929
ds-mon-db-log-utilization-max: <number>
ds-mon-db-log-utilization-min: <number>
ds-mon-db-version: <version>
```

Server health (LDAP)

Anonymous clients can monitor the health status of the DS server by reading the `alive` attribute of the root DSE:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN "" \
 --searchScope base \
 "(&)" \
 alive

dn:
alive: true
```

When `alive` is `true`, the server's internal tests have not found any errors requiring administrative action. When it is `false`, fix the errors and either restart or replace the server.

If the server returns `false` for this attribute, get error information, as described in Server health details (LDAP).

Server health details (LDAP)

The default monitor user can check whether the server is alive and able to handle requests on `cn=health status,cn=monitor`:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN "cn=health status,cn=monitor" \
 --searchScope base \
 "(&)"

dn: cn=health status,cn=monitor
ds-mon-alive: true
ds-mon-healthy: true
objectClass: top
objectClass: ds-monitor
objectClass: ds-monitor-health-status
cn: health status
```

When the server is either not alive or not able to handle requests, this entry includes error diagnostics as strings on the `ds-mon-alive-errors` and `ds-mon-healthy-errors` attributes.

Replication delay (LDAP)

The following example uses the default monitor user account to check the delay in replication:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN cn=monitor \
 "(ds-mon-receive-delay=*)" \
 ds-mon-receive-delay

dn: ds-mon-domain-name=cn=schema,cn=replicas,cn=replication,cn=monitor
ds-mon-receive-delay: <delay>

dn: ds-mon-domain-name=dc=example\,dc=com,cn=replicas,cn=replication,cn=monitor
ds-mon-receive-delay: <delay>

dn: ds-mon-domain-name=uid=monitor,cn=replicas,cn=replication,cn=monitor
ds-mon-receive-delay: <delay>
```

DS replicas measure replication delay as the local delay when receiving and replaying changes. A replica calculates these local delays based on changes received from other replicas. Therefore, a replica can only calculate delays based on changes it has received. Network outages cause inaccuracy in delay metrics.

A replica calculates delay metrics based on times reflecting the following events:

- $\mathbf{t_0}$: the remote replica records the change in its data
- $\mathbf{t_1}$: the remote replica sends the change to a replica server
- $\mathbf{t_2}$: the local replica receives the change from a replica server
- $\mathbf{t_3}$: the local replica applies the change to its data

This figure illustrates when these events occur:

repl-delay

Replication keeps track of changes using change sequence numbers (CSNs), opaque and unique identifiers for each change that indicate when and where each change first occurred. The $\mathbf{t_n}$ values are CSNs.

When the CSNs for the last change received and the last change replayed are identical, the replica has applied all the changes it has received. In this case, there is no known delay. The receive and replay delay metrics are set to 0 (zero).

When the last received and last replayed CSNs differ:

- Receive delay is set to the time $t_2 - t_0$ for the last change received.

  Another name for receive delay is current delay.

- Replay delay is approximately $t_3 - t_2$ for the last change replayed. In other words, it is an approximation of how long it took the last change to be replayed.

As long as replication delay tends toward zero regularly and over the long term, temporary spikes and increases in delay measurements are normal. When all replicas remain connected and yet replication delay remains high and increases over the long term, the high replication delay indicates a problem. Steadily high and increasing replication delay shows that replication is not converging, and the service is failing to achieve eventual consistency.

For a current snapshot of replication delays, you can also use the `dsrepl status` command. For details, see replication status.

Replication status (LDAP)

The following example uses the default monitor user account to check the replication status of the local replica:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN cn=monitor \
 "(ds-mon-status=*)" \
 ds-mon-status

dn: ds-mon-domain-name=dc=example\,dc=com,cn=replicas,cn=replication,cn=monitor
ds-mon-status: Normal
```

If the status is not `Normal`, how you react depends on the value of the `ds-mon-status` attribute:

Status	Explanation	Actions to Take
`Bad generation id`	Replication is broken. Internally, DS replicas store a shorthand form of the initial state called a generation ID. The generation ID is a hash of the first 1000 entries in a backend. If the replicas' generation IDs match, the servers can replicate data without user intervention. If the replicas' generation IDs do not match for a given backend, you must manually initialize replication between them to force the same initial state on all replicas. This status arises for one of the following reasons: • The replica and the replication server have different generation IDs for the data because the replica was not initialized with the same data as its peer replicas. • The replica has fallen further behind the replication server than allowed by the replication-purge-delay. In other words, the replica is missing too many changes, and lacks the historical information required to synchronize with peer replicas. • The fractional replication configuration for this replica does not match the backend data. For example, you reconfigured fractional replication to include or exclude different attributes, or you configured fractional replication in an incompatible way on different peer replicas.	Whenever you see this status: 1. If fractional replication is configured, make sure the configuration is compatible on all peer replicas. For details, see Fractional replication. 2. Reinitialize replication to fix the bad generation IDs. For details, see Manual initialization.

Status	Explanation	Actions to Take
`Degraded`	Unless this status is persistent, replication is operating normally. The replica has fallen further behind peer replicas than the degraded-status-threshold. By default, the threshold is 5000, meaning this state is triggered if the replica falls 5000 or more changes behind. Additionally, the number of pending changes to apply is an *approximation* calculated internally using change sequence numbers that are not necessarily sequential. This status can arise periodically during normal operation when, for example, replication absorbs a burst of updates. In a directory service that sustains 5000 updates a second, a temporary `Degraded` status can represent a one-second delay.	If the `Degraded` status persists: 1. Make sure peer replica systems are sized appropriately. If some replicas are on more powerful systems with faster I/O than others, the replicas on the smaller systems can fall behind as load increases. 2. Consider raising the `degraded-status-threshold` setting.
`Full update`	Replication is operating normally. You have chosen to initialize replication over the network. The time to complete the operation depends on the network bandwidth and volume of data to synchronize.	Monitor the server output and wait for initialization to complete.
`Invalid`	This status arises for one of the following reasons: • The replica has encountered a replication protocol error. This status can arise due to faulty network communication between the replica and the replication server. • The replica has just started, and is initializing.	If this status happens during normal operation: 1. Review the replica and replication server error logs, described in About logs, for network-related replication error messages. 2. Independently verify network communication between the replica and the replication server systems.
`Normal`	Replication is operating normally.	Nothing to do.

Status	Explanation	Actions to Take
`Not connected`	This status arises for one of the following reasons: • The replica has just started and is not yet connected to the replication server. • The replica cannot connect to a replication server.	If this status happens during normal operation: 1. Review the replica and replication server error logs for network-related replication error messages. 2. Independently verify network communication between the replica and the replication server systems.

Request statistics (LDAP)

DS server connection handlers respond to client requests. The following example uses the default monitor user account to read statistics about client operations on each of the available connection handlers:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN "cn=connection handlers,cn=monitor" \
 "(&)"
```

For details about the content of metrics returned, see Metric types reference.

Work queue (LDAP)

DS servers have a work queue to track request processing by worker threads, and whether the server has rejected any requests due to a full queue. If enough worker threads are available, then no requests are rejected. The following example uses the default monitor user account to read statistics about the work queue:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN "cn=work queue,cn=monitor" \
 "(&)"
```

For details about the content of metrics returned, see Metric types reference. To adjust the number of worker threads, see the settings for Traditional Work Queue.

Database size (LDAP)

DS servers maintain counts of the number of entries in each backend and under each base DN. The following example uses the default monitor user account to read the counts:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN cn=monitor \
 "(|(ds-mon-backend-entry-count=*)(ds-mon-base-dn-entry-count=*))" \
 ds-mon-backend-entry-count ds-mon-base-dn-entry-count
```

Active users (LDAP)

DS server connection handlers respond to client requests. The following example uses the default monitor user account to read the metrics about active connections on each connection handler:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN cn=monitor \
 "(objectClass=ds-monitor-connection*)" \
 ds-mon-active-connections-count ds-mon-active-persistent-searches ds-mon-connection ds-mon-listen-address
```

For details about the content of metrics returned, see Metric types reference.

Index use (LDAP)

DS maintains metrics about index use. The metrics indicate how often an index was accessed since the DS server started.

The following example demonstrates how to read the metrics for all monitored indexes:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=monitor \
 --bindPassword password \
 --baseDN cn=monitor \
 "(objectClass=ds-monitor-backend-index)" ds-mon-index ds-mon-index-uses
```

SNMP-based monitoring

> **Note**
>
> The interface stability of this legacy feature is *deprecated*.

DS servers support SNMP, including the Management Information Base described in RFC 2605: Directory Server Monitoring MIB.

SNMP is not enabled by default. SNMP-based monitoring depends on an OpenDMK library. The OpenDMK binary bundle containing this library ships with DS servers as `snmp/opendmk.jar`. Installation requires that you accept the OpenDMK Binary License. OpenDMK installation is a separate step that you must perform before you can use SNMP.

1. Run the OpenDMK installer and accept the license, use the self-extracting .jar:

   ```
   $ java -jar /path/to/opendj/snmp/opendmk.jar
   ```

2. Install OpenDMK, and then copy the libraries to the `/path/to/opendj/extlib` directory. For example, if you install OpenDMK in the `/path/to` directory, copy the libraries from the `/path/to/OpenDMK-bin/lib` directory:

   ```
   $ cp /path/to/OpenDMK-bin/lib/* /path/to/opendj/extlib/
   ```

3. Set up an SNMP connection handler:

   ```
   $ dsconfig \
    set-connection-handler-prop \
    --handler-name SNMP \
    --set enabled:true \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

4. If the server does not have access to the default ports, change them.

   By default, the SNMP connection handler listens on port `161`, and uses port `162` for traps. On UNIX and Linux systems, only root can normally open these ports. The following command installs as a normal user, changing the listen and trap ports:

```
$ dsconfig \
 set-connection-handler-prop \
 --handler-name SNMP \
 --set listen-port:11161 \
 --set trap-port:11162 \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

5. Restart the SNMP connection handler to take the changes into account:

```
$ dsconfig \
 set-connection-handler-prop \
 --handler-name SNMP \
 --set enabled:false \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-connection-handler-prop \
 --handler-name SNMP \
 --set enabled:true \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

6. Check that connection handler works as expected.

   The following command reads the response on the SNMP listen port:

```
$ snmpwalk -v 2c -c OpenDJ@OpenDJ localhost:11161

iso.3.6.1.2.1.66.1.1.1.1 = STRING: "ForgeRock Directory Services version"
iso.3.6.1.2.1.66.1.1.2.1 = STRING: "/path/to/opendj" ...
```

JMX-based monitoring

A number of tools support Java Management Extensions (JMX), including the `jconsole` command bundled with the Java platform, and VisualVM. JMX is not configured by default.

Configure JMX

1. Set server Java arguments appropriately to avoid regular full garbage collection (GC) events.

   JMX is based on Java Remote Method Invocation (RMI), which uses references to objects. By default, the JMX client and server perform a full GC periodically to clean up stale references. As a result, the default settings cause JMX to cause a full GC every hour.

   To prevent hourly full GCs when using JMX, add the `-XX:+DisableExplicitGC` option to the list of `start-ds.java-args` arguments. You can do this by editing the `config/java.properties` file and restarting the server.

   Avoid using this argument when importing LDIF online using the `import-ldif` command. The import process uses GC to work around memory management issues.

2. Configure the server to activate JMX access.

   The following example uses the reserved port number, `1689`:

   ```
   $ dsconfig \
    create-connection-handler \
    --handler-name JMX \
    --type jmx \
    --set enabled:true \
    --set listen-port:1689 \
    --hostname localhost \
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --no-prompt
   ```

   The change takes effect immediately.

Connect over JMX

1. Add appropriate privileges to access JMX monitoring information.

   By default, no users have privileges to access the JMX connection. The following commands create a user with JMX privileges, who can authenticate over an insecure connection:

```
# Create a password policy to allow the user to authenticate insecurely:
$ dsconfig \
 create-password-policy \
 --policy-name "Allow insecure authentication" \
 --type password-policy \
 --set default-password-storage-scheme:PBKDF2-HMAC-SHA256 \
 --set password-attribute:userPassword \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

# Create a backend for the JMX monitor user entry:
$ dsconfig \
create-backend \
 --backend-name jmxMonitorUser \
 --type ldif \
 --set enabled:true \
 --set base-dn:"uid=JMX Monitor" \
 --set ldif-file:db/jmxMonitorUser/jmxMonitorUser.ldif \
 --set is-private-backend:true \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

# Prepare the JMX monitor user entry.
# Notice the privileges and password policy settings:
$ cat > /tmp/jmxMonitorUser.ldif << EOF
dn: uid=JMX Monitor
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: JMX Monitor
sn: User
uid: JMX Monitor
userPassword: password
ds-privilege-name: monitor-read
ds-privilege-name: jmx-notify
ds-privilege-name: jmx-read
ds-privilege-name: jmx-write
ds-pwp-password-policy-dn: cn=Allow insecure authentication,cn=Password Policies,cn=config
EOF

# Import the JMX monitor user:
$ import-ldif \
 --backendID jmxMonitorUser \
 --includeBranch "uid=JMX Monitor" \
 --ldifFile /tmp/jmxMonitorUser.ldif \
 --hostname localhost \
```

```
    --port 4444 \
    --bindDN uid=admin \
    --bindPassword password \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

2. Connect using the service URI, username, and password:

Service URI

Full URI to the service including the hostname or IP address and port number for JMX where the DS server listens for connections.

For example, if the server hostname is `localhost`, and the DS server listens for JMX connections on port `1689`, then the service URI is:

```
service:jmx:rmi:///jndi/rmi://localhost:1689/org.opends.server.protocols.jmx.client-unknown
```

Username

The full DN of the user with privileges to connect over JMX, such as `uid=JMX Monitor`.

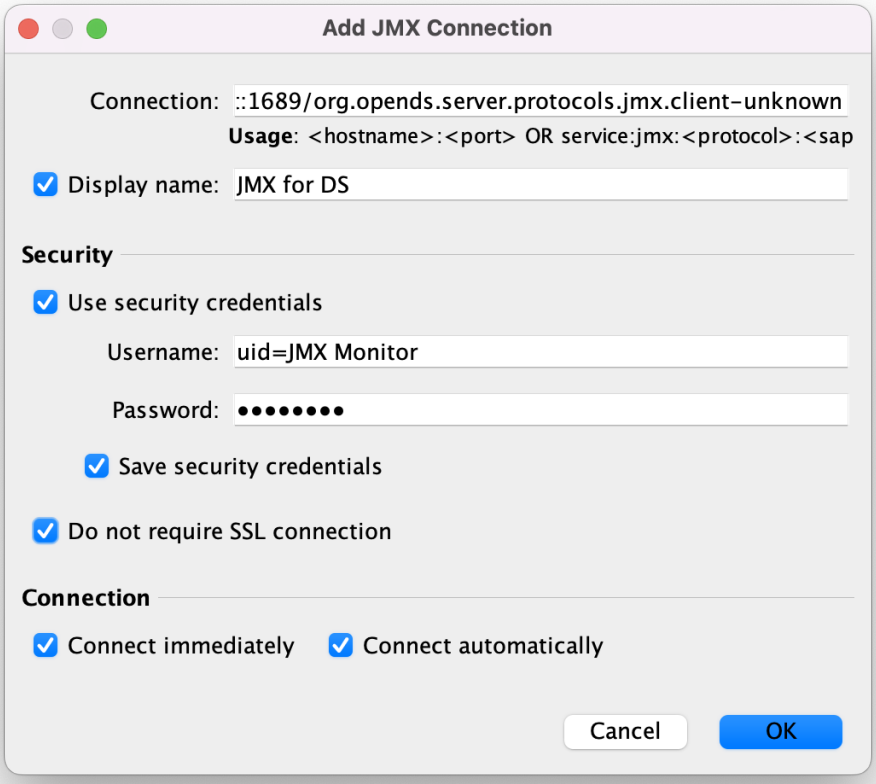
Password

The bind password for the user.

3. Connect remotely.

The following steps show how you connect using VisualVM:

1. Start VisualVM.
2. Select **File** > **Add JMX Connection...** to configure the connection:



3. Select the connection in the left menu to view JMX monitoring information.

For additional details, see Monitoring and Management Using JMX Technology.

Status and tasks

The `status` command functions in offline mode, but provides more information with the server is running. The command describes the server's capabilities, including the ports and disks it uses, and the backends it serves. With the `--script-friendly` option, the command returns JSON output. The command requires administrative credentials to read a running server's configuration:

```
$ status \
 --bindDn uid=admin \
 --bindPassword password \
 --hostname localhost \
 --port 4444 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --script-friendly
```

The `manage-tasks` command lets you manage tasks scheduled on a server, such as regular backup. The command connects to the administration port of a local or remote server:

```
$ manage-tasks \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Push to Graphite

The Graphite application stores numeric time-series data of the sort produced by monitoring metrics, and allows you to render graphs of that data.

Your applications, in this case DS servers, push data into Graphite. You do this by configuring the Graphite Monitor Reporter Plugin with the host and port number of the Graphite service, and with a prefix for your server, such as its FQDN. By default, the plugin pushes all metrics it produces to the Graphite service. You can opt to limit this by setting the `excluded-metric-pattern` or `included-metric-pattern` properties.

The following example configures the plugin to push metrics to Graphite at `graphite.example.com:2004` every 10 seconds (default):

```
$ dsconfig \
 create-plugin \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --plugin-name Graphite \
 --type graphite-monitor-reporter \
 --set enabled:true \
 --set graphite-server:graphite.example.com:2004 \
 --set metric-name-prefix:ds.example.com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

To view metrics stored in Graphite, you can use the Graphite render API or Grafana, for example. See the Graphite and Grafana documentation for details.

Alerts

DS servers can send alerts for significant server events.

JMX alerts

The following example enables JMX alert notifications:

```
$ dsconfig \
 set-alert-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name "JMX Alert Handler" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Mail alerts

The following example sets up an SMTP server, and configures email alerts:

```
$ dsconfig \
 create-mail-server \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --server-name "SMTP server" \
 --set enabled:true \
 --set auth-username:mail.user \
 --set auth-password:password \
 --set smtp-server:smtp.example.com:587 \
 --set trust-manager-provider:"JVM Trust Manager" \
 --set use-start-tls:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 create-alert-handler \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name "SMTP Alert Handler" \
 --type smtp \
 --set enabled:true \
 --set message-subject:"DS Alert, Type: %%alert-type%%, ID: %%alert-id%%" \
 --set message-body:"%%alert-message%%" \
 --set recipient-address:kvaughan@example.com \
 --set sender-address:ds@example.com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Alert types

DS servers use the following alert types. For alert types that indicate server problems, check `logs/errors` for details:

`org.opends.server.AccessControlDisabled`

The access control handler has been disabled.

`org.opends.server.AccessControlEnabled`

The access control handler has been enabled.

`org.opends.server.authentiation.dseecompat.ACIParseFailed`

The dseecompat access control subsystem failed to correctly parse one or more ACI rules when the server first started.

`org.opends.server.BackupFailure`

A backup has failed.

`org.opends.server.BackupSuccess`

A backup has completed successfully.

`org.opends.server.CannotCopySchemaFiles`

A problem has occurred while attempting to create copies of the existing schema configuration files before making a schema update, and the schema configuration has been left in a potentially inconsistent state.

`org.opends.server.CannotRenameCurrentTaskFile`

The server is unable to rename the current tasks backing file in the process of trying to write an updated version.

`org.opends.server.CannotRenameNewTaskFile`

The server is unable to rename the new tasks backing file into place.

`org.opends.server.CannotScheduleRecurringIteration`

The server is unable to schedule an iteration of a recurring task.

`org.opends.server.CannotWriteConfig`

The server is unable to write its updated configuration for some reason and therefore the server may not exhibit the new configuration if it is restarted.

`org.opends.server.CannotWriteNewSchemaFiles`

A problem has occurred while attempting to write new versions of the server schema configuration files, and the schema configuration has been left in a potentially inconsistent state.

`org.opends.server.CannotWriteTaskFile`

The server is unable to write an updated tasks backing file for some reason.

org.opends.server.DirectoryServerShutdown

The server has begun the process of shutting down.

org.opends.server.DirectoryServerStarted

The server has completed its startup process.

org.opends.server.DiskFull

Free disk space has reached the full threshold.

Default is 6% of the size of the file system.

org.opends.server.DiskSpaceLow

Free disk space has reached the low threshold.

Default is 10% of the size of the file system.

org.opends.server.EnteringLockdownMode

The server is entering lockdown mode, wherein only root users are allowed to perform operations and only over the loopback address.

org.opends.server.LDAPHandlerDisabledByConsecutiveFailures

Consecutive failures have occurred in the LDAP connection handler and have caused it to become disabled.

org.opends.server.LDAPHandlerUncaughtError

Uncaught errors in the LDAP connection handler have caused it to become disabled.

org.opends.server.LDIFBackendCannotWriteUpdate

An LDIF backend was unable to store an updated copy of the LDIF file after processing a write operation.

org.opends.server.LDIFConnectionHandlerIOError

The LDIF connection handler encountered an I/O error that prevented it from completing its processing.

org.opends.server.LDIFConnectionHandlerParseError

The LDIF connection handler encountered an unrecoverable error while attempting to parse an LDIF file.

org.opends.server.LeavingLockdownMode

The server is leaving lockdown mode.

org.opends.server.ManualConfigEditHandled

The server detects that its configuration has been manually edited with the server online, and those changes were overwritten by another change made through the server. The manually edited configuration will be copied to another location.

`org.opends.server.ManualConfigEditLost`

The server detects that its configuration has been manually edited with the server online, and those changes were overwritten by another change made through the server. The manually edited configuration could not be preserved due to an unexpected error.

`org.opends.server.replication.UnresolvedConflict`

Multimaster replication cannot resolve a conflict automatically.

`org.opends.server.UncaughtException`

A server thread has encountered an uncaught exception that caused that thread to terminate abnormally. The impact that this problem has on the server depends on which thread was impacted and the nature of the exception.

`org.opends.server.UniqueAttributeSynchronizationConflict`

A unique attribute conflict has been detected during synchronization processing.

`org.opends.server.UniqueAttributeSynchronizationError`

An error occurred while attempting to perform unique attribute conflict detection during synchronization processing.

Metric types reference

The following monitoring metrics are available in each interface:

Type	Description
Counter	Cumulative metric for a numerical value that only increases while the server is running. Counts that reflect volatile data, such as the number of requests, are reset to 0 when the server starts up.
Gauge	Metric for a numerical value that can increase or decrease.

Type	Description
Histogram	Metric that samples observations, and counts them in buckets, as well as providing a sum of all observed values. Common REST and LDAP views show histograms as JSON objects. JSON histograms for entry sizes (in bytes) have the following fields:(1) `{ "count": number, // Number of events since the server started "sum": number, // Sum of quantities measured for each event // since the server started // The buckets in a histogram depend on what the server observes. // Each bucket for an entry size measurement has a ceiling size. // The first field shows the number of 500-byte or smaller entries, // and the second shows the number of 1000-byte or smaller entries. // The final field shows the number of entries larger than // 1,000,000 bytes: "less-than-or-equal-to-500": number, "less-than-or-equal-to-1000": number, "less-than-or-equal-to-5000": number, "less-than-or-equal-to-10000": number, "less-than-or-equal-to-50000": number, "less-than-or-equal-to-100000": number, "less-than-or-equal-to-500000": number, "less-than-or-equal-to-1000000": number, "less-than-or-equal-to-inf": number }`

Type	Description
Summary	Metric that samples observations, providing a count of observations, sum total of observed amounts, average rate of events, and moving average rates across sliding time windows. Common REST and LDAP views show summaries as JSON objects. JSON summaries have the following fields:(1) `{ "count": number, // Number of events since the server started "total": number, // Sum of quantities measured for each event // since the server started // The following are related to the "count": "mean_rate": number, // Average event rate per second // since the server started "m1_rate": number, // One-minute average event rate per second // (exponentially decaying) "m5_rate": number, // Five-minute average event rate per second // (exponentially decaying) "m15_rate": number, // Fifteen-minute average event rate per second // (exponentially decaying) }` The `"total"` depends on the type of events measured. For example, if the `"count"` is the number of requests, then the `"total"` is the total etime in milliseconds to process all the requests. If the `"count"` is the number of times the server read bytes of data, then the `"total"` is the total number of bytes read. The Prometheus view does not provide time-based statistics, as rates can be calculated from the time-series data. Instead, the Prometheus view includes summary metrics whose names have the following suffixes or labels: • `_count` : number of events since the server started • `_total` : sum of quantities measured for each event since the server started • `{quantile="0.5"}` : 50% at or below this value since the server started • `{quantile="0.75"}` : 75% at or below this value since the server started • `{quantile="0.95"}` : 95% at or below this value since the server started • `{quantile="0.98"}` : 98% at or below this value since the server started • `{quantile="0.99"}` : 99% at or below this value since the server started • `{quantile="0.999"}` : 99.9% at or below this value since the server started

Type	Description
Timer	Metric combining a summary with other statistics. Common REST and LDAP views show summaries as JSON objects. JSON summaries have the following fields[1] `{ "count": number, // Number of events since the server started "total": number, // Total duration for all events // since the server started, in ms // (for requests, sum of the etimes // since the server started, in ms) // The following are related to the "count": "mean_rate": number, // Average event rate per second // since the server started "m1_rate": number, // One-minute average event rate per second // (exponentially decaying) "m5_rate": number, // Five-minute average event rate per second // (exponentially decaying) "m15_rate": number, // Fifteen-minute average event rate per second // (exponentially decaying) // The following are related to the "total": "mean": number, // Average duration over all events // since the server started, in ms "min": number, // Minimum duration recorded // since the server started, in ms "max": number, // Maximum duration recorded // since the server started, in ms "stddev": number, // Standard deviation of durations // since the server started, in ms "p50": number, // 50% durations at or below this value // (median) since the server started, in ms "p75": number, // 75% durations at or below this value // since the server started, in ms "p95": number, // 95% durations at or below this value // since the server started, in ms "p98": number, // 98% durations at or below this value // since the server started, in ms "p99": number, // 99% durations at or below this value // since the server started, in ms "p999": number, // 99.9% durations at or below this value // since the server started, in ms "p9999": number, // 99.99% durations at or below this value // since the server started, in ms "p99999": number // 99.999% durations at or below this value // since the server started, in ms }` The Prometheus view does not provide time-based statistics. Rates can be calculated from the time-series data.

[1] Monitoring metrics reflect sample observations made while the server is running. The values are not saved when the server shuts down. As a result, metrics of this type reflect data recorded since the server started.

Metrics that show etime measurements in milliseconds (ms) continue to show values in ms even if the server is configured to log etimes in nanoseconds.

The calculation of moving averages is intended to be the same as that of the `uptime` and `top` commands, where the moving average plotted over time is smoothed by weighting that decreases exponentially. For an explanation of the mechanism, see the Wikipedia section, Exponential moving average.

LDAP metrics reference

LDAP metrics are exposed as LDAP attributes on entries under `cn=monitor`. Metrics entry object class names start with `ds-monitor`. Metrics attribute names start with `ds-mon`. For details, see the About This Reference.

For examples of common monitoring requests, see LDAP-based monitoring.

> **Note**
>
> Some `ds-mon-jvm-*` metrics depend on the JVM version and configuration. In particular, GC-related metrics depend on the garbage collector that the server uses. The GC metric names are *unstable*, and can change even in a minor JVM release.

Name	Syntax	Description
`ds-mon-abandoned-requests`	Counter metric	Total number of abandoned operations since startup
`ds-mon-active-connections-count`	Integer	Number of active client connections
`ds-mon-active-persistent-searches`	Integer	Number of active persistent searches
`ds-mon-admin-hostport`	Host port	The administrative host and port
`ds-mon-alive`	Boolean	Indicates whether the server is alive
`ds-mon-alive-errors`	Directory String	Lists server errors preventing the server from operating correctly that require administrative action
`ds-mon-backend-degraded-index-count`	Integer	Number of degraded indexes in the backend
`ds-mon-backend-degraded-index`	Directory String	Backend degraded index
`ds-mon-backend-entry-count`	Integer	Number of entries contained in the backend
`ds-mon-backend-entry-size-read`	Histogram	Histogram of entry sizes being read from the underlying storage
`ds-mon-backend-entry-size-written`	Histogram	Histogram of entry sizes being written to the underlying storage
`ds-mon-backend-filter-use-indexed`	Integer	Number of indexed searches performed against the backend
`ds-mon-backend-filter-use-start-time`	Generalized Time	Time when recording started for statistical information about the simple search filters processed against the backend

Name	Syntax	Description
`ds-mon-backend-filter-use-unindexed`	Integer	Number of unindexed searches performed against the backend
`ds-mon-backend-filter-use`	Json	Information about the simple search filter processed against the backend
`ds-mon-backend-is-private`	Boolean	Whether the base DNs of this backend should be considered public or private
`ds-mon-backend-proxy-base-dn`	DN	Base DNs routed to remote LDAP servers by the proxy backend
`ds-mon-backend-proxy-shard`	Summary metric	Remote LDAP servers that the proxy backend forwards requests to
`ds-mon-backend-ttl-entries-deleted`	Summary metric	Summary for entries purged by time-to-live
`ds-mon-backend-ttl-is-running`	Boolean	Indicates whether time-to-live is in the process of purging expired entries
`ds-mon-backend-ttl-last-run-time`	Generalized Time	Last date and time when time-to-live finished purging expired entries
`ds-mon-backend-ttl-queue-size`	Integer	Number of entries queued for purging by the time-to-live service
`ds-mon-backend-ttl-thread-count`	Integer	Number of active time-to-live threads
`ds-mon-backend-writability-mode`	Directory String	Current backend behavior when processing write operations, can either be "disabled", "enabled" or "internal-only"
`ds-mon-base-dn-entry-count`	Integer	Number of subordinate entries of the base DN, including the base DN
`ds-mon-base-dn`	DN	Base DN handled by a backend
`ds-mon-build-number`	Integer	Build number of the Directory Server
`ds-mon-build-time`	Generalized Time	Build date and time of the Directory Server
`ds-mon-bytes-read`	Summary metric	Network bytes read summary
`ds-mon-bytes-written`	Summary metric	Network bytes written summary
`ds-mon-cache-entry-count`	Integer	Current number of entries held in this cache
`ds-mon-cache-max-entry-count`	Integer	Maximum number of entries allowed in this cache

Name	Syntax	Description
`ds-mon-cache-max-size-bytes`	Size in bytes	Memory limit for this cache
`ds-mon-cache-misses`	Summary metric	Number of attempts to retrieve an entry that was not held in this cache
`ds-mon-cache-total-tries`	Summary metric	Number of attempts to retrieve an entry from this cache
`ds-mon-certificate-expires-at`	Generalized Time	Certificate expiration date and time
`ds-mon-certificate-issuer-dn`	DN	Certificate issuer DN
`ds-mon-certificate-serial-number`	Integer	Certificate serial number
`ds-mon-certificate-subject-dn`	DN	Certificate subject DN
`ds-mon-changelog-id`	Directory String	Changelog identifier
`ds-mon-changelog-hostport`	Host port	The host and port of the changelog server
`ds-mon-changelog-purge-delay`	Duration in milli-seconds	The purge delay of the changelog
`ds-mon-compact-version`	Directory String	Compact version of the Directory Server
`ds-mon-config-dn`	DN	DN of the configuration entry
`ds-mon-connected-to-server-hostport`	Host port	Host and replication port of the server that this server is connected to
`ds-mon-connected-to-server-id`	Integer	Identifier of the server that this server is connected to
`ds-mon-connection`	Json	Client connection summary information
`ds-mon-connections`	Summary metric	Connection summary
`ds-mon-current-connections`	Integer	Number of client connections currently established with the Directory Server
`ds-mon-current-delay`	Duration in milli-seconds	Current local delay in receiving replicated operations
`ds-mon-current-receive-window`	Integer	Current replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size

Name	Syntax	Description
`ds-mon-current-send-window`	Integer	Current replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds-mon-current-time`	Generalized Time	Current date and time
`ds-mon-db-cache-evict-internal-nodes-count`	Integer	Number of internal nodes evicted from the database cache
`ds-mon-db-cache-evict-leaf-nodes-count`	Integer	Number of leaf nodes (data records) evicted from the database cache
`ds-mon-db-cache-leaf-nodes`	Boolean	Whether leaf nodes are cached
`ds-mon-db-cache-misses-internal-nodes`	Integer	Number of internal nodes requested by btree operations that were not in the database cache
`ds-mon-db-cache-misses-leaf-nodes`	Integer	Number of leaf nodes (data records) requested by btree operations that were not in the database cache
`ds-mon-db-cache-size-active`	Size in bytes	Size of the database cache
`ds-mon-db-cache-size-total`	Size in bytes	Maximum size of the database cache
`ds-mon-db-cache-total-tries-internal-nodes`	Integer	Number of internal nodes requested by btree operations
`ds-mon-db-cache-total-tries-leaf-nodes`	Integer	Number of leaf nodes (data records) requested by btree operations
`ds-mon-db-checkpoint-count`	Integer	Number of checkpoints run so far
`ds-mon-db-log-cleaner-file-deletion-count`	Integer	Number of cleaner file deletions
`ds-mon-db-log-files-open`	Integer	Number of files currently open in the database file cache
`ds-mon-db-log-files-opened`	Integer	Number of times a log file has been opened
`ds-mon-db-log-size-active`	Size in bytes	Estimate of the amount in bytes of live data in all data files (i.e., the size of the DB, ignoring garbage)
`ds-mon-db-log-size-total`	Size in bytes	Size used by all data files on disk
`ds-mon-db-log-utilization-max`	Integer	Current maximum (upper bound) log utilization as a percentage

Name	Syntax	Description
`ds-mon-db-log-utilization-min`	Integer	Current minimum (lower bound) log utilization as a percentage
`ds-mon-db-version`	Directory String	Database version used by the backend
`ds-mon-disk-dir`	Filesystem path	A monitored directory containing data that may change over time
`ds-mon-disk-free`	Size in bytes	Amount of free disk space
`ds-mon-disk-full-threshold`	Size in bytes	Effective full disk space threshold
`ds-mon-disk-low-threshold`	Size in bytes	Effective low disk space threshold
`ds-mon-disk-root`	Filesystem path	Monitored disk root
`ds-mon-disk-state`	Directory String	Current disk state, can be either "normal", "low" or "full"
`ds-mon-domain-generation-id`	Integer	Replication domain generation identifier
`ds-mon-domain-name`	DN	Replication domain name
`ds-mon-entries-awaiting-updates-count`	Duration in milli-seconds	Number of entries for which an update operation has been received but not replayed yet by this replica
`ds-mon-fix-ids`	Directory String	IDs of issues that have been fixed in this Directory Server build
`ds-mon-full-version`	Directory String	Full version of the Directory Server
`ds-mon-group-id`	Directory String	Unique identifier of the group in which the directory server belongs
`ds-mon-healthy`	Boolean	Indicates whether the server is able to handle requests
`ds-mon-healthy-errors`	Directory String	Lists transient server errors preventing the server from temporarily handling requests
`ds-mon-index`	Directory String	The name of the index
`ds-mon-index-uses`	Summary metric	Current number of index reads
`ds-mon-install-path`	Filesystem path	Directory Server root installation path
`ds-mon-instance-path`	Filesystem path	Directory Server instance path
`ds-mon-je-environment-nbytes-evicted-critical`	Size in bytes	Number of bytes evicted by the DB worker threads For details, see Cache internal nodes.

Name	Syntax	Description
`ds-mon-jvm-architecture`	Directory String	Java virtual machine architecture (e.g. 32-bit, 64-bit)
`ds-mon-jvm-arguments`	Directory String	Input arguments passed to the Java virtual machine
`ds-mon-jvm-available-cpus`	Integer	Number of processors available to the Java virtual machine
`ds-mon-jvm-class-path`	Filesystem path	Path used to find directories and JAR archives containing Java class files
`ds-mon-jvm-classes-loaded`	Integer	Number of classes loaded since the Java virtual machine started
`ds-mon-jvm-classes-unloaded`	Integer	Number of classes unloaded since the Java virtual machine started
`ds-mon-jvm-java-home`	Filesystem path	Installation directory for Java runtime environment (JRE)
`ds-mon-jvm-java-vendor`	Directory String	Java runtime environment (JRE) vendor
`ds-mon-jvm-java-version`	Directory String	Java runtime environment (JRE) version
`ds-mon-jvm-memory-heap-init`	Size in bytes	Amount of heap memory that the Java virtual machine initially requested from the operating system
`ds-mon-jvm-memory-heap-max`	Size in bytes	Maximum amount of heap memory that the Java virtual machine will attempt to use
`ds-mon-jvm-memory-heap-reserved`	Size in bytes	Amount of heap memory that is committed for the Java virtual machine to use
`ds-mon-jvm-memory-heap-used`	Size in bytes	Amount of heap memory used by the Java virtual machine
`ds-mon-jvm-memory-init`	Size in bytes	Amount of memory that the Java virtual machine initially requested from the operating system
`ds-mon-jvm-memory-max`	Size in bytes	Maximum amount of memory that the Java virtual machine will attempt to use
`ds-mon-jvm-memory-non-heap-init`	Size in bytes	Amount of non-heap memory that the Java virtual machine initially requested from the operating system
`ds-mon-jvm-memory-non-heap-max`	Size in bytes	Maximum amount of non-heap memory that the Java virtual machine will attempt to use
`ds-mon-jvm-memory-non-heap-reserved`	Size in bytes	Amount of non-heap memory that is committed for the Java virtual machine to use

Name	Syntax	Description
`ds-mon-jvm-memory-non-heap-used`	Size in bytes	Amount of non-heap memory used by the Java virtual machine
`ds-mon-jvm-memory-reserved`	Size in bytes	Amount of memory that is committed for the Java virtual machine to use
`ds-mon-jvm-memory-used`	Size in bytes	Amount of memory used by the Java virtual machine
`ds-mon-jvm-supported-tls-ciphers`	Directory String	Transport Layer Security (TLS) cipher suites supported by this Directory Server
`ds-mon-jvm-supported-tls-protocols`	Directory String	Transport Layer Security (TLS) protocols supported by this Directory Server
`ds-mon-jvm-threads-blocked-count`	Integer	Number of threads in the BLOCKED state
`ds-mon-jvm-threads-count`	Integer	Number of live threads including both daemon and non-daemon threads
`ds-mon-jvm-threads-daemon-count`	Integer	Number of live daemon threads
`ds-mon-jvm-threads-deadlock-count`	Integer	Number of deadlocked threads
`ds-mon-jvm-threads-deadlocks`	Directory String	Diagnostic stack traces for deadlocked threads
`ds-mon-jvm-threads-new-count`	Integer	Number of threads in the NEW state
`ds-mon-jvm-threads-runnable-count`	Integer	Number of threads in the RUNNABLE state
`ds-mon-jvm-threads-terminated-count`	Integer	Number of threads in the TERMINATED state
`ds-mon-jvm-threads-timed-waiting-count`	Integer	Number of threads in the TIMED_WAITING state
`ds-mon-jvm-threads-waiting-count`	Integer	Number of threads in the WAITING state
`ds-mon-jvm-vendor`	Directory String	Java virtual machine vendor
`ds-mon-jvm-version`	Directory String	Java virtual machine version
`ds-mon-last-seen`	Generalized Time	Time that this server was last seen
`ds-mon-ldap-hostport`	Host port	The host and port to connect using LDAP (no support for start TLS)
`ds-mon-ldap-starttls-hostport`	Host port	The host and port to connect using LDAP (with support for start TLS)

Name	Syntax	Description
`ds-mon-ldaps-hostport`	Host port	The host and port to connect using LDAPS
`ds-mon-listen-address`	Directory String	Host and port
`ds-mon-lost-connections`	Duration in milli-seconds	Number of times the replica lost its connection to the replication server
`ds-mon-major-version`	Integer	Major version number of the Directory Server
`ds-mon-max-connections`	Integer	Maximum number of simultaneous client connections that have been established with the Directory Server
`ds-mon-max-receive-window`	Integer	Maximum replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds-mon-max-send-window`	Integer	Maximum replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds-mon-minor-version`	Integer	Minor version number of the Directory Server
`ds-mon-newest-change-number`	Integer	Newest change number present in the change number index database
`ds-mon-newest-csn-timestamp`	Generalized Time	Timestamp of the newest CSN present in the replica database
`ds-mon-newest-csn`	CSN (Change Sequence Number)	Newest CSN present in the replica database
`ds-mon-oldest-change-number`	Integer	Oldest change number present in the change number index database
`ds-mon-oldest-csn-timestamp`	Generalized Time	Timestamp of the oldest CSN present in the replica database
`ds-mon-oldest-csn`	CSN (Change Sequence Number)	Oldest CSN present in the replica database
`ds-mon-os-architecture`	Directory String	Operating system architecture
`ds-mon-os-name`	Directory String	Operating system name
`ds-mon-os-version`	Directory String	Operating system version

Name	Syntax	Description
`ds-mon-point-version`	Integer	Point version number of the Directory Server
`ds-mon-process-id`	UUID	Process ID of the running directory server
`ds-mon-product-name`	Directory String	Full name of the Directory Server
`ds-mon-protocol`	Directory String	Network protocol
`ds-mon-receive-delay`	Duration in milli-seconds	Current local delay in receiving replicated operations
`ds-mon-replay-delay`	Duration in milli-seconds	Current local delay in replaying replicated operations
`ds-mon-replayed-updates-conflicts-resolved`	Counter metric	Number of updates replayed on this replica for which replication naming conflicts have been resolved
`ds-mon-replayed-updates-conflicts-unresolved`	Counter metric	Number of updates replayed on this replica for which replication naming conflicts have not been resolved
`ds-mon-replayed-internal-updates`	Counter metric	Number of updates replayed on this replica which modify the internal state but not user data
`ds-mon-replayed-updates`	Timer metric	Timer for updates that have been replayed on this replica
`ds-mon-replica-hostport`	Host port	Host and port of a replica server
`ds-mon-replication-domain`	DN	The replication domain
`ds-mon-replication-protocol-version`	Integer	The protocol version used for replication
`ds-mon-requests-abandon`	Timer metric	Abandon request timer
`ds-mon-requests-add`	Timer metric	Add request timer
`ds-mon-requests-bind`	Timer metric	Bind request timer
`ds-mon-requests-compare`	Timer metric	Compare request timer
`ds-mon-requests-delete`	Timer metric	Delete request timer
`ds-mon-requests-extended`	Timer metric	Extended request timer

Name	Syntax	Description
`ds-mon-requests-failure-client-invalid-request`	Timer metric	Timer for requests that failed because there was a problem while attempting to perform the associated operation (associated LDAP result codes: 1, 2, 12, 15, 16, 17, 18, 19, 20, 21, 23, 34, 35, 36, 37, 38, 39; associated HTTP status codes: client error (4xx) except 401 and 403)
`ds-mon-requests-failure-client-redirect`	Timer metric	Timer for requests that could not complete because further action is required (associated HTTP status codes: redirection (3xx))
`ds-mon-requests-failure-client-referral`	Timer metric	Timer for requests that failed because the server did not hold the request targeted entry (but was able to provide alternative servers that may) (associated LDAP result code: 10)
`ds-mon-requests-failure-client-resource-limit`	Timer metric	Timer for requests that failed because they were trying to exceed the resource limits allocated to the associated clients (associated LDAP result codes: time, size and admin limit exceeded (respectively 4, 5 and 11)
`ds-mon-requests-failure-client-security`	Timer metric	Timer for requests that failed for security reasons (associated LDAP result codes: 8, 9, 13, 25, 26, 27; associated HTTP status codes: unauthorized (401) and forbidden (403))
`ds-mon-requests-failure-server`	Timer metric	Timer for apparently valid requests that failed because the server was not able to process them (associated LDAP result codes: busy (51), unavailable (52), unwilling to perform (53) and other (80); associated HTTP status codes: server error (5xx))
`ds-mon-requests-failure-uncategorized`	Timer metric	Timer for requests that failed due to uncategorized reasons
`ds-mon-requests-get`	Timer metric	GET request timer
`ds-mon-requests-in-queue`	Integer	Number of requests in the work queue that have not yet been picked up for processing
`ds-mon-requests-modify-dn`	Timer metric	Modify DN request timer
`ds-mon-requests-modify`	Timer metric	Modify request timer
`ds-mon-requests-patch`	Timer metric	PATCH request timer
`ds-mon-requests-post`	Timer metric	POST request timer
`ds-mon-requests-put`	Timer metric	PUT request timer

Name	Syntax	Description
`ds-mon-requests-search-base`	Timer metric	Base object search request timer
`ds-mon-requests-search-one`	Timer metric	One level search request timer
`ds-mon-requests-search-sub`	Timer metric	Subtree search request timer
`ds-mon-requests-submitted`	Summary metric	Summary for operations that have been successfully submitted to the work queue
`ds-mon-requests-unbind`	Timer metric	Unbind request timer
`ds-mon-requests-uncategorized`	Timer metric	Uncategorized request timer
`ds-mon-revision`	Directory String	Revision ID in the source repository from which the Directory Server is build
`ds-mon-sent-updates`	Counter metric	Number of replication updates sent by this replica
`ds-mon-server-hostport`	Host port	Host and port of a server
`ds-mon-server-id`	Integer	Server identifier
`ds-mon-server-is-local`	Boolean	Indicates whether this is the topology server that has handled the monitoring request
`ds-mon-server-state`	CSN (Change Sequence Number)	Replication server state
`ds-mon-short-name`	Directory String	Short name of the Directory Server
`ds-mon-ssl-encryption`	Boolean	Whether SSL encryption is used when exchanging messages with this server
`ds-mon-start-time`	Generalized Time	Start date and time for the Directory Server
`ds-mon-status-last-changed`	Generalized Time	Last date and time the replication status of the local replica changed
`ds-mon-status`	Directory String	Replication status of the local replica, can either be "Invalid", "Not connected", "Normal", "Degraded", "Full update", "Bad generation id"
`ds-mon-system-name`	Directory String	Fully qualified domain name of the system where the Directory Server is running
`ds-mon-total-connections`	Integer	Total number of client connections that have been established with the Directory Server since it started

Name	Syntax	Description
`ds-mon-updates-already-in-progress`	Counter metric	Number of duplicate updates: updates received by this replica which cannot be applied because they are already in progress. Can happen when a directory server fails over to another replication server
`ds-mon-updates-inbound-queue`	Integer	Number of remote updates received from the replication server but not replayed yet on this replica
`ds-mon-updates-outbound-queue`	Integer	Number of local updates that are waiting to be sent to the replication server once they complete
`ds-mon-updates-totals-per-replay-thread`	Json	JSON array of the number of updates replayed per replay thread
`ds-mon-vendor-name`	Directory String	Vendor name of the Directory Server
`ds-mon-version-qualifier`	Directory String	Version qualifier of the Directory Server
`ds-mon-working-directory`	Filesystem path	Current working directory of the user running the Directory Server

Prometheus metrics reference

The following list puts Prometheus labels in braces. For example, the labels in `ds_backend_db_cache_misses_internal_nodes{backend,type}` are `backend` and `type`.

For examples of common monitoring requests, see HTTP-based monitoring.

> **Note**
>
> Some `ds_jvm_*` metrics depend on the JVM version and configuration. In particular, GC-related metrics depend on the garbage collector that the server uses. The GC metric names are *unstable*, and can change even in a minor JVM release.

Name	Type	Description
`ds_all_entry_caches_cache_entry_count`	Gauge	Current number of entries held in this cache
`ds_all_entry_caches_cache_misses_count`	Summary	Number of attempts to retrieve an entry that was not held in this cache
`ds_all_entry_caches_cache_misses_total`	Summary	Number of attempts to retrieve an entry that was not held in this cache

Name	Type	Description
`ds_all_entry_caches_cache_total_tries_count`	Summary	Number of attempts to retrieve an entry from this cache
`ds_all_entry_caches_cache_total_tries_total`	Summary	Number of attempts to retrieve an entry from this cache
`ds_backend_db_cache_evict_internal_nodes_count{backend,type}`	Gauge	Number of internal nodes evicted from the database cache
`ds_backend_db_cache_evict_leaf_nodes_count{backend,type}`	Gauge	Number of leaf nodes (data records) evicted from the database cache
`ds_backend_db_cache_leaf_nodes{backend,type}`	Gauge	Whether leaf nodes are cached
`ds_backend_db_cache_misses_internal_nodes{backend,type}`	Gauge	Number of internal nodes requested by btree operations that were not in the database cache
`ds_backend_db_cache_misses_leaf_nodes{backend,type}`	Gauge	Number of leaf nodes (data records) requested by btree operations that were not in the database cache
`ds_backend_db_cache_size_active_bytes{backend,type}`	Gauge	Size of the database cache
`ds_backend_db_cache_size_total_bytes{backend,type}`	Gauge	Maximum size of the database cache
`ds_backend_db_cache_total_tries_internal_nodes{backend,type}`	Gauge	Number of internal nodes requested by btree operations
`ds_backend_db_cache_total_tries_leaf_nodes{backend,type}`	Gauge	Number of leaf nodes (data records) requested by btree operations
`ds_backend_db_checkpoint_count{backend,type}`	Gauge	Number of checkpoints run so far
`ds_backend_db_log_cleaner_file_deletion_count{backend,type}`	Gauge	Number of cleaner file deletions
`ds_backend_db_log_files_open{backend,type}`	Gauge	Number of files currently open in the database file cache
`ds_backend_db_log_files_opened{backend,type}`	Gauge	Number of times a log file has been opened
`ds_backend_db_log_size_active_bytes{backend,type}`	Gauge	Estimate of the amount in bytes of live data in all data files (i.e., the size of the DB, ignoring garbage)

Name	Type	Description
`ds_backend_db_log_size_total_bytes{backend,type}`	Gauge	Size used by all data files on disk
`ds_backend_db_log_utilization_max{backend,type}`	Gauge	Current maximum (upper bound) log utilization as a percentage
`ds_backend_db_log_utilization_min{backend,type}`	Gauge	Current minimum (lower bound) log utilization as a percentage
`ds_backend_degraded_index_count{backend,type}`	Gauge	Number of degraded indexes in the backend
`ds_backend_entry_count{backend,base_dn,dc,type}`	Gauge	Number of subordinate entries of the base DN, including the base DN
`ds_backend_entry_count{backend,base_dn,type}`	Gauge	Number of subordinate entries of the base DN, including the base DN
`ds_backend_entry_size_read_bucket{backend,type,le}`	Histogram	Histogram of entry sizes being read from the underlying storage
`ds_backend_entry_size_written_bucket{backend,type,le}`	Histogram	Histogram of entry sizes being written to the underlying storage
`ds_backend_filter_use_indexed{backend,type}`	Gauge	Number of indexed searches performed against the backend
`ds_backend_filter_use_start_time_seconds{backend,type}`	Gauge	Time when recording started for statistical information about the simple search filters processed against the backend
`ds_backend_filter_use_unindexed{backend,type}`	Gauge	Number of unindexed searches performed against the backend
`ds_backend_index_uses_count{backend_id,base_dn,index}`	Summary	Current number of index reads
`ds_backend_index_uses_total{backend_id,base_dn,index}`	Summary	Total number of index reads
`ds_backend_is_private{backend,type}`	Gauge	Whether the base DNs of this backend should be considered public or private
`ds_backend_ttl_entries_deleted_count{backend,type}`	Summary	Summary for entries purged by time-to-live
`ds_backend_ttl_entries_deleted_total{backend,type}`	Summary	Summary for entries purged by time-to-live

Name	Type	Description
`ds_backend_ttl_is_running{backend,type}`	Gauge	Indicates whether time-to-live is in the process of purging expired entries
`ds_backend_ttl_last_run_time_seconds{backend,type}`	Gauge	Last date and time when time-to-live finished purging expired entries
`ds_backend_ttl_queue_size{backend,type}`	Gauge	Number of entries queued for purging by the time-to-live service
`ds_backend_ttl_thread_count{backend,type}`	Gauge	Number of active time-to-live threads
`ds_certificates_certificate_expires_at_seconds{alias,key_manager}`	Gauge	Certificate expiration date and time
`ds_connection_handlers_http_active_connections_count{http_handler}`	Gauge	Number of active client connections
`ds_connection_handlers_http_bytes_read_count{http_handler}`	Summary	Network bytes read summary
`ds_connection_handlers_http_bytes_read_total{http_handler}`	Summary	Network bytes read summary
`ds_connection_handlers_http_bytes_written_count{http_handler}`	Summary	Network bytes written summary
`ds_connection_handlers_http_bytes_written_total{http_handler}`	Summary	Network bytes written summary
`ds_connection_handlers_http_requests_count{http_handler,type}`	Summary	Delete request timer
`ds_connection_handlers_http_requests_count{http_handler,type}`	Summary	GET request timer
`ds_connection_handlers_http_requests_count{http_handler,type}`	Summary	PATCH request timer
`ds_connection_handlers_http_requests_count{http_handler,type}`	Summary	POST request timer
`ds_connection_handlers_http_requests_count{http_handler,type}`	Summary	PUT request timer
`ds_connection_handlers_http_requests_count{http_handler,type}`	Summary	Uncategorized request timer

Name	Type	Description
`ds_connection_handlers_http_requests_failure_count{http_handler,type}`	Summary	Timer for apparently valid requests that failed because the server was not able to process them (associated LDAP result codes: busy (51), unavailable (52), unwilling to perform (53) and other (80); associated HTTP status codes: server error (5xx))
`ds_connection_handlers_http_requests_failure_count{http_handler,type}`	Summary	Timer for requests that could not complete because further action is required (associated HTTP status codes: redirection (3xx))
`ds_connection_handlers_http_requests_failure_count{http_handler,type}`	Summary	Timer for requests that failed because there was a problem while attempting to perform the associated operation (associated LDAP result codes: 1, 2, 12, 15, 16, 17, 18, 19, 20, 21, 23, 34, 35, 36, 37, 38, 39; associated HTTP status codes: client error (4xx) except 401 and 403)
`ds_connection_handlers_http_requests_failure_count{http_handler,type}`	Summary	Timer for requests that failed due to uncategorized reasons
`ds_connection_handlers_http_requests_failure_count{http_handler,type}`	Summary	Timer for requests that failed for security reasons (associated LDAP result codes: 8, 9, 13, 25, 26, 27; associated HTTP status codes: unauthorized (401) and forbidden (403))
`ds_connection_handlers_http_requests_failure_seconds_total{http_handler,type}`	Summary	Timer for apparently valid requests that failed because the server was not able to process them (associated LDAP result codes: busy (51), unavailable (52), unwilling to perform (53) and other (80); associated HTTP status codes: server error (5xx))
`ds_connection_handlers_http_requests_failure_seconds_total{http_handler,type}`	Summary	Timer for requests that could not complete because further action is required (associated HTTP status codes: redirection (3xx))
`ds_connection_handlers_http_requests_failure_seconds_total{http_handler,type}`	Summary	Timer for requests that failed because there was a problem while attempting to perform the associated operation (associated LDAP result codes: 1, 2, 12, 15, 16, 17, 18, 19, 20, 21, 23, 34, 35, 36, 37, 38, 39; associated HTTP status codes: client error (4xx) except 401 and 403)
`ds_connection_handlers_http_requests_failure_seconds_total{http_handler,type}`	Summary	Timer for requests that failed due to uncategorized reasons
`ds_connection_handlers_http_requests_failure_seconds_total{http_handler,type}`	Summary	Timer for requests that failed for security reasons (associated LDAP result codes: 8, 9, 13, 25, 26, 27; associated HTTP status codes: unauthorized (401) and forbidden (403))

Name	Type	Description
`ds_connection_handlers_http_requests_failure_seconds{http_handler,type,quantile}`	Summary	Timer for apparently valid requests that failed because the server was not able to process them (associated LDAP result codes: busy (51), unavailable (52), unwilling to perform (53) and other (80); associated HTTP status codes: server error (5xx))
`ds_connection_handlers_http_requests_failure_seconds{http_handler,type,quantile}`	Summary	Timer for requests that could not complete because further action is required (associated HTTP status codes: redirection (3xx))
`ds_connection_handlers_http_requests_failure_seconds{http_handler,type,quantile}`	Summary	Timer for requests that failed because there was a problem while attempting to perform the associated operation (associated LDAP result codes: 1, 2, 12, 15, 16, 17, 18, 19, 20, 21, 23, 34, 35, 36, 37, 38, 39; associated HTTP status codes: client error (4xx) except 401 and 403)
`ds_connection_handlers_http_requests_failure_seconds{http_handler,type,quantile}`	Summary	Timer for requests that failed due to uncategorized reasons
`ds_connection_handlers_http_requests_failure_seconds{http_handler,type,quantile}`	Summary	Timer for requests that failed for security reasons (associated LDAP result codes: 8, 9, 13, 25, 26, 27; associated HTTP status codes: unauthorized (401) and forbidden (403))
`ds_connection_handlers_http_requests_seconds_total{http_handler,type}`	Summary	Delete request timer
`ds_connection_handlers_http_requests_seconds_total{http_handler,type}`	Summary	GET request timer
`ds_connection_handlers_http_requests_seconds_total{http_handler,type}`	Summary	PATCH request timer
`ds_connection_handlers_http_requests_seconds_total{http_handler,type}`	Summary	POST request timer
`ds_connection_handlers_http_requests_seconds_total{http_handler,type}`	Summary	PUT request timer
`ds_connection_handlers_http_requests_seconds_total{http_handler,type}`	Summary	Uncategorized request timer

Name	Type	Description
`ds_connection_handlers_http_requests_seconds{http_handler,type,quantile}`	Summary	Delete request timer
`ds_connection_handlers_http_requests_seconds{http_handler,type,quantile}`	Summary	GET request timer
`ds_connection_handlers_http_requests_seconds{http_handler,type,quantile}`	Summary	PATCH request timer
`ds_connection_handlers_http_requests_seconds{http_handler,type,quantile}`	Summary	POST request timer
`ds_connection_handlers_http_requests_seconds{http_handler,type,quantile}`	Summary	PUT request timer
`ds_connection_handlers_http_requests_seconds{http_handler,type,quantile}`	Summary	Uncategorized request timer
`ds_connection_handlers_ldap_abandoned_requests{ldap_handler}`	Counter	Total number of abandoned operations since startup
`ds_connection_handlers_ldap_active_connections_count{ldap_handler}`	Gauge	Number of active client connections
`ds_connection_handlers_ldap_active_persistent_searches{ldap_handler}`	Gauge	Number of active persistent searches
`ds_connection_handlers_ldap_bytes_read_count{ldap_handler}`	Summary	Network bytes read summary
`ds_connection_handlers_ldap_bytes_read_total{ldap_handler}`	Summary	Network bytes read summary
`ds_connection_handlers_ldap_bytes_written_count{ldap_handler}`	Summary	Network bytes written summary
`ds_connection_handlers_ldap_bytes_written_total{ldap_handler}`	Summary	Network bytes written summary
`ds_connection_handlers_ldap_connections_count{ldap_handler}`	Summary	Connection summary

Name	Type	Description
`ds_connection_handlers_ldap_connections_total{ldap_handler}`	Summary	Connection summary
`ds_connection_handlers_ldap_requests_count{ldap_handler,scope,type}`	Summary	Base object search request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,scope,type}`	Summary	One level search request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,scope,type}`	Summary	Subtree search request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,type}`	Summary	Abandon request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,type}`	Summary	Add request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,type}`	Summary	Bind request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,type}`	Summary	Compare request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,type}`	Summary	Delete request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,type}`	Summary	Extended request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,type}`	Summary	Modify DN request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,type}`	Summary	Modify request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,type}`	Summary	Unbind request timer
`ds_connection_handlers_ldap_requests_count{ldap_handler,type}`	Summary	Uncategorized request timer
`ds_connection_handlers_ldap_requests_failure_count{ldap_handler,type}`	Summary	Timer for apparently valid requests that failed because the server was not able to process them (associated LDAP result codes: busy (51), unavailable (52), unwilling to perform (53) and other (80); associated HTTP status codes: server error (5xx))

Name	Type	Description
`ds_connection_handlers_ldap_requests_failure_count{ldap_handler,type}`	Summary	Timer for requests that failed because the server did not hold the request targeted entry (but was able to provide alternative servers that may) (associated LDAP result code: 10)
`ds_connection_handlers_ldap_requests_failure_count{ldap_handler,type}`	Summary	Timer for requests that failed because there was a problem while attempting to perform the associated operation (associated LDAP result codes: 1, 2, 12, 15, 16, 17, 18, 19, 20, 21, 23, 34, 35, 36, 37, 38, 39; associated HTTP status codes: client error (4xx) except 401 and 403)
`ds_connection_handlers_ldap_requests_failure_count{ldap_handler,type}`	Summary	Timer for requests that failed because they were trying to exceed the resource limits allocated to the associated clients (associated LDAP result codes: time, size and admin limit exceeded (respectively 4, 5 and 11)
`ds_connection_handlers_ldap_requests_failure_count{ldap_handler,type}`	Summary	Timer for requests that failed due to uncategorized reasons
`ds_connection_handlers_ldap_requests_failure_count{ldap_handler,type}`	Summary	Timer for requests that failed for security reasons (associated LDAP result codes: 8, 9, 13, 25, 26, 27; associated HTTP status codes: unauthorized (401) and forbidden (403))
`ds_connection_handlers_ldap_requests_failure_seconds_total{ldap_handler,type}`	Summary	Timer for apparently valid requests that failed because the server was not able to process them (associated LDAP result codes: busy (51), unavailable (52), unwilling to perform (53) and other (80); associated HTTP status codes: server error (5xx))
`ds_connection_handlers_ldap_requests_failure_seconds_total{ldap_handler,type}`	Summary	Timer for requests that failed because the server did not hold the request targeted entry (but was able to provide alternative servers that may) (associated LDAP result code: 10)
`ds_connection_handlers_ldap_requests_failure_seconds_total{ldap_handler,type}`	Summary	Timer for requests that failed because there was a problem while attempting to perform the associated operation (associated LDAP result codes: 1, 2, 12, 15, 16, 17, 18, 19, 20, 21, 23, 34, 35, 36, 37, 38, 39; associated HTTP status codes: client error (4xx) except 401 and 403)
`ds_connection_handlers_ldap_requests_failure_seconds_total{ldap_handler,type}`	Summary	Timer for requests that failed because they were trying to exceed the resource limits allocated to the associated clients (associated LDAP result codes: time, size and admin limit exceeded (respectively 4, 5 and 11)

Name	Type	Description
`ds_connection_handlers_ldap_requests_failure_seconds_total{ldap_handler,type}`	Summary	Timer for requests that failed due to uncategorized reasons
`ds_connection_handlers_ldap_requests_failure_seconds_total{ldap_handler,type}`	Summary	Timer for requests that failed for security reasons (associated LDAP result codes: 8, 9, 13, 25, 26, 27; associated HTTP status codes: unauthorized (401) and forbidden (403))
`ds_connection_handlers_ldap_requests_failure_seconds{ldap_handler,type,quantile}`	Summary	Timer for apparently valid requests that failed because the server was not able to process them (associated LDAP result codes: busy (51), unavailable (52), unwilling to perform (53) and other (80); associated HTTP status codes: server error (5xx))
`ds_connection_handlers_ldap_requests_failure_seconds{ldap_handler,type,quantile}`	Summary	Timer for requests that failed because the server did not hold the request targeted entry (but was able to provide alternative servers that may) (associated LDAP result code: 10)
`ds_connection_handlers_ldap_requests_failure_seconds{ldap_handler,type,quantile}`	Summary	Timer for requests that failed because there was a problem while attempting to perform the associated operation (associated LDAP result codes: 1, 2, 12, 15, 16, 17, 18, 19, 20, 21, 23, 34, 35, 36, 37, 38, 39; associated HTTP status codes: client error (4xx) except 401 and 403)
`ds_connection_handlers_ldap_requests_failure_seconds{ldap_handler,type,quantile}`	Summary	Timer for requests that failed because they were trying to exceed the resource limits allocated to the associated clients (associated LDAP result codes: time, size and admin limit exceeded (respectively 4, 5 and 11)
`ds_connection_handlers_ldap_requests_failure_seconds{ldap_handler,type,quantile}`	Summary	Timer for requests that failed due to uncategorized reasons
`ds_connection_handlers_ldap_requests_failure_seconds{ldap_handler,type,quantile}`	Summary	Timer for requests that failed for security reasons (associated LDAP result codes: 8, 9, 13, 25, 26, 27; associated HTTP status codes: unauthorized (401) and forbidden (403))
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,scope,type}`	Summary	Base object search request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,scope,type}`	Summary	One level search request timer

Name	Type	Description
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,scope,type}`	Summary	Subtree search request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,type}`	Summary	Abandon request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,type}`	Summary	Add request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,type}`	Summary	Bind request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,type}`	Summary	Compare request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,type}`	Summary	Delete request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,type}`	Summary	Extended request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,type}`	Summary	Modify DN request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,type}`	Summary	Modify request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,type}`	Summary	Unbind request timer
`ds_connection_handlers_ldap_requests_seconds_total{ldap_handler,type}`	Summary	Uncategorized request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,scope,type,quantile}`	Summary	Base object search request timer

Name	Type	Description
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,scope,type,quantile}`	Summary	One level search request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,scope,type,quantile}`	Summary	Subtree search request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Abandon request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Add request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Bind request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Compare request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Delete request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Extended request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Modify DN request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Modify request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Unbind request timer
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Uncategorized request timer

Name	Type	Description
`ds_current_connections`	Gauge	Number of client connections currently established with the Directory Server
`ds_current_time_seconds`	Gauge	Current date and time
`ds_disk_free_space_bytes{disk}`	Gauge	Amount of free disk space
`ds_disk_free_space_full_threshold_bytes{disk}`	Gauge	Effective full disk space threshold
`ds_disk_free_space_low_threshold_bytes{disk}`	Gauge	Effective low disk space threshold
`ds_entry_cache_entry_count{cache}`	Gauge	Current number of entries held in this cache
`ds_entry_cache_max_entry_count{cache}`	Gauge	Maximum number of entries allowed in this cache
`ds_entry_cache_max_size_bytes{cache}`	Gauge	Memory limit for this cache
`ds_entry_cache_misses_count{cache}`	Summary	Number of attempts to retrieve an entry that was not held in this cache
`ds_entry_cache_misses_total{cache}`	Summary	Number of attempts to retrieve an entry that was not held in this cache
`ds_entry_cache_total_tries_count{cache}`	Summary	Number of attempts to retrieve an entry from this cache
`ds_entry_cache_total_tries_total{cache}`	Summary	Number of attempts to retrieve an entry from this cache
`ds_health_status_alive`	Gauge	Indicates whether the server is alive
`ds_health_status_healthy`	Gauge	Indicates whether the server is able to handle requests
`ds_jvm_available_cpus`	Gauge	Number of processors available to the Java virtual machine
`ds_jvm_classes_loaded`	Gauge	Number of classes loaded since the Java virtual machine started
`ds_jvm_classes_unloaded`	Gauge	Number of classes unloaded since the Java virtual machine started
`ds_jvm_memory_heap_init_bytes`	Gauge	Amount of heap memory that the Java virtual machine initially requested from the operating system

Name	Type	Description
`ds_jvm_memory_heap_max_bytes`	Gauge	Maximum amount of heap memory that the Java virtual machine will attempt to use
`ds_jvm_memory_heap_reserved_bytes`	Gauge	Amount of heap memory that is committed for the Java virtual machine to use
`ds_jvm_memory_heap_used_bytes`	Gauge	Amount of heap memory used by the Java virtual machine
`ds_jvm_memory_init_bytes`	Gauge	Amount of memory that the Java virtual machine initially requested from the operating system
`ds_jvm_memory_max_bytes`	Gauge	Maximum amount of memory that the Java virtual machine will attempt to use
`ds_jvm_memory_non_heap_init_bytes`	Gauge	Amount of non-heap memory that the Java virtual machine initially requested from the operating system
`ds_jvm_memory_non_heap_max_bytes`	Gauge	Maximum amount of non-heap memory that the Java virtual machine will attempt to use
`ds_jvm_memory_non_heap_reserved_bytes`	Gauge	Amount of non-heap memory that is committed for the Java virtual machine to use
`ds_jvm_memory_non_heap_used_bytes`	Gauge	Amount of non-heap memory used by the Java virtual machine
`ds_jvm_memory_reserved_bytes`	Gauge	Amount of memory that is committed for the Java virtual machine to use
`ds_jvm_memory_used_bytes`	Gauge	Amount of memory used by the Java virtual machine
`ds_jvm_threads_blocked_count`	Gauge	Number of threads in the BLOCKED state
`ds_jvm_threads_count`	Gauge	Number of live threads including both daemon and non-daemon threads
`ds_jvm_threads_daemon_count`	Gauge	Number of live daemon threads
`ds_jvm_threads_deadlock_count`	Gauge	Number of deadlocked threads
`ds_jvm_threads_new_count`	Gauge	Number of threads in the NEW state
`ds_jvm_threads_runnable_count`	Gauge	Number of threads in the RUNNABLE state
`ds_jvm_threads_terminated_count`	Gauge	Number of threads in the TERMINATED state
`ds_jvm_threads_timed_waiting_count`	Gauge	Number of threads in the TIMED_WAITING state

Name	Type	Description
`ds_jvm_threads_waiting_count`	Gauge	Number of threads in the WAITING state
`ds_max_connections`	Gauge	Maximum number of simultaneous client connections that have been established with the Directory Server
`ds_replication_changelog_connected_changelogs_current_receive_window{changelog_id,domain_name,dc}`	Gauge	Current replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds_replication_changelog_connected_changelogs_current_receive_window{changelog_id,domain_name}`	Gauge	Current replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds_replication_changelog_connected_changelogs_current_send_window{changelog_id,domain_name,dc}`	Gauge	Current replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds_replication_changelog_connected_changelogs_current_send_window{changelog_id,domain_name}`	Gauge	Current replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds_replication_changelog_connected_changelogs_domain_generation_id{changelog_id,domain_name,dc}`	Gauge	Replication domain generation identifier
`ds_replication_changelog_connected_changelogs_domain_generation_id{changelog_id,domain_name}`	Gauge	Replication domain generation identifier
`ds_replication_changelog_connected_changelogs_max_receive_window{changelog_id,domain_name,dc}`	Gauge	Maximum replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds_replication_changelog_connected_changelogs_max_receive_window{changelog_id,domain_name}`	Gauge	Maximum replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds_replication_changelog_connected_changelogs_max_send_window{changelog_id,domain_name,dc}`	Gauge	Maximum replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size

Name	Type	Description
`ds_replication_changelog_connected_changelogs_max_send_window{changelog_id,domain_name}`	Gauge	Maximum replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds_replication_changelog_connected_changelogs_missing_changes{changelog_id,domain_name,dc}`	Gauge	Missing changes for replication
`ds_replication_changelog_connected_changelogs_missing_changes{changelog_id,domain_name}`	Gauge	Missing changes for replication
`ds_replication_changelog_connected_changelogs_ssl_encryption{changelog_id,domain_name,dc}`	Gauge	Whether SSL encryption is used when exchanging messages with this server
`ds_replication_changelog_connected_changelogs_ssl_encryption{changelog_id,domain_name}`	Gauge	Whether SSL encryption is used when exchanging messages with this server
`ds_replication_changelog_connected_replicas_current_receive_window{domain_name,dc,server_id}`	Gauge	Current replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds_replication_changelog_connected_replicas_current_receive_window{domain_name,server_id}`	Gauge	Current replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds_replication_changelog_connected_replicas_current_send_window{domain_name,dc,server_id}`	Gauge	Current replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds_replication_changelog_connected_replicas_current_send_window{domain_name,server_id}`	Gauge	Current replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds_replication_changelog_connected_replicas_domain_generation_id{domain_name,dc,server_id}`	Gauge	Replication domain generation identifier

Name	Type	Description
`ds_replication_changelog_connected_replicas_domain_generation_id{domain_name,server_id}`	Gauge	Replication domain generation identifier
`ds_replication_changelog_connected_replicas_max_receive_window{domain_name,dc,server_id}`	Gauge	Maximum replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds_replication_changelog_connected_replicas_max_receive_window{domain_name,server_id}`	Gauge	Maximum replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds_replication_changelog_connected_replicas_max_send_window{domain_name,dc,server_id}`	Gauge	Maximum replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds_replication_changelog_connected_replicas_max_send_window{domain_name,server_id}`	Gauge	Maximum replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds_replication_changelog_connected_replicas_ssl_encryption{domain_name,dc,server_id}`	Gauge	Whether SSL encryption is used when exchanging messages with this server
`ds_replication_changelog_connected_replicas_ssl_encryption{domain_name,server_id}`	Gauge	Whether SSL encryption is used when exchanging messages with this server
`ds_replication_changelog_domain_generation_id{domain_name,dc}`	Gauge	Replication domain generation identifier
`ds_replication_changelog_domain_generation_id{domain_name}`	Gauge	Replication domain generation identifier
`ds_replication_changelog_missing_changes{domain_name,dc}`	Gauge	Missing changes for replication
`ds_replication_changelog_missing_changes{domain_name}`	Gauge	Missing changes for replication
`ds_replication_changelog_newest_change_number`	Gauge	Newest change number present in the change number index database

Name	Type	Description
`ds_replication_changelog_oldest_change_number`	Gauge	Oldest change number present in the change number index database
`ds_replication_changelog_replica_dbs_newest_csn_timestamp_seconds{domain_name,dc,server_id}`	Gauge	Timestamp of the newest CSN present in the replica database
`ds_replication_changelog_replica_dbs_oldest_csn_timestamp_seconds{domain_name,dc,server_id}`	Gauge	Timestamp of the oldest CSN present in the replica database
`ds_replication_replica_current_receive_window`	Gauge	Current replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds_replication_replica_current_send_window`	Gauge	Current replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds_replication_replica_domain_generation_id`	Gauge	Replication domain generation identifier
`ds_replication_replica_entries_awaiting_updates_count`	Gauge	Number of entries for which an update operation has been received but not replayed yet by this replica
`ds_replication_replica_lost_connections`	Gauge	Number of times the replica lost its connection to the replication server
`ds_replication_replica_max_receive_window`	Gauge	Maximum replication window size for receiving messages, indicating the number of replication messages a remote server can send before waiting on acknowledgement from this server. This does not depend on the TCP window size
`ds_replication_replica_max_send_window`	Gauge	Maximum replication window size for sending messages, indicating the number of replication messages this server can send before waiting on acknowledgement from the receiving server. This does not depend on the TCP window size
`ds_replication_replica_remote_replicas_current_delay_seconds{domain_name,dc,remote_server_id,server_id}`	Gauge	Current local delay in receiving replicated operations

Name	Type	Description
`ds_replication_replica_remote_replicas_receive_delay_seconds{domain_name,dc,remote_server_id,server_id}`	Gauge	Current local delay in receiving replicated operations
`ds_replication_replica_remote_replicas_replay_delay_seconds{domain_name,dc,remote_server_id,server_id}`	Gauge	Current local delay in replaying replicated operations
`ds_replication_replica_remote_replicas_replayed_updates_count{domain_name,dc,remote_server_id,server_id}`	Summary	Timer for updates that have been replayed on this replica
`ds_replication_replica_remote_replicas_replayed_updates_seconds_total{domain_name,dc,remote_server_id,server_id}`	Summary	Timer for updates that have been replayed on this replica
`ds_replication_replica_remote_replicas_replayed_updates_seconds{domain_name,dc,remote_server_id,server_id,quantile}`	Summary	Timer for updates that have been replayed on this replica
`ds_replication_replica_replayed_internal_updates{domain_name,server_id}`	Counter	Number of updates replayed on this replica which modify the internal state but not user data
`ds_replication_replica_replayed_updates_conflicts_resolved`	Counter	Number of updates replayed on this replica for which replication naming conflicts have been resolved
`ds_replication_replica_replayed_updates_conflicts_unresolved`	Counter	Number of updates replayed on this replica for which replication naming conflicts have not been resolved
`ds_replication_replica_replayed_updates_count`	Summary	Timer for updates that have been replayed on this replica
`ds_replication_replica_replayed_updates_seconds_total`	Summary	Timer for updates that have been replayed on this replica
`ds_replication_replica_replayed_updates_seconds{quantile}`	Summary	Timer for updates that have been replayed on this replica
`ds_replication_replica_sent_updates`	Counter	Number of replication updates sent by this replica
`ds_replication_replica_ssl_encryption`	Gauge	Whether SSL encryption is used when exchanging messages with this server

Name	Type	Description
`ds_replication_replica_status_last_changed_seconds`	Gauge	Last date and time the replication status of the local replica changed
`ds_replication_replica_updates_already_in_progress{domain_name,server_id}`	Counter	Number of duplicate updates: updates received by this replica which cannot be applied because they are already in progress. Can happen when a directory server fails over to another replication server
`ds_replication_replica_updates_inbound_queue`	Gauge	Number of remote updates received from the replication server but not replayed yet on this replica
`ds_replication_replica_updates_outbound_queue`	Gauge	Number of local updates that are waiting to be sent to the replication server once they complete
`ds_start_time_seconds`	Gauge	Start date and time for the Directory Server
`ds_topology_servers_server_is_local{server_id}`	Gauge	Indicates whether this is the topology server that has handled the monitoring request
`ds_total_connections`	Gauge	Total number of client connections that have been established with the Directory Server since it started
`ds_work_queue_requests_in_queue`	Gauge	Number of requests in the work queue that have not yet been picked up for processing
`ds_work_queue_requests_submitted_count`	Summary	Summary for operations that have been successfully submitted to the work queue
`ds_work_queue_requests_submitted_total`	Summary	Summary for operations that have been successfully submitted to the work queue

Use LDAP

This guide shows you how to use DS LDAP features and command-line tools.

Tools

Find DS tools.

Authentication

Understand LDAP binds.

Searches

Look up DS entries.

Comparisons

Compare LDAP attributes.

Updates

Add, modify, rename, delete.

Passwords

Manage passwords.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

About DS tools

Client tools

- Add DS client command-line tools to your PATH:

Bash

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

PowerShell

```
PS C:\path\to> $env:PATH += ";C:\path\to\opendj\bat"
```

- For reference information, use the `--help` option with any DS tool.
- All commands call Java programs. This means every command starts a JVM, so it takes longer to start than a native binary.

Command[1]	Description
`addrate`	Measure add and delete throughput and response time.
`authrate`	Measure bind throughput and response time.
`base64`	Encode and decode data in base64 format. Base64-encoding represents binary data in ASCII, and can be used to encode character strings in LDIF, for example.
`ldapcompare`	Compare the attribute values you specify with those stored on entries in the directory.
`ldapdelete`	Delete entries from the directory.
`ldapmodify`	Modify the specified attribute values for the specified entries.
`ldappasswordmodify`	Modify user passwords.
`ldapsearch`	Search a branch of directory data for entries that match the LDAP filter you specify.
`ldifdiff`	Display differences between two LDIF files, with the resulting output having LDIF format.

Command[1]	Description
`ldifmodify`	Modify specified attribute values for specified entries in an LDIF file.
`ldifsearch`	Search a branch of data in LDIF for entries matching the LDAP filter you specify.
`makeldif`	Generate directory data in LDIF based on templates that define how the data should appear. Also see makeldif-template.
`modrate`	Measure modification throughput and response time.
`searchrate`	Measure search throughput and response time.

[1] UNIX names for the commands. Equivalent Windows commands have .bat extensions.

Trusted certificates

When a client tool initiates a secure connection to a server, the server presents its digital certificate.

The tool must determine whether it trusts the server certificate and continues to negotiate a secure connection, or does not trust the server certificate and drops the connection. To trust the server certificate, the tool's truststore must contain the trusted certificate. The trusted certificate is a CA certificate, or the self-signed server certificate.

The following table explains how the tools locate the truststore.

Truststore Option	Truststore Used
None	The default truststore, `user.home/.opendj/keystore`, where *user.home* is the Java system property. *user.home* is `$HOME` on Linux and UNIX, and `%USERPROFILE%` on Windows. The keystore password is `OpenDJ`. Neither the file name, nor the password can be changed. • In interactive mode, DS command-line tools prompt for approval to trust an unrecognized certificate, and whether to store it in the default truststore for future use. • In silent mode, the tools rely on the default truststore.
`--use<Type>TrustStore {trustStorePath}`	Only the specified truststore is used. The <*Type*> in the option name reflects the trust store type. The tool fails with an error if it cannot trust the server certificate.

Default settings

You can set defaults in the `~/.opendj/tools.properties` file as in the following example:

```
hostname=localhost
port=1636
bindDN=uid=kvaughan,ou=People,dc=example,dc=com
useSsl=true
```

The file location on Windows is `%UserProfile%\.opendj\tools.properties`.

To override the settings, use the `--noPropertiesFile` option.

Authentication (binds)

Authentication is the act of confirming the identity of a principal. Authorization is the act of determining whether to grant or to deny access to a principal. Authentication is performed to make authorization decisions.

DS servers implement fine-grained access control for authorization. Authorization for an operation depends on who is requesting the operation. DS servers must authenticate the principal before making an authorization decision. In LDAP, the bind operation authenticates the principal.

Clients bind by providing a means to find their principal's entry, and credentials to check against the entry:

- In a simple bind operation, the client provides an LDAP name, usually the DN identifying its entry, and the corresponding password stored in the entry.

  In the simplest bind operation, the client provides a zero-length name and a zero-length password. This results in an anonymous bind, meaning the client is authenticated as an anonymous user of the directory. LDAP servers may allow anonymous binds to read public information, such as root DSE attributes.

- Other bind mechanisms involve digital certificates, Kerberos tickets, or challenge response mechanisms that prove the client knows a password.

A user rarely knows, let alone enters, their DN. Instead, a user provides a client application with an identifying string stored in their entry, such as a user ID or an email address. The client application builds the DN directly from the user's identity string, or searches for the user entry based on the user's identity string to find the DN. The client application performs a simple bind with the resulting DN.

For example, suppose Babs Jensen enters her email address, `bjensen@example.com`, and password. The client application might search for the entry matching `(mail=bjensen@example.com)` under base DN `dc=example,dc=com`. Alternatively, the client application might extract the user ID `bjensen` from the address, then build the corresponding DN, `uid=bjensen,ou=people,dc=example,dc=com`, without a lookup.

Identity mappers

When the mapping from the user identifier to the DN is known, DS servers can use an *identity mapper* to do the translation. Identity mappers are used to perform PLAIN SASL authentication (with a user name and password), SASL GSSAPI authentication (Kerberos V5), SASL CRAM MD5, and DIGEST MD5 authentication. They also map authorization IDs to DNs for password modify extended operations and proxied authorization.

One use of PLAIN SASL is to translate user names from HTTP Basic authentication to LDAP authentication. The following example shows PLAIN SASL authentication using the default exact match identity mapper. In this example, Babs Jensen has access to read the hashed value of her password. Notice the authentication ID is her user ID, `u:bjensen`, rather than the DN of her entry:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN dc=example,dc=com \
 --saslOption mech=PLAIN \
 --saslOption authid=u:bjensen \
 --bindPassword hifalutin \
 "(cn=Babs Jensen)" \
 userPassword

dn: uid=bjensen,ou=People,dc=example,dc=com
userPassword: {PBKDF2-HMAC-SHA256}10:<hash>
```

The Exact Match Identity Mapper searches for a match between the string (here, `bjensen`), and the value of a specified attribute (by default, the `uid` attribute). By default, the identity mapper searches all public naming contexts local to the server. If duplicate entries exist, or if the required indexes are not available for all backends, this behavior can be restricted using the `match-base-dn` property.

You can configure multiple identity mappers, if necessary. When resolving the identity, the server uses the first identity mapper that finds a match. If multiple identity mappers match different entries, however, then the server returns LDAP error code 19, Constraint Violation.

If you know that users are entering their email addresses, you could create an exact match identity mapper for email addresses, then use that for PLAIN SASL authentication:

```
$ dsconfig \
 create-identity-mapper \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --mapper-name "Email Mapper" \
 --type exact-match \
 --set match-attribute:mail \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-sasl-mechanism-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name PLAIN \
 --set identity-mapper:"Email Mapper" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN dc=example,dc=com \
 --saslOption mech=PLAIN \
 --saslOption authid=u:bjensen@example.com \
 --bindPassword hifalutin \
 "(cn=Babs Jensen)" \
 userPassword

dn: uid=bjensen,ou=People,dc=example,dc=com
userPassword: {PBKDF2-HMAC-SHA256}10:<hash>
```

A Regular Expression Identity Mapper uses a regular expression to extract a substring from the string provided. The server searches for a match between the substring and the value of a specified attribute. When an email address is user ID + @ + domain, you can use the default regular expression identity mapper in the same way as the email mapper in the example above. The default regular expression pattern is `^([^@]+)@.+$`, and the part of the identity string matching `([^@]+)` is used to find the entry by user ID:

```
$ dsconfig \
 set-sasl-mechanism-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name PLAIN \
 --set identity-mapper:"Regular Expression" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN dc=example,dc=com \
 --saslOption mech=PLAIN \
 --saslOption authid=u:bjensen@example.com \
 --bindPassword hifalutin \
 "(cn=Babs Jensen)" \
 userPassword

dn: uid=bjensen,ou=People,dc=example,dc=com
userPassword: {PBKDF2-HMAC-SHA256}10:<hash>
```

Use the `dsconfig` command interactively to experiment with `match-pattern` and `replace-pattern` settings for the regular expression identity mapper. The `match-pattern` can be any `javax.util.regex.Pattern` regular expression.

Like the exact match identity mapper, the regular expression identity mapper searches all public naming contexts local to the server by default. If duplicate entries exist, this behavior can be restricted using the `match-base-dn` property.

LDAP search

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

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

Whereas a phone book has only one index (alphabetical order by name), the directory has many indexes. When performing a search, you specify which attributes to use, and the server derives the corresponding indexes.

The phone book might be divided into white pages for residential subscribers and yellow pages for businesses. If you look up an individual's phone number, you limit your search to the white pages. Directory services divide entries in various ways. For example, they can store organizations and groups in different locations from user entries or printer accounts. When searching the directory, you therefore also specify where to search.

The `ldapsearch` command requires arguments for at least the search base DN option and an LDAP filter. The search base DN identifies where in the directory to search for entries that match the filter. For example, if you are looking for printers, you might use `ou=Printers,dc=example,dc=com`. In the `GNB00` office, you could look up a printer as shown in the following example:

```
$ ldapsearch --baseDN ou=Printers,dc=example,dc=com "(printerLocation=GNB00)"
```

In the example above, the LDAP filter matches printer entries where the `printerLocation` attribute is equal to `GNB00`.

You also specify the host and port to access directory services, and the protocol to use, such as LDAP or LDAPS. If the directory service does not allow anonymous access to the data you want to search, you supply credentials, such as a username and password, or a public key certificate. You can optionally specify a list of attributes to return. If you do not specify attributes, then the search returns all user attributes for the entry.

For details about the operators that can be used in search filters, see LDAP filter operators.

Simple LDAP filter

The following example searches for entries with user IDs (`sn`) equal to `hall`, returning only DNs and user ID values:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(sn=hall)" \
 uid

dn: uid=ahall,ou=People,dc=example,dc=com
uid: ahall

dn: uid=bhal2,ou=People,dc=example,dc=com
uid: bhal2

dn: uid=bhall,ou=People,dc=example,dc=com
uid: bhall
```

Complex LDAP filter

The following example returns entries with `sn` equal to `jensen` for users located in San Francisco:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN ou=people,dc=example,dc=com \
 "(&(sn=jensen)(l=San Francisco))" \
 @person

dn: uid=bjensen,ou=People,dc=example,dc=com
objectClass: person
objectClass: cos
objectClass: oauth2TokenObject
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: posixAccount
objectClass: top
cn: Barbara Jensen
cn: Babs Jensen
description: Original description
sn: Jensen
telephoneNumber: +1 408 555 1862

dn: uid=rjensen,ou=People,dc=example,dc=com
objectClass: person
objectClass: cos
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: posixAccount
objectClass: top
cn: Richard Jensen
description: Description on ou=People
sn: Jensen
telephoneNumber: +1 408 555 5957
```

The command returns the attributes associated with the `person` object class.

Complex filters can use both "and" syntax, `(&(filtercomp)(filtercomp))`, and "or" syntax, `(|(filtercomp)(filtercomp))`.

Return operational attributes

Operational attributes are returned only when explicitly requested. Use `+` in the attribute list after the filter to return all operational attributes:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 +

dn: uid=bjensen,ou=People,dc=example,dc=com
entryDN: uid=bjensen,ou=People,dc=example,dc=com
entryUUID: <uuid>
etag: <etag>
hasSubordinates: false
isMemberOf: cn=Carpoolers,ou=Self Service,ou=Groups,dc=example,dc=com
numSubordinates: 0
structuralObjectClass: inetOrgPerson
subschemaSubentry: cn=schema
```

Alternatively, specify operational attributes by name.

Return attributes of an object class

Use `@objectClass` in the attribute list to return all attributes of a particular object class:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
  @person

dn: uid=bjensen,ou=People,dc=example,dc=com
objectClass: person
objectClass: cos
objectClass: oauth2TokenObject
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: posixAccount
objectClass: top
cn: Barbara Jensen
cn: Babs Jensen
description: Original description
sn: Jensen
telephoneNumber: +1 408 555 1862
```

Approximate match

DS servers support searches for an approximate match of the filter. Approximate match searches use the `~=` comparison operator, described in LDAP filter operators. They rely on `approximate` type indexes, which are configured as shown in Approximate index.

The following example configures an approximate match index for the surname (`sn`) attribute, and then rebuilds the index:

```
$ dsconfig \
 set-backend-index-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name sn \
 --set index-type:approximate \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ rebuild-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN dc=example,dc=com \
 --index sn
```

Once the index is built, it is ready for use in searches. The following example shows a search using the approximate comparison operator:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(sn~=jansen)" \
 sn

dn: uid=ajensen,ou=People,dc=example,dc=com
sn: Jensen

dn: uid=bjense2,ou=People,dc=example,dc=com
sn: Jensen

dn: uid=bjensen,ou=People,dc=example,dc=com
sn: Jensen

dn: uid=ejohnson,ou=People,dc=example,dc=com
sn: Johnson

dn: uid=gjensen,ou=People,dc=example,dc=com
sn: Jensen

dn: uid=jjensen,ou=People,dc=example,dc=com
sn: Jensen

dn: uid=kjensen,ou=People,dc=example,dc=com
sn: Jensen

dn: uid=rjense2,ou=People,dc=example,dc=com
sn: Jensen

dn: uid=rjensen,ou=People,dc=example,dc=com
sn: Jensen

dn: uid=tjensen,ou=People,dc=example,dc=com
sn: Jensen
```

Notice that `jansen` matches `Jensen` and `Johnson`.

Escape characters in filters

RFC 4515, *Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters*, mentions a number of characters that require special handing in search filters.

For a filter like `(attr=value)`, the following list indicates characters that you must replace with a backslash (`\`) followed by two hexadecimal digits when using them as part of the value string:

- Replace `*` with `\2a`.
- Replace `(` with `\28`.
- Replace `)` with `\29`.

- Replace `\` with `\5c`.
- Replace NUL (0x00) with `\00`.

The following example shows a filter with escaped characters matching an actual value:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(cn=\28A \5cgreat\5c name\2a\29)" \
 cn

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
cn: (A \great\ name*)
```

Active accounts

DS servers support extensible matching rules. Use a filter that specifies a matching rule OID that extends the matching operator.

DS servers support time-based matching rules for use with attributes that hold timestamp values:

Name: `relativeTimeOrderingMatch.gt`

Greater-than relative time matching rule for time-based searches.

Use this in a filter to match attributes with values greater than the current time +/- an offset.

The filter `(pwdExpirationTime:1.3.6.1.4.1.26027.1.4.5:=5d)` matches entries where the password expiration time is greater than the current time plus five days. In other words, entries whose passwords expire in more than five days.

Name: `relativeTimeOrderingMatch.lt`

Less-than relative time matching rule for time-based searches.

Use this in a filter to match attributes with values less than the current time +/- an offset.

The filter `(ds-last-login-time:1.3.6.1.4.1.26027.1.4.6:=-4w)` matches entries where the last login time is less than the current time minus four weeks. In other words, accounts that have not been active in the last four weeks.

Name: `partialDateAndTimeMatchingRule`

Partial date and time matching rule for matching parts of dates in time-based searches.

The filter `(ds-last-login-time:1.3.6.1.4.1.26027.1.4.7:=2020)` matches entries where the last login time was in 2020.

The following example uses the `ds-last-login-time` attribute, which is an operational attribute (`USAGE directoryOperation`) with Generalized Time syntax (`SYNTAX 1.3.6.1.4.1.1466.115.121.1.24`).

When checking schema compliance, the server skips operational attributes. The server can therefore add operational attributes to an entry without changing the entry's object classes.

Operational attributes hold information for the directory, rather than information targeting client applications. The server returns operational attributes only when explicitly requested, and client applications generally should not be able to modify them.

As the `ds-last-login-time` attribute is operational, it has limited visibility. This helps prevent client applications from modifying its value unless specifically allowed to.

Configure the applicable password policy to write the last login timestamp when a user authenticates.

The following command configures a subentry password policy. On successful authentication, the policy causes the server to write a timestamp in generalized time format to the user's `ds-last-login-time` operational attribute:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Record last login,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
cn: Record last login
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-last-login-time-attribute: ds-last-login-time
ds-pwp-last-login-time-format: yyyyMMddHH'Z'
subtreeSpecification: { base "ou=people" }
EOF
```

The `ds-pwp-last-login-time-format` setting must:

- Match the syntax of the `ds-pwp-last-login-time-attribute` attribute, which in this example is `GeneralizedTime`.
- Be a valid format string for the `java.text.SimpleDateFormat` class.

Configure and build an index for time-based searches on the `ds-last-login-time` attribute:

```
$ dsconfig \
 create-backend-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --set index-type:extensible \
 --set index-extensible-matching-rule:1.3.6.1.4.1.26027.1.4.5 \
 --set index-extensible-matching-rule:1.3.6.1.4.1.26027.1.4.6 \
 --set index-extensible-matching-rule:1.3.6.1.4.1.26027.1.4.7 \
 --index-name ds-last-login-time \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ rebuild-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDN dc=example,dc=com \
 --index ds-last-login-time
```

Make sure you have some users who have authenticated recently:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 1.1

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 1.1
```

The following search returns users who have authenticated in the last 13 weeks:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(ds-last-login-time:1.3.6.1.4.1.26027.1.4.5:=-13w)" \
 1.1

dn: uid=bjensen,ou=People,dc=example,dc=com

dn: uid=kvaughan,ou=People,dc=example,dc=com
```

The following search returns users who have authenticated this year:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 "(ds-last-login-time:1.3.6.1.4.1.26027.1.4.7:=$(date +%Y))" \
 1.1

dn: uid=bjensen,ou=People,dc=example,dc=com

dn: uid=kvaughan,ou=People,dc=example,dc=com
```

Language subtypes

DS servers support the language subtypes listed in Support for languages and locales.

When you perform a search you can request the language subtype by OID or by language subtype string. For example, the following search gets the French version of a common name. The example uses the DS `base64` command to decode the attribute value:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(cn=Frederique Dupont)" cn\;lang-fr

dn: uid=fdupont,ou=People,dc=example,dc=com
cn;lang-fr:: RnJlZMOpcmlxdWUgRHVwb250

$ base64 decode --encodedData RnJlZMOpcmlxdWUgRHVwb250

Fredérique Dupont
```

At the end of the OID or language subtype, further specify the matching rule as follows:

- Add `.1` for less than
- Add `.2` for less than or equal to
- Add `.3` for equal to (default)
- Add `.4` for greater than or equal to
- Add `.5` for greater than
- Add `.6` for substring

LDAP filter operators

Operator	Definition	Example
`=`	Equality comparison, as in `(sn=Jensen)`. This can also be used with substring matches. For example, to match last names starting with `Jen`, use the filter `(sn=Jen*)`. Substrings are more expensive for the directory server to index. Substring searches might not be permitted, depending on the attribute.	`"(cn=My App)"` matches entries with common name `My App`. `"(sn=Jen*)"` matches entries with surname starting with `Jen`.
`<=`	Less than or equal to comparison, which works alphanumerically.	`"(cn<=App)"` matches entries with `commonName` up to those starting with App (case-insensitive) in alphabetical order.
`>=`	Greater than or equal to comparison, which works alphanumerically.	`"(uidNumber>=1151)"` matches entries with `uidNumber` greater than 1151.

Operator	Definition	Example
`=*`	Presence comparison. For example, to match all entries with a `userPassword` attribute, use the filter `(userPassword=*)`.	`"(member=*)"` matches entries with a `member` attribute.
`~=`	Approximate comparison, matching attribute values similar to the value you specify.	`"(sn~=jansen)"` matches entries with a surname that sounds similar to `Jansen` (Johnson, Jensen, and other surnames).
`[:dn][:oid]:=`	Extensible match comparison. At the end of the OID or language subtype, you further specify the matching rule as follows: • Add `.1` for less than • Add `.2` for less than or equal to • Add `.3` for equal to (default) • Add `.4` for greater than or equal to • Add `.5` for greater than • Add `.6` for substring	`(uid:dn:=bjensen)` matches entries with DN component `uid=bjensen`. `(ds-last-login-time:1.3.6.1.4.1.26027.1.4.5:=-13w)` matches entries with a last login time more recent than 13 weeks. Extensible match filters work with localized values. DS servers support internationalized locales, each of which has an OID for collation order, such as `1.3.6.1.4.1.42.2.27.9.4.76.1` for French. DS software lets you use the language subtype, such as `fr`, instead of the OID. `"(cn:dn:=My App)"` matches entries with `cn:My App` and DN component `cn=My App`.
`!`	NOT operator, to find entries that do not match the specified filter component. Take care to limit your search when using `!` to avoid matching so many entries that the server treats your search as unindexed.	`'!(objectclass=person)'` matches non-person entries.
`&`	AND operator, to find entries that match all specified filter components.	`'(&(l=San Francisco)(!(uid=bjensen)))'` matches entries for users in San Francisco other than the user with ID `bjensen`.
`|`	OR operator, to find entries that match one of the specified filter components.	`"|(sn=Jensen)(sn=Johnson)"` matches entries with surname Jensen or surname Johnson.

JSON query filters

DS servers support attribute values that have JSON syntax. This makes it possible to index JSON values, and to search for them using Common REST query filters, as described in DS REST APIs.

The following examples depend on settings applied with the `ds-evaluation` setup profile.

The first example uses a custom JSON query index for an `oauth2Token` JSON attribute. The index lets you search with Common REST query filters. The search finds the entry with `"access_token": "123"`:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(oauth2Token=access_token eq '123')" \
 oauth2Token

dn: uid=bjensen,ou=People,dc=example,dc=com
oauth2Token: {"access_token":"123","expires_in":59,"token_type":"Bearer","refresh_token":"456"}
```

You can combine Common REST query filter syntax filters with other LDAP search filter to form complex filters, as demonstrated in Complex LDAP filter. For example, `(&(oauth2Token=access_token eq '123')(mail=bjensen@example.com))`.

The next example relies on a default JSON query index for equality, part of the `ds-evaluation` setup profile. The index applies to a `json` attribute that holds arbitrary JSON objects. The search finds an entry with a `json` attribute that has an "array" field containing an array of objects:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(json=array[x eq 1 and y eq 2])" \
 json

dn: uid=abarnes,ou=People,dc=example,dc=com
json: {"array":[{"x":1,"y":2},{"x":3,"y":4}]}
```

Notice the value of the `json` attribute: `{"array":[{"x":1,"y":2},{"x":3,"y":4}]}`:

- The filter `"(json=array[x eq 1 and y eq 2])"` matches because it matches the first object of the array.
- The filter `"(array[x eq 1] and array[y eq 4])"` matches because it matches both objects in the array.
- The filter `"(json=array[x eq 1 and y eq 4])"` fails to match, because the array has no object `{"x":1,"y":4}`.

JSON assertions

In addition to searches with query filters, JSON attributes can be matched with filters using JSON in the assertion. This example demonstrates a case where JSON objects are considered equal if their "id" fields match. This example depends on settings applied with the `ds-evaluation` setup profile.

Search for entries with a `jsonToken` attribute:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 '(jsonToken={"id":"HgAaB6xDhLom4JbM"})' \
 jsonToken

jsonToken: {"id":"HgAaB6xDhLom4JbM","scopes":["read","write"],"expires":"2018-01-10T10:08:34Z"}
```

Server-side sort

If permitted by the directory administrator, you can request that the server sort the search results. When your application requests a server-side sort, the server retrieves the entries matching your search, and then returns the whole set of entries in sorted order.

The following example grants access to use the server-side sort control:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --bindPassword password << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetcontrol = "ServerSideSort")
 (version 3.0;acl "Allow Server-Side Sort for Kirsten Vaughan";
 allow (read)(userdn = "ldap:///uid=kvaughan,ou=People,dc=example,dc=com");)
EOF
```

This process consumes memory resources on the server, so the best practice is to sort results on the client side, or to browse results with a search that matches a virtual list view index, as demonstrated in Virtual list view index.

DS supports the following sort key forms. The `ldapsearch` command `--sortOrder` option takes these forms as arguments:

`[+|-]attr`

Use this form with standard LDAP attributes.

The optional plus or minus sign defines the order, and *attr* is the name of the LDAP attribute to sort on.

For example, `cn` and `+cn` sort by common name in ascending order. `-sn` sorts by surname in descending order.

The following example sorts the results in ascending order by surname using `--sortOrder +sn`:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDn dc=example,dc=com \
 --sortOrder +sn \
 "(&(sn=*)(cn=babs*))" \
 cn

dn: uid=user.94643,ou=People,dc=example,dc=com
cn: Babs Bautista

dn: uid=user.81225,ou=People,dc=example,dc=com
cn: Babs Bawek

dn: uid=user.67807,ou=People,dc=example,dc=com
cn: Babs Baxter

dn: uid=user.54389,ou=People,dc=example,dc=com
cn: Babs Bayer

dn: uid=user.40971,ou=People,dc=example,dc=com
cn: Babs Bayerkohler

dn: uid=user.27553,ou=People,dc=example,dc=com
cn: Babs Bayless

dn: uid=user.14135,ou=People,dc=example,dc=com
cn: Babs Bayley

dn: uid=user.717,ou=People,dc=example,dc=com
cn: Babs Bayly

dn: uid=bjensen,ou=People,dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen

dn: uid=user.89830,ou=People,dc=example,dc=com
cn: Babs Pdesupport

dn: uid=user.76412,ou=People,dc=example,dc=com
cn: Babs Peacemaker

dn: uid=user.62994,ou=People,dc=example,dc=com
cn: Babs Peacocke

dn: uid=user.49576,ou=People,dc=example,dc=com
cn: Babs Peake

dn: uid=user.36158,ou=People,dc=example,dc=com
cn: Babs Pearce
```

```
dn: uid=user.22740,ou=People,dc=example,dc=com
cn: Babs Pearcy

dn: uid=user.9322,ou=People,dc=example,dc=com
cn: Babs Pearse
```

[+|-]jsonAttr:customJsonOrderingMatchingRule

Use this form to sort on predefined fields in LDAP attributes whose values are JSON objects.

Here, *jsonAttr* is the attribute name of the JSON attribute, and *customJsonOrderingMatchingRule* is one defined in the LDAP schema and backed by a custom schema provider. For details, see Schema and JSON.

The following example sorts the results in ascending order by the "id" field of the `jsonToken` attribute. The custom matching rule, `caseIgnoreJsonTokenIDMatch`, is defined by the `ds-evaluation` setup profile:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDn dc=example,dc=com \
 --sortOrder +jsonToken:caseIgnoreJsonTokenIDMatch \
 "(objectClass=jsonTokenObject)" \
 jsonToken

dn: uid=mjablons,ou=People,dc=example,dc=com
jsonToken: {"id":"HgAaB6xDhLom4JbM","scopes":["read","write"],"expires":"2018-01-10T10:08:34Z"}

dn: uid=awhite,ou=People,dc=example,dc=com
jsonToken: {"id":"HkV5KzDrOgqN4prp","scopes":["read"],"expires":"2018-01-10T11:09:12Z"}
```

[+|-]jsonAttr:extensibleJsonOrderingMatch:caseSensitive?:ignoreSpace?:/jsonPath[:/jsonPath…]

Use this form to sort on arbitrary fields in LDAP attributes whose values are JSON objects.

DS creates a matching rule on demand, if necessary. In that case, the search is unindexed.

The following example uses this sort key form to mimic the previous example that used a custom JSON ordering rule:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDn dc=example,dc=com \
 --sortOrder +jsonToken:extensibleJsonOrderingMatch:true:true:/id \
 "(objectClass=jsonTokenObject)" \
 jsonToken

dn: uid=mjablons,ou=People,dc=example,dc=com
jsonToken: {"id":"HgAaB6xDhLom4JbM","scopes":["read","write"],"expires":"2018-01-10T10:08:34Z"}

dn: uid=awhite,ou=People,dc=example,dc=com
jsonToken: {"id":"HkV5KzDrOgqN4prp","scopes":["read"],"expires":"2018-01-10T11:09:12Z"}
```

This form has these required parameters:

- Start with `extensibleJsonOrderingMatch`, or the OID `1.3.6.1.4.1.36733.2.1.4.6`.
- Set *caseSensitive?* to `true` to respect case when comparing values, `false` otherwise.
- Set *ignoreSpace?* to `true` to ignore whitespace when comparing values, `false` otherwise.
- Each */jsonPath* specifies a field inside the JSON object. Specify at least one */jsonPath*.

DN patterns

LDAP attributes such as `manager` have DN values. After adding an extensible match index for these attributes, you can use wildcards to find matches for specific RDNs in the DN, for example.

The following example demonstrates adding an index, so you can search for Torrey Rigden's (`uid=trigden`) employees, regardless of which company Torrey works for now.

The example that follows creates an extensible match index using the DN pattern matching rule, `distinguishedNamePatternMatch`, which has numeric OID `1.3.6.1.4.1.36733.2.1.4.13`. This supports searches that include wildcards.

The example also indexes `manager` for equality for search filters. The equality index is not required, but can be useful for searches the match entire DNs:

```
# Create and rebuild the new index:
$ dsconfig \
 create-backend-index \
 --backend-name dsEvaluation \
 --index-name manager \
 --set index-extensible-matching-rule:1.3.6.1.4.1.36733.2.1.4.13 \
 --set index-type:equality \
 --set index-type:extensible \
 --type generic \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
$ rebuild-index \
 --index manager \
 --baseDn dc=example,dc=com \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

# Search for Torrey Ridgden's employees:
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDn dc=example,dc=com \
 "(manager:distinguishedNamePatternMatch:=uid=trigden,**)" \
 manager
...
dn: uid=bjensen,ou=People,dc=example,dc=com
manager: uid=trigden, ou=People, dc=example,dc=com
...
```

Notice the search filter, `(manager:distinguishedNamePatternMatch:=uid=trigden,**)`. In DN pattern matching filters:

- `*` matches a single RDN component, or a single RDN component value.
- `**` matches multiple RDN components, or a single RDN component value.
- `+` is the separator for multiple attribute value assertions (AVAs) in the same RDN component, as in `sn=smith+givenName=jane,ou=people,dc=example,dc=com`, which matches `sn=*+givenName=*,ou=people,dc=example,dc=com`, for example.

For details, see distinguishedNamePatternMatch.

LDAP compare

The LDAP compare operation checks whether an attribute value you specify matches the attribute value stored on one or more directory entries.

In this example, Kirsten Vaughan uses the `ldapcompare` command to check whether the value matches the value of the `description` attribute:

```
$ ldapcompare \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 'description:Description on ou=People' \
 uid=kvaughan,ou=people,dc=example,dc=com

# Comparing type description with value Description on ou=People in entry uid=kvaughan,ou=people,dc=example,dc=com
# Compare operation returned true for entry uid=kvaughan,ou=people,dc=example,dc=com
```

LDAP updates

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

For details on the LDIF format shown in the examples that follow, see RFC 2849.

Add entries

Add users

The following example adds two new users:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: cn=Arsene Lupin,ou=Special Users,dc=example,dc=com
objectClass: person
objectClass: top
cn: Arsene Lupin
telephoneNumber: +33 1 23 45 67 89
sn: Lupin

dn: cn=Horace Velmont,ou=Special Users,dc=example,dc=com
objectClass: person
objectClass: top
cn: Horace Velmont
telephoneNumber: +33 1 12 23 34 45
sn: Velmont
EOF
```

Bulk adds

The following example adds 10,000 generated entries, using the `--numConnections` option to perform multiple add operations in parallel:

```
# Generate user entries with user IDs larger than those that exist,
# and remove container entries from the output:
$ makeldif \
 --outputLdif output.ldif \
 <(sed "s/<sequential:0>/<sequential:100000>/" /path/to/opendj/config/MakeLDIF/example.template)

$ sed '1,10d' output.ldif > /tmp/generated-users.ldif

# Bulk add the generated user entries:
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --numConnections 64 \
 /tmp/generated-users.ldif
```

When you use the `--numConnections` option, the number of connection is rounded up to the nearest power of two for performance reasons.

Modify entries

Add attributes

The following example shows you how to add a description and JPEG photo to Sam Carter's entry:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=scarter,ou=people,dc=example,dc=com
changetype: modify
add: description
description: Accounting Manager
-
add: jpegphoto
jpegphoto:<file:///tmp/picture.jpg
EOF
```

Change an attribute

The following example replaces the description on Sam Carter's entry:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=scarter,ou=people,dc=example,dc=com
changetype: modify
replace: description
description: New description
EOF
```

Delete an attribute

The following example deletes the JPEG photo on Sam Carter's entry:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=scarter,ou=people,dc=example,dc=com
changetype: modify
delete: jpegphoto
EOF
```

Delete one attribute value

The following example deletes a single CN value on Barbara Jensen's entry:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
delete: cn
cn: Barbara Jensen
EOF
```

From standard input

A double dash, `--`, signifies the end of command options. After the double dash, only trailing arguments are allowed. To indicate standard input as a trailing argument, use a bare dash, `-`, after the double dash.

Consider the following changes expressed in LDIF:

```
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: New description from standard input
```

To send these changes to the `ldapmodify` command on standard input, use either of the following equivalent constructions:

```
# With dashes:
$ cat bjensen-stdin-description.ldif | ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 -- -
```

```
# Without dashes:
$ cat bjensen-stdin-description.ldif | ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery
```

Optimistic concurrency (MVCC)

Consider an application that lets end users update user profiles through a browser. It stores user profiles as DS entries. End users can look up user profiles and modify them. The application assumes that the end users can tell the right information when they see it, and updates profiles exactly as users see them on their screens.

Suppose two users, Alice and Bob, are busy and often interrupted. Alice has Babs Jensen's new phone and room numbers. Bob has Babs's new location and description. Both assume that they have all the information that has changed. What can you do to make sure that your application applies the right changes when Alice and Bob simultaneously update Babs Jensen's profile?

DS servers have two features to help you in this situation. One of the features is the LDAP Assertion Control, described in Supported LDAP controls, used to tell the directory server to perform the modification only if an assertion you make stays true. The other feature is DS support for entity tag (ETag) attributes, making it easy to check whether the entry in the directory is the same as the entry you read.

Alice and Bob both get Babs's entry. In LDIF, the relevant attributes from the entry look like the following. The ETag is a generated value:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 telephoneNumber roomNumber l ETag

dn: uid=bjensen,ou=People,dc=example,dc=com
l: San Francisco
roomNumber: 0209
telephoneNumber: +1 408 555 1862
ETag: ETAG
```

Bob prepares his changes in your application. Bob is almost ready to submit the new location and description when Carol stops by to ask Bob a few questions.

Alice starts just after Bob, but manages to submit her changes without interruption. Now Babs's entry has a new phone number, room number, and ETag:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 telephoneNumber roomNumber l ETag

dn: uid=bjensen,ou=People,dc=example,dc=com
telephoneNumber: +47 2108 1746
roomNumber: 1389
l: San Francisco
ETag: NEW_ETAG
```

In your application, you use the ETag value with the assertion control to prevent Bob's update from succeeding. The application tries the equivalent of the following commands with Bob's updates:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --assertionFilter "(ETag=${ETAG})" << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: l
l: Grenoble
-
add: description
description: Employee of the Month
EOF

# The LDAP modify request failed: 122 (Assertion Failed)
# Additional Information:  Entry uid=bjensen,ou=People,dc=example,dc=com cannot be modified because the request
contained an LDAP assertion control and the associated filter did not match the contents of the entry
```

The application reloads Babs's entry with the new ETag value, and tries Bob's update again. This time Bob's changes do not collide with other changes. Babs's entry is successfully updated:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --assertionFilter "(ETag=${NEW_ETAG})" << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: l
l: Grenoble
-
add: description
description: Employee of the Month
EOF

# MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com
```

JSON attribute

DS servers support attribute values that have JSON syntax as demonstrated in JSON query filters.

This example depends on the configuration and sample data used in JSON query matching rule index. Unless you have installed the server with the evaluation profile, perform the commands in that example to prepare the server before trying this one.

The following example replaces the existing JSON value with a new JSON value:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=bjensen,ou=people,dc=example,dc=com
changetype: modify
add: objectClass
objectClass: jsonObject
-
add: json
json: {"stuff":["things","devices","paraphernalia"]}
EOF
```

Notice that the JSON object is replaced entirely.

When DS servers receive update requests for `Json` syntax attributes, they expect valid JSON objects. By default, `Json` syntax attribute values must comply with *The JavaScript Object Notation (JSON) Data Interchange Format*, described in RFC 7159. You can use the advanced core schema configuration option `json-validation-policy` to have the server be more lenient in what it accepts, or to disable JSON syntax checking.

The following example relaxes JSON syntax checking to allow comments, single quotes, and unquoted control characters such as newlines, in strings:

```
$ dsconfig \
 set-schema-provider-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --provider-name "Core Schema" \
 --set json-validation-policy:lenient \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Change incoming updates

Some client applications send updates including attributes with names that differ from the attribute names defined in the LDAP schema. Other client applications might try to update attributes they should not update, such as the operational attributes `creatorsName`, `createTimestamp`, `modifiersName`, and `modifyTimestamp`. Ideally, you would fix the client application behavior, but that is not always possible. You can configure the attribute cleanup plugin to filter add and modify requests, rename attributes in requests using incorrect names, and remove attributes that applications should not change.

Rename attributes

The following example renames incoming `email` attributes to `mail` attributes:

1. Configure the attribute cleanup plugin to rename the inbound attribute:

```
$ dsconfig \
 create-plugin \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --type attribute-cleanup \
 --plugin-name "Rename email to mail" \
 --set enabled:true \
 --set rename-inbound-attributes:email:mail \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Confirm that it worked as expected:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << 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
email: newuser@example.com
userPassword: chngthspwd
EOF
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=newuser)" \
  mail

dn: uid=newuser,ou=People,dc=example,dc=com
mail: newuser@example.com
```

Remove attributes

The following example prevents client applications from adding or modifying `creatorsName`, `createTimestamp`, `modifiersName`, and `modifyTimestamp` attributes.

1. Set up the attribute cleanup plugin:

```
$ dsconfig \
 create-plugin \
 --type attribute-cleanup \
 --plugin-name "Remove attrs" \
 --set enabled:true \
 --set remove-inbound-attributes:creatorsName \
 --set remove-inbound-attributes:createTimestamp \
 --set remove-inbound-attributes:modifiersName \
 --set remove-inbound-attributes:modifyTimestamp \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Confirm that it worked as expected:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=badattr,ou=People,dc=example,dc=com
uid: newuser
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: top
cn: Bad Attr
sn: Attr
ou: People
mail: badattr@example.com
userPassword: chngthspwd
creatorsName: cn=Bad Attr
createTimestamp: Never in a million years.
modifiersName: uid=admin
modifyTimestamp: 20110930164937Z
EOF

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 "(uid=badattr)" \
 creatorsName createTimestamp modifiersTimestamp modifyTimestamp

dn: uid=badattr,ou=People,dc=example,dc=com
createTimestamp: <timestamp>
creatorsName: uid=kvaughan,ou=People,dc=example,dc=com
```

Rename entries

The Relative Distinguished Name (RDN) refers to the part of an entry's DN that differentiates it from all other DNs at the same level in the directory tree. For example, `uid=bjensen` is the RDN of the entry with the DN `uid=bjensen,ou=People,dc=example,dc=com`. When you change the RDN of the entry, you rename the entry, modifying the naming attribute and DN.

In this example, Sam Carter is changing her last name to Jensen, and changing her login from `scarter` to `sjensen`. The following example shows you how to rename and change Sam Carter's entry. Notice the boolean field, `deleteoldrdn: 1`, which indicates that the previous RDN, `uid: scarter`, should be removed. Setting `deleteoldrdn: 0` instead would preserve `uid: scarter` on the entry:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=scarter,ou=people,dc=example,dc=com
changetype: modrdn
newrdn: uid=sjensen
deleteoldrdn: 1

dn: uid=sjensen,ou=people,dc=example,dc=com
changetype: modify
replace: cn
cn: Sam Jensen
-
replace: sn
sn: Jensen
-
replace: homeDirectory
homeDirectory: /home/sjensen
-
replace: mail
mail: sjensen@example.com
EOF
```

Move entries

When you rename an entry with child entries, the directory has to move all the entries underneath it.

> **Note**
>
> DS directory servers support the modify DN operation only for moving entries in the same backend, under the same base DN. Depending on the number of entries you move, this can be a resource-intensive operation.

Move a branch

The following example moves all entries at and below `ou=People,dc=example,dc=com` under `ou=Subscribers,dc=example,dc=com`. All the entries in this example are in the same backend. The line `deleteoldrdn: 1` indicates that the old RDN, `ou: People`, should be removed:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: ou=People,dc=example,dc=com
changetype: modrdn
newrdn: ou=Subscribers
deleteoldrdn: 1
newsuperior: dc=example,dc=com
EOF
```

Be aware that the move does not modify ACIs and other values that depend on `ou=People`. You must also edit any affected entries.

Move an entry

The following example moves an application entry that is under `dc=example,dc=com` under `ou=Apps,dc=example,dc=com` instead. The line `deleteoldrdn: 0` indicates that old RDN, `cn`, should be preserved:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: cn=New App,dc=example,dc=com
changetype: moddn
newrdn: cn=An App
deleteoldrdn: 0
newsuperior: ou=Apps,dc=example,dc=com
EOF
```

Delete entries

Remove a branch

> **Note**
>
> This can be a resource-intensive operation. The resources required to remove a branch depend on the number of entries to delete.

The following example shows you how to give an administrator access to use the subtree delete control, and to use the subtree delete option to remove an entry and its child entries:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --add global-aci:"(targetcontrol=\"SubtreeDelete\")\
 (version 3.0; acl \"Allow Subtree Delete\"; allow(read) \
 userdn=\"ldap:///uid=kvaughan,ou=People,dc=example,dc=com\";)" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ ldapdelete \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
--bindDN "uid=kvaughan,ou=People,dc=example,dc=com" \
--bindPassword bribery \
--deleteSubtree "ou=Special Users,dc=example,dc=com"
```

From standard input

A double dash, `--`, signifies the end of command options. After the double dash, only trailing arguments are allowed. To indicate standard input as a trailing argument, use a bare dash, `-`, after the double dash.

Consider the following list of users to delete:

```
$ cat users-to-delete.txt

uid=sfarmer,ou=People,dc=example,dc=com
uid=skellehe,ou=People,dc=example,dc=com
uid=slee,ou=People,dc=example,dc=com
uid=smason,ou=People,dc=example,dc=com
uid=speterso,ou=People,dc=example,dc=com
uid=striplet,ou=People,dc=example,dc=com
```

To send this list to the `ldapdelete` command on standard input, use either of the following equivalent constructions:

```
# With dashes:
$ cat users-to-delete.txt | ldapdelete \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 -- -
```

```
# Without dashes:
$ cat users-to-delete.txt | ldapdelete \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery
```

LDIF tools

Generate test data

The makeldif command uses templates to generate sample data with great flexibility. Default templates are located in the `opendj/config/MakeLDIF/` directory.

> **Tip**
>
> The quickest way to generate user entries is to use the `ds-evaluation` setup profile. The profile lets you generate an arbitrary number of Example.com users as part of the setup process.
> For details, see Install DS for evaluation.

1. Write a template file for your generated LDIF.

   The `example.template` file used in the examples creates `inetOrgPerson` entries. To learn how to generate test data that matches your production data more closely, read makeldif-template.

2. Create additional data files for your template.

   Additional data files are located in the same directory as your template file.

3. Decide whether to generate the same test data each time you use the same template.

   If so, provide the same `randomSeed` integer each time you run the command.

4. Run the `makeldif` command to generate your LDIF file.

   The following command demonstrates use of the example MakeLDIF template:

   ```
   $ makeldif \
    --outputLdif example.ldif \
    --randomSeed 42 \
    /path/to/opendj/config/MakeLDIF/example.template

   LDIF processing complete.
   ```

Search LDIF

The `ldifsearch` command searches for entries in LDIF files:

```
$ ldifsearch \
 --baseDN dc=example,dc=com \
 example.ldif \
 "(sn=Grenier)" \
 uid

dn: uid=user.4630,ou=People,dc=example,dc=com
uid: user.4630
```

Update LDIF

The `ldifmodify` command applies changes, generating a new version of the LDIF.

In the example that follows, the `changes.ldif` file contains the following LDIF:

```
dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: New description.
-
replace: initials
initials: ZZZ
```

The resulting target LDIF file is approximately the same size as the source LDIF file, but the order of entries in the file is not guaranteed to be identical:

```
$ ldifmodify \
 --outputLdif new.ldif \
 example.ldif \
 changes.ldif
```

Compare LDIF

The `ldifdiff` command reports differences between two LDIF files in LDIF format:

```
$ ldifdiff example.ldif new.ldif

dn: uid=user.0,ou=People,dc=example,dc=com
changetype: modify
delete: description
description: This is the description for Aaccf Amar.
-
add: description
description: New description.
-
delete: initials
initials: AAA
-
add: initials
initials: ZZZ
-
```

The `ldifdiff` command reads files into memory to compare their contents. The command is designed to work with small files and fragments, and can quickly run out of memory when calculating the differences between large files.

Use standard input

For each LDIF tool, a double dash, `--`, signifies the end of command options. After the double dash, only trailing arguments are allowed.

To indicate standard input as a trailing argument, use a bare dash, `-`, after the double dash. How bare dashes are used after a double dash depends on the tool:

`ldifdiff`

The bare dash can replace either the source LDIF file, or the target LDIF file argument.

To take the source LDIF from standard input, use the following construction:

```
ldifdiff [options] -- - target.ldif
```

To take the target LDIF from standard input, use the following construction:

```
ldifdiff [options] -- source.ldif -
```

`ldifmodify`

The bare dash can replace either the source.ldif or changes.ldif file arguments.

To take the source LDIF from standard input, use the following construction:

```
ldifmodify [options] -- - changes.ldif [changes.ldif ...]
```

To take the changes in LDIF from standard input, use the following construction:

```
ldifmodify [options] -- source.ldif -
```

`ldifsearch`

The bare dash lets you take the source LDIF from standard input with the following construction:

```
ldifsearch [options] -- - filter [attributes ...]
```

LDAP schema

LDAP services are based on X.500 Directory Services, which are telecommunications standards. In telecommunications, interoperability is paramount. Competitors must cooperate to the extent that they use each others' systems. For directory services, the protocols for exchanging data and the descriptions of the data are standardized. LDAP defines *schema* that describe what attributes a given LDAP entry must have and may optionally have, and what attribute values can contain and how they can be matched. Formal schema definitions protect interoperability when many applications read and write to the same directory service. Directory data are much easier to share when you understand how to use LDAP schema.

LDAP schema covers LDAP schema from the server administrator's perspective. Administrators can update LDAP directory schema. DS servers support a large number of standard schema definitions by default. Administrators can also adjust how strictly each DS server applies schema definitions. For the list of standard definitions that DS servers provide, see Standard Schema.

As a script developer, you use the available schema, and accept the server's application of schema when updating directory entries.

Read schema

Directory servers publish information about services they provide as operational attributes of the *root DSE*. The root DSE is the entry with an empty string DN, `""`. DSE is an acronym for DSA-Specific Entry. DSA is an acronym for Directory System Agent. The DSE differs by server, but is generally nearly identical for replicas.

DS servers publish the DN of the entry holding schema definitions as the value of the attribute `subschemaSubentry`:

Find LDAP schema

Look up the schema DN:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 --searchScope base \
 "(&)" \
 subschemaSubentry

dn: dc=example,dc=com
subschemaSubentry: cn=schema
```

By default, the DN for the schema entry is `cn=schema`.

The schema entry has the following attributes whose values are schema definitions:

`attributeTypes`

Attribute type definitions describe attributes of directory entries, such as `givenName` or `mail`.

Consider the following features of LDAP attributes:

- Attributes can have multiple names.

  Many common attributes take advantage of this feature. For example, `cn` and `commonName` both refer to the same attribute type. The same is true of `dc` and `domainComponent`, `l` and `localityName`, and others.

- The definition specifies the attribute's syntax and the matching rules for indexing the attribute and searching its values for matches.

  The index for a telephone number is not the same as the index for a digital certificate. By default, you must take the attribute's syntax into account when adding or updating its values.

- LDAP attributes can have multiple values by default.

  Think of the values as a set, rather than an array. LDAP does not require directory servers to order the values in any particular way, and it does not allow duplicates.

  The definition must label the attribute type as `SINGLE-VALUE` to change this, even for boolean attributes. Keep this in mind when defining your own attributes.

- Some attributes are intended for use by the directory server, rather than external applications.

  This is the case for example when you see `NO-USER-MODIFICATION` in the definition. These definitions also set `USAGE` to an operational attribute type: `directoryOperation`, `distributedOperation`, or `dSAOperation`.

`objectClasses`

Object class definitions identify the attribute types that an entry must have, and may have. Examples of object classes include `person` and `organizationalUnit`. Object classes inherit from other object classes. For example, `inetOrgPerson` inherits from `person`.

Object classes are specified as values of an entry's `objectClass` attribute.

An object class can be one of the following:

- *Structural* object classes define the core structure of the entry, generally representing a real-world object.

  By default, DS directory entries have a single structural object class or at least a single line of structural object class inheritance.

  The `person` object class is structural, for example.

- *Auxiliary* object classes define additional characteristics of entries.

  The `posixAccount` object class is auxiliary, for example.

- *Abstract* object classes define base characteristics for other object classes to inherit, and cannot themselves inherit from other object classes.

  The `top` object class from which others inherit is abstract, for example.

`ldapSyntaxes`

An *attribute syntax* constrains what directory clients can store as attribute values.

`matchingRules`

A `Matching rule` determines how the directory server compares attribute values to assertion values for LDAP search and LDAP compare operations.

For example, in a search having the filter `(uid=bjensen)` the assertion value is `bjensen`.

`nameForms`

A *name form* specifies which attribute can be used as the relative DN (RDN) for a structural object class.

`dITStructureRules`

A *DIT structure rule* defines a relationship between directory entries by identifying the name form allowed for subordinate entries of a given superior entry.

Object class schema

The schema entry in a server is large because it contains all of the schema definitions. Filter the results when reading a specific schema definition.

The example below reads the definition for the `person` object class:

```
$ grep \'person\' <(ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN "cn=schema" \
 --searchScope base \
 "(objectClass=subschema)" \
 objectClasses)

objectClasses: ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL MUST ( sn $ cn ) MAY ( userPassword $ telephoneNumber $
seeAlso $ description ) X-ORIGIN 'RFC 4519' X-SCHEMA-FILE '00-core.ldif' )
```

Notice the use of the object class name in `grep \'person\'` to filter search results.

The object class defines which attributes an entry of that object class *must* have, and which attributes the entry *may* optionally have. A `person` entry must have a `cn` and an `sn` attribute. A `person` entry may optionally have `userPassword`, `telephoneNumber`, `seeAlso`, and `description` attributes.

To determine definitions of those attributes, read the LDAP schema:

Attribute schema

The following example shows you how to read the schema definition for the `cn` attribute:

```
$ grep \'cn\' <(ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN "cn=schema" \
 --searchScope base \
 "(objectClass=subschema)" \
 attributeTypes)

attributeTypes: ( 2.5.4.3 NAME ( 'cn' 'commonName' ) SUP name X-ORIGIN 'RFC 4519' X-SCHEMA-FILE '00-core.ldif' )
```

The `cn` attribute inherits its definition from the `name` attribute. That attribute definition indicates attribute syntax and matching rules as shown in the following example:

```
$ grep \'name\' <(ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN "cn=schema" \
 --searchScope base \
 "(objectClass=subschema)" \
 attributeTypes)

attributeTypes: ( 2.5.4.41 NAME 'name' EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'RFC 4519' X-SCHEMA-FILE '00-core.ldif' )
```

This means that the server ignores case when matching a common name value. Use the OID to read the syntax as shown in the following example:

```
$ grep 1.3.6.1.4.1.1466.115.121.1.15 <(ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --baseDN "cn=schema" \
 --searchScope base \
 "(objectClass=subschema)" \
 ldapSyntaxes)

ldapSyntaxes: ( 1.3.6.1.4.1.1466.115.121.1.15 DESC 'Directory String' X-ORIGIN 'RFC 4517' )
```

Taken together with the information for the `name` attribute, the common name attribute value is a Directory String of at most 32,768 characters. For details about syntaxes, read RFC 4517, Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules. That document describes a Directory String as one or more UTF-8 characters.

Schema errors

For the sake of interoperability and to avoid polluting directory data, scripts and applications should respect LDAP schema. In the simplest case, scripts and applications can use the schemas already defined.

DS servers do accept updates to schema definitions over LDAP while the server is running. This means that when a new application calls for attributes that are not yet defined by existing directory schemas, the directory administrator can easily add them, as described in Update LDAP schema, as long as the new definitions do not conflict with existing definitions.

General purpose applications handle many different types of data. Such applications must manage schema compliance at run time. Software development kits provide mechanisms for reading schema definitions at run time, and checking whether entry data is valid according to the schema definitions.

Many scripts do not require run time schema checking. When schema checking is not required, it is sufficient to check schema-related LDAP result codes when writing to the directory:

LDAP result code: 17 (Undefined attribute type)

The requested operation failed because it referenced an attribute that is not defined in the server schema.

LDAP result code: 18 (Inappropriate matching)

The requested operation failed because it attempted to perform an inappropriate type of matching against an attribute.

LDAP result code: 20 (Attribute or value exists)

The requested operation failed because it would have resulted in a conflict with an existing attribute or attribute value in the target entry.

For example, the request tried to add a second value to a single-valued attribute.

LDAP result code: 21 (Invalid attribute syntax)

The requested operation failed because it violated the syntax for a specified attribute.

LDAP result code: 34 (Invalid DN syntax)

The requested operation failed because it would have resulted in an entry with an invalid or malformed DN.

LDAP result code: 64 (Naming violation)

The requested operation failed because it would have violated the server's naming configuration.

For example, the request did not respect a name form definition.

LDAP result code: 65 (Object class violation)

The requested operation failed because it would have resulted in an entry that violated the server schema.

For example, the request tried to remove a required attribute, or tried to add an attribute that is not allowed.

LDAP result code: 69 (Object class mods prohibited)

> The requested operation failed because it would have modified the object classes associated with an entry in an illegal manner.

When you encounter an error, take the time to read the additional information. The additional information from a server is often sufficient to allow you to resolve the problem directly.

Object class violations, and Invalid attribute syntax show some common problems that can result from schema violations.

Object class violations

A number of schema violations show up as object class violations. The following request fails to add an `undefined` attribute:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
add: undefined
undefined: This attribute is not defined.
EOF

# The LDAP modify request failed: 65 (Object Class Violation)
# Additional Information:  Entry uid=bjensen,ou=People,dc=example,dc=com cannot be modified because the resulting
entry would have violated the server schema: Entry "uid=bjensen,ou=People,dc=example,dc=com" violates the schema
because it contains attribute "undefined" which is not allowed by any of the object classes in the entry
```

The solution is to define the `undefined` attribute, and to ensure that it is allowed by one of the object classes defined for the entry.

The following request fails to add a second structural object class:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
add: objectClass
objectClass: organizationalUnit
EOF

# The LDAP modify request failed: 65 (Object Class Violation)
# Additional Information:  Entry uid=bjensen,ou=People,dc=example,dc=com cannot be modified because the resulting
entry would have violated the server schema: Entry "uid=bjensen,ou=People,dc=example,dc=com" violates the schema
because it contains multiple conflicting structural object classes "inetOrgPerson" and "organizationalUnit". Only a
single structural object class is allowed in an entry
```

The solution in this case is to define only one structural object class for the entry. Either Babs Jensen is a person or an organizational unit, but not both.

Invalid attribute syntax

The following request fails to add an empty string as a common name attribute value:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
add: cn
cn:
EOF

# The LDAP modify request failed: 21 (Invalid Attribute Syntax)
# Additional Information:  When attempting to modify entry uid=bjensen,ou=People,dc=example,dc=com to add one or more
values for attribute cn, value "" was found to be invalid according to the associated syntax: The operation attempted
to assign a zero-length value to an attribute with the directory string syntax
```

As mentioned in Attribute schema, a Directory String has one or more UTF-8 characters.

Workarounds

Follow the suggestions in Schema errors as much as possible. In particular follow these rules of thumb:

- Test with a private DS server to resolve schema issues before going live.

- Adapt your scripts and applications to avoid violating schema definitions.
- When existing schemas are not sufficient, request schema updates to add definitions that do not conflict with any already in use.

When it is not possible to respect the schema definitions, you can sometimes work around LDAP schema constraints without changing the server configuration. The schema defines an `extensibleObject` object class. The `extensibleObject` object class is auxiliary. It effectively allows entries to hold any user attribute, even attributes that are not defined in the schema.

ExtensibleObject

The following example adds one attribute that is undefined and another that is not allowed:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
add: objectClass
objectClass: extensibleObject
-
add: undefined
undefined: This attribute is not defined in the LDAP schema.
-
add: serialNumber
serialNumber: This attribute is not allowed according to the object classes.
EOF

# MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com
```

Use of the `extensibleObject` object class can be abused and can prevent interoperability. Restrict its use to cases where no better alternative is available.

Passwords and accounts

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

The `ldappasswordmodify` command lets authorized users change their own passwords and reset other users' passwords.

Reset a password

Whenever one user changes another user's password, DS servers consider it a password reset. Often password policies specify that users must change their passwords again after a password reset.

Assume password administrator Kirsten Vaughan has the `password-reset` privilege. The following example shows Kirsten resetting Andy Hall's password:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery \
 --authzID "dn:uid=ahall,ou=people,dc=example,dc=com"

The LDAP password modify operation was successful
Generated Password:  <password>
```

If a client application performs the LDAP password modify extended operation on a connection that is bound to a user (in other words, when a user first does a bind on the connection, then requests the LDAP Password Modify extended operation), then the operation is performed as the user associated with the connection. If the user associated with the connection is not the same user whose password is being changed, then DS servers consider it a password reset.

To change, rather than reset, the password as the user while binding as an application or an administrator, use the LDAP Password Modify extended operation with an authorization ID. Alternatively, use proxied authorization, as described in Proxied authorization.

If you reset a password, and do not want it to count as a password reset, use the `manage-account` command with the `set-password-is-reset` hidden option, supported only for testing:

```
$ manage-account \
 set-password-is-reset \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --targetDN uid=ahall,ou=people,dc=example,dc=com \
 --operationValue true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

Change your password

Users can change their own passwords with the `ldappasswordmodify` command as long as they know their current password:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=ahunter,ou=people,dc=example,dc=com" \
 --bindPassword egregious \
 --newPassword chngthspwd
```

The same operation works for directory superusers, such as `uid=admin`:

```
$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --authzID dn:uid=admin \
 --currentPassword password \
 --newPassword OzNOkkfkTJDSW9Bg
```

Check password quality

The `ldappasswordmodify` and `ldapmodify` commands support password quality advice controls to get additional information about why a password update failed. When you use the request control and a password update fails, the server can send the response control with details indicating which validators rejected the new password.

You can use this as a means to test a password, and to evaluate the effectiveness of a new password policy.

> **Note**
>
> The new LDAP control has interface stability: *Evolving*.

The following commands demonstrate how the tools show the information from the response control:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetcontrol="PasswordQualityAdvice") (version 3.0; acl
  "Authenticated users can check password quality";
  allow(read) userdn="ldap:///all";)
EOF

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Minimum length policy,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
objectClass: ds-pwp-validator
objectClass: ds-pwp-length-based-validator
cn: Minimum length policy
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA512
ds-pwp-length-based-min-password-length: 8
subtreeSpecification: {base "ou=people", specificationFilter "(uid=pshelton)" }
EOF

$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=pshelton,ou=People,dc=example,dc=com \
 --bindPassword nosedive \
 --control PasswordQualityAdvice:true \
 --control NoOp \
 --newPassword passwd

The LDAP password modify operation failed: 19 (Constraint Violation)
Additional Information:  The provided new password failed the validation
checks defined in the server: The provided password is shorter than the
minimum required length of 8 characters

The new password was rejected by the password policy located in "cn=Minimum
length policy,dc=example,dc=com"

The following password quality criteria were not satisfied:
* length-based with parameters {max-password-length=0, min-password-length=8}
```

Notice that the check can be performed as a no-op.

Passwords with special characters

DS servers expect passwords to be UTF-8 encoded and base64-encoded when included in LDIF. UTF-8 characters such as `à` or `ô` must be correctly encoded:

```
$ export LANG=en_US.UTF-8

$ ldappasswordmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=wlutz,ou=People,dc=example,dc=com \
 --bindPassword bassinet \
 --newPassword pàsswôrd

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=wlutz,ou=People,dc=example,dc=com \
 --bindPassword pàsswôrd \
 --baseDN dc=example,dc=com \
 "(uid=wlutz)" \
 1.1

dn: uid=wlutz,ou=People,dc=example,dc=com
```

Check account usability

The account usability control lets a password administrator read information about whether the user can authenticate to the directory:

- The remote LDAP directory service must support the LDAP control, which has OID `1.3.6.1.4.1.42.2.27.9.5.8`.
- The password administrator must be able to use the LDAP control.

To try the account usability control:

1. Enable the password administrator to use the LDAP account usability control.

   The following example sets a global ACI for Kirsten Vaughan:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --add global-aci:"(targetcontrol=\"AccountUsability\")\
 (version 3.0; acl \"Account usability access\"; allow(read) \
 userdn=\"ldap:///uid=kvaughan,ou=People,dc=example,dc=com\";)" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Use a password policy that produces results for account usability, as in the following example:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Lockout with max age and grace logins,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
cn: Lockout with max age and grace logins
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-lockout-failure-expiration-interval: 10 m
ds-pwp-grace-login-count: 3
ds-pwp-lockout-duration: 5 m
ds-pwp-lockout-failure-count: 3
ds-pwp-max-password-age: 30 d
subtreeSpecification: { base "ou=people", specificationFilter "(uid=bjensen)" }
EOF
```

3. Use the account usability control to get information about an account:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 --control AccountUsability:true \
 "(uid=bjensen)" \
 1.1
# Account Usability Response Control
# The account is usable
# Time until password expiration:  <time>
dn: uid=bjensen,ou=People,dc=example,dc=com
```

4. Perform actions to change the account usability information on the account:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword wrong-password \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 1.1

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword wrong-password \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 1.1

$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=bjensen,ou=people,dc=example,dc=com \
 --bindPassword wrong-password \
 --baseDN dc=example,dc=com \
 "(uid=bjensen)" \
 1.1
```

5. Use the account usability control again to see what has changed:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=people,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN dc=example,dc=com \
 --control AccountUsability:true \
 "(uid=bjensen)" \
 1.1
# Account Usability Response Control
# The account is not usable
# The account is locked
# Time until the account is unlocked:  <time>
```

Proxied authorization

Proxied authorization, defined in RFC 4370, provides a mechanism for binding as a proxy, and making requests on behalf of other users. For example, an application binds with its credentials, but each request is made as a user who logs in through the application.

To use proxied authorization, the proxy user must have:

- Permission to use the LDAP Proxy Authorization Control.

  Grant access to this control using an ACI with a `targetcontrol` list that includes the Proxy Authorization Control OID `ProxiedAuthV2` (`2.16.840.1.113730.3.4.18`). The ACI must grant `allow(read)` permission to the proxy.

  This calls for an ACI with a target scope that includes the entry of the proxy user binding to the directory.

- Permission to proxy as the given authorization user.

  This calls for an ACI with a target scope that includes the entry of the authorization user. The ACI must grant `allow(proxy)` permission to the proxy.

- The privilege to use proxied authorization.

  Add `ds-privilege-name: proxied-auth` to the proxy's entry.

The following table shows whether proxied authorization allows an operation on the target.

	Bind DN no access	Bind DN has access
Proxy ID no access	No	No
Proxy ID has access	Yes	Yes

The following steps rely on the access settings available in the evaluation setup profile, described in Learn about the evaluation setup profile, to demonstrate proxied authorization for an Example.com application. In the evaluation profile, `kvaughan` is a directory administrator user with access to modify `bjensen`'s entry.

If you are using a different profile, make sure you have granted access to the bind DN user and the proxy ID user:

1. Grant access to applications to use the Proxy Authorization control, and to use proxied authorization:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetcontrol="ProxiedAuthV2")
  (version 3.0; acl "Apps can use the Proxy Authorization Control";
  allow(read) userdn="ldap:///cn=*,ou=Apps,dc=example,dc=com";)
aci: (target="ldap:///dc=example,dc=com") (targetattr ="*")
  (version 3.0; acl "Allow apps proxied auth";
  allow(proxy) (userdn = "ldap:///cn=*,ou=Apps,dc=example,dc=com");)
EOF
```

   The latter ACI allows any user whose DN matches `cn=*,ou=Apps,dc=example,dc=com` to proxy as any user under the ACI target of `dc=example,dc=com`. For example, `cn=My App,ou=Apps,dc=example,dc=com` can proxy as any Example.com user, but cannot proxy as the directory superuser `uid=admin`. The target of the ACI does not include `uid=admin`.

2. Grant My App the privilege to use proxied authorization:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=My App,ou=Apps,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: proxied-auth
EOF
```

   Other applications without this privilege cannot yet use proxied authorization.

3. Test that My App can use proxied authorization:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "cn=My App,ou=Apps,dc=example,dc=com" \
 --bindPassword password \
 --proxyAs "dn:uid=kvaughan,ou=People,dc=example,dc=com" << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Changed through proxied auth
EOF

# MODIFY operation successful for DN uid=bjensen,ou=People,dc=example,dc=com
```

Use an identity mapper if identifiers have the `u:authzid` (user ID) form rather than `dn:authzid` form. Specify the identity mapper with the global configuration setting, `proxied-authorization-identity-mapper`.

For details, see Identity mappers.

Notification of changes

Applications that need change notification can use a persistent search or read the external change log.

Use persistent search

Defined in the Internet-Draft, Persistent Search: A Simple LDAP Change Notification Mechanism, a persistent search is like a regular search that never returns. Every time a change happens in the scope of the search, the server returns an additional response:

1. Grant access to perform a persistent search, by adding an ACI to use the persistent search control.

   Persistent searches consume server resources, so servers do not allow them by default. If an application does not have access, the request fails with an unavailable critical extension error:

   ```
   The LDAP search request failed: 12 (Unavailable Critical Extension)
   Additional Information:  The request control with Object Identifier (OID) "2.16.840.1.113730.3.4.3"
   cannot be used due to insufficient access rights
   ```

   The following command grants access under `dc=example,dc=com` to `My App`:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetcontrol = "PSearch")
 (version 3.0;acl "Allow Persistent Search for My App";
 allow (read)(userdn = "ldap:///cn=My App,ou=Apps,dc=example,dc=com");)
EOF
```

2. Start the persistent search.

   The following example initiates a persistent search, where notifications are sent for all update operations, only notifications about changed entries are returned, and no additional information are returned:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN 'cn=My App,ou=Apps,dc=example,dc=com' \
 --bindPassword password \
 --baseDN dc=example,dc=com \
 --persistentSearch ps:all:true:false \
 '(&)' >> /tmp/psearch.txt &

$ export PSEARCH_PID=$!
```

   Notice the search filter, `(&)`, which is always true, meaning that it matches all entries. For details on settings for a persistent search, see the `--persistentSearch` option in ldapsearch Options.

3. Make changes that impact the persistent search results.

   To prepare to modify an entry, save the following LDIF in a file named `description.ldif`:

```
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Hello, persistent search
```

   The following commands perform a modify operation, and a delete operation:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: Hello, persistent search
EOF

$ ldapdelete \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 uid=tpierce,ou=People,dc=example,dc=com
```

The result is the following responses to the persistent search:

```
dn: uid=bjensen,ou=People,dc=example,dc=com
objectClass: person
objectClass: cos
objectClass: oauth2TokenObject
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: posixAccount
objectClass: top
classOfService: bronze
cn: Barbara Jensen
cn: Babs Jensen
description: Hello, persistent search
facsimileTelephoneNumber: +1 408 555 1992
gidNumber: 1000
givenName: Barbara
homeDirectory: /home/bjensen
l: San Francisco
mail: bjensen@example.com
manager: uid=trigden, ou=People, dc=example,dc=com
oauth2Token: {"access_token":"123","expires_in":59,"token_type":"Bearer","refresh_token":"456"}
ou: Product Development
ou: People
preferredLanguage: en, ko;q=0.8
roomNumber: 0209
sn: Jensen
telephoneNumber: +1 408 555 1862
uid: bjensen
uidNumber: 1076
userPassword: {PBKDF2-HMAC-SHA256}10:<hash>

dn: uid=tpierce,ou=People,dc=example,dc=com
objectClass: person
objectClass: cos
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: posixAccount
objectClass: top
classOfService: gold
cn: Tobias Pierce
departmentNumber: 1000
description: Description on ou=People
diskQuota: 100 GB
facsimileTelephoneNumber: +1 408 555 9332
gidNumber: 1000
givenName: Tobias
homeDirectory: /home/tpierce
l: Bristol
mail: tpierce@example.com
mailQuota: 10 GB
manager: uid=scarter, ou=People, dc=example,dc=com
ou: Accounting
ou: People
preferredLanguage: en-gb
roomNumber: 1383
sn: Pierce
street: Broad Quay House, Prince Street
telephoneNumber: +1 408 555 1531
uid: tpierce
uidNumber: 1042
userPassword: {PBKDF2-HMAC-SHA256}10:<hash>
```

If the data is replicated, the results include the entry `dc=example,dc=com`. Replication updates the `ds-sync-*` operational attributes on `dc=example,dc=com`, and those changes appear in the results because the entry is in the scope of the persistent search.

4. Terminate the persistent search.

   Interrupt the command with CTRL + C (`SIGINT`) or `SIGTERM`:

   ```
   $ kill -s SIGTERM $PSEARCH_PID
   ```

Use the external change log

You read the external change log over LDAP. When you poll the change log, you can get the list of updates that happened since your last request.

The external change log mechanism uses an LDAP control with OID `1.3.6.1.4.1.26027.1.5.4`. This control allows the client application to bookmark the last changes seen. The control returns a cookie that the application sends to the server to read the next batch of changes.

These steps show the client binding as `uid=admin` to read the change log. Other accounts require sufficient access and privileges to read the change log. For instructions, see Let a user read the changelog:

1. Send an initial search request using the LDAP control with no cookie value.

   In this example, two changes appear in the changelog:

   ```
   $ ldapsearch \
    --hostname localhost \
    --port 1636 \
    --useSsl \
    --usePkcs12TrustStore /path/to/opendj/config/keystore \
    --trustStorePassword:file /path/to/opendj/config/keystore.pin \
    --bindDN uid=admin \
    --bindPassword password \
    --baseDN cn=changelog \
    --control "ecl:false" \
    "(&)" \
    changes changeLogCookie targetDN

   dn: cn=changelog

   dn: replicationCSN=<CSN1>,dc=example,dc=com,cn=changelog
   changes:: <base64Changes1>
   targetDN: uid=bjensen,ou=People,dc=example,dc=com
   changeLogCookie: <COOKIE1>

   dn: replicationCSN=<CSN2>,dc=example,dc=com,cn=changelog
   changes:: <base64Changes2>
   targetDN: uid=bjensen,ou=People,dc=example,dc=com
   changeLogCookie: <COOKIE2>
   ```

   The changes are base64-encoded. You can decode them using the `base64` command. The following example decodes a change:

```
$ base64 decode --encodedData
cmVwbGFjZTogZGVzY3JpcHRpb24KZGVzY3JpcHRpb246IE5ldyBkZXNjcmlwdGlvbgotCnJlcGxhY2U6IG1vZGlmaWVyc05hbWUKbW9kaWZpZX
JzTmFtZTogdWlkPWJqZW5zZW4sb3U9UGVvcGxlLGRjPWV4YW1wbGUsZGM9Y29tCi0KcmVwbGFjZTogbW9kaWZ5VGltZXN0YW1wCm1vZGlmeVRp
bWVzdGFtcDogMjAxNjEwMTQxNTA5MTJaCi0K

replace: description
description: New description
-
replace: modifiersName
modifiersName: uid=bjensen,ou=People,dc=example,dc=com
-
replace: modifyTimestamp
modifyTimestamp: <timestamp>
-
```

Notice the `changeLogCookie` value, which has the form `base-dn:CSN`. Here, *CSN* is a change sequence number.

2. To start reading a particular change in the changelog, provide the cookie with the control:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --baseDN cn=changelog \
 --control "ecl:false:$COOKIE1" \
 "(&)" \
 changes changeLogCookie targetDN

dn: replicationCSN=<CSN2>,dc=example,dc=com,cn=changelog
changes:: <base64Changes2>
targetDN: uid=bjensen,ou=People,dc=example,dc=com
changeLogCookie: <COOKIE2>
```

The following command decodes the change returned:

```
$ base64 decode --encodedData
cmVwbGFjZTogZGVzY3JpcHRpb24KZGVzY3JpcHRpb246IE5ldywgaW1wcm92ZWQgZGVzY3JpcHRpb24KLQpyZXBsYWNlOiBtb2RpZmllcnNOYW
1lCm1vZGlmaWVyc05hbWU6IHVpZD1iamVuc2VuLG91PVBlb3BsZSxkYz1leGFtcGxlLGRjPWNvbQotCnJlcGxhY2U6IG1vZGlmeVRpbWVzdGFt
cAptb2RpZnlUaW1lc3RhbXA6IDIwMTYxMDE0MTUwOTE5WgotCg==

replace: description
description: New, improved description
-
replace: modifiersName
modifiersName: uid=bjensen,ou=People,dc=example,dc=com
-
replace: modifyTimestamp
modifyTimestamp: <timestamp>
-
```

3. If you lose the cookie, start over from the earliest available change by sending a request with no cookie.

Use REST/HTTP

This guide shows you how to configure and use DS REST APIs to access directory services over HTTP. The RESTful HTTP APIs return directory data as JSON resources. DS APIs are based on the ForgeRock® Common REST API framework.

The common REST API provides ForgeRock Identity Platform software common ways to access web resources and collections of resources.

Create

Create resources over HTTP.

Read

Read resources over HTTP.

Update

Update resources over HTTP.

Delete

Delete resources over HTTP.

Search

Query DS over HTTP.

Passwords/Accounts

Manage passwords and accounts.

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

DS REST APIs

DS REST APIs offer HTTP access to directory data as JSON resources. DS software maps JSON resources onto LDAP entries.

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

The examples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the examples, generate the CA certificate in PEM format from the server deployment ID and password:

```
$ dskeymgr \
 export-ca-cert \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --outputFile ca-cert.pem
```

About Common REST

Common REST is a common REST API framework. It provides ForgeRock Identity Platform software common ways to access web resources and collections of resources. Adapt the examples in this section to your resources and deployment.

> **Note**
>
> This page describes the full Common REST framework. Some platform component products do not implement all Common REST behaviors exactly as described. For details, refer to the product-specific examples and reference information.

Common REST resources

Servers generally return JSON-format resources, though resource formats can depend on the implementation.

Resources in collections can be found by their unique identifiers (IDs). IDs are exposed in the resource URIs. For example, if a server has a user collection under `/users`, then you can access a user at `/users/user-id`. The ID is also the value of the `_id` field of the resource.

Resources are versioned using revision numbers. A revision is specified in the resource's `_rev` field. Revisions make it possible to figure out whether to apply changes without resource locking and without distributed transactions.

Common REST verbs

The Common REST APIs use the following verbs, sometimes referred to collectively as `CRUDPAQ`. For details and HTTP-based examples of each, follow the links to the sections for each verb.

Create

Add a new resource.

This verb maps to HTTP PUT or HTTP POST.

For details, see Create.

Read

Retrieve a single resource.

This verb maps to HTTP GET.

For details, see Read.

Update

Replace an existing resource.

This verb maps to HTTP PUT.

For details, see Update.

Delete

Remove an existing resource.

This verb maps to HTTP DELETE.

For details, see Delete.

Patch

Modify part of an existing resource.

This verb maps to HTTP PATCH.

For details, see Patch.

Action

Perform a predefined action.

This verb maps to HTTP POST.

For details, see Action.

Query

Search a collection of resources.

This verb maps to HTTP GET.

For details, see Query.

Common REST parameters

Common REST reserved query string parameter names start with an underscore, `_`. Reserved query string parameters include, but are not limited to, the following names:

- `_action`

- `_api`
- `_crestapi`
- `_fields`
- `_mimeType`
- `_pageSize`
- `_pagedResultsCookie`
- `_pagedResultsOffset`
- `_prettyPrint`
- `_queryExpression`
- `_queryFilter`
- `_queryId`
- `_sortKeys`
- `_totalPagedResultsPolicy`

> **Note**
>
> Some parameter values are not safe for URLs, so URL-encode parameter values as necessary.

Continue reading for details about how to use each parameter.

Common REST extension points

The *action* verb is the main vehicle for extensions. For example, to create a new user with HTTP POST rather than HTTP PUT, you might use `/users?_action=create`. A server can define additional actions. For example, `/tasks/1?_action=cancel`.

A server can define *stored queries* to call by ID. For example, `/groups?_queryId=hasDeletedMembers`. Stored queries can call for additional parameters. The parameters are also passed in the query string. Which parameters are valid depends on the stored query.

Common REST headers

Accept-API-Version

Common REST APIs use the `Accept-API-Version` header to specify protocol and resource versions:

```
Accept-API-Version: protocol=version,resource=version
```

`protocol`

The version reflects changes in the Common REST protocol, such as common method parameters and headers specified by the protocol itself, or the input or response conventions it prescribes.

For example, protocol version 2.2 introduced the `_countOnly` parameter.

`resource`

The version reflects changes in the resource implementation, including JSON representation of resources, input parameters required, and incompatible behavior changes.

For example, the version changes when `errorMessage` changes to `message` in a JSON response.

Whether this header is required depends on the product and API you make the request to.

X-ForgeRock-TransactionId

Common REST APIs use the `X-ForgeRock-TransactionId` header to track related requests through ForgeRock Identity Platform.

```
X-ForgeRock-TransactionId: transactionID
```

The *transactionID* consists of a unique identifier for the transaction optionally followed by a sequence number for the individual request.

This header is optional. In self-managed deployments, you configure products to trust transaction IDs and let them propagate for audit purposes.

Common REST API documentation

Common REST APIs often depend at least in part on runtime configuration. Many Common REST endpoints therefore serve *API descriptors* at runtime. An API descriptor documents the actual API as it is configured.

Use the following query string parameters to retrieve API descriptors:

`_api`

Serves an API descriptor that complies with the OpenAPI specification.

This API descriptor represents the API accessible over HTTP. It is suitable for use with popular tools such as Swagger UI.

`_crestapi`

Serves a native Common REST API descriptor.

This API descriptor provides a compact representation that is not dependent on the transport protocol. It requires a client that understands Common REST, as it omits many Common REST defaults.

> **Note**
>
> Consider limiting access to API descriptors in production environments in order to avoid unnecessary traffic.
> To provide documentation in production environments, see To publish OpenAPI documentation instead.

To publish OpenAPI documentation

In production systems, developers expect stable, well-documented APIs. Rather than retrieving API descriptors at runtime through Common REST, prepare final versions, and publish them alongside the software in production.

Use the OpenAPI-compliant descriptors to provide API reference documentation for your developers:

1. Configure the software to produce production-ready APIs.

   In other words, configure the software as for production so that the APIs match exactly.

2. Retrieve the OpenAPI-compliant descriptor.

   The following command saves the descriptor to a file. :

   ```
   $ curl -o <filename>.json <endpoint>?_api
   ```

   > **Note**
   >
   > The *endpoint* must be a valid endpoint. For example:
   >
   > ```
   > $ curl -o myapi.json https://am.example.com:8443/am/json/realms/root/authenticate?_api
   > ```

3. If necessary, edit the descriptor.

   For example, add security definitions to describe the API protection.

4. Publish the descriptor using a tool such as Swagger UI.

Create

There are two ways to create a resource, HTTP POST or HTTP PUT.

To create a resource using POST, perform an HTTP POST with the query string parameter `_action=create`, and the JSON resource as a payload. Accept a JSON response. The server creates the identifier if not specified:

```
POST /users?_action=create HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
{ JSON resource }
```

To create a resource using PUT, perform an HTTP PUT including the case-sensitive identifier for the resource in the URL path, and the JSON resource as a payload. Use the `If-None-Match: *` header. Accept a JSON response:

```
PUT /users/some-id HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
If-None-Match: *
{ JSON resource }
```

The `_id` and content of the resource depend on the server implementation. The server is not required to use the `_id` that the client provides. The server response to the request indicates the resource location as the value of the `Location` header.

If you include the `If-None-Match` header, you must use `If-None-Match: *`. In this case, the request creates the object if it does not exist, and fails if the object does exist. If you include any value other `If-None-Match: *`, the server returns an HTTP 400 Bad Request error. For example, creating an object with `If-None-Match: revision` returns a bad request error.

If you do not include `If-None-Match: *`, the request creates the object if it does not exist, and *updates* the object if it does exist.

Parameters

`_fields=field[,field…]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

If the `field` is left blank, the server returns all default values.

`_prettyPrint=true`

Format the body of the response.

Read

To retrieve a single resource, perform an HTTP GET on the resource by its case-sensitive identifier (`_id`), and accept a JSON response:

```
GET /users/some-id HTTP/1.1
Host: example.com
Accept: application/json
```

Parameters

`_fields=field[,field…]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

If the `field` is left blank, the server returns all default values.

`_mimeType=mime-type`

Some resources have fields whose values are multi-media resources, such as a profile photo.

If the feature is enabled for the endpoint, you can read a single field that is a multi-media resource by specifying the *field* and *mime-type*.

In this case, the content type of the field value returned matches the *mime-type* that you specify, and the body of the response is the multi-media resource.

Do not use the `Accept` header in this case. For example, `Accept: image/png` does not work. Use the `_mimeType` query string parameter instead.

`_prettyPrint=true`

Format the body of the response.

Update

To update a resource, perform an HTTP PUT including the case-sensitive identifier (`_id`) as the final element of the path to the resource, and the JSON resource as the payload. Use the `If-Match: _rev` header to check that you are actually updating the version you modified. Use `If-Match: *` if the version does not matter. Accept a JSON response:

```
PUT /users/some-id HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
If-Match: _rev
{ JSON resource }
```

When updating a resource, include all the attributes to retain. Omitting an attribute in the resource amounts to deleting the attribute unless it is not under the control of your application. Attributes not under the control of your application include private and read-only attributes. In addition, virtual attributes and relationship references might not be under the control of your application.

> **Note**
>
> Product-specific implementations may differ. Not all products use the payload to replace the state of the resource in its entirety. For example, attributes that are omitted from the request payload to AM will not be deleted. Instead, you need to specify the attribute and set the value to an empty array to delete the attribute from the resource.
> For more information, see the product-specific examples and reference information.

Parameters

`_fields=field[,field…]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}` , `parent/child` refers to the `"child":"value"` .

If the `field` is left blank, the server returns all default values.

`_prettyPrint=true`

Format the body of the response.

Delete

To delete a single resource, perform an HTTP DELETE by its case-sensitive identifier (`\_id`) and accept a JSON response:

```
DELETE /users/some-id HTTP/1.1
Host: example.com
Accept: application/json
```

Parameters

`_fields=field[,field…]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

If the `field` is left blank, the server returns all default values.

`_prettyPrint=true`

Format the body of the response.

Patch

To patch a resource, send an HTTP PATCH request with the following parameters:

- `operation`
- `field`
- `value`
- `from` (optional with copy and move operations)

You can include these parameters in the payload for a PATCH request, or in a JSON PATCH file. If successful, you'll see a JSON response similar to the following:

```
PATCH /users/some-id HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
If-Match: _rev
{ JSON array of patch operations }
```

PATCH operations apply to three types of targets:

- **single-valued**, such as an object, string, boolean, or number.
- **list semantics array**, where the elements are ordered, and duplicates are allowed.
- **set semantics array**, where the elements are not ordered, and duplicates are not allowed.

Common REST PATCH supports multiple `operations`:

Patch operation: add

The `add` operation ensures that the target field contains the value provided, creating parent fields as necessary.

If the target field is single-valued, then the value you include in the PATCH replaces the value of the target. A single-valued field is an `object`, `string`, `boolean`, or `number`.

An `add` operation has different results on two standard types of arrays:

- **List semantic arrays**: you can run any of these `add` operations on that type of array:
  - If you `add` an array of values, the PATCH operation appends it to the existing list of values.
  - If you `add` a single value, specify an ordinal element in the target array, or use the `{-}` special index to add that value to the end of the list.
- **Set semantic arrays**: The value included in the patch is merged with the existing set of values. Any duplicates within the array are removed.

As an example, start with the following list semantic array resource:

```
{
    "fruits" : [ "orange", "apple" ]
}
```

The following add operation includes the pineapple to the end of the list of fruits, as indicated by the `-` at the end of the `fruits` array.

```
{
    "operation" : "add",
    "field" : "/fruits/-",
    "value" : "pineapple"
}
```

The following is the resulting resource:

```
{
    "fruits" : [ "orange", "apple", "pineapple" ]
}
```

You can add only one array element one at a time, as per the corresponding JSON Patch specification. If you add an array of elements, for example:

```
{
    "operation" : "add",
    "field" : "/fruits/-",
    "value" : ["pineapple", "mango"]
}
```

The resulting resource would have the following invalid JSON structure:

```
{
    "fruits" : [ "orange", "apple", ["pineapple", "mango"]]
}
```

Patch operation: copy

The copy operation takes one or more existing values from the source field. It then adds those same values on the target field. Once the values are known, it is equivalent to performing an `add` operation on the target field.

The following `copy` operation takes the value from a field named `mail`, and then runs a `replace` operation on the target field, `another_mail`.

```
[
  {
     "operation":"copy",
     "from":"mail",
     "field":"another_mail"
  }
]
```

If the source and target field values are arrays, the result depends on whether the array has list semantics or set semantics, as described in Patch operation: add.

Patch operation: increment

The `increment` operation changes the value or values of the target field by the amount you specify. The value that you include must be one number, and may be positive or negative. The value of the target field must accept numbers. The following `increment` operation adds `1000` to the target value of `/user/payment`.

```
[
  {
    "operation" : "increment",
    "field" : "/user/payment",
    "value" : "1000"
  }
]
```

Since the `value` of the `increment` is a single number, arrays do not apply.

Patch operation: move

The move operation removes existing values on the source field. It then adds those same values on the target field. This is equivalent to a `remove` operation on the source, followed by an `add` operation with the same values, on the target.

The following `move` operation is equivalent to a `remove` operation on the source field, `surname`, followed by a `replace` operation on the target field value, `lastName`. If the target field does not exist, it is created:

```
[
  {
     "operation":"move",
     "from":"surname",
     "field":"lastName"
  }
]
```

To apply a `move` operation on an array, you need a compatible single-value, list semantic array, or set semantic array on both the source and the target. For details, see the criteria described in Patch operation: add.

Patch operation: remove

The `remove` operation ensures that the target field no longer contains the value provided. If the remove operation does not include a value, the operation removes the field. The following `remove` deletes the value of the `phoneNumber`, along with the field.

```
[
  {
    "operation" : "remove",
    "field" : "phoneNumber"
  }
]
```

If the object has more than one `phoneNumber`, those values are stored as an array.

A `remove` operation has different results on two standard types of arrays:

- **List semantic arrays**: A `remove` operation deletes the specified element in the array. For example, the following operation removes the first phone number, based on its array index (zero-based):

  ```
  [
     {
        "operation" : "remove",
        "field" : "/phoneNumber/0"
     }
  ]
  ```

- **Set semantic arrays**: The list of values included in a patch are removed from the existing array.

Patch operation: replace

The `replace` operation removes any existing value(s) of the targeted field, and replaces them with the provided value(s). It is essentially equivalent to a `remove` followed by a `add` operation. If the arrays are used, the criteria is based on Patch operation: add. However, indexed updates are not allowed, even when the target is an array.

The following `replace` operation removes the existing `telephoneNumber` value for the user, and then adds the new value of `+1 408 555 9999`.

```
[
  {
    "operation" : "replace",
    "field" : "/telephoneNumber",
    "value" : "+1 408 555 9999"
  }
]
```

A PATCH replace operation on a list semantic array works as a PATCH remove operation. The following example demonstrates how the effect of both operations. Start with the following resource:

```
{
    "fruits" : [ "apple", "orange", "kiwi", "lime" ],
}
```

Apply the following operations on that resource:

```
[
  {
    "operation" : "remove",
    "field" : "/fruits/0",
    "value" : ""
  },
  {
    "operation" : "replace",
    "field" : "/fruits/1",
    "value" : "pineapple"
  }
]
```

The PATCH operations are applied sequentially. The `remove` operation removes the first member of that resource, based on its array index, (`fruits/0`), with the following result:

```
[
  {
    "fruits" : [ "orange", "kiwi", "lime" ],
  }
]
```

The second PATCH operation, a `replace`, is applied on the second member (`fruits/1`) of the intermediate resource, with the following result:

```
[
  {
    "fruits" : [ "orange", "pineapple", "lime" ],
  }
]
```

Patch operation: transform

The `transform` operation changes the value of a field based on a script, or some other data transformation command. The following `transform` operation takes the value from the field named `/objects`, and applies the `something.js` script as shown:

```
[
  {
    "operation" : "transform",
    "field" : "/objects",
    "value" : {
      "script" : {
        "type" : "text/javascript",
        "file" : "something.js"
      }
    }
  }
]
```

Patch operation limitations

Some HTTP client libraries do not support the HTTP PATCH operation. Make sure that the library you use supports HTTP PATCH before using this REST operation.

For example, the Java Development Kit HTTP client does not support PATCH as a valid HTTP method. Instead, the method `HttpURLConnection.setRequestMethod("PATCH")` throws `ProtocolException`.

Parameters

`_fields=field[,field…]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

If the `field` is left blank, the server returns all default values.

`_prettyPrint=true`

Format the body of the response.

Action

Actions are a means of extending Common REST APIs and are defined by the resource provider, so the actions you can use depend on the implementation.

The standard action indicated by `_action=create` is described in Create.

Parameters

In addition to these parameters, specific action implementations have their own parameters:

`_fields=field[,field…]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

If the `field` is left blank, the server returns all default values.

`_prettyPrint=true`

Format the body of the response.

Query

To query a resource collection (or resource container), perform an HTTP GET, and accept a JSON response, including either a `_queryExpression`, `_queryFilter`, or `_queryId` parameter. The parameters cannot be used together:

```
GET /users?_queryFilter=true HTTP/1.1
Host: example.com
Accept: application/json
```

The server returns the result as a JSON object including a `"results"` array, and other fields that depend on the parameters.

Parameters

`_countOnly=true`

Return a count of query results without returning the resources.

This parameter requires protocol version 2.2 or later.

`_fields=field[,field…]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

If the `field` is left blank, the server returns all default values.

`_queryFilter=filter-expression`

Query filters request that the server return entries that match the filter expression. You must URL-escape the filter expression.

The string representation is summarized as follows. Continue reading for additional explanation:

```
Expr           = OrExpr
OrExpr         = AndExpr ( 'or' AndExpr ) *
AndExpr        = NotExpr ( 'and' NotExpr ) *
NotExpr        = '!' PrimaryExpr | PrimaryExpr
PrimaryExpr    = '(' Expr ')' | ComparisonExpr | PresenceExpr | LiteralExpr
ComparisonExpr = Pointer OpName JsonValue
PresenceExpr   = Pointer 'pr'
LiteralExpr    = 'true' | 'false'
Pointer        = JSON pointer
OpName         = 'eq' |  # equal to
                 'co' |  # contains
                 'sw' |  # starts with
                 'lt' |  # less than
                 'le' |  # less than or equal to
                 'gt' |  # greater than
                 'ge' |  # greater than or equal to
                 STRING  # extended operator
JsonValue      = NUMBER | BOOLEAN | '"' UTF8STRING '"'
STRING         = ASCII string not containing white-space
UTF8STRING     = UTF-8 string possibly containing white-space
```

JsonValue components of filter expressions follow RFC 7159: The JavaScript Object Notation (JSON) Data Interchange Format. In particular, as described in section 7 of the RFC, the escape character in strings is the backslash character. For example, to match the identifier `test\`, use `_id eq 'test\\'`. In the JSON resource, the `\` is escaped the same way: `"_id":"test\\"`.

When using a query filter in a URL, the filter expression is part of a query string parameter. A query string parameter must be URL encoded, as described in RFC 3986: Uniform Resource Identifier (URI): Generic Syntax. For example, white space, double quotes (`"`), parentheses, and exclamation characters must be URL encoded in HTTP query strings. The following rules apply to URL query components:

```
query       = *( pchar / "/" / "?" )
pchar       = unreserved / pct-encoded / sub-delims / ":" / "@"
unreserved  = ALPHA / DIGIT / "-" / "." / "_" / "~"
pct-encoded = "%" HEXDIG HEXDIG
sub-delims  = "!" / "$" / "&amp;" / "'" / "(" / ")"
                  / "*" / "+" / "," / ";" / "="
```

`ALPHA`, `DIGIT`, and `HEXDIG` are core rules of RFC 5234: Augmented BNF for Syntax Specifications:

```
ALPHA       =  %x41-5A / %x61-7A   ; A-Z / a-z
DIGIT       =  %x30-39             ; 0-9
HEXDIG      =  DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
```

As a result, a backslash escape character in a *JsonValue* component is percent-encoded in the URL query string parameter as `%5C`. To encode the query filter expression `_id eq 'test\\'`, use `_id+eq+'test%5C%5C'`, for example.

A simple filter expression can represent a comparison, presence, or a literal value.

For comparison expressions, use `json-pointer comparator json-value`, where the *comparator* is one of the following:

```
eq (equals)
co (contains)
sw (starts with)
lt (less than)
le (less than or equal to)
gt (greater than)
ge (greater than or equal to)
```

For presence, use `json-pointer pr` to match resources where the JSON pointer is present, and the value it points to is not `null`.

Literal values include `true` (match anything) and `false` (match nothing).

Complex expressions employ `and`, `or`, and `!` (not), with parentheses, `(expression)`, to group expressions.

`_queryId=identifier`

Specify a query by its identifier.

Specific queries can take their own query string parameter arguments, which depend on the implementation.

`_pagedResultsCookie=string`

The string is an opaque cookie used by the server to keep track of the position in the search results. The server returns the cookie in the JSON response as the value of `pagedResultsCookie`.

In the request `_pageSize` must also be set and non-zero. You receive the cookie value from the provider on the first request, and then supply the cookie value in subsequent requests until the server returns a `null` cookie, meaning the final page of results has been returned.

The `_pagedResultsCookie` parameter is supported when used with the `_queryFilter` parameter. The `_pagedResultsCookie` parameter is not guaranteed to work with the `_queryExpression` or `_queryId` parameters.

The `_pagedResultsCookie` and `_pagedResultsOffset` parameters are mutually exclusive, and not to be used together.

`_pagedResultsOffset=integer`

When `_pageSize` is non-zero, use this as an index in the result set indicating the first page to return.

The `_pagedResultsCookie` and `_pagedResultsOffset` parameters are mutually exclusive, and not to be used together.

`_pageSize=integer`

Return query results in pages of this size. After the initial request, use `_pagedResultsCookie` or `_pageResultsOffset` to page through the results.

`_prettyPrint=true`

Format the body of the response.

`_totalPagedResultsPolicy=string`

When a `_pageSize` is specified, and non-zero, the server calculates the `"totalPagedResults"`, in accordance with the `totalPagedResultsPolicy`, and provides the value as part of the response.

The `"totalPagedResults"` is either an estimate of the total number of paged results (`_totalPagedResultsPolicy=ESTIMATE`), or the exact total result count (`_totalPagedResultsPolicy=EXACT`). If no count policy is specified in the query, or if `_totalPagedResultsPolicy=NONE`, result counting is disabled, and the server returns value of -1 for `"totalPagedResults"`.

`_sortKeys=(|-)__field__[,(|-)field…]`

Sort the resources returned based on the specified field(s), either in `+` (ascending, default) order, or in `-` (descending) order.

Because ascending order is the default, including the `` `` character in the query is unnecessary. If you do include the `` `` character, it must be URL-encoded as `%2B`, for example:

```
http://localhost:8080/api/users?_queryFilter=true&_sortKeys=%2Bname/givenName
```

The `_sortKeys` parameter is not supported for predefined queries (`_queryId`).

HTTP status codes

When working with a Common REST API over HTTP, client applications should expect at least these HTTP status codes. Not all servers necessarily return all status codes identified here:

200 OK

The request was successful and a resource returned, depending on the request.

201 Created

The request succeeded and the resource was created.

204 No Content

The action request succeeded, and there was no content to return.

304 Not Modified

The read request included an `If-None-Match` header, and the value of the header matched the revision value of the resource.

400 Bad Request

The request was malformed.

401 Unauthorized

The request requires user authentication.

403 Forbidden

Access was forbidden during an operation on a resource.

404 Not Found

The specified resource could not be found, perhaps because it does not exist.

405 Method Not Allowed

The HTTP method is not allowed for the requested resource.

406 Not Acceptable

The request contains parameters that are not acceptable, such as a resource or protocol version that is not available.

409 Conflict

The request would have resulted in a conflict with the current state of the resource.

410 Gone

The requested resource is no longer available, and will not become available again. This can happen when resources expire for example.

412 Precondition Failed

The resource's current version does not match the version provided.

415 Unsupported Media Type

The request is in a format not supported by the requested resource for the requested method.

428 Precondition Required

The resource requires a version, but no version was supplied in the request.

500 Internal Server Error

The server encountered an unexpected condition that prevented it from fulfilling the request.

501 Not Implemented

The resource does not support the functionality required to fulfill the request.

503 Service Unavailable

The requested resource was temporarily unavailable. The service may have been disabled, for example.

API versions

DS REST APIs support versioning. If DS exposes multiple versions of a REST API, then you must select the version. In your HTTP requests, set a version header to specify the resource version:

```
Accept-API-Version: resource=version
```

Here, version is the `version` in the REST to LDAP mapping file. For details, see API configuration.

If you do not set a version header, then the latest version is returned.

The default example configuration includes only one API, whose version is `1.0`. In this case, the header can be omitted. If the version were specified in the examples that follow, the appropriate header would be `Accept-API-Version: resource=1.0`.

Authentication

When you first try to read a resource without authenticating, the request results in an error:

```
$ curl --cacert ca-cert.pem https://localhost:8443/api/users/bjensen

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

HTTP status code 401 indicates that the request requires user authentication.

To disable the requirement to authenticate, set the Rest2ldap endpoint `authorization-mechanism` to map anonymous HTTP requests to LDAP requests performed by an authorized user. The following example uses Kirsten Vaughan's identity:

```
$ dsconfig \
 set-http-authorization-mechanism-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --mechanism-name "HTTP Anonymous" \
 --set enabled:true \
 --set user-dn:uid=kvaughan,ou=people,dc=example,dc=com \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-http-endpoint-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --endpoint-name "/api" \
 --set authorization-mechanism:"HTTP Anonymous" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

By default, both the Rest2ldap endpoint and also the REST to LDAP gateway allow HTTP Basic authentication and HTTP header-based authentication in the style of IDM software. The authentication mechanisms map HTTP credentials to LDAP credentials.

When you set up a directory server with the `ds-evaluation` profile, the relative distinguished name (RDN) attribute is the user ID (`uid`). For example, the DN and user ID for Babs Jensen are:

```
dn: uid=bjensen,ou=People,dc=example,dc=com
uid: bjensen
```

Given this pattern in the user entries, the default sample REST to LDAP configuration translates the HTTP user name to the LDAP user ID. The default sample finds user entries under `ou=People,dc=example,dc=com`. (The default sample mapping requires that LDAP entries be immediate subordinates of the mapping's base DN.) Babs Jensen authenticates as `bjensen` (password: `hifalutin`) over HTTP. The corresponding LDAP bind DN is `uid=bjensen,ou=People,dc=example,dc=com`:

- Example using HTTP Basic authentication:

```
$ curl \
 --user bjensen:hifalutin \
 --cacert ca-cert.pem \
 --silent \
 https://localhost:8443/api/users/bjensen?_fields=userName

{"_id":"bjensen","_rev":"<revision>","userName":"bjensen@example.com"}
```

- Example using HTTP Basic authentication with credentials in the URL (`username:password@` form):

```
$ curl \
 --cacert ca-cert.pem \
 --silent \
 https://bjensen:hifalutin@localhost:8443/api/users/bjensen?_fields=userName

{"_id":"bjensen","_rev":"<revision>","userName":"bjensen@example.com"}
```

- Example using HTTP header based authentication:

```
$ curl \
 --header "X-OpenIDM-Username: bjensen" \
 --header "X-OpenIDM-Password: hifalutin" \
 --cacert ca-cert.pem \
 --silent \
 https://localhost:8443/api/users/bjensen?_fields=userName

{"_id":"bjensen","_rev":"<revision>","userName":"bjensen@example.com"}
```

If the directory data is arranged differently, or if the user names are email addresses rather than user IDs, then you must update the configuration for authentication to work.

The REST to LDAP gateway can translate HTTP username/password authentication to LDAP PLAIN SASL authentication. The gateway falls back to proxied authorization, binding as the directory superuser by default. For details, see REST to LDAP reference.

DS query parameters

REST to LDAP supports the following additional query string parameters:

- `dryRun` (boolean) to discover how a server will react to a operation, without performing the operation.

  This parameter relies on the LDAP no-op control, OID `1.3.6.1.4.1.4203.1.10.2`.

You can use with parameter with CREST create, update, patch, and the password-related actions, `modifyPassword` and `resetPassword`.

Default: `false`

- `passwordQualityAdvice` (boolean) to get additional information for a failed password update.

  The `passwordQualityAdvice` parameter relies on the DS LDAP password quality advice control, OID `1.3.6.1.4.1.36733.2.1.5.5`.

  You can use with parameter with CREST create, update, patch, and the password-related actions, `modifyPassword` and `resetPassword`.

  Default: `false`

Create

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

Create (HTTP PUT)

To choose the identifier when creating a resource, perform an HTTP PUT request with the headers `Content-Type: application/json` and `If-None-Match: *`, and the JSON content of your resource:

```
$ curl \
 --request PUT \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --header "Content-Type: application/json" \
 --header "If-None-Match: *" \
 --data '{
  "_id": "newuser",
  "_schema":"frapi:opendj:rest2ldap:user:1.0",
  "contactInformation": {
    "telephoneNumber": "+1 408 555 1212",
    "emailAddress": "newuser@example.com"
  },
  "name": {
    "familyName": "New",
    "givenName": "User"
  },
  "displayName": ["New User"],
  "manager": {
    "_id": "kvaughan"
  }
 }' \
 --silent \
 https://localhost:8443/api/users/newuser?_prettyPrint=true

{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<datestamp>"
  },
  "userName" : "newuser@example.com",
  "displayName" : [ "New User" ],
  "name" : {
    "givenName" : "User",
    "familyName" : "New"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  },
  "manager" : {
    "_id" : "kvaughan",
    "_rev": "<revision>"
  }
}
```

Create (HTTP POST)

To let the server choose the identifier when creating a resource, perform an HTTP POST with the header `Content-Type: application/json`, and the JSON content of the resource.

Including `_action=create` in the query string is optional:

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --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": {
    "familyName": "New",
    "givenName": "User"
  },
  "displayName": ["New User"],
  "manager": {"_id": "kvaughan"}
 }' \
 --silent \
 https://localhost:8443/api/users?_prettyPrint=true

{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<datestamp>"
  },
  "userName" : "newuser@example.com",
  "displayName" : [ "New User" ],
  "name" : {
    "givenName" : "User",
    "familyName" : "New"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  },
  "manager" : {
    "_id" : "kvaughan",
    "_rev" : "<revision>"
  }
}
```

Read

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

Read a resource

Perform an HTTP GET:

```
$ curl \
 --request GET \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --silent \
 "https://localhost:8443/api/users/newuser?_prettyPrint=true"

{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<datestamp>"
  },
  "userName" : "newuser@example.com",
  "displayName" : [ "New User" ],
  "name" : {
    "givenName" : "User",
    "familyName" : "New"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  },
  "manager" : {
    "_id" : "kvaughan",
    "_rev": "<revision>"
  }
}
```

Referenced resource fields

Notice in the preceding example that the operation returns only the manager's `_id` and `_rev` fields. The fields returned depend on how the reference is configured in the REST to LDAP mapping.

To return additional fields from resources referenced with a resource path in the REST to LDAP mapping, explicitly specify the field names:

```
$ curl \
 --request GET \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --silent \
 "https://localhost:8443/api/users/newuser?_fields=/displayName,/manager/displayName&_prettyPrint=true"

{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "displayName" : [ "New User" ],
  "manager" : {
    "_id" : "kvaughan",
    "displayName" : [ "Kirsten Vaughan" ],
    "_rev" : "<revision>"
  }
}
```

To return all configured fields for the resource and the manager, use `_fields=/,/manager`. This returns all fields of the referenced manager resource that are configured for the REST to LDAP mapping.

Reverse references

When you read a manager's entry, the `"reverseReference"` field in the mapping returns the list of users reporting to the manager when the reverse reference field is explicitly requested.

The search is not indexed by default, so the directory superuser makes the request:

```
$ curl \
 --request GET \
 --cacert ca-cert.pem \
 --user admin:password \
 --silent \
 "https://localhost:8443/api/users/kvaughan?_fields=/reports/displayName&_prettyPrint=true"

{
  "_id" : "kvaughan",
  "_rev" : "<revision>",
  "reports" : [ {
    "_id" : "ashelton",
    "displayName" : [ "Alexander Shelton" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "btalbot",
    "displayName" : [ "Brad Talbot" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "dakers",
    "displayName" : [ "David Akers" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "dsmith",
    "displayName" : [ "Daniel Smith" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "eward",
    "displayName" : [ "Eric Ward" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "gjensen",
    "displayName" : [ "Gern Jensen" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "hmiller",
    "displayName" : [ "Harry Miller" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "jburrell",
    "displayName" : [ "James Burrell" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "jcampai2",
    "displayName" : [ "Jeffrey Campaigne" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "jfalena",
    "displayName" : [ "John Falena" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "jvaughan",
    "displayName" : [ "Jeff Vaughan" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "kcarter",
    "displayName" : [ "Karen Carter" ],
    "_rev" : "<revision>"
  }, {
    "_id" : "mreuter",
```

```
      "displayName" : [ "Matthew Reuter" ],
      "_rev" : "<revision>"
    }, {
      "_id" : "newuser",
      "displayName" : [ "New User" ],
      "_rev" : "<revision>"
    }, {
      "_id" : "pworrell",
      "displayName" : [ "Pete Worrell" ],
      "_rev" : "<revision>"
    }, {
      "_id" : "rbannist",
      "displayName" : [ "Richard Bannister" ],
      "_rev" : "<revision>"
    }, {
      "_id" : "rdaugherty",
      "displayName" : [ "Robert Daugherty" ],
      "_rev" : "<revision>"
    }, {
      "_id" : "rschneid",
      "displayName" : [ "Rachel Schneider" ],
      "_rev" : "<revision>"
    }, {
      "_id" : "striplet",
      "displayName" : [ "Stephen Triplett" ],
      "_rev" : "<revision>"
    }, {
      "_id" : "tclow",
      "displayName" : [ "Torrey Clow" ],
      "_rev" : "<revision>"
    }, {
      "_id" : "tmason",
      "displayName" : [ "Torrey Mason" ],
      "_rev" : "<revision>"
    }, {
      "_id" : "tschmith",
      "displayName" : [ "Tobias Schmith" ],
      "_rev" : "<revision>"
    }, {
      "_id" : "tward",
      "displayName" : [ "Tobias Ward" ],
      "_rev" : "<revision>"
    } ]
  }
```

Notice that the example explicitly requests reports with the `_fields` query parameter.

Update

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

To update a resource, replace it with an HTTP PUT that includes the case-sensitive identifier (`_id`) as the final element of the path, and a JSON payload that contains _all_ writable fields of the resource that you want to retain. Use an `If-Match` header to ensure the resource already exists. For read-only fields, either include unmodified versions, or omit them from your updated version.

To update a resource regardless of the revision, use an `If-Match: *` header:

```
$ curl \
 --request PUT \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --header "Content-Type: application/json" \
 --header "If-Match: *" \
 --data '{
   "contactInformation": {
     "telephoneNumber": "+1 408 555 4798",
     "emailAddress": "scarter@example.com"
   },
   "name": {
     "familyName": "Carter",
     "givenName": "Sam"
   },
   "userName": "scarter@example.com",
   "displayName": ["Sam Carter", "Samantha Carter"],
   "groups": [
     {
       "_id": "Accounting Managers"
     }
   ],
   "manager": {
     "_id": "trigden"
   },
  "uidNumber": 1002,
  "gidNumber": 1000,
  "homeDirectory": "/home/scarter"
 }' \
 --silent \
 https://localhost:8443/api/users/scarter?_prettyPrint=true

{
  "_id" : "scarter",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta" : {
    "lastModified" : "<datestamp>"
  },
  "userName" : "scarter@example.com",
  "displayName" : [ "Sam Carter", "Samantha Carter" ],
  "name" : {
    "givenName" : "Sam",
    "familyName" : "Carter"
  },
  "description" : "Description on ou=People",
  "manager" : {
    "_id" : "trigden",
    "_rev" : "<revision>"
  },
  "groups" : [ {
    "_id" : "Accounting Managers",
    "_rev" : "<revision>"
  } ],
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 4798",
    "emailAddress" : "scarter@example.com"
```

```
    },
    "uidNumber" : 1002,
    "gidNumber" : 1000,
    "homeDirectory" : "/home/scarter"
  }
```

To update a resource only if the resource matches a particular version, use an `If-Match: revision` header as shown in the following example:

```
$ export REVISION=$(cut -d \" -f 8 <(curl --silent \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 https://localhost:8443/api/users/scarter?_fields=_rev))

$ curl \
 --request PUT \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --header "If-Match: $REVISION" \
 --header "Content-Type: application/json" \
 --data '{
   "_id" : "scarter",
   "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
   "contactInformation": {
     "telephoneNumber": "+1 408 555 4798",
     "emailAddress": "scarter@example.com"
   },
   "name": {
     "familyName": "Carter",
     "givenName": "Sam"
   },
   "userName": "scarter@example.com",
   "displayName": ["Sam Carter", "Samantha Carter"],
  "uidNumber": 1002,
  "gidNumber": 1000,
  "homeDirectory": "/home/scarter"
 }' \
 --silent \
 https://localhost:8443/api/users/scarter?_prettyPrint=true

{
  "_id" : "scarter",
  "_rev" : "<new-revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta" : {
    "lastModified" : "<datestamp>"
  },
  "userName" : "scarter@example.com",
  "displayName" : [ "Sam Carter", "Samantha Carter" ],
  "name" : {
    "givenName" : "Sam",
    "familyName" : "Carter"
  },
  "description" : "Description on ou=People",
  "groups" : [ {
    "_id" : "Accounting Managers",
    "_rev" : "<revision>"
  } ],
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 4798",
    "emailAddress" : "scarter@example.com"
  },
  "uidNumber" : 1002,
  "gidNumber" : 1000,
  "homeDirectory" : "/home/scarter"
}
```

Delete

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

To delete a resource, perform an HTTP DELETE on the resource URL. The operation returns the resource you deleted:

```
$ curl \
 --request DELETE \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --silent \
 https://localhost:8443/api/users/newuser?_prettyPrint=true

{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<datestamp>"
  },
  "userName" : "newuser@example.com",
  "displayName" : [ "New User" ],
  "name" : {
    "givenName" : "User",
    "familyName" : "New"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  },
  "manager" : {
    "_id" : "kvaughan",
    "_rev" : "<revision>"
  }
}
```

To delete a resource only if the resource matches a particular version, use an `If-Match: revision` header:

```
$ export REVISION=$(cut -d \" -f 8 <(curl --silent \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 https://localhost:8443/api/users/newuser?_fields=_rev))

$ curl \
 --request DELETE \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --header "If-Match: $REVISION" \
 --silent \
 https://localhost:8443/api/users/newuser?_prettyPrint=true

{
  "_id" : "newuser",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:user:1.0",
  "_meta" : {
    "created" : "<datestamp>"
  },
  "userName" : "newuser@example.com",
  "displayName" : [ "New User" ],
  "name" : {
    "givenName" : "User",
    "familyName" : "New"
  },
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  },
  "manager" : {
    "_id" : "kvaughan",
    "_rev" : "<revision>"
  }
}
```

To delete a resource and all of its children, follow these high-level steps:

- Make sure that the REST to LDAP configuration does map the resources to delete to LDAP entries.

  For an example, see Nested resources.

- If you are using the gateway, this requires the default setting of true for `useSubtreeDelete` in `WEB-INF/classes/rest2ldap/rest2ldap.json`.

  > **Note**
  >
  > Only users who have access to request a tree delete can delete resources with children.

- Allow the REST user to use the subtree delete control:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --add global-aci:"(targetcontrol=\"SubtreeDelete\")\
 (version 3.0; acl \"Allow Subtree Delete\"; allow(read) \
 userdn=\"ldap:///uid=kvaughan,ou=People,dc=example,dc=com\";)" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

- Request the delete as a user who has rights to perform a subtree delete on the resource.

> **Note**
>
> This can be a resource-intensive operation. The resources required to remove a branch depend on the number of LDAP entries to delete.

Patch

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

Patching is updating part of the resource rather than replacing the resource. For example, you could change Babs Jensen's email address with an HTTP PATCH request:

```
$ curl \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 --request PATCH \
 --header "Content-Type: application/json" \
 --data '[
  {
    "operation": "replace",
    "field": "/contactInformation/emailAddress",
    "value": "babs@example.com"
  }
 ]' \
 --silent \
 https://localhost:8443/api/users/bjensen?_prettyPrint=true

{
  "_id" : "bjensen",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta" : {
    "lastModified" : "<datestamp>"
  },
  "userName" : "babs@example.com",
  "displayName" : [ "Barbara Jensen", "Babs Jensen" ],
  "name" : {
    "givenName" : "Barbara",
    "familyName" : "Jensen"
  },
  "description" : "Original description",
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1862",
    "emailAddress" : "babs@example.com"
  },
  "uidNumber" : 1076,
  "gidNumber" : 1000,
  "homeDirectory" : "/home/bjensen",
  "manager" : {
    "_id" : "trigden",
    "_rev" : "<revision>"
  },
  "groups" : [ {
    "_id" : "Carpoolers"
  } ]
}
```

Notice in the example that the data sent specifies the type of patch operation, the field to change, and a value that depends on the field you change and on the operation. A single-valued field takes an object, boolean, string, or number depending on its type, whereas a multi-valued field takes an array of values. Getting the type wrong results in an error. Notice that the patch data is itself an array. This makes it possible to patch more than one part of the resource by using a set of patch operations in the same request.

DS software supports these patch operations:

add

The add operation ensures that the target field contains the value provided, creating parent fields as necessary.

If the target field is single-valued and a value already exists, then that value is replaced with the value you provide. *Note that you do not get an error when adding a value to a single-valued field that already has a value.* A single-valued field is one whose value is not an array (an object, string, boolean, or number).

If the target field is multi-valued, then the array of values you provide is merged with the set of values already in the resource. New values are added, and duplicate values are ignored. A multi-valued field takes an array value.

`remove`

The remove operation ensures that the target field does not contain the value provided. If you do not provide a value, the entire field is removed if it already exists.

If the target field is single-valued and a value is provided, then the provided value must match the existing value to remove, otherwise the field is left unchanged.

If the target field is multi-valued, then values in the array you provide are removed from the existing set of values.

`replace`

The replace operation removes existing values on the target field, and replaces them with the values you provide. It is equivalent to performing a remove on the field, then an add with the values you provide.

`increment`

The increment operation increments or decrements the value or values in the target field by the amount you specify, which is positive to increment and negative to decrement. The target field must take a number or a set of numbers. The value you provide must be a single number.

One key nuance in how a patch works with DS software concerns multi-valued fields. Although JSON resources represent multi-valued fields as *arrays*, DS software treats those values as *sets*. In other words, values in the field are unique, and the ordering of an array of values is not meaningful in the context of patch operations. If you reference array values by index, DS software returns an error. DS software does, however, allow use of a hyphen to add an element to a set. Include the hyphen as the last element of the `field` JSON pointer path as in the `"/members/-"` field of this example patch: `[{ "operation" : "add", "field" : "/members/-", "value" : { "_id" : "bjensen" } }]`.

Perform patch operations as if arrays values were sets. The following example includes Barbara Jensen in a group by adding her to the set of members:

```
$ curl \
 --user kvaughan:bribery \
 --request PATCH \
 --cacert ca-cert.pem \
 --header "Content-Type: application/json" \
 --data '[{
     "operation": "add",
     "field": "/members",
     "value": [{"_id": "bjensen"}]
   }]' \
 --silent \
 https://localhost:8443/api/groups/Directory%20Administrators?_prettyPrint=true

{
  "_id" : "Directory Administrators",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:group:1.0",
  "_meta" : {
    "lastModified" : "<datestamp>"
  },
  "displayName" : "Directory Administrators",
  "members" : [ {
    "_id" : "kvaughan"
  }, {
    "_id" : "rdaugherty"
  }, {
    "_id" : "hmiller"
  }, {
    "_id" : "bjensen"
  } ]
}
```

The following example removes Barbara Jensen from the group:

```
$ curl \
 --user kvaughan:bribery \
 --request PATCH \
 --cacert ca-cert.pem \
 --header "Content-Type: application/json" \
 --data '[{
    "operation": "remove",
    "field": "/members",
    "value": [{"_id": "bjensen"}]
  }]' \
 --silent \
 https://localhost:8443/api/groups/Directory%20Administrators?_prettyPrint=true

{
  "_id" : "Directory Administrators",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:group:1.0",
  "_meta" : {
    "lastModified" : "<datestamp>"
  },
  "displayName" : "Directory Administrators",
  "members" : [ {
    "_id" : "kvaughan"
  }, {
    "_id" : "rdaugherty"
  }, {
    "_id" : "hmiller"
  } ]
}
```

To change the value of more than one attribute in a patch operation, include multiple operations in the body of the JSON patch, as shown in the following example:

```
$ curl \
 --user kvaughan:bribery \
 --request PATCH \
 --cacert ca-cert.pem \
 --header "Content-Type: application/json" \
 --data '[
  {
    "operation": "replace",
    "field": "/contactInformation/telephoneNumber",
    "value": "+1 408 555 9999"
  },
  {
    "operation": "add",
    "field": "/contactInformation/emailAddress",
    "value": "barbara.jensen@example.com"
  }
 ]' \
 --silent \
 https://localhost:8443/api/users/bjensen?_prettyPrint=true

{
  "_id" : "bjensen",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta" : {
    "lastModified" : "<datestamp>"
  },
  "userName" : "barbara.jensen@example.com",
  "displayName" : [ "Barbara Jensen", "Babs Jensen" ],
  "name" : {
    "givenName" : "Barbara",
    "familyName" : "Jensen"
  },
  "description" : "Original description",
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 9999",
    "emailAddress" : "barbara.jensen@example.com"
  },
  "uidNumber" : 1076,
  "gidNumber" : 1000,
  "homeDirectory" : "/home/bjensen",
  "manager" : {
    "_id" : "trigden",
    "_rev" : "<revision>"
  },
  "groups" : [ {
    "_id" : "Carpoolers"
  } ]
}
```

Notice that for a multi-valued attribute, the `value` field takes an array, whereas the `value` field takes a single value for a single-valued field. For single-valued fields, an `add` operation has the same effect as a `replace` operation.

You can use resource revision numbers in `If-Match: revision` headers to patch the resource only if the resource matches a particular version, as shown in the following example:

```
$ export REVISION=$(cut -d \" -f 8 <(curl --silent \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 https://localhost:8443/api/users/bjensen?_fields=_rev))

$ curl \
 --user kvaughan:bribery \
 --request PATCH \
 --cacert ca-cert.pem \
 --header "If-Match: $REVISION" \
 --header "Content-Type: application/json" \
 --data '[
  {
    "operation": "add",
    "field": "/contactInformation/emailAddress",
    "value": "babs@example.com"
  }
 ]' \
 --silent \
 https://localhost:8443/api/users/bjensen?_prettyPrint=true

{
  "_id" : "bjensen",
  "_rev" : "<new-revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta" : {
    "lastModified" : "<datestamp>"
  },
  "userName" : "babs@example.com",
  "displayName" : [ "Barbara Jensen", "Babs Jensen" ],
  "name" : {
    "givenName" : "Barbara",
    "familyName" : "Jensen"
  },
  "description" : "Original description",
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 9999",
    "emailAddress" : "babs@example.com"
  },
  "uidNumber" : 1076,
  "gidNumber" : 1000,
  "homeDirectory" : "/home/bjensen",
  "groups" : [ {
    "_id" : "Carpoolers"
  } ],
  "manager" : {
    "_id" : "trigden",
    "_rev" : "<revision>"
  }
}
```

The resource revision changes when the patch is successful.

Actions

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

Change your password

> **Note**
>
> This action *requires HTTPS* to avoid sending the password over an insecure connection.

Perform an HTTPS POST with the header `Content-Type: application/json`, `_action=modifyPassword` in the query string, and the old and new passwords in JSON format as the POST data.

The JSON POST DATA must include the following fields:

`oldPassword`

The value of this field is the current password as a UTF-8 string.

`newPassword`

The value of this field is the new password as a UTF-8 string.

On success, the HTTP status code is 200 OK, and the response body is an empty JSON resource:

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --header "Content-Type: application/json" \
 --data '{"oldPassword": "hifalutin", "newPassword": "chngthspwd"}' \
 --silent \
 https://localhost:8443/api/users/bjensen?_action=modifyPassword

{}
```

Check password quality

The `passwordQualityAdvice` and `dryRun` query string parameters let you get additional information for a password update that might fail. The `passwordQualityAdvice` parameter relies on the LDAP password quality advice control, OID `1.3.6.1.4.1.36733.2.1.5.5`, which users must have access to request. The `dryRun` parameter relies on the LDAP no-op control, OID `1.3.6.1.4.1.4203.1.10.2`.

You can use this as a means to test a password, and to evaluate the effectiveness of a new password policy.

> **Note**
>
> The password quality advice control and the `passwordQualityAdvice` parameter have interface stability: *Evolving*.

The following commands demonstrate how the parameters cause the server to return information. On failure, the status code is HTTP 400 Bad Request, and the response is a JSON object listing what passed validation and what failed:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Minimum length policy,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
objectClass: ds-pwp-validator
objectClass: ds-pwp-length-based-validator
cn: Minimum length policy
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA512
ds-pwp-length-based-min-password-length: 8
subtreeSpecification: {base "ou=people", specificationFilter "(uid=bjensen)" }

dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetcontrol="PasswordQualityAdvice")
  (version 3.0; acl "Authenticated users can check password quality";
  allow(read) userdn="ldap:///all";)
EOF

$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user bjensen:chngthspwd \
 --header "Content-Type: application/json" \
 --data '{"oldPassword": "chngthspwd", "newPassword": "passwd"}' \
 --silent \
 "https://localhost:8443/api/users/bjensen?_action=modifyPassword&dryRun=true&passwordQualityAdvice=true"

{
  "code" : 400,
  "reason" : "Bad Request",
  "message" : "Constraint Violation: The provided new password failed the validation checks defined in the server:
The provided password is shorter than the minimum required length of 8 characters",
  "detail" : {
    "passwordQualityAdvice" : {
      "passingCriteria" : [ ],
      "failingCriteria" : [ {
        "type" : "length-based",
        "parameters" : {
          "max-password-length" : 0,
          "min-password-length" : 8
        }
      } ],
      "attributeType" : "userPassword"
    }
  }
}
```

You can use `passwordQualityAdvice` without the `dryRun` parameter:

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user bjensen:password \
 --header "Content-Type: application/json" \
 --data '{"oldPassword": "chngthspwd", "newPassword": "hifalutin"}' \
 --silent \
 "https://localhost:8443/api/users/bjensen?_action=modifyPassword&passwordQualityAdvice=true"
```

On success, the HTTP status code is 200 OK, and the response body is an empty JSON resource.

Reset a password

Whenever one user changes another user's password, DS servers consider it a password reset. Often, password policies specify that users must change their passwords again after a password reset.

> **Note**
>
> This action *requires HTTPS* to avoid sending the password over an insecure connection.

Perform an HTTPS POST with the header `Content-Type: application/json`, `_action=resetPassword` in the query string, and an empty JSON document (`{}`) as the POST data.

The following example demonstrates an administrator changing a user's password. Before trying this example, make sure the password administrator has been given the `password-reset` privilege. Otherwise, the password administrator has insufficient access. On success, the HTTP status code is 200 OK, and the response body is a JSON resource with a `generatedPassword` containing the new password:

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --header "Content-Type: application/json" \
 --data '{}' \
 --silent \
 https://localhost:8443/api/users/bjensen?_action=resetPassword

{"generatedPassword":"<new-password>"}
```

As password administrator, provide the new, generated password to the user.

Use this feature in combination with a password policy that forces the user to change their password after a reset. For an example, see Require password change on add or reset.

Account usability action

The `accountUsability` action lets a password administrator read information about whether the user can authenticate to the directory. This mirrors the LDAP Account usability control:

- The `"supportedActions"` list in the REST to LDAP mapping for the user must include the `"accountUsability"` action.

This action is not in the `"supportedActions"` list by default.

- The remote LDAP directory service must support the LDAP control, which has OID `1.3.6.1.4.1.42.2.27.9.5.8`.
- The password administrator must be able to use the LDAP control.

Try the `accountUsability` action:

1. Edit the mapping configuration to include the `"accountUsability"` action in the list for the user resource:

```
"supportedActions": [ "accountUsability","modifyPassword", "resetPassword" ],
```

2. Enable the password administrator to use the LDAP account usability control.

   The following example sets a global ACI for Kirsten Vaughan:

```
$ dsconfig \
 set-access-control-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --add global-aci:"(targetcontrol=\"AccountUsability\")\
 (version 3.0; acl \"Account usability access\"; allow(read) \
 userdn=\"ldap:///uid=kvaughan,ou=People,dc=example,dc=com\";)" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. Use a password policy that produces results for account usability, as in the following example:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=Lockout with max age and grace logins,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: ds-pwp-password-policy
cn: Lockout with max age and grace logins
ds-pwp-password-attribute: userPassword
ds-pwp-default-password-storage-scheme: PBKDF2-HMAC-SHA256
ds-pwp-lockout-failure-expiration-interval: 10 m
ds-pwp-grace-login-count: 3
ds-pwp-lockout-duration: 5 m
ds-pwp-lockout-failure-count: 3
ds-pwp-max-password-age: 30 d
subtreeSpecification: { base "ou=people", specificationFilter "(uid=bjensen)" }
EOF
```

4. Produce some account usability information on a user account:

```
$ curl \
 --user bjensen:wrong-password \
 --cacert ca-cert.pem \
 --silent \
 https://localhost:8443/api/users/bjensen?_fields=userName

$ curl \
 --user bjensen:wrong-password \
 --cacert ca-cert.pem \
 --silent \
 https://localhost:8443/api/users/bjensen?_fields=userName

$ curl \
 --user bjensen:wrong-password \
 --cacert ca-cert.pem \
 --silent \
 https://localhost:8443/api/users/bjensen?_fields=userName
```

5. Use the action to get account usability information:

```
$ curl \
 --request POST \
 --user kvaughan:bribery \
 --header "Content-Type: application/json" \
 --data '{}' \
 --cacert ca-cert.pem \
 --silent \
 https://localhost:8443/api/users/bjensen?_action=accountUsability

{"status":"locked","unlockIn":<seconds>}
```

The JSON response can contain these fields. Only the `"status"` property is always present in the response. Other fields are optional:

```
{
  "status": "string",              // One of "disabled", "locked", "passwordExpired",
                                   // "mustChangePassword", or "valid"
  "unlockIn": integer,             // Seconds until locked account is unlocked
  "graceLoginsRemaining": integer, // Number of remaining authentications allowed with
                                   // an expired password
  "passwordExpiresIn": integer,    // Seconds until password expires
}
```

Query

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

To search, perform an HTTP GET with a `_queryFilter=expression` parameter. For details about the query filter expression, see Query.

> **Note**
>
> The `_queryId` parameter, described in Query, is not used in DS software at present.

The following table shows LDAP search filters and corresponding query filter expressions:

LDAP Filter	REST Filter
`(&)`	`_queryFilter=true`
`(uid=*)`	`_queryFilter=_id+pr`
`(uid=bjensen)`	`_queryFilter=_id+eq+'bjensen'`
`(uid=*jensen*)`	`_queryFilter=_id+co+'jensen'`
`(uid=jensen*)`	`_queryFilter=_id+sw+'jensen'`
`(&(uid=*jensen*)(cn=babs*))`	`_queryFilter=(_id+co+'jensen'and+displayName+sw'babs')`
`(|(uid=*jensen*)(cn=sam*))`	`_queryFilter=(_id+co+'jensen'or+displayName+sw'sam')`
`(!(uid=*jensen*))`	`_queryFilter=!(_id+co+'jensen')`
`(uid<=jensen)`	`_queryFilter=_id+le+'jensen'`
`(uid>=jensen)`	`_queryFilter=_id+ge+'jensen'`

For query operations, the filter expression is constructed from the following building blocks. Make sure you URL-encode the filter expressions, which are shown here without URL-encoding to make them easier to read.

In filter expressions, the simplest json-pointer is a field of the JSON resource, such as `userName` or `id`. A json-pointer can also point to nested elements, as described in the JSON Pointer Internet-Draft:

Comparison expressions

Build filters using the following comparison expressions:

`json-pointer eq json-value`

Matches when the pointer equals the value, as in the following example:

```
$ curl \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=userName+eq+'babs@example.com'&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "bjensen",
    "_rev" : "<revision>",
    "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
    "_meta" : {
      "lastModified" : "<datestamp>"
    },
    "userName" : "babs@example.com",
    "displayName" : [ "Barbara Jensen", "Babs Jensen" ],
    "name" : {
      "givenName" : "Barbara",
      "familyName" : "Jensen"
    },
    "description" : "Original description",
    "manager" : {
      "_id" : "trigden",
      "_rev" : "<revision>"
    },
    "contactInformation" : {
      "telephoneNumber" : "+1 408 555 9999",
      "emailAddress" : "babs@example.com"
    },
    "uidNumber" : 1076,
    "gidNumber" : 1000,
    "homeDirectory" : "/home/bjensen",
    "groups" : [ {
      "_id" : "Carpoolers"
    } ]
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

`json-pointer co json-value`

Matches when the pointer contains the value, as in the following example:

```
$ curl \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?
_queryFilter=userName+co+'jensen'&_fields=userName&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "ajensen",
    "_rev" : "<revision>",
    "userName" : "ajensen@example.com"
  }, {
    "_id" : "gjensen",
    "_rev" : "<revision>",
    "userName" : "gjensen@example.com"
  }, {
    "_id" : "jjensen",
    "_rev" : "<revision>",
    "userName" : "jjensen@example.com"
  }, {
    "_id" : "kjensen",
    "_rev" : "<revision>",
    "userName" : "kjensen@example.com"
  }, {
    "_id" : "rjensen",
    "_rev" : "<revision>",
    "userName" : "rjensen@example.com"
  }, {
    "_id" : "tjensen",
    "_rev" : "<revision>",
    "userName" : "tjensen@example.com"
  } ],
  "resultCount" : 6,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

`json-pointer sw json-value`

Matches when the pointer starts with the value, as in the following example:

```
$ curl \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=userName+sw+'ab'&_fields=userName&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "abarnes",
    "_rev" : "<revision>",
    "userName" : "abarnes@example.com"
  }, {
    "_id" : "abergin",
    "_rev" : "<revision>",
    "userName" : "abergin@example.com"
  } ],
  "resultCount" : 2,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

`json-pointer lt json-value`

Matches when the pointer is less than the value, as in the following example:

```
$ curl \
 --user admin:password \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=userName+lt+'ac'&_fields=userName&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "abarnes",
    "_rev" : "<revision>",
    "userName" : "abarnes@example.com"
  }, {
    "_id" : "abergin",
    "_rev" : "<revision>",
    "userName" : "abergin@example.com"
  } ],
  "resultCount" : 2,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

`json-pointer le json-value`

Matches when the pointer is less than or equal to the value, as in the following example:

```
$ curl \
 --user admin:password \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=userName+le+'ad'&_fields=userName&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "abarnes",
    "_rev" : "<revision>",
    "userName" : "abarnes@example.com"
  }, {
    "_id" : "abergin",
    "_rev" : "<revision>",
    "userName" : "abergin@example.com"
  }, {
    "_id" : "achassin",
    "_rev" : "<revision>",
    "userName" : "achassin@example.com"
  } ],
  "resultCount" : 3,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

`json-pointer gt json-value`

Matches when the pointer is greater than the value, as in the following example:

```
$ curl \
 --user admin:password \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=userName+gt+'wa'&_fields=userName&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "wlutz",
    "_rev" : "<revision>",
    "userName" : "wlutz@example.com"
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

`json-pointer ge json-value`

Matches when the pointer is greater than or equal to the value, as in the following example:

```
$ curl \
 --user admin:password \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=userName+ge+'va'&_fields=userName&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "wlutz",
    "_rev" : "<revision>",
    "userName" : "wlutz@example.com"
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

Presence expression

`json-pointer pr` matches any resource on which the json-pointer is present:

```
$ curl \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/groups?_queryFilter=displayName+pr&_fields=displayName&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "Accounting Managers",
    "_rev" : "<revision>",
    "displayName" : "Accounting Managers"
  }, {
    "_id" : "Directory Administrators",
    "_rev" : "<revision>",
    "displayName" : "Directory Administrators"
  }, {
    "_id" : "HR Managers",
    "_rev" : "<revision>",
    "displayName" : "HR Managers"
  }, {
    "_id" : "PD Managers",
    "_rev" : "<revision>",
    "displayName" : "PD Managers"
  }, {
    "_id" : "QA Managers",
    "_rev" : "<revision>",
    "displayName" : "QA Managers"
  } ],
  "resultCount" : 5,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

Literal expressions

`true` matches any resource in the collection.

`false` matches no resource in the collection.

In other words, you can list all resources in a collection as in the following example:

```
$ curl \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/groups?_queryFilter=true&_fields=_id&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "Accounting Managers",
    "_rev" : "<revision>"
  }, {
    "_id" : "Directory Administrators",
    "_rev" : "<revision>"
  }, {
    "_id" : "HR Managers",
    "_rev" : "<revision>"
  }, {
    "_id" : "PD Managers",
    "_rev" : "<revision>"
  }, {
    "_id" : "QA Managers",
    "_rev" : "<revision>"
  } ],
  "resultCount" : 5,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

Complex expressions

Combine expressions using boolean operators `and`, `or`, and `!` (not), and by using parentheses `(expression)` with group expressions. The following example queries resources with last name Jensen and manager name starting with `Sam`:

```
$ curl \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=\
(name/familyName+co+'jensen'+and+manager/displayName+sw+'Sam')&_fields=name/familyName,manager/
displayName&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "bjense2",
    "_rev" : "<revision>",
    "name" : {
      "familyName" : "Jensen"
    },
    "manager" : {
      "displayName" : [ "Sam Carter", "Samantha Carter" ],
      "_id" : "scarter",
      "_rev" : "<revision>"
    }
  }, {
    "_id" : "jjensen",
    "_rev" : "<revision>",
    "name" : {
      "familyName" : "Jensen"
    },
    "manager" : {
      "displayName" : [ "Sam Carter", "Samantha Carter" ],
      "_id" : "scarter",
      "_rev" : "<revision>"
    }
  }, {
    "_id" : "tjensen",
    "_rev" : "<revision>",
    "name" : {
      "familyName" : "Jensen"
    },
    "manager" : {
      "displayName" : [ "Sam Carter", "Samantha Carter" ],
      "_id" : "scarter",
      "_rev" : "<revision>"
    }
  } ],
  "resultCount" : 3,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

Notice the filters use the JSON pointers `name/familyName` and `manager/displayName` to identify the fields nested inside the `name` and `manager` objects.

Graph-like queries

The default REST to LDAP sample mapping defines references with `resourcePath` fields. When the REST to LDAP mapping defines references to other resources in this way, the mapping supports recursion. Client applications can use query filters that traverse a "graph" of resources.

The following example gets users whose manager belongs to the `Directory Administrators` group. The search is not indexed by default, so the directory superuser makes the request:

```
$ curl \
 --user admin:password \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users/?_queryFilter=/manager/groups/
_id+eq+'Directory%20Administrators'&_fields=_id&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "ashelton",
    "_rev" : "<revision>"
  }, {
    "_id" : "btalbot",
    "_rev" : "<revision>"
  }, {
    "_id" : "dakers",
    "_rev" : "<revision>"
  }, {
    "_id" : "dsmith",
    "_rev" : "<revision>"
  }, {
    "_id" : "eward",
    "_rev" : "<revision>"
  }, {
    "_id" : "gjensen",
    "_rev" : "<revision>"
  }, {
    "_id" : "hmiller",
    "_rev" : "<revision>"
  }, {
    "_id" : "jburrell",
    "_rev" : "<revision>"
  }, {
    "_id" : "jcampai2",
    "_rev" : "<revision>"
  }, {
    "_id" : "jfalena",
    "_rev" : "<revision>"
  }, {
    "_id" : "jvaughan",
    "_rev" : "<revision>"
  }, {
    "_id" : "kcarter",
    "_rev" : "<revision>"
  }, {
    "_id" : "mreuter",
    "_rev" : "<revision>"
  }, {
    "_id" : "newuser",
    "_rev" : "<revision>"
  }, {
    "_id" : "pworrell",
    "_rev" : "<revision>"
  }, {
    "_id" : "rbannist",
    "_rev" : "<revision>"
  }, {
    "_id" : "rdaugherty",
    "_rev" : "<revision>"
  }, {
```

```
      "_id" : "rschneid",
      "_rev" : "<revision>"
    }, {
      "_id" : "striplet",
      "_rev" : "<revision>"
    }, {
      "_id" : "tclow",
      "_rev" : "<revision>"
    }, {
      "_id" : "tmason",
      "_rev" : "<revision>"
    }, {
      "_id" : "tschmith",
      "_rev" : "<revision>"
    }, {
      "_id" : "tward",
      "_rev" : "<revision>"
    } ],
    "resultCount" : 23,
    "pagedResultsCookie" : null,
    "totalPagedResultsPolicy" : "NONE",
    "totalPagedResults" : -1,
    "remainingPagedResults" : -1
  }
```

REST to LDAP translates such graph-like HTTP queries into a series of corresponding LDAP requests. Complex graph-like queries can have a significant server-side performance impact.

Complex JSON queries

The default JSON query index works for JSON attributes that hold arbitrary JSON objects. This includes JSON with nested objects, such as `{"array":[{"x":1,"y":2},{"x":3,"y":4}]}`.

To try this feature:

1. Install a DS server with the `ds-evaluation` setup profile.
2. Replace the `"frapi:opendj:rest2ldap:posixUser:1.0"` resource type definition with a resource type for a user with a `json` attribute in the mapping configuration file, `/path/to/opendj/config/rest2ldap/endpoints/api/example-v1.json`:

   ```
   // A user with a "json" attribute.
   "frapi:opendj:rest2ldap:jsonUser:1.0": {
       "superType": "frapi:opendj:rest2ldap:user:1.0",
       "objectClasses": [ "jsonObject" ],
       "properties": {
           "json": {
               "type": "json",
               "isMultiValued": true
           }
       }
   },
   ```

   Alternatively, download jsonUser.json to replace the existing file.

3. Restart DS to load the new API configuration.
4. Perform searches using the configuration.

   This search finds an entry with a `json` attribute that has an "array" field containing an array of objects:

```
$ curl \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 --silent \
 --get \
 --data-urlencode "_queryFilter=json/array[x eq 1] and json/array[y eq 4]" \
 --data-urlencode "_fields=json" \
 --data-urlencode "_prettyPrint=true" \
 "https://localhost:8443/api/users/"

{
  "result" : [ {
    "_id" : "abarnes",
    "_rev" : "<revision>",
    "json" : [ {
      "array" : [ {
        "x" : 1,
        "y" : 2
      }, {
        "x" : 3,
        "y" : 4
      } ]
    } ]
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

   - The filter `json/array[x eq 1] and json/array[y eq 4]` matches because it matches both objects in the array.
   - The filter `json/array[x eq 1 and y eq 2]` matches because it matches the first object of the array.
   - The filter `json/array[x eq 1 and y eq 4]` fails to match, because the array has no object `{"x":1,"y":4}`.

Paged results

You can page through search results using the following query string parameters that are further described in Query:

- `_pagedResultsCookie=string`
- `_pagedResultsOffset=integer`
- `_pageSize=integer`

The following example demonstrates how paged results are used:

```
# Request five results per page, and retrieve the first page:
$ curl \
 --user admin:password \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=true&_fields=userName&_pageSize=5&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "abarnes",
    "_rev" : "<revision>",
    "userName" : "abarnes@example.com"
  }, {
    "_id" : "abergin",
    "_rev" : "<revision>",
    "userName" : "abergin@example.com"
  }, {
    "_id" : "achassin",
    "_rev" : "<revision>",
    "userName" : "achassin@example.com"
  }, {
    "_id" : "ahall",
    "_rev" : "<revision>",
    "userName" : "ahall@example.com"
  }, {
    "_id" : "ahel",
    "_rev" : "<revision>",
    "userName" : "ahel@example.com"
  } ],
  "resultCount" : 5,
  "pagedResultsCookie" : "<cookie>",
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}

$ export COOKIE=$(cut -d \" -f 4 <(grep pagedResultsCookie \
 <(curl --cacert ca-cert.pem \
 --user admin:password \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=true&_fields=userName&_pageSize=5&_prettyPrint=true")))

# Provide the cookie to request the next five results:
$ curl \
 --user admin:password \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=true&_fields=userName&_pageSize=5\
&_pagedResultsCookie=$COOKIE&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "ahunter",
    "_rev" : "<revision>",
    "userName" : "ahunter@example.com"
  }, {
    "_id" : "ajensen",
    "_rev" : "<revision>",
    "userName" : "ajensen@example.com"
  }, {
```

```
      "_id" : "aknutson",
      "_rev" : "<revision>",
      "userName" : "aknutson@example.com"
    }, {
      "_id" : "alangdon",
      "_rev" : "<revision>",
      "userName" : "alangdon@example.com"
    }, {
      "_id" : "alutz",
      "_rev" : "<revision>",
      "userName" : "alutz@example.com"
    } ],
    "resultCount" : 5,
    "pagedResultsCookie" : "<new-cookie>",
    "totalPagedResultsPolicy" : "NONE",
    "totalPagedResults" : -1,
    "remainingPagedResults" : -1
  }

  # Request the tenth page of five results:
  $ curl \
   --user admin:password \
   --cacert ca-cert.pem \
   --silent \
   "https://localhost:8443/api/users?_queryFilter=true&_fields=userName\
  &_pageSize=5&_pagedResultsOffset=10&_prettyPrint=true"

  {
    "result" : [ {
      "_id" : "ashelton",
      "_rev" : "<revision>",
      "userName" : "ashelton@example.com"
    }, {
      "_id" : "awalker",
      "_rev" : "<revision>",
      "userName" : "awalker@example.com"
    }, {
      "_id" : "awhite",
      "_rev" : "<revision>",
      "userName" : "awhite@example.com"
    }, {
      "_id" : "aworrell",
      "_rev" : "<revision>",
      "userName" : "aworrell@example.com"
    }, {
      "_id" : "bfrancis",
      "_rev" : "<revision>",
      "userName" : "bfrancis@example.com"
    } ],
    "resultCount" : 5,
    "pagedResultsCookie" : "<cookie>",
    "totalPagedResultsPolicy" : "NONE",
    "totalPagedResults" : -1,
    "remainingPagedResults" : -1
  }
```

The LDAP searches corresponding to the queries in these examples are not indexed. Notice the following features of the responses:

- `"remainingPagedResults" : -1` means that the number of remaining results is not calculated.

- `"totalPagedResults" : -1` means that the total number of paged results is not calculated.
- `"totalPagedResultsPolicy" : "NONE"` means that result counting is disabled.

When the LDAP search corresponding to the query is indexed, and `_pageSize` is set to an integer greater than zero, the response contains an estimated number of `"totalPagedResults"`, and `"totalPagedResultsPolicy" : "ESTIMATE"`:

```
$ curl \
 --user kvaughan:bribery \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?
_queryFilter=userName+co+'jensen'&_pageSize=2&_fields=displayName&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "ajensen",
    "_rev" : "<revision>",
    "displayName" : [ "Allison Jensen" ]
  }, {
    "_id" : "gjensen",
    "_rev" : "<revision>",
    "displayName" : [ "Gern Jensen" ]
  } ],
  "resultCount" : 2,
  "pagedResultsCookie" : "<cookie>",
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

Notice that `"remainingPagedResults" : -1` in this case, too. REST to LDAP never calculates the value for this field, as doing so would require a potentially costly calculation to determine the current position in the total result set.

The estimated number of results can be useful when the LDAP search uses a big index or a VLV index, and the total number of results is large.

Server-side sort

You can use the `_sortKeys` parameter, described in Query, to request that the server sort the results it returns.

The following example sorts results by given name:

```
$ curl \
 --user admin:password \
 --cacert ca-cert.pem \
 --silent \
 "https://localhost:8443/api/users?_queryFilter=name/familyName+eq+'jensen'\
&_sortKeys=name/givenName&_fields=name&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "ajensen",
    "_rev" : "<revision>",
    "name" : {
      "givenName" : "Allison",
      "familyName" : "Jensen"
    }
  }, {
    "_id" : "bjensen",
    "_rev" : "<revision>",
    "name" : {
      "givenName" : "Barbara",
      "familyName" : "Jensen"
    }
  }, {
    "_id" : "bjense2",
    "_rev" : "<revision>",
    "name" : {
      "givenName" : "Bjorn",
      "familyName" : "Jensen"
    }
  }, {
    "_id" : "gjensen",
    "_rev" : "<revision>",
    "name" : {
      "givenName" : "Gern",
      "familyName" : "Jensen"
    }
  }, {
    "_id" : "jjensen",
    "_rev" : "<revision>",
    "name" : {
      "givenName" : "Jody",
      "familyName" : "Jensen"
    }
  }, {
    "_id" : "kjensen",
    "_rev" : "<revision>",
    "name" : {
      "givenName" : "Kurt",
      "familyName" : "Jensen"
    }
  }, {
    "_id" : "user.5814",
    "_rev" : "<revision>",
    "name" : {
      "givenName" : "Mollie",
      "familyName" : "Jensen"
    }
  }, {
    "_id" : "user.19233",
    "_rev" : "<revision>",
```

```
      "name" : {
        "givenName" : "Molly",
        "familyName" : "Jensen"
      }
    }, {
      "_id" : "user.32652",
      "_rev" : "<revision>",
      "name" : {
        "givenName" : "Mommy",
        "familyName" : "Jensen"
      }
    }, {
      "_id" : "user.46071",
      "_rev" : "<revision>",
      "name" : {
        "givenName" : "Mona",
        "familyName" : "Jensen"
      }
    }, {
      "_id" : "user.59490",
      "_rev" : "<revision>",
      "name" : {
        "givenName" : "Monah",
        "familyName" : "Jensen"
      }
    }, {
      "_id" : "user.72909",
      "_rev" : "<revision>",
      "name" : {
        "givenName" : "Monica",
        "familyName" : "Jensen"
      }
    }, {
      "_id" : "user.86328",
      "_rev" : "<revision>",
      "name" : {
        "givenName" : "Moniek",
        "familyName" : "Jensen"
      }
    }, {
      "_id" : "user.99747",
      "_rev" : "<revision>",
      "name" : {
        "givenName" : "Monika",
        "familyName" : "Jensen"
      }
    }, {
      "_id" : "rjense2",
      "_rev" : "<revision>",
      "name" : {
        "givenName" : "Randy",
        "familyName" : "Jensen"
      }
    }, {
      "_id" : "rjensen",
      "_rev" : "<revision>",
      "name" : {
        "givenName" : "Richard",
        "familyName" : "Jensen"
      }
    }, {
      "_id" : "tjensen",
```

```
      "_rev" : "<revision>",
      "name" : {
        "givenName" : "Ted",
        "familyName" : "Jensen"
      }
    } ],
    "resultCount" : 17,
    "pagedResultsCookie" : null,
    "totalPagedResultsPolicy" : "NONE",
    "totalPagedResults" : -1,
    "remainingPagedResults" : -1
  }
```

To sort in reverse order, use `_sortKeys=-field`.

To specify multiple sort keys, use a comma-separated list of fields.

The sort key fields that you specify must exist in the result entries.

The server must store and then sort the result set for your search. If you expect a large result set for your search, use paged results, described in Paged results, to limit the impact on the server and get your results more quickly.

Binary resources

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

Map a binary resource

1. Edit the attributes section for a resource in the configuration file `/path/to/opendj/config/rest2ldap/endpoints/api/example-v1.json`.

   The following JSON excerpt maps the user `photo` property to the `jpegPhoto` LDAP attribute:

   ```
   "photo" : { "type" : "simple", "ldapAttribute" : "jpegPhoto", "isBinary" : true },
   ```

2. Force the Rest2ldap endpoint to reread the updated configuration file:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --endpoint-name "/api" \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

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

Update a binary resource

1. Ensure that the application has a resource to upload.

   For example, copy a JPEG photo `picture.jpg` to the current directory.

2. Upload the binary resource as a base64-encoded JSON string.

   The following example patches Babs Jensen's resource to add a profile photo:

```
$ base64 encode --rawDataFile picture.jpg

/9j/4AAQSkZJRgABAQEAYABgAAD/4QAWRXhpZgAASUkqAAgAAAAAAAAAAD/
2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
wAARCAABAAEDASIAAhEBAxEB/8QAFQABAQAAAAAAAAAAAAAAAAAAAAr/xAAUEAEAAAAAAAAAAAAAAAAAAAA/
8QAFAEBAAAAAAAAAAAAAAAAAAAAP/EABQRAQAAAAAAAAAAAAAAAAAAAD/2gAMAwEAAhEDEQA/AL+AAf/Z

$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --header "Content-Type: application/json" \
 --data "[{\"operation\": \"add\", \"field\": \"/photo\", \"value\": \"/9j/4AAQSkZJRgABAQEAYABgAAD/
4QAWRXhpZgAASUkqAAgAAAAAAAAAAD/
2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
wAARCAABAAEDASIAAhEBAxEB/8QAFQABAQAAAAAAAAAAAAAAAAAAAAr/xAAUEAEAAAAAAAAAAAAAAAAAAAA/
8QAFAEBAAAAAAAAAAAAAAAAAAAAP/EABQRAQAAAAAAAAAAAAAAAAAAAD/2gAMAwEAAhEDEQA/AL+AAf/Z\"}]" \
 --silent \
 "https://kvaughan:bribery@localhost:8443/api/users/bjensen?_prettyPrint=true"

{
  "_id" : "bjensen",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "_meta" : {
    "lastModified" : "<datestamp>"
  },
  "userName" : "babs@example.com",
  "displayName" : [ "Barbara Jensen", "Babs Jensen" ],
  "name" : {
    "givenName" : "Barbara",
    "familyName" : "Jensen"
  },
  "description" : "Original description",
  "manager" : {
    "_id" : "trigden",
    "_rev" : "<revision>"
  },
  "groups" : [ {
    "_id" : "Carpoolers",
    "_rev" : "<revision>"
  } ],
  "photo" : "<base64-encoded-photo>",
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 9999",
    "emailAddress" : "babs@example.com"
  },
  "uidNumber" : 1076,
  "gidNumber" : 1000,
  "homeDirectory" : "/home/bjensen"
}
```

Read a binary resource

1. Read the binary resource as a base64-encoded JSON string.

The following example reads Babs Jensen's profile photo:

```
$ curl \
 --cacert ca-cert.pem \
 --silent \
 "https://kvaughan:bribery@localhost:8443/api/users/bjensen?_fields=photo"

{"_id":"bjensen","_rev":"<revision>","photo":"<base64-photo>"}
```

Configuration examples

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile.
> For details, see Learn about the evaluation setup profile.

Custom object

The example REST to LDAP mapping configuration file, `config/rest2ldap/endpoints/api/example-v1.json`, works well with the `ds-evaluation` setup profile. For most deployments, you must customize the mapping to expose additional information.

This example demonstrates how to configure an additional mapping for a custom LDAP object called `affiliateObject`, whose `affiliate` attribute references user entries. This demonstration extends a server set up with the `ds-evaluation` profile. The mappings will also work if you are using the REST to LDAP gateway.

To begin, add the schema for the custom LDAP object, and an example LDAP entry with the object class:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( affiliate-oid  NAME 'affiliate' SUP distinguishedName X-SCHEMA-FILE '99-example-mods.ldif' )
-
add: objectClasses
objectClasses: ( affiliateObject-oid NAME 'affiliateObject' SUP top STRUCTURAL MUST cn MAY ( description $
affiliate ) X-SCHEMA-FILE '99-example-mods.ldif' )
EOF

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN "uid=kvaughan,ou=people,dc=example,dc=com" \
 --bindPassword bribery << EOF
dn: ou=affiliates,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: affiliates

dn: cn=My Affiliates,ou=affiliates,dc=example,dc=com
objectClass: top
objectClass: affiliateObject
cn: My Affiliates
affiliate: uid=bjensen,ou=People,dc=example,dc=com
affiliate: uid=kvaughan,ou=People,dc=example,dc=com
EOF
```

For details, see LDAP schema and Add entries.

In the REST to LDAP mapping configuration file, define an `affiliates` collection subresource, and an `examples:affiliates:1.0` resource type that describes the objects in the `affiliates` collection. This causes the REST API to add an `/api/affiliates` path next to `/api/users` and `/api/groups`. Under `/api/affiliates`, REST clients access the affiliate JSON resources.

The following definition for the `affiliates` collection subresource belongs in the set of `subResources` in the mapping configuration file:

```
// This sub-resource definition maps JSON resources under /api/affiliates
// to LDAP entries under ou=affiliates,dc=example,dc=com.
"affiliates" : {
    "type": "collection",
    "dnTemplate": "ou=affiliates,dc=example,dc=com",
    "resource": "examples:affiliates:1.0",
    "namingStrategy": {
        "type": "clientDnNaming",
        "dnAttribute": "cn"
    }
}
```

The parser for the REST to LDAP mapping configuration file is lenient. It lets you include comments in the JSON, although the JSON standard does not allow comments.

The following definition for the `examples:affiliates:1.0` resource type belongs in the set of `example-v1` resource types in the mapping configuration file:

```
// An "affiliate" resource includes property mappings for an "affiliateObject",
// which is identified by the "cn" attribute. The affiliate DNs reference person entries.
// Rather than return DNs in JSON resources, a mapper returns identifiers and display names.
"examples:affiliates:1.0": {
    "superType": "frapi:opendj:rest2ldap:object:1.0",
    "objectClasses": [ "affiliateObject" ],
    "properties": {
        "displayName": {
            "type": "simple",
            "ldapAttribute": "cn",
            "isRequired": true,
            "writability": "createOnly"
        },
        "affiliate": {
            "type": "reference",
            "resourcePath": "../../users",
            "ldapAttribute": "affiliate",
            "isMultiValued": true
        },
        "description": {
            "type": "simple"
        }
    }
}
```

Alternatively, download example-v1.json to replace the existing file.

Replace the directory configuration file, `config/rest2ldap/endpoints/api/example-v1.json`, with the updated mapping configuration file. If you have not already done so, enable HTTP access. For details, see Configure HTTP user APIs. The following example forces the Rest2ldap endpoint to reread its configuration:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --endpoint-name "/api" \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

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

With the updated REST to LDAP mapping configuration loaded, REST clients can access affiliates:

```
$ curl \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --silent \
 "https://localhost:8443/api/affiliates/my%20affiliates?_fields=affiliate/id&_prettyPrint=true"

{
  "_id" : "My Affiliates",
  "_rev" : "<revision>",
  "affiliate" : [ {
    "_id" : "bjensen",
    "_rev" : "<revision>"
  }, {
    "_id" : "kvaughan",
    "_rev" : "<revision>"
  } ]
}
```

Per-server password policies

This example demonstrates how to add a per-server password policy over REST. Per-server password policies are set in the server configuration, and not replicated. You must create them on each replica.

The password policy in this example includes:

- A password history setting to retain the last five password values.
- The same password storage scheme as the default password policy.

- The default random password generator.
- The default length-based password validator.
- The default dictionary password validator, which is available, but not enabled by default.

With the default password policy, a user can change their password to `password`. A password policy with the default dictionary validator would not allow this:

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --header "Content-Type: application/json" \
 --data '{"oldPassword": "hifalutin", "newPassword": "password"}' \
 --silent \
 "https://localhost:8443/api/users/bjensen?_action=modifyPassword&dryRun=true&passwordQualityAdvice=true"

{}
```

Update the server configuration to enable the password policy for all Example.com users:

1. Enable the default dictionary password validator:

```
$ curl \
 --request PATCH \
 --user admin:password \
 --data '[{"operation": "replace", "field": "/enabled", "value": true}]' \
 --cacert ca-cert.pem \
 --header "Content-Type: application/json" \
 --silent \
 "https://localhost:8443/admin/config/password-validators/Dictionary"
```

2. Add the password policy:

```
$ curl \
 --request POST \
 --user admin:password \
 --data '{
  "_id": "Per-Server Password Policy",
  "_schema": "password-policy",
  "password-attribute": "userPassword",
  "default-password-storage-scheme": [{"_id": "PBKDF2-HMAC-SHA256"}],
  "password-generator": { "_id": "Random Password Generator" },
  "password-validator": [{"_id": "Dictionary"}, {"_id": "Length-Based Password Validator"}],
  "password-history-count": 5
 }' \
 --cacert ca-cert.pem \
 --header "Content-Type: application/json" \
 --silent \
 "https://localhost:8443/admin/config/password-policies/"
```

3. Assign the password policy to users:

The following command adds a virtual attribute that assigns the password policy to all Example.com users:

```
$ curl \
 --request POST \
 --user admin:password \
 --data '{
  "_id": "Password Policy Virtual Attribute",
  "_schema": "user-defined-virtual-attribute",
  "enabled": true,
  "base-dn": [ "ou=people,dc=example,dc=com" ],
  "filter": [ "(objectClass=person)" ],
  "attribute-type": "ds-pwp-password-policy-dn",
  "value": [ "cn=Per-Server Password Policy,cn=Password Policies,cn=config" ]
 }' \
 --cacert ca-cert.pem \
 --header "Content-Type: application/json" \
 --silent \
 "https://localhost:8443/admin/config/virtual-attributes/"
```

Check that the new policy does not let a user change their password to `password`:

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --header "Content-Type: application/json" \
 --data '{"oldPassword": "hifalutin", "newPassword": "password"}' \
 --silent \
 "https://localhost:8443/api/users/bjensen?_action=modifyPassword&dryRun=true&passwordQualityAdvice=true"

{
  "code" : 400,
  "reason" : "Bad Request",
  "message" : "Constraint Violation: The provided new password failed the validation checks defined in the server:
The provided password contained a word from the server's dictionary",
  "detail" : {
    "passwordQualityAdvice" : {
      "passingCriteria" : [ {
        "type" : "length-based",
        "parameters" : {
          "min-password-length" : 6,
          "max-password-length" : 0
        }
      } ],
      "failingCriteria" : [ {
        "type" : "dictionary",
        "parameters" : {
          "case-sensitive-validation" : false,
          "min-substring-length" : 5,
          "test-reversed-password" : true,
          "check-substrings" : true
        }
      } ]
    }
  }
}
```

For details on password policy settings, see Per-server password policies.

Subentry password policies

This example demonstrates how to configure REST to LDAP to manage Internet-Draft subentry password policies. This demonstration extends a server set up with the `ds-evaluation` profile. The mappings will also work if you are using the REST to LDAP gateway.

To begin, edit the default REST to LDAP mapping file, `config/rest2ldap/endpoints/api/example-v1.json`.

Define an `/api/subentries` endpoint that exposes LDAP subentries over REST, including password policies, by adding the following under `subResources`:

```
// This subresource definition maps JSON resources under /api/subentries
// to subentries under dc=example,dc=com.
"subentries" : {
  "type": "collection",
  "dnTemplate": "dc=example,dc=com",
  "resource": "examples:subentry:1.0",
  "namingStrategy": {
    "type": "clientDnNaming",
    "dnAttribute": "cn"
  }
}
```

Define the subentry and password policy under the other `example-v1` resource types:

```
// A subentry resource maps to a subentry object.
"examples:subentry:1.0": {
  "superType": "frapi:opendj:rest2ldap:object:1.0",
  "isAbstract": true,
  "objectClasses": [ "top", "subentry" ],
  "properties": {
    "_id": {
      "type": "simple",
      "ldapAttribute": "cn",
      "isRequired": true,
      "writability": "createOnly"
    },
    "subtreeSpecification": {
      "type": "simple"
    }
  }
},
// A passwordPolicy resource maps to a subentry password policy object.
// This mapping uses the LDAP attribute names,
// except for passwordValidator which maps to the validator in the configuration.
"examples:passwordPolicy:1.0": {
  "superType": "examples:subentry:1.0",
  "objectClasses": [ "pwdPolicy", "pwdValidatorPolicy", "subentry" ],
  "properties": {
    "pwdAttribute": {
      "type": "simple",
      "isRequired": true
    },
    "pwdAllowUserChange": {
      "type": "simple"
    },
    "pwdCheckQuality": {
      "type": "simple"
    },
    "pwdExpireWarning": {
      "type": "simple"
    },
    "pwdFailureCountInterval": {
      "type": "simple"
    },
    "pwdGraceAuthNLimit": {
      "type": "simple"
    },
    "pwdInHistory": {
      "type": "simple"
    },
    "pwdLockout": {
      "type": "simple"
    },
    "pwdLockoutDuration": {
      "type": "simple"
    },
    "pwdMaxAge": {
      "type": "simple"
    },
    "pwdMaxFailure": {
      "type": "simple"
    },
    "pwdMinAge": {
      "type": "simple"
```

```
      },
      "pwdMinLength": {
        "type": "simple"
      },
      "pwdMustChange": {
        "type": "simple"
      },
      "pwdSafeModify": {
        "type": "simple"
      },
      "passwordValidator" : {
        "type": "reference",
        "baseDn": "cn=Password Validators,cn=config",
        "ldapAttribute": "ds-cfg-password-validator",
        "primaryKey": "cn",
        "isMultiValued": true,
        "mapper": {
          "type": "object",
          "properties": {
            "_id": {
              "type": "simple",
              "ldapAttribute": "cn",
              "isRequired": true
            }
          }
        }
      }
    }
```

In LDAP, you can edit user entries to assign password policies. For REST to LDAP, add corresponding password policy properties to the `frapi:opendj:rest2ldap:user:1.0` resource type:

```
// Administrators can read the current password policy.
"pwdPolicy": {
  "type": "reference",
  "baseDn": "..,..",
  "ldapAttribute": "pwdPolicySubentry",
  "primaryKey": "cn",
  "searchFilter": "(&(objectclass=subentry)(objectclass=pwdPolicy))",
  "mapper": {
    "type": "object",
    "properties": {
      "_id": {
        "type": "simple",
        "ldapAttribute": "cn",
        "writability": "readOnlyDiscardWrites"
      }
    }
  }
},
// Administrators can set a new password policy.
"newPwdPolicy": {
  "type": "reference",
  "baseDn": "..,..",
  "ldapAttribute": "ds-pwp-password-policy-dn",
  "primaryKey": "cn",
  "searchFilter": "(&(objectclass=subentry)(objectclass=pwdPolicy))",
  "mapper": {
    "type": "object",
    "properties": {
      "_id": {
        "type": "simple",
        "ldapAttribute": "cn",
        "isRequired": true
      }
    }
  }
},
```

Alternatively, download subentries.json to replace the existing `example-v1.json` file.

Replace the directory configuration file, `config/rest2ldap/endpoints/api/example-v1.json`, with the updated mapping configuration file. If you have not already done so, enable HTTP access. For details, see Configure HTTP user APIs. The following example forces the Rest2ldap endpoint to reread its configuration:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --endpoint-name "/api" \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

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

With the augmented mapping in place, grant access for password administrators to edit and assign password policies:

- Grant the necessary privileges to edit subentry password policies:
  - The `subentry-write` privilege lets password administrators edit subentries.
  - The `config-read` privilege lets password administrators reference password validator entries stored in the server configuration.
- Grant access to assign password policies, letting password administrators read users' `pwdPolicySubentry` attributes and write users' `ds-pwp-password-policy-dn` attributes.
- Grant access to assign password policies, letting password administrators write the subentry attributes, `ds-pwp-password-validator` and `subtreeSpecification`.

For this demonstration, grant access to the administrator user Kirsten Vaughan:

```
# Assign privileges:
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password  << EOF
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: config-read
EOF

$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=kvaughan,ou=People,dc=example,dc=com
changetype: modify
add: ds-privilege-name
ds-privilege-name: subentry-write
EOF

# Grant access to subentry policies
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "pwdPolicySubentry||ds-pwp-password-policy-dn||ds-pwp-password-validator||subtreeSpecification")
 (version 3.0;acl "Allow Administrators to manage users' password policies";
 allow (all) (groupdn = "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com");)
EOF

# Grant access to configuration-based policy elements:
$ dsconfig \
 set-access-control-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --add "global-aci:(target=\"ldap:///cn=config\")(targetattr=\"*||+\")\
(version 3.0; acl \"Config read for Kirsten Vaughan\"; allow (read,search,compare)\
userdn=\"ldap:///uid=kvaughan,ou=People,dc=example,dc=com\";)" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Create and assign a subentry password policy:

```
$ curl \
 --request PUT \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --header "Content-Type: application/json" \
 --header "If-None-Match: *" \
 --data '{
  "_id": "Subentry Password Policy with Validators",
  "_schema": "examples:passwordPolicy:1.0",
  "pwdAttribute": "userPassword",
  "pwdAllowUserChange": true,
  "pwdFailureCountInterval": 300,
  "pwdLockout": true,
  "pwdLockoutDuration": 300,
  "pwdMaxFailure": 3,
  "pwdSafeModify": true,
  "passwordValidator": [{"_id": "Character Set"}, {"_id": "Length-Based Password Validator"}],
  "subtreeSpecification": "{base \"ou=people\", specificationFilter \"(isMemberOf=cn=Directory
Administrators,ou=Groups,dc=example,dc=com)\" }"
 }' \
 --silent \
 https://localhost:8443/api/subentries/Subentry%20Password%20Policy%20with%20Validators?_prettyPrint=true
```

Observe that the password administrator can view which password policy applies:

```
$ curl \
 --cacert ca-cert.pem \
 --user kvaughan:bribery \
 --silent \
 "https://localhost:8443/api/users/kvaughan?_fields=pwdPolicy&_prettyPrint=true"

{
  "_id" : "kvaughan",
  "_rev" : "<revision>",
  "pwdPolicy" : {
    "_id" : "Subentry Password Policy with Validators"
  }
}
```

Observe that the field is not visible to regular users:

```
$ curl \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --silent \
 "https://localhost:8443/api/users/bjensen?_fields=pwdPolicy&_prettyPrint=true"

{
  "_id" : "bjensen",
  "_rev" : "<revision>"
}
```

Map LDAP entries

This example demonstrates how to map an additional existing resource at the root of the REST API.

For example, the `ds-evaluation` setup profile includes localities:

```
dn: l=Bristol,ou=Locations,dc=example,dc=com
objectClass: top
objectClass: locality
l: Bristol
street: Broad Quay House, Prince Street

dn: l=Montbonnot,ou=Locations,dc=example,dc=com
objectClass: top
l: Montbonnot
street: 55 Rue Blaise Pascal

dn: l=Lysaker,ou=Locations,dc=example,dc=com
objectClass: top
objectClass: locality
l: Lysaker
street: Lysaker Torg 2

dn: l=San Francisco,ou=Locations,dc=example,dc=com
objectClass: top
objectClass: locality
l: San Francisco
street: 201 Mission Street Suite 2900

dn: l=Vancouver,ou=Locations,dc=example,dc=com
objectClass: top
objectClass: locality
l: Vancouver
street: 201 Northeast Park Plaza Drive
```

This demonstration adds an `/api/locations` next to `/api/users` and `/api/groups`.

Edit your copy of the default REST API mapping configuration. Add a location subresource at the top level next to users and groups subresource definitions:

```
"locations": {
    "type": "collection",
    "dnTemplate": "ou=locations,dc=example,dc=com",
    "resource": "frapi:opendj:rest2ldap:location:1.0",
    "namingStrategy": {
        "type": "clientDnNaming",
        "dnAttribute": "l"
    }
}
```

Add a location schema after the other schema definitions:

```
"frapi:opendj:rest2ldap:location:1.0": {
    "superType": "frapi:opendj:rest2ldap:object:1.0",
    "objectClasses": ["locality"],
    "properties": {
        "_id": {
            "type": "simple",
            "ldapAttribute": "l",
            "isRequired": true,
            "writability": "createOnly"
        },
        "displayName": {
            "type": "simple",
            "ldapAttribute": "l",
            "isRequired": true,
            "writability": "readOnly"
        },
        "description": {
            "type": "simple"
        },
        "state": {
            "type": "simple",
            "ldapAttribute": "st"
        },
        "street": {
            "type": "simple"
        }
    }
}
```

Alternatively, download example-locations.json to replace the existing `example-v1.json` file.

Replace the directory configuration file, `config/rest2ldap/endpoints/api/example-v1.json`, with the updated mapping configuration file. If you have not already done so, enable HTTP access. For details, see Configure HTTP user APIs. The following example forces the Rest2ldap endpoint to reread its configuration:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --endpoint-name "/api" \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

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

With the updated configuration loaded, view the results:

```
# List locations:
$ curl \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --silent \
 "https://localhost:8443/api/locations/?_queryFilter=true&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "Bristol",
    "_rev" : "<revision>",
    "_schema" : "frapi:opendj:rest2ldap:location:1.0",
    "displayName" : "Bristol",
    "street" : "Broad Quay House, Prince Street"
  }, {
    "_id" : "Montbonnot",
    "_rev" : "<revision>",
    "_schema" : "frapi:opendj:rest2ldap:location:1.0",
    "displayName" : "Montbonnot",
    "street" : "55 Rue Blaise Pascal"
  }, {
    "_id" : "Lysaker",
    "_rev" : "<revision>",
    "_schema" : "frapi:opendj:rest2ldap:location:1.0",
    "displayName" : "Lysaker",
    "street" : "Lysaker Torg 2"
  }, {
    "_id" : "San Francisco",
    "_rev" : "<revision>",
    "_schema" : "frapi:opendj:rest2ldap:location:1.0",
    "displayName" : "San Francisco",
    "street" : "201 Mission Street Suite 2900"
  }, {
    "_id" : "Vancouver",
    "_rev" : "<revision>",
    "_schema" : "frapi:opendj:rest2ldap:location:1.0",
    "displayName" : "Vancouver",
    "street" : "201 Northeast Park Plaza Drive"
  } ],
  "resultCount" : 5,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}

# Read a single location:
$ curl \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --silent \
 "https://localhost:8443/api/locations/montbonnot?_prettyPrint=true"

{
  "_id" : "Montbonnot",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:location:1.0",
  "displayName" : "Montbonnot",
  "street" : "55 Rue Blaise Pascal"
}
```

Nested resources

This example demonstrates how to add device resources under user resources. Use this pattern when you have LDAP child entries under parent entries.

For example, the `ds-evaluation` setup profile includes a user with two subordinate devices:

```
dn: uid=nbohr,ou=People,dc=example,dc=com
objectClass: person
objectClass: cos
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: posixAccount
objectClass: top
uid: nbohr
classOfService: gold
userpassword: password
facsimileTelephoneNumber: +1 408 555 1213
givenName: Niels
cn: Niels Bohr
telephoneNumber: +1 408 555 1212
sn: Bohr
roomNumber: 0007
homeDirectory: /home/nbohr
mail: nbohr@example.com
l: Vancouver
ou: People
uidNumber: 1111
gidNumber: 1000
description: Quantum device example

dn: cn=quantum dot,uid=nbohr,ou=People,dc=example,dc=com
objectClass: device
objectClass: top
cn: quantum dot
serialNumber: WI-3005
owner: uid=nbohr,ou=People,dc=example,dc=com

dn: cn=qubit generator,uid=nbohr,ou=People,dc=example,dc=com
objectClass: device
objectClass: top
cn: qubit generator
serialNumber: XF551426
owner: uid=nbohr,ou=People,dc=example,dc=com
```

In your REST API, the users collection contains zero or more user resources. Each user resource can have a devices collection that contains zero or more device resources. The path to a device is `/users/uid/devices/cn`, where *uid* is the user UID, and *cn* is the device CN. For example, `/users/nbohr/devices/quantum dot`.

Edit your copy of the default REST API mapping configuration. Add the subresources definition to the `frapi:opendj:rest2ldap:user:1.0` schema:

```
"subResources": {
    "devices": {
        "type": "collection",
        "resource": "frapi:opendj:rest2ldap:device:1.0",
        "namingStrategy": {
            "type": "clientDnNaming",
            "dnAttribute": "cn"
        }
    }
}
```

Add a device schema after the other schema definitions:

```
"frapi:opendj:rest2ldap:device:1.0": {
    "superType": "frapi:opendj:rest2ldap:object:1.0",
    "objectClasses": ["device"],
    "properties": {
        "_id": {
            "type": "simple",
            "ldapAttribute": "cn",
            "isRequired": true,
            "writability": "createOnly"
        },
        "displayName": {
            "type": "simple",
            "ldapAttribute": "cn",
            "isRequired": true,
            "writability": "readOnly"
        },
        "description": {
            "type": "simple"
        },
        "owner": {
            "type": "reference",
            "resourcePath": "../..",
            "ldapAttribute": "owner"
        },
        "serialNumber": {
            "type": "simple"
        }
    }
}
```

Alternatively, download example-subresources.json to replace the default `example-v1.json` file.

Replace the directory configuration file, `config/rest2ldap/endpoints/api/example-v1.json`, with the updated mapping configuration file. If you have not already done so, enable HTTP access. For details, see Configure HTTP user APIs. The following example forces the Rest2ldap endpoint to reread its configuration:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --endpoint-name "/api" \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

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

With the updated configuration loaded, view the results:

```
# Read the user resource:
$ curl \
 --cacert ca-cert.pem \
 --user nbohr:password \
 --silent \
 "https://localhost:8443/api/users/nbohr?_prettyPrint=true"

{
  "_id" : "nbohr",
  "_rev" : "<revision>",
  "_schema" : "frapi:opendj:rest2ldap:posixUser:1.0",
  "userName" : "nbohr@example.com",
  "displayName" : [ "Niels Bohr" ],
  "name" : {
    "givenName" : "Niels",
    "familyName" : "Bohr"
  },
  "description" : "Quantum device example",
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "nbohr@example.com"
  },
  "uidNumber" : 1111,
  "gidNumber" : 1000,
  "homeDirectory" : "/home/nbohr"
}

# List devices:
$ curl \
 --cacert ca-cert.pem \
 --user nbohr:password \
 --silent \
 "https://localhost:8443/api/users/nbohr/devices/?_queryFilter=true&_prettyPrint=true"

{
  "result" : [ {
    "_id" : "quantum dot",
    "_rev" : "<revision>",
    "_schema" : "frapi:opendj:rest2ldap:device:1.0",
    "displayName" : "quantum dot",
    "owner" : {
      "_id" : "nbohr",
      "_rev" : "<revision>"
    },
    "serialNumber" : "WI-3005"
  }, {
    "_id" : "qubit generator",
    "_rev" : "<revision>",
    "_schema" : "frapi:opendj:rest2ldap:device:1.0",
    "displayName" : "qubit generator",
    "owner" : {
      "_id" : "nbohr",
      "_rev" : "<revision>"
    },
    "serialNumber" : "XF551426"
  } ],
  "resultCount" : 2,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
```

```
  "remainingPagedResults" : -1
}

# Read a device resource:
$ curl \
 --cacert ca-cert.pem \
 --user nbohr:password \
 --silent \
 "https://localhost:8443/api/users/nbohr/devices/quantum%20dot?_prettyPrint=true"

{
  "_id" : "quantum dot",
  "_rev" : "000000007ba330d8",
  "_schema" : "frapi:opendj:rest2ldap:device:1.0",
  "displayName" : "quantum dot",
  "owner" : {
    "_id" : "nbohr",
    "_rev" : "<revision>"
  },
  "serialNumber" : "WI-3005"
}m
```

JSON to LDAP

This example demonstrates simple user profiles, starting from JSON and working back to LDAP. The following JSON is a example profile:

```
{
    "externalId": "newuser",
    "userName": "newuser",
    "name": {
        "formatted": "New User",
        "givenName": "User",
        "familyName": "New"
    },
    "phoneNumbers": [{
        "value": "+1 408 555 1212",
        "type": "work"
    }],
    "emails": [
        {
            "value": "newuser@example.com",
            "type": "work",
            "primary": true
        },
        {
            "value": "newuser@example.org",
            "type": "personal",
            "primary": false
        }
    ]
}
```

Notice these key features of the example profile:

- Unlike Common REST JSON objects, this profile has no `"_id"` field.

- The `"phoneNumbers"` and `"emails"` fields are multi-valued, similar to the LDAP `telephoneNumber` and `mail` attributes.

  However, both JSON fields hold arrays of nested objects. The similar LDAP attributes hold simple values. The other fields have a straightforward mapping to standard LDAP attributes, but these fields do not.

  The `"type"` field in a phone number or email is an arbitrary label assigned by the user.

This example assumes that the user profile does not exist in LDAP yet. The aim is therefore to translate JSON objects into LDAP entries.

LDAP attributes generally hold values with tightly defined syntaxes, not arbitrary objects. JSON syntax attributes are an exception. The example profile holds a mix of fields that map directly to LDAP attributes, and fields that do not.

When determining them LDAP attributes to use, the first step is to look for natural matches with attributes defined in the default schema. Based on the *About This Reference*, you can derive the following table of correspondences:

Profile JSON Field	LDAP Attribute
`"externalId"`	`uid`
`"userName"`	`uid`
`"name/formatted"`	`cn`
`"name/givenName"`	`givenName`
`"name/familyName"`	`sn`
`"emails"`	No direct match.
`"phoneNumbers"`	No direct match.

For the `"emails"` and `"phoneNumbers"` fields, there is no natural match. The `uddiEMail` and `uddiPhone` attributes come the closest because they optionally include a user-defined type. However, avoid those attributes for the following reasons:

- These user profiles are unrelated to Universal Description, Discovery, and Integration (UDDI), which is focused instead on storing data for a SOAP-based web services discovery.
- All client applications would have to know how to consume the UDDI-specific format. REST to LDAP has no means to transform `type#value` into `{"type": "type", "value": "value"}`.
- The `uddiEmail` has no provision for the `"primary"` boolean value.

It would also be possible, but not advisable, to store `"emails"` and `"phoneNumbers"` in separate LDAP entries. The LDAP user entry could include attributes that reference email address and phone number entries by their DNs. The email and phone number entries could in turn reference users with the standard `owner` attribute. Unfortunately, there is no real value in storing email addresses and phone numbers separately from a user's entry. The email and phone settings will not be shared with other entries, but instead belong only to the profile. Client applications will expect to update them atomically with the user profile. A profile change should not require updating an arbitrary number of LDAP entries. Even if they could be conveniently configured as single updates in the REST API, it would mean that updates to a JSON resource could partially fail in LDAP, a source of potential confusion.

An apt choice for the `"emails"` and `"phoneNumbers"` fields is therefore JSON syntax attributes. This example defines the following LDAP schema definitions for appropriate JSON attributes:

```
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( example-email-oid
  NAME 'email'
  DESC 'An email address with an optional user-defined type and primary boolean'
  EQUALITY caseIgnoreJsonValueQueryMatch
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1
  USAGE userApplications
  X-SCHEMA-FILE '99-example.ldif'
  X-ORIGIN 'DS Documentation Examples' )
-
add: attributeTypes
attributeTypes: ( example-phone-number-oid
  NAME 'phoneNumber'
  DESC 'A phone number with an optional user-defined type'
  EQUALITY caseIgnoreJsonValueQueryMatch
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1
  USAGE userApplications
  X-SCHEMA-FILE '99-example.ldif'
  X-ORIGIN 'DS Documentation Examples' )
-
add: objectClasses
objectClasses: ( example-profile-user-oid
  NAME 'profileUser'
  DESC 'A user with optional profile components having user-defined types'
  SUP top
  AUXILIARY
  MAY ( email $ phoneNumber )
  X-SCHEMA-FILE '99-example.ldif'
  X-ORIGIN 'DS Documentation Examples' )
```

An auxiliary object class lets an entry that already has a structural object class hold additional attributes. For background information about LDAP schema, read LDAP schema.

The following corresponding LDIF uses these definitions to define Example.com data with one user profile. Babs Jensen is the administrator and sole initial user. She can create other profiles and reset passwords:

```
dn: dc=example,dc=com
objectClass: domain
objectClass: top
dc: example
aci: (target ="ldap:///dc=example,dc=com")
 (targetattr != "userPassword")
 (version 3.0;acl "Authenticated users have read-search access";
 allow (read, search, compare)(userdn = "ldap:///all");)
aci: (target ="ldap:///dc=example,dc=com")
 (targetattr = "*")
 (version 3.0;acl "Allow Babs to administer other entries";
 allow (all)(userdn = "ldap:///uid=bjensen,ou=People,dc=example,dc=com");)

dn: ou=People,dc=example,dc=com
objectClass: organizationalunit
objectClass: top
ou: People
aci: (target ="ldap:///ou=People,dc=example,dc=com")
 (targetattr = "*")
 (version 3.0; acl "Allow self management of profiles";
 allow (delete, write)(userdn = "ldap:///self");)

dn: uid=bjensen,ou=People,dc=example,dc=com
objectClass: profileUser
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
uid: bjensen
ou: People
cn: Barbara Jensen
cn: Babs Jensen
givenname: Barbara
sn: Jensen
userpassword: hifalutin
email: {"value": "bjensen@example.com", "type": "work", "primary": true}
phoneNumber: {"value": "+1 408 555 1862", "type": "work"}
ds-privilege-name: password-reset
```

Client applications might look up user profiles based on email addresses or phone numbers. They are not likely to look up user profiles based on user-defined labels for emails and phone numbers. Therefore, instead of indexing all fields of each JSON attribute value, index only the values:

1. Install a directory server with the `ds-evaluation` setup profile.
2. Configure a custom schema provider to allow the server to index only the JSON `"values"`:

```
$ dsconfig \
 create-schema-provider \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --provider-name "Value JSON Query Matching Rule" \
 --type json-query-equality-matching-rule \
 --set enabled:true \
 --set case-sensitive-strings:false \
 --set ignore-white-space:true \
 --set matching-rule-name:caseIgnoreJsonValueQueryMatch \
 --set matching-rule-oid:1.3.6.1.4.1.36733.2.1.4.1.2 \
 --set indexed-field:value \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. Update the LDAP schema:

```
$ ldapmodify \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( example-email-oid
  NAME 'email'
  DESC 'An email address with an optional user-defined type and primary boolean'
  EQUALITY caseIgnoreJsonValueQueryMatch
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1
  USAGE userApplications
  X-SCHEMA-FILE '99-example.ldif'
  X-ORIGIN 'DS Documentation Examples' )
-
add: attributeTypes
attributeTypes: ( example-phone-number-oid
  NAME 'phoneNumber'
  DESC 'A phone number with an optional user-defined type'
  EQUALITY caseIgnoreJsonValueQueryMatch
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1
  USAGE userApplications
  X-SCHEMA-FILE '99-example.ldif'
  X-ORIGIN 'DS Documentation Examples' )
-
add: objectClasses
objectClasses: ( example-profile-user-oid
  NAME 'profileUser'
  DESC 'A user with optional profile components having user-defined types'
  SUP top
  AUXILIARY
  MAY ( email $ phoneNumber )
  X-SCHEMA-FILE '99-example.ldif'
  X-ORIGIN 'DS Documentation Examples' )
EOF
```

4. Add indexes for the `email` and `phoneNumber` JSON attributes:

```
$ dsconfig \
 create-backend-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name email \
 --set index-type:equality \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 create-backend-index \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backend-name dsEvaluation \
 --index-name phoneNumber \
 --set index-type:equality \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

5. Import the LDAP data with Babs's profile:

```
$ import-ldif \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --backendID dsEvaluation \
 --ldifFile /path/to/opendjprofiles.ldif \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

6. To check your work, read Babs's profile over LDAP:

```
$ ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDn dc=example,dc=com \
 "(uid=bjensen)"

dn: uid=bjensen,ou=People,dc=example,dc=com
objectClass: profileUser
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: top
cn: Barbara Jensen
cn: Babs Jensen
email: {"value": "bjensen@example.com", "type": "work", "primary": true}
givenName: Barbara
ou: People
phoneNumber: {"value": "+1 408 555 1862", "type": "work"}
sn: Jensen
uid: bjensen
```

Now that you have sample data in LDAP, create the REST to LDAP mapping.

The REST to LDAP mapping defines the HTTP API and JSON resources that are backed by the LDAP service and entries.

Rather than patch the default example mapping, this example adds a separate API with a new mapping for the profile view.

The new mapping uses the following features:

- A `"serverNaming"` naming strategy ensures proper handing of Common REST `"_id"` values. LDAP entries use `uid` as the RDN. JSON profiles use server-generated `"_id"` values.

  The JSON profiles have `"externalId"` values corresponding to the LDAP `uid`. However, the JSON profiles do not define `"_id"` values. Client applications will use the `"_id"` values when uniquely identifying a profile, but not when looking up a profile.

  The `"_id"` maps to the `entryUUID` LDAP operational attribute. The `entryUUID` attribute is generated on creation and managed by the LDAP server.

- As in the default example, passwords are not visible in profiles.

  Client applications use dedicated REST actions to manage password reset by administrators and to manage password modifications by users. The REST actions for password management require HTTPS.

- The `"externalId"` and `"userName"` fields both map to LDAP `uid` attributes.

  You could make it possible to update these fields after creation. This example assumes that these fields might be expected not to change, and so restricts client applications from changing them.

- The `"formatted"` and `"familyName"` fields of the name map to attributes that are required by the LDAP object class, `inetOrgPerson`. They are therefore required in the JSON profile as well.

The full mapping file is shown in the following listing:

```
{
    "version": "1.0",
    "resourceTypes": {
        "example-mapping": {
            "subResources": {
                "users": {
                    "type": "collection",
                    "dnTemplate": "ou=people,dc=example,dc=com",
                    "resource": "examples:user:1.0",
                    "namingStrategy": {
                        "type": "serverNaming",
                        "dnAttribute": "uid",
                        "idAttribute": "entryUUID"
                    }
                }
            }
        },
        "frapi:opendj:rest2ldap:object:1.0": {
            "isAbstract": true,
            "objectClasses": [
                "top"
            ],
            "properties": {
                "_schema": {
                    "type": "resourceType"
                },
                "_rev": {
                    "type": "simple",
                    "ldapAttribute": "etag",
                    "writability": "readOnly"
                },
                "_meta": {
                    "type": "object",
                    "properties": {
                        "created": {
                            "type": "simple",
                            "ldapAttribute": "createTimestamp",
                            "writability": "readOnly"
                        },
                        "lastModified": {
                            "type": "simple",
                            "ldapAttribute": "modifyTimestamp",
                            "writability": "readOnly"
                        }
                    }
                }
            }
        },
        "examples:user:1.0": {
            "superType": "frapi:opendj:rest2ldap:object:1.0",
            "objectClasses": [
                "profileUser",
                "person",
                "organizationalPerson",
                "inetOrgPerson"
            ],
            "supportedActions": [
                "modifyPassword",
                "resetPassword"
            ],
```

```
            "properties": {
                "externalId": {
                    "type": "simple",
                    "ldapAttribute": "uid",
                    "isRequired": true,
                    "writability": "createOnlyDiscardWrites"
                },
                "userName": {
                    "type": "simple",
                    "ldapAttribute": "uid",
                    "isRequired": true,
                    "writability": "createOnlyDiscardWrites"
                },
                "name": {
                    "type": "object",
                    "properties": {
                        "formatted": {
                            "type": "simple",
                            "ldapAttribute": "cn",
                            "isRequired": true
                        },
                        "givenName": {
                            "type": "simple"
                        },
                        "familyName": {
                            "type": "simple",
                            "ldapAttribute": "sn",
                            "isRequired": true
                        }
                    }
                },
                "emails": {
                    "type": "json",
                    "ldapAttribute": "email",
                    "isMultiValued": true
                },
                "phoneNumbers": {
                    "type": "json",
                    "ldapAttribute": "phoneNumber",
                    "isMultiValued": true
                }
            }
        }
    }
}
```

1. Download the mapping file, example-mapping.json.
2. Copy the mapping file to the appropriate server REST to LDAP configuration directory:

   ```
   $ mkdir /path/to/opendj/config/rest2ldap/endpoints/rest

   $ cp example-mapping.json /path/to/opendj/config/rest2ldap/endpoints/rest/
   ```

   The default example defines a REST API under `/api`. This example uses `/rest` instead.
3. Enable a Rest2ldap endpoint for the API:

```
$ dsconfig \
 create-http-endpoint \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set authorization-mechanism:HTTP\ Basic \
 --set config-directory:config/rest2ldap/endpoints/rest \
 --set enabled:true \
 --type rest2ldap-endpoint \
 --endpoint-name /rest \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

Look Up a Profile by Email Address

The following query looks for profiles with email addresses containing `bjensen`:

```
$ curl \
 --request GET \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --silent \
 "https://localhost:8443/rest/users/?_queryFilter=emails/value+co+'bjensen'"

{
  "result" : [ {
    "_id" : "<generated-id>",
    "_rev" : "<revision>",
    "_schema" : "examples:user:1.0",
    "externalId" : "bjensen",
    "userName" : "bjensen",
    "name" : {
      "formatted" : [ "Barbara Jensen", "Babs Jensen" ],
      "givenName" : "Barbara",
      "familyName" : "Jensen"
    },
    "emails" : [ {
      "value" : "bjensen@example.com",
      "type" : "work",
      "primary" : true
    } ],
    "phoneNumbers" : [ {
      "value" : "+1 408 555 1862",
      "type" : "work"
    } ]
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

Look Up a Profile by Phone Number

The following query looks for profiles with the phone number `+1 408 555 1862`:

```
$ curl \
 --request GET \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --silent \
 "https://localhost:8443/rest/users/?_queryFilter=phoneNumbers/value+eq+'%2B1%20408%20555%201862'"
```

Notice that the telephone number is URL encoded as `%2B1%20408%20555%201862`.

Create a User Profile

This example creates a profile using the JSON in newuser.json:

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user bjensen:hifalutin \
 --header "Content-Type: application/json" \
 --data @newuser.json \
 --silent \
 https://localhost:8443/rest/users/
```

Upon initial creation, the user has no password, and so cannot authenticate, yet.

Set a Password

Set a password to allow the user to authenticate.

This example uses Bash shell and jq to manipulate JSON on the command line.

The first operation gets the user profile `"_id"` and holds it in an `ID` environment variable.

The second operation performs an administrative password reset as Babs and holds the resulting generated password in a `PASSWORD` environment variable.

The third and final operation uses the generated password to authenticate as the user and modify the password:

```
$ export ID=$(jq --raw-output '.result[0] ._id' \
 <(curl --silent --request GET --cacert ca-cert.pem --user bjensen:hifalutin 2>/dev/null \
 "https://localhost:8443/rest/users/?_queryFilter=externalId+eq+'newuser'"))

$ export PASSWORD=$(jq --raw-output '.generatedPassword' \
 <(curl --silent --request POST --cacert ca-cert.pem --user bjensen:hifalutin \
 --header "Content-Type: application/json" --data '{}' \
 "https://localhost:8443/rest/users/${ID}?_action=resetPassword"))

$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user "newuser:${PASSWORD}" \
 --header "Content-Type: application/json" \
 --data "{\"oldPassword\": \"${PASSWORD}\", \"newPassword\": \"password\"}" \
 --silent \
 "https://localhost:8443/rest/users/${ID}?_action=modifyPassword"
```

Read Your Profile

This example employs the user profile `"_id"`:

```
$ curl \
 --request GET \
 --cacert ca-cert.pem \
 --user newuser:password \
 --silent \
 "https://localhost:8443/rest/users/${ID}"
```

Changing a User's Own User-Defined Email Type

This example employs the user profile `"_id"`.

Download the JSON patch payload, changeEmailType.json:

```
$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --user newuser:password \
 --header "Content-Type: application/json" \
 --data @changeEmailType.json \
 --silent \
 "https://localhost:8443/rest/users/${ID}"

{
  "_id" : "<generated-user-id>",
  "_rev" : "<revision>",
  "_schema" : "examples:user:1.0",
  "_meta" : {
    "created" : "<datestamp>"
  },
  "externalId" : "newuser",
  "userName" : "newuser",
  "name" : {
    "formatted" : "New User",
    "givenName" : "User",
    "familyName" : "New"
  },
  "emails" : [ {
    "value" : "newuser@example.com",
    "type" : "work",
    "primary" : true
  }, {
    "value" : "newuser@example.org",
    "type" : "home",
    "primary" : false
  } ],
  "phoneNumbers" : [ {
    "value" : "+1 408 555 1212",
    "type" : "work"
  } ]
}
```

Remove Your Email Address

This example employs the user profile `"_id"`.

Download the JSON patch payload, removeWorkEmail.json:

```
$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --user newuser:password \
 --header "Content-Type: application/json" \
 --data @removeWorkEmail.json \
 --silent \
 "https://localhost:8443/rest/users/${ID}"

{
  "_id" : "<generated-user-id>",
  "_rev" : "<revision>",
  "_schema" : "examples:user:1.0",
  "_meta" : {
    "created" : "<datestamp>"
  },
  "externalId" : "newuser",
  "userName" : "newuser",
  "name" : {
    "formatted" : "New User",
    "givenName" : "User",
    "familyName" : "New"
  },
  "emails" : [ {
    "value" : "newuser@example.org",
    "type" : "home",
    "primary" : true
  } ],
  "phoneNumbers" : [ {
    "value" : "+1 408 555 1212",
    "type" : "work"
  } ]
}
```

Add Your Cell Phone Number

This example employs the user profile `"_id"`.

Download the JSON patch payload, addPersonalCell.json:

```
$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --user newuser:password \
 --header "Content-Type: application/json" \
 --data @addPersonalCell.json \
 --silent \
 "https://localhost:8443/rest/users/${ID}"

{
  "_id" : "<generated-user-id>",
  "_rev" : "<revision>",
  "_schema" : "examples:user:1.0",
  "_meta" : {
    "created" : "<datestamp>"
  },
  "externalId" : "newuser",
  "userName" : "newuser",
  "name" : {
    "formatted" : "New User",
    "givenName" : "User",
    "familyName" : "New"
  },
  "emails" : [ {
    "value" : "newuser@example.org",
    "type" : "home",
    "primary" : true
  } ],
  "phoneNumbers" : [ {
    "value" : "+1 408 555 1212",
    "type" : "work"
  }, {
    "value": "+1 408 555 1234",
    "type": "personal cell"
  } ]
}
```

Delete Your Profile

This example employs the user profile `"_id"`:

```
$ curl \
 --request DELETE \
 --cacert ca-cert.pem \
 --user newuser:password \
 --header "Content-Type: application/json" \
 --silent \
 "https://localhost:8443/rest/users/${ID}"
```

REST API documentation

API descriptors provide runtime documentation for REST APIs. Requests for API descriptors use the reserved query string parameters, `_api` and `_crestapi`. By default, DS servers do not return descriptors, but respond instead with HTTP status code `501 Not Implemented`.

> **Note**
>
> Although it is possible to serve the descriptors at runtime, do not use production servers for this purpose. Instead, prepare the documentation by reading API descriptors from a server with the same API as production servers. Publish the documentation separately.

Preparing documentation for a Rest2ldap endpoint is an iterative process:

1. Enable API descriptors for the connection handler you use:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name HTTPS \
 --set api-descriptor-enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

2. Restart the connection handler to take the configuration change into account:

```
$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name HTTPS \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

$ dsconfig \
 set-connection-handler-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --handler-name HTTPS \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

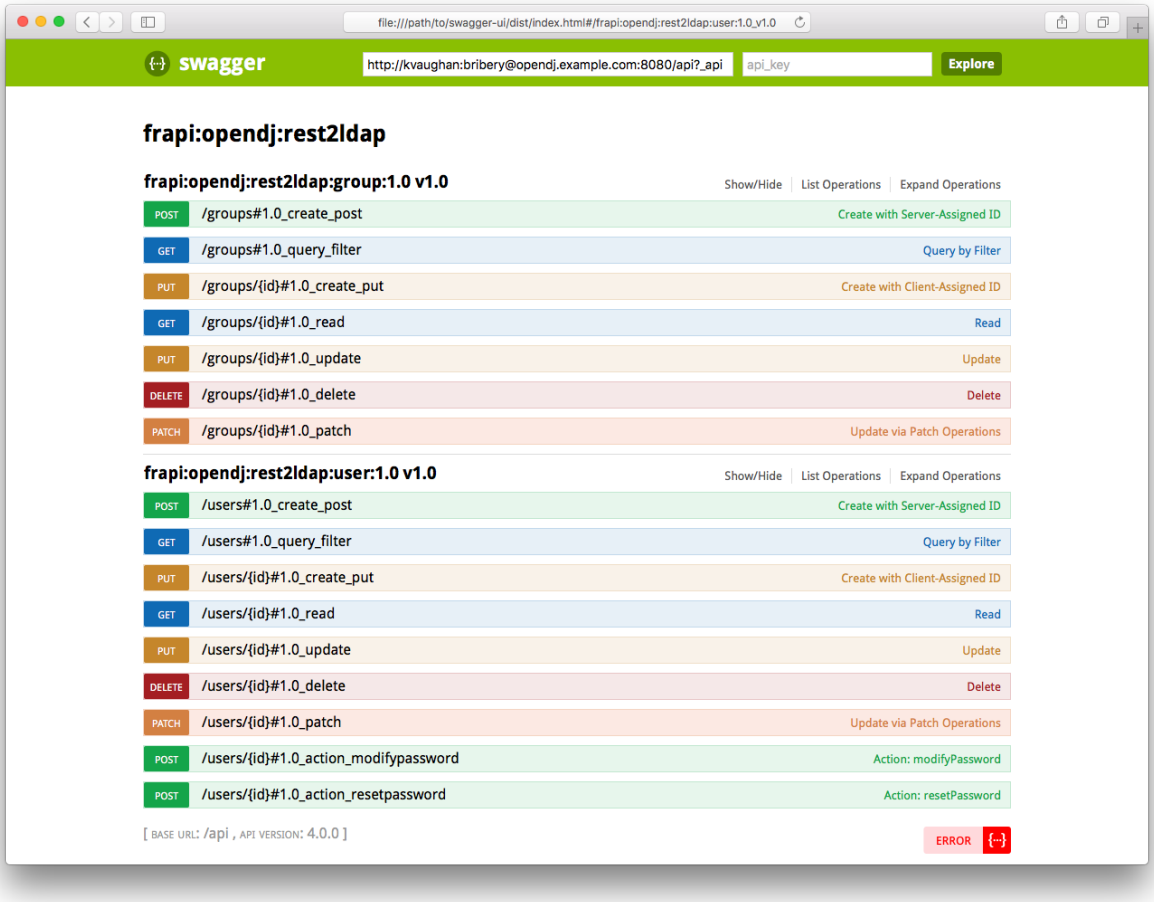
3. Configure the API.

4. Run a local copy of a tool for viewing OpenAPI documentation, such as Swagger UI.

5. View the generated documentation through the tool by reading the OpenAPI format descriptor.

For example, read the descriptor for the `/api` endpoint with a URL such as `https://kvaughan:bribery@localhost:8443/api?_api` for directory data, or `https://admin:password@localhost:8443/admin?_api` for the server configuration.

The following screenshot shows example documentation:



If your browser does not display the generated documentation, disable CORS settings. See your browser's documentation or search the web for details.

6. Update the API configuration.

7. Force the Rest2ldap endpoint to reread the updated configuration file:

```
$ dsconfig \
 set-http-endpoint-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --endpoint-name "/api" \
 --set enabled:false \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt

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

8. Edit the descriptor.
9. Publish the final descriptor alongside your production service.

REST to LDAP reference

DS software offers these alternatives for HTTP access to directory data:

	REST to LDAP Gateway	DS Server
Implementation	The gateway is a servlet that connects to remote LDAP server(s).	The DS server exposes RESTful HTTP APIs to its directory data.
Base Configuration Directory	`WEB-INF/classes`	`opendj/config`
Connection Configuration	A single configuration file defines how the gateway connects and authenticates to LDAP servers. For details, see Gateway LDAP connections.	The server uses an internal connection. Identity mappers define how HTTP user identities map to LDAP user identities. For details, see Identity mappers.
Gateway Configuration	A single configuration file defines which LDAP features the gateway uses. For details, see Gateway LDAP features.	The server configuration defines which LDAP features are enabled.

	REST to LDAP Gateway	DS Server
API Configuration	In both cases, one or more configuration files define the HTTP APIs that map JSON resources to LDAP entries. For details, see API configuration.	

Unlike standard JSON, REST to LDAP JSON configuration files permit `//`-style comments.

Gateway LDAP connections

A single `config.json` file defines how the gateway connects and authenticates to LDAP servers. The top-level fields are:

```
{
  // Secure connections to remote LDAP servers.
  "security": {},
  // Connect and authenticate to remote LDAP servers.
  "ldapConnectionFactories": {},
  // Set authorization policies for access to directory data.
  "authorization": {}
}
```

Security

The `"security"` field covers the parameters for secure connections to remote servers:

```
{
  "security": {
    // Specifies the policy for trusting server certificates exchanged
    // during SSL/StartTLS negotiation. This setting and the following
    // trust policy settings will be ignored if there is no connection
    // security. Acceptable values are:
    //
    // "trustAll" - blindly trust all server certificates (not secure)
    // "jvm"      - only certificates signed by the authorities
    //              associated with the host JVM will be accepted (default)
    // "file"     - use a file-based trust store for validating
    //              certificates. This option requires the following
    //              "fileBasedTrustManager*" settings to be configured.
    //
    "trustManager": "jvm",

    // File based trust manager configuration (see above).
    "fileBasedTrustManagerType": "JKS",
    "fileBasedTrustManagerFile": "/path/to/truststore",
    "fileBasedTrustManagerPasswordFile": "/path/to/pinfile",

    // Specifies the key-manager for TLS client authentication
    // (mutual TLS, mTLS) against the remote server.
    // Acceptable values are:
    //
    // "none"     - disables TLS client authentication,
    //              no client certificates will be used. (default)
    // "jvm"      - use the JVM's default keystore
    //              for retrieving client certificates.
    // "file"     - use a file-based key store
    //              for retrieving client certificates.
    // "pkcs11"   - use a PKCS#11 token
    //              for retrieving client certificates.
    "keyManager": "none",

    // Keystore based key manager configuration (see above).
    "fileBasedKeyManagerType": "JKS",
    "fileBasedKeyManagerFile": "/path/to/keystore",
    "fileBasedKeyManagerPasswordFile": "/path/to/pinfile",

    // PKCS11 based key manager configuration
    "pkcs11KeyManagerPasswordFile": "/path/to/pinfile"
  }
}
```

LDAP connection factories

The `"ldapConnectionFactories"` field configures connection and authentication to remote servers:

```
{
  "ldapConnectionFactories": {
    // Unauthenticated connections used for performing bind requests.
    "bind": {
      // Indicates whether LDAP connections should be secured using
      // SSL or StartTLS. Acceptable values are:
      //
      // "none"     - use plain LDAP connections (default)
      // "ssl"      - secure connection using LDAPS
      // "startTLS" - secure connection using LDAP+StartTLS
      //
      "connectionSecurity": "none",

      // This alias references a client SSL key-pair which will be used
      // for performing client authentication (mutual TLS or mTLS) between
      // between this gateway and the remote LDAP server.
      "sslCertAlias": "client-cert",

      // Re-usable pool of 24 connections per server.
      "connectionPoolSize": 24,

      // Specifies the timeout for connections.
      // If a request for a connection cannot be fulfilled,
      // it times out after this interval.
      "connectionTimeoutMilliSeconds": 10000,

      // Ensure connections are kept alive by sending a periodic search request.
      // Fractional parts of interval and timeout values will not be considered.
      "heartBeatIntervalSeconds": 300,
      "heartBeatTimeoutMilliSeconds": 3000,
      "heartBeatRequestBaseDn": "",
      "heartBeatRequestFilter": "(&)",

      // The failover load-balancing pools in priority order:
      "failoverLdapServers": [
        // Connect to these servers first:
        [{
            "hostname": "localhost",
            "port": 1389
        }],
        // (Optional) Connect to these servers next:
        [
            // Empty
        ],
        // (Optional) Connect to these servers last:
        [
            // Empty
        ]
      ],
      // When optional failover load-balancing pools are configured,
      // availability checks for the load balancer can be configured as well.
      // The availability check is a search request which is sent periodically
      // on a single connection in order to determine whether the server is
      // healthy or not. The server is considered healthy if the search returns
      // a single entry.
      // Fractional parts of interval and timeout values will not be considered.
      "availabilityCheckIntervalSeconds": 5,
      "availabilityCheckTimeoutMilliSeconds": 3000,
      "availabilityCheckRequestBaseDn": "",
      "availabilityCheckRequestFilter": "(healthy=true)"
```

```
    },

    // Authenticated connections which will be used for searches during
    // authentication and proxied operations (if enabled). This factory
    // will re-use the server "bind" configuration.
    "root": {
      "inheritFrom": "bind",

      // Defines how authentication should be performed. Only simple
      // and SASL/External authentication are supported at the moment.
      // If the OAuth 2.0 authorization policy is configured below,
      // then the directory service must be configured
      // to allow the user configured here to perform proxied authorization.
      "authentication": {
        // The type of LDAP bind request to use, which must be "simple"
        // (the default), "sasl-scram" or "sasl-external".
        "policy": "simple",

        "simple": {
            "bindDn": "uid=admin",
            "bindPassword": "password"
        },

        // SASL SCRAM mechanisms require the user's password
        // to be stored using a compatible storage scheme.
        "sasl-scram": {
            // The SCRAM mechanism name, which defaults to SCRAM-SHA-256.
            "scramMechanism": "SCRAM-SHA-256",
            "bindDn": "uid=admin",
            "bindPassword": "password"
            // Optionally: "scramMinIterations": 2048
            // to reduce the accepted minimum number of iterations.
        }

        // SASL/External does not have any configurable parameters.
      }
    }
  }
}
```

Authorization

The `"authorization"` field sets authorization policies for access to directory data. It has the following top-level fields:

```
{
  "authorization": {
    // Authorization policies to use. One of "anonymous", "basic", or "oauth2".
    "policies": [],
    // Perform all operations using a pre-authorized connection.
    "anonymous": {},
    // Use HTTP Basic authentication's information to bind to the LDAP server.
    "basic": {},
    // Use an OAuth2 authorization method.
    "oauth2": {}
  }
}
```

The `"anonymous"` object has the following settings:

```
{
  "anonymous": {
    // Specify the connection factory to use to perform LDAP operations.
    // If missing, the "root" factory will be used.
    "ldapConnectionFactory": "root"
  }
}
```

The `"basic"` object has the following settings:

```
{
  "basic": {
    // Indicates whether the filter should allow alternative authentication
    // and, if so, which HTTP headers it should obtain the username and
    // password from.
    "supportAltAuthentication": true,
    "altAuthenticationUsernameHeader": "X-OpenIDM-Username",
    "altAuthenticationPasswordHeader": "X-OpenIDM-Password",

    // Define which LDAP bind mechanism to use
    // Supported mechanisms are "simple", "sasl-plain", "sasl-scram", or "search"
    "bind": "search",

    // Bind to the LDAP server using the DN built from the HTTP Basic's username
    "simple": {
      // Connection factory used to perform the bind operation.
      // If missing, "bind" factory will be used.
      "ldapConnectionFactory": "bind",

      // The Bind DN Template containing a single {username},
      // which will be replaced by the authenticating user's name.
      // (For example: uid={username},ou=People,dc=example,dc=com)
      // If missing, "{username}" is used.
      "bindDnTemplate": "uid={username},ou=People,dc=example,dc=com"
    },

    // Bind to the LDAP server using a SASL Plain bind request
    "sasl-plain": {
      // Connection factory used to perform the bind operation.
      // If missing, "bind" factory will be used.
      "ldapConnectionFactory": "bind",

      // Authorization identity template containing a single {username},
      // which will be replaced by the authenticating user's name.
      // (For example: u:{username})
      "authzIdTemplate": "u:{username}"
    },

    // Bind to the LDAP server using a SASL SCRAM bind request.
    // SASL SCRAM mechanisms require the user's password to be stored
    // using a compatible storage scheme.
    "sasl-scram": {
      // Connection factory used to perform the bind operation.
      // If missing, "bind" factory will be used.
      "ldapConnectionFactory": "bind",

      // The SCRAM mechanism name, which defaults to SCRAM-SHA-256.
      "scramMechanism": "SCRAM-SHA-256",

      // Authorization identity template containing a single {username}
      // which will be replaced by the authenticating user's name.
      // (For example: u:{username})
      "authzIdTemplate": "u:{username}"
    },

    // Bind to the LDAP server using the resulting DN of a search request.
    "search": {
      // Connection factory used to perform the search operation.
      // If missing, "root" factory will be used.
      "searchLdapConnectionFactory": "root",
```

```
        // Connection factory used to perform the bind operation.
        // If missing, "bind" factory will be used.
        "bindLdapConnectionFactory": "bind",

        // The {username} filter format parameters will be substituted
        // with the client-provided username,
        // using LDAP filter string character escaping.
        "baseDn": "ou=people,dc=example,dc=com",
        "scope": "sub", // Or "one".
        "filterTemplate": "(&(uid={username})(objectClass=inetOrgPerson))"
      }
    }
  }
```

The `"oauth2"` object has the following settings:

```
{
  "oauth2": {
    // Access tokens associated realm.
    // This attribute is optional and has a string syntax.
    "realm": "myrealm",

    // Defines the list of required scopes required to access the service.
    // This field is required and cannot be empty.
    "requiredScopes": [ "read", "write", "uid" ],

    // Specify the resolver to use to resolve OAuth2 access token.
    // This attribute is required and its value must be one of "openam", "rfc7662", "cts".
    // Note that the JSON object corresponding to this attribute value must be present
    // and well formed in the "oauth2" JSON attribute.
    "resolver": "openam",

    // Configures caching of access token introspection results.
    // This attribute is optional, if it is not present, no token caching
    // will be performed.
    "accessTokenCache": {
      // Indicates whether the access token caching should be used.
      // This attribute is optional (default value is false)
      // and must have a boolean syntax.
      "enabled": false,

      // Specifies the maximal caching duration for an access token.
      // Once this delay is over, token will be refreshed from an access token resolver
      // (see "oauth2/resolver").
      // This attribute is optional, its default value is "5 minutes".
      // The duration syntax supports all human readable notations from day
      // ("days", "day", "d") to nanosecond ("nanoseconds", "nanosecond", "nanosec",
      // "nanos", "nano", "ns")
      // Any negative or zero values are incorrect.
      "cacheExpiration": "5 minutes"
    },

    // The OpenAM access token resolver configuration.
    // This attribute must be present if the "oauth2/resolver" is equal to "openam".
    // If "oauth2/resolver" is set to another resolver, this attribute will be ignored.
    "openam": {
      // Defines the OpenAM endpoint URL where the request should be sent.
      // This attribute is required and must have a string syntax.
      "endpointUrl": "http://openam.example.com:8080/openam/oauth2/tokeninfo",

      // This alias points at an existing certificate that is used for TLS authentication
      // for secure communication between this gateway
      // and the OpenAM access-token resolver.
      "sslCertAlias": "client-cert",

      // The default authzIdTemplate demonstrates how an authorization DN
      // may be constructed from the "uid" field in the following example
      // OpenAM tokeninfo response:
      // {
      //     "scope":["uid"],
      //     "realm":"/",
      //     "expires_in":45,
      //     "uid" : "bjensen",
      // }
      // This attribute is required and has a string syntax.
      // It must start with either 'dn:' or 'u:'.
```

```
      "authzIdTemplate": "dn:uid={uid},ou=People,dc=example,dc=com"
    },

    // The RFC-7662 (see https://www.rfc-editor.org/info/rfc7662)
    // access token resolver configuration.
    // This attribute must be present if the "oauth2/resolver" is equal to "rfc7662".
    // If "oauth2/resolver" is set to another resolver, this attribute will be ignored.
    "rfc7662": {
      // Defines the token introspection endpoint URL where the request should be sent.
      // This attribute is required and must have a string syntax.
      "endpointUrl": "http://openam.example.com:8080/openam/oauth2/myrealm/introspect",

      // This alias points at an existing certificate that is used for TLS authentication
      // for secure communication between this gateway and the introspection
      // access-token resolver.
      "sslCertAlias": "client-cert",

      // Token introspect endpoint requires authentication.
      // It should support HTTP basic authorization
      // (a base64-encoded string of clientId:clientSecret)
      // These attributes are mandatory.
      "clientId": "client_id",
      "clientSecret": "client_secret",

      // The default authzIdTemplate demonstrates how an authorization DN
      // may be constructed from the "username" field in the following example
      // introspect response:
      // {
      //     "active": true,
      //     "token_type": "access_token",
      //     "exp": 3524,
      //     "username" : "bjensen",
      // }
      // This attribute is required and has a string syntax.
      // It must start with either 'dn:' or 'u:'.
      "authzIdTemplate": "dn:uid={username},ou=People,dc=example,dc=com"
    },

    // The CTS access token resolver.
    // This attribute must be present if the "oauth2/resolver" is equal to "cts".
    // If "oauth2/resolver" is set to another resolver, this attribute will be ignored.
    // Note: You can use {userName/0} in authzIdTemplate configuration to access
    // user id from the default CTS access token content config.
    "cts": {
      // The connection factory to use to access CTS.
      // This attribute must reference a connection factory
      // defined in the "ldapConnectionFactories" section.
      // Default value: "root"
      // (That is, the "root" connection factory will be used to access the CTS).
      "ldapConnectionFactory": "root",

      // The access token base DN.
      // This attribute is required and must have a string syntax.
      "baseDn": "ou=famrecords,ou=openam-session,ou=tokens,dc=example,dc=com",

      // The default authzIdTemplate demonstrates how an authorization DN
      // may be constructed from the "userName" field in the following example
      // CTS access token entry:
      // {
      //     "active": true,
      //     "tokenName": ["access_token"],
      //     "exp": [3524],
```

```
        //     "userName" : ["bjensen"],
        // }
        // This attribute is required and has a string syntax.
        // It must start with either 'dn:' or 'u:'.
        "authzIdTemplate": "dn:uid={userName/0},ou=People,dc=example,dc=com"
      },

      // ONLY FOR TESTING: A file-based access token resolver.
      // This attribute must be present if the "oauth2/resolver" is equal to "file".
      // If "oauth2/resolver" is set to another resolver, this attribute will be ignored.
      "file": {
        // Directory containing token files.
        // You can test the rest2ldap OAuth2 authorization support
        // by providing JSON token files under
        // the directory set in the configuration below.
        // File names must be equal to the token strings.
        // The file content must a JSON object with the following attributes:
        // 'scope', 'expireTime' and all the field(s) needed to resolve the authzIdTemplate.
        "folderPath": "/path/to/test/folder",

        // The default authzIdTemplate demonstrates an authorization DN constructed
        // from the "uid" field in a fake token file:
        // {
        //     "scope": ["read", "uid", "write"],
        //     "expireTime": 1961336698000,
        //     "uid": "bjensen"
        // }
        // This attribute is required and has string syntax.
        // It must start with either 'dn:' or 'u:'.
        "authzIdTemplate": "dn:uid={uid},ou=People,dc=example,dc=com"
      }
    }
  }
```

Gateway LDAP features

A single `rest2ldap/rest2ldap.json` file defines the LDAP features that the gateway uses. The settings have the following defaults:

```
{
  "useMvcc": true,
  "mvccAttribute": "etag",
  "readOnUpdatePolicy": "controls",
  "useSubtreeDelete": true,
  "usePermissiveModify": true,
  "useServerSideSortForJson": true,
  "returnNullForMissingProperties": false,
  "localSortMaxEntries" : 1000
}
```

Field	Description
`"useMvcc"`	Whether the gateway supports multi-version concurrency control (MVCC). If `true`, specify the `"mvccAttribute"`.
`"mvccAttribute"`	The LDAP attribute to use for MVCC. This lets a client application check whether this is the correct version of the resource with the header: `If-Match: mvcc-value`
`"readOnUpdatePolicy"`	Specifies the policy for reading an entry after addition, after modification, and before deletion. One of the following: • `"controls"`: Use RFC 4527 read-entry controls to reflect the state of the resource on update. The remote LDAP servers must support RFC 4527. • `"disabled"`: Do not read the entry or return the resource on update. • `"search"`: Perform an LDAP search to retrieve the entry after addition, after modification, and before deletion. The JSON resource returned might differ from the updated LDAP entry.
`"useSubtreeDelete"`	Whether to use the LDAP Subtree Delete request control (OID: `1.2.840.113556.1.4.805`) for LDAP delete operations resulting from delete operations on resources. Client applications deleting resources with children must have access to use the control. When `true`, the gateway attempts to use the control. It falls back to searching for and deleting children if the server rejects the request.
`"usePermissiveModify"`	Whether to use the LDAP Permissive Modify request control (OID: `1.2.840.113556.1.4.1413`) for the LDAP modifications corresponding to PATCH and PUT requests. Set this to `false` when remote LDAP servers do not support the control.
`"useServerSideSortForJson"`	Whether to use the LDAP Server-Side Sort request control (OID: `1.2.840.113556.1.4.473`) to request that the server sort the results before returning them. When `false`, the gateway sorts search results locally. In this case, set `localSortMaxEntries`, which effectively limits the maximum `_pageSize` that the gateway accepts.

Field	Description
`"returnNullForMissingProperties"`	Whether missing (unmapped) JSON properties should be included in JSON resources. By default, a REST to LDAP mapping omits JSON fields for LDAP attributes that have no values.
`"localSortMaxEntries"`	The maximum number of entries supported by the local sort mechanism. When a request includes a `_sortKey` parameter, the gateway does the following. If the `_sortKey` parameter targets: • A normal LDAP attribute, the gateway includes a server-side sort request to the LDAP server. • A JSON syntax LDAP attribute, the action depends on the `useServerSideSortForJson` setting. • An attribute that is in a referenced entry, the gateway sorts the results locally.

API configuration

Mapping files define APIs by defining how JSON resources map to LDAP entries. REST to LDAP lets you define multiple APIs, and multiple versions for each API.

A mapping file name has the form `rest2ldap/endpoints/base-path/root-resource.json`:

- The base-path must match the Rest2ldap endpoint `base-path` setting in the DS server configuration.
- The root-resource matches the root resource name in the API's `"resourceTypes"`.
- Each file defines a single version of the API.

The sample API file `rest2ldap/endpoints/api/example-v1.json` has the following structure:

```
{
  "version": "1.0",         // (Optional) version for this API.
  "resourceTypes": {        // (Required) resources for this API.
    "example-v1": {         // (Required) root resource type. Name must match file basename.
      "subResources": {     // (Required) The base path, /api, is implicit.
        "users": {},        // All resource collections, such as /api/users/ and
        "groups": {}        // /api/groups/ are explicitly defined here.
      }
    },

    // The sample also defines the resource types used by the root's "subResources".
    // This is optional, but critical to readability of the "subResources" definition.
    // Keep resource type names unique to avoid clashes with definitions elsewhere.
    // See the full text of the sample to view how inheritance works.
    "frapi:opendj:rest2ldap:object:1.0": {},    // Parent type of all objects.
    "frapi:opendj:rest2ldap:user:1.0": {},      // Basic user type, parent of
    "frapi:opendj:rest2ldap:posixUser:1.0": {}, // user with uid, gid, home dir.
    "frapi:opendj:rest2ldap:group:1.0": {}      // Basic group type.
  }
}
```

The example API defines the following resource collections:

- `"users"`
- `"groups"`

For reference information, see Subresources.

The example API defines the following resource types:

- `"frapi:opendj:rest2ldap:object:1.0"`
- `"frapi:opendj:rest2ldap:user:1.0"`
- `"frapi:opendj:rest2ldap:posixUser:1.0"`
- `"frapi:opendj:rest2ldap:group:1.0"`

For reference information, see Resource types.

Version

The optional `"version"` field specifies the version of the root resource of the API. For multiple versions of the same API, use multiple mapping files.

Valid version strings include:

- `"*"` (Default, no version specified.)
- `"integer"`
- `"integer.integer"`

Each integer is a positive decimal integer.

A client application requests a specific version by setting the request header:

```
Accept-API-Version: resource=version
```

If more than one version of the API is available, but the client does not specify a version header, REST to LDAP uses the latest version of the API.

Subresources

The `"subResources"` object specifies the API under the current path.

A `"subResources"` object has the following fields, shown with their default values:

```
{
  "name": {
    "type": "collection || singleton",  // Required
    "dnTemplate": "",
    "glueObjectClasses": [],
    "isReadOnly": false,
    "namingStrategy": {},
    "resource": ""
  }
}
```

Field	Description

`"type"`	The type, either `"collection"` or `"singleton"`.
	A collection is a container for other resources that clients can create, read, update, delete, patch, and query. When `"type": "collection"`, the definition has these settings:
	`{ "type": "collection", "namingStrategy": {}, // Required "resource": "", // Required "dnTemplate": "", // Optional "glueObjectClasses": [], // Optional "isReadOnly": false // Optional }`
	A singleton is a resource with no children. When `"type": "singleton"`, the definition has these settings:
	`{ "type": "singleton", "resource": "", // Required "dnTemplate": "", // Optional "isReadOnly": false // Optional }`

Field	Description
`"dnTemplate"`	The relative DN template where the LDAP entries are located. If this is an empty string, the LDAP entries are located directly beneath the parent LDAP entry. DN templates can use variables in braces `{}`. REST to LDAP substitutes DN template variables with values extracted from the URL template.
`"glueObjectClasses"`	One or more LDAP object classes associated with intermediate "glue" entries forming the DN template. Required if the DN template contains one or more RDNs.
`"isReadOnly"`	Whether this resource is read-only.

Field	Description

`"namingStrategy"`	How to map LDAP entry names to JSON resources.
	LDAP entries mapped to JSON resources must be immediate subordinates of the `"baseDn"`.
	REST to LDAP supports the following naming strategies:
	• `"type": "clientDnNaming"`: The RDN and resource ID are both derived from a single user attribute.
	In the following example, the `uid` attribute is the RDN. Its value is the JSON resource ID:
	`{ "namingStrategy": { "type": "clientDnNaming", "dnAttribute": "uid" } }`
	• `"type": "clientNaming"`: The RDN and resource ID are derived from separate user attributes.
	In the following example, the RDN attribute is `uid`. The JSON resource ID is the value of the `mail` attribute:
	`{ "namingStrategy": { "type": "clientNaming", "dnAttribute": "uid", "idAttribute": "mail" } }`
	• `"type": "serverNaming"`: The RDN is derived from a user attribute and the resource ID from an operational attribute.
	In the following example, the RDN attribute is `uid`. The JSON resource ID is the value of the `entryUUID` operational attribute:
	`{ "namingStrategy": { "type": "serverNaming", "dnAttribute": "uid", "idAttribute": "entryUUID" } }`

Field	Description
`"resource"`	The resource type name of the subresource. A collection can contain objects of different types as long as all types inherit from the same super type. In that case, set `resource` to the super type name.

Resource types

The required `"resourceTypes"` object maps resource type names to resource type definitions.

One of the resource type names must match the basename of the mapping file. This resource is referred to as the *root resource* of the API.

You can reuse a resource type name when specifying the same definition in different mapping files. For example, all mapping files might define the abstract base type `frapi:opendj:rest2ldap:object:1.0`, and use this everywhere as the `superType` for other resource types.

If the definitions differ, however, you must use a different resource type name for the changed resource type. For example, if you add a new user type in another API based on `frapi:opendj:rest2ldap:user:1.0`, but with a different definition. If you still use the original resource type in your APIs, you must also use a different name for your new user type.

Examples in the sample mapping file use the namespace prefix `frapi:opendj:rest2ldap:`. You may use any string allowed in JSON.

REST to LDAP does not provide a mechanism for inheriting resource types between mapping files. To reuse a resource type from another mapping file, manually copy its name and definition, taking care to change the name if you change the definition.

A resource type definition object has the following fields. All fields are optional, and have the default values shown here:

```
{
  "unique-name": {
    "properties": {},
    "subResources": {},
    "isAbstract": false,
    "superType": "",
    "objectClasses": [],
    "supportedActions": [],
    "includeAllUserAttributesByDefault": false,
    "excludedDefaultUserAttributes": []
  }
}
```

Field	Description
`"properties"`	Map of property names to property definitions. For details, see Properties.

Field	Description
`"subResources"`	Map of subresource names to subresource definitions. The names are URL path templates, setting the relative path where the subresources are located. URL path templates can set variables in braces `{}`. For example, suppose LDAP entries for devices are under the following base DNs: • `ou=others,ou=devices,dc=example,dc=com` • `ou=pcs,ou=devices,dc=example,dc=com` • `ou=phones,ou=devices,dc=example,dc=com` • `ou=tablets,ou=devices,dc=example,dc=com` You can define a DN template, `ou={type},ou=devices,dc=example,dc=com`. Given the paths relative paths: • `devices/others` • `devices/pcs` • `devices/phones` • `devices/tablets` REST to LDAP substitutes `{type}` with the last path element to resolve the DN template and locate the entries in the correct LDAP organizational unit. For details, see Subresources.
`"isAbstract"`	Whether this is an abstract resource type used only for inheritance.
`"superType"`	The resource type that this resource type extends. Resource types that extend another type inherit properties and subresource definitions. Make sure each `superType` has, at most, one child type. Inheritance in REST to LDAP does not support the equivalent of LDAP AUXILIARY object classes.
`"objectClasses"`	LDAP object classes names for the LDAP entries underlying the JSON resource. On creation, REST to LDAP adds these to the object classes on the LDAP entry. The LDAP object classes are never visible in the JSON resource.

Field	Description
`"supportedActions"`	Names of the DS-specific ForgeRock® Common REST actions that this resource type supports. Action names must match the actions allowed on the resource in the underlying implementation: • `"accountUsability"` : Return account usability information. For an example, see Account usability action. • `"create"` : Create a resource by HTTP POST. This action is implicitly supported. Do not include it in the list. For an example, see Create (HTTP POST). • `"modifyPassword"` : Change one's own password given the old and new password. For an example, see Change your password. • `"resetPassword"` : Reset a user's password to a generated value. For an example, see Reset a password.
`"includeAllUserAttributesByDefault"`	Whether to include all LDAP user attributes as properties of the JSON resource. When `true`, the JSON field names match the LDAP attribute names.
`"excludedDefaultUserAttributes"`	LDAP user attributes to exclude when `"includeAllUserAttributesByDefault": true`.

Properties

The `"properties"` object specifies how the JSON resource maps to the underlying LDAP entries.

A `"properties"` object has the following fields, shown with their default values:

```
{
  "name": {
    "type": "See the list below.", // Required
    "baseDn": "",
    "defaultJsonValue": "",
    "extensibleJsonOrderingMatchingRule": "",
    "isBinary": false,
    "isMultiValued": false,
    "isRequired": false,
    "jsonQueryEqualityMatchingRule": "caseIgnoreJsonQueryMatch",
    "ldapAttribute": "name",
    "mapper": {},
    "optionalJsonPropertyName": "",
    "primaryKey": "",
    "propertyName": "",
    "resourcePath": "",
    "schema": "",
    "searchFilter": "(objectClass=*)",
    "value": "",
    "writability": "readWrite"
  }
}
```

The `"type"` must be one of the following:

- `"constant"`
- `"json"`
- `"object"`
- `"reference"`
- `"reverseReference"`
- `"resourceType"`
- `"simple"`

`"type"`

The type, which determines the fields the object has:

`"constant"`

A fixed value:

```
{
  "name": {
    "type": "constant",
    "value": "A valid JSON value."
  }
}
```

`"json"`

A `Json` syntax LDAP attribute:

```
{
  "type": "json",
  "ldapAttribute": "name", // Required
  "baseDn": "",
  "defaultJsonValue": "",
  "extensibleJsonOrderingMatchingRule": "",
  "isBinary": false,
  "isMultiValued": false,
  "isRequired": false,
  "jsonQueryEqualityMatchingRule": "caseIgnoreJsonQueryMatch",
  "schema": "",
  "writability": "readWrite"
}
```

The `"ldapAttribute"` must be a `Json` syntax LDAP attribute.

`"object"`

An object with its own mapping:

```
{
  "type": "object",
  "properties": {}
}
```

`"reference"`

An LDAP entry found by reference.

This is useful with LDAP attributes that take another entry's DN, such as `manager`, and (group) `member`.

The mapping can define the location of the reference by:

- Resource path, the path to another JSON resource:

```
{
  "type": "reference",
  "resourcePath": "", // Required
  "isMultiValued": false,
  "isRequired": false,
  "ldapAttribute": "name",
  "optionalJsonPropertyName": "",
  "searchFilter": "(objectClass=*)",
  "writability": "readWrite"
}
```

When REST to LDAP reads the reference, it returns the `_id` and `_rev` fields by default.

- Base DN, the LDAP base DN under which to find entries referenced by the JSON resource:

```
{
  "type": "reference",
  "baseDn": "",     // Required
  "mapper": {},     // Required
  "primaryKey": "", // Required
  "isMultiValued": false,
  "isRequired": false,
  "ldapAttribute": "name",
  "optionalJsonPropertyName": "",
  "searchFilter": "(objectClass=*)",
  "writability": "readWrite"
}
```

When REST to LDAP reads the reference, it returns all fields by default.

REST to LDAP follows these rules to determine field types for references:

- If the LDAP attribute is defined in the LDAP schema, then the REST to LDAP mapping uses the most appropriate type in JSON.
  For example, numbers appear as JSON numbers, and booleans as booleans.
- If the LDAP attribute only has one value, it maps to a scalar.
- If the LDAP attribute has multiple values, it maps to an array.

`"reverseReference"`

An LDAP entry that references this one:

```
{
  "type": "reverseReference",
  // Required:
  "resourcePath": "",
  // Required:
  "propertyName": "",
  // Optional:
  "searchFilter": "(objectClass=*)"
}
```

The `"propertyName"` is the name of the property that references this resource.

Field	Description
	When REST to LDAP reads the resource, it does not return reverse references by default. Use the `_fields` parameter to explicitly request reverse reference fields. **`"resourceType"`** The name of a resource type defined elsewhere in this mapping file. **`"simple"`** An LDAP attribute: `{ "type": "simple", "ldapAttribute": "name", // Required "defaultJsonValue": "", "isBinary": false, "isMultiValued": false, "isRequired": false, "writability": "readWrite" }` Use simple mappings where the correspondence between JSON properties and LDAP attributes is one-to-one.
`"baseDn"`	The base LDAP DN where REST to LDAP finds entries referenced by the JSON resource. Base DN values can be literal values, such as `dc=example,dc=com`. Alternatively, they can use the following notation: **`{url-template}`** REST to LDAP replaces this with the literal value used in the request. For example, if the path resource path is `/{tenant}/users`, and the base DN is `ou=people,dc={tenant},dc=com`, a request for `/example/users`, references `ou=people,dc=example,dc=com`. **`".."`** This is like `..` in a path, meaning "the parent directory." Paths are big-endian, whereas DNs are little-endian. To reference a `"resourcePath"`, you write `../../groups`. To reference a `"baseDn"`, you write `ou=groups,..,..`. Notice a limitation in this reference to group member entries: all group members must be people. The configuration does not support nested groups and members in other locations.
`"defaultJsonValue"`	A JSON value to return if no corresponding LDAP attribute is present. To generate a non-static value based on other attributes of the LDAP entry, use Template-based virtual attributes instead.

Field	Description
`"extensibleJsonOrderingMatchingRule"`	The JSON ordering matching rule to use when requesting an extensible server-side sort. The default rule will ignore case and whitespace when sorting values of JSON fields. For a description of the extended server-side sort syntax, see Server-side sort.
`"isBinary"`	Whether the underlying LDAP attribute holds a binary value, such as a JPEG photo or a digital certificate. When `"isBinary": true`, the JSON resource holds the base64-encoded value. For details, see Binary resources.
`"isMultiValued"`	Whether the field can take an array value. Most LDAP attributes are multi-valued. A literal-minded mapping from LDAP to JSON would be full of array properties, many with only one value. To minimize inconvenience, REST to LDAP returns single value scalars by default, even when the underlying LDAP attribute is multi-valued. By default, the JSON resource gets the first value returned for a multi-valued LDAP attribute. When `"isMultiValued": true`, a single value is still returned as a scalar. If the LDAP attribute has multiple values, REST to LDAP returns an array.
`"isRequired"`	Whether the LDAP attribute is mandatory and must be provided to create the resource.
`"jsonQueryEqualityMatchingRule"`	When a query filter in the HTTP request uses a JSON path that points to a field in a JSON attribute value, it uses the matching rule specified by this property to compare the query filter with attribute values: • `"caseIgnoreJsonQueryMatch"`: Ignore case when finding matches (default). • `"caseExactJsonQueryMatch"`: Respect case when finding matches.
`"ldapAttribute"`	The attribute in the LDAP entry underlying the JSON resource. By default, REST to LDAP assumes the JSON field name matches the LDAP attribute name.
`"mapper"`	How the referenced entry content maps to the content of this JSON field. A mapper object can have all the fields described in this table.

Field	Description
`"optionalJsonPropertyName"`	Use this when creating a reference for an attribute that has `NameAndOptionalJSON` syntax. REST to LDAP returns optional JSON from the attribute as the value of a property having the name you specify. For example, if you configure `"optionalJsonPropertyName": "_refProperties"`, then the `"_refProperties"` field of the reference contains the optional JSON. Suppose the LDAP entry contains an attribute with this syntax, `relationship: {"optional": "JSON"}cn=Some relationship,dc=example,dc=com`. The configuration specifies `"optionalJsonPropertyName": "_refProperties"` in the `relationship` reference. The JSON resource has a relationship field like the following: `{ "relationship": { "_id": "Some relationship", "_refProperties": {"optional": "JSON"} } }`
`"primaryKey"`	Indicates which LDAP attribute in the mapper holds the primary key to the referenced entry.
`"propertyName"`	Name of another field of the JSON resource.

`"resourcePath"`	A path to another JSON resource.
	Resource path values use the following notation:
	`{url-template}`
	REST to LDAP replaces this with the literal value used in the request.
	`/path`
	The absolute path to the resource.
	`..`
	The `..` refers to the relative parent path. Use `..` to reference an arbitrary resource in the same collection.
	For example, according to the default sample API, `/api/users/bjensen` refers to a user, and `/api/groups/Directory Administrators` refers to a group. To refer to Babs Jensen's manager from her account, the resource path is to some other user, `..`. To refer to groups Babs belongs to, the resource path is to some group, `../../groups`.
	The following example shows how `..` references a manager entry next to the current entry:
	`{ "manager": { "type": "reference", "resourcePath": ".." } }`
	The following example shows a reference to group member entries:
	`{ "members": { "type": "reference", "resourcePath": "../../users", "ldapAttribute": "uniqueMember", "isMultiValued": true } }`
	Notice a limitation in this reference to group member entries: all group members must be people. The configuration does not handle nested groups and members in other locations.
	The following example uses a resource path to define a manager's reports. All users, managers and their reports, are under `users/`:
	`{ "reports": { "type": "reverseReference", "resourcePath": "..", "propertyName": "manager" } }`

Field	Description
`"schema"`	Specifies a JSON Schema that applies values of type `"json"`. When no schema is specified, REST to LDAP accepts arbitrary JSON values.
`"searchFilter"`	The LDAP filter to use when searching for a referenced entry.
`"value"`	Use with `"type": "constant"` to specify the constant value.
`"writability"`	Whether the mapping supports updates: • `"createOnly"`: This attribute can be set only when the entry is created. Attempts to update this attribute thereafter result in errors. • `"createOnlyDiscardWrites"`: This attribute can be set only when the entry is created. Attempts to update this attribute thereafter do not result in errors. Instead, the update value is discarded. • `"readOnly"`: This attribute cannot be written. Attempts to write this attribute result in errors. • `"readOnlyDiscardWrites"`: This attribute cannot be written. Attempts to write this attribute do not result in errors. Instead, the value to write is discarded. • `"readWrite"`: This attribute can be set at creation and updated thereafter.

Configuration reference

About this reference

This reference describes server configuration settings that you can view and edit with the `dsconfig` command. The `dsconfig` command is the primary tool for managing the server configuration, which follows an object-oriented configuration model. Each configuration object has its own properties. Configuration objects can be related to each other by inheritance and by reference.

The server configuration model exposes a wide range of configurable features. As a consequence, the `dsconfig` command has many subcommands.

Subcommands exist to create, list, and delete configuration objects, and to get and set properties of configuration objects. Their names reflect these five actions:

- `create-` *object*
- `list-` *objects*
- `delete-` *object*
- `get-` *object* `-prop`
- `set-` *object* `-prop`

Each configuration *object* has a user-friendly name, such as `Connection Handler`. Subcommand names use lower-case, hyphenated versions of the friendly names, as in `create-connection-handler`.

Subcommands

Core Server

Administration Connector

- get-administration-connector-prop
- set-administration-connector-prop

Alert Handler

- create-alert-handler
- delete-alert-handler
- get-alert-handler-prop
- list-alert-handlers

- set-alert-handler-prop

Connection Handler

- create-connection-handler
- delete-connection-handler
- get-connection-handler-prop
- list-connection-handlers
- set-connection-handler-prop

Extended Operation Handler

- create-extended-operation-handler
- delete-extended-operation-handler
- get-extended-operation-handler-prop
- list-extended-operation-handlers
- set-extended-operation-handler-prop

Global Configuration

- get-global-configuration-prop
- set-global-configuration-prop

HTTP Endpoint

- create-http-endpoint
- delete-http-endpoint
- get-http-endpoint-prop
- list-http-endpoints
- set-http-endpoint-prop

Plugin Root

- get-plugin-root-prop
- set-plugin-root-prop

Plugin

- create-plugin
- delete-plugin
- get-plugin-prop

- list-plugins
- set-plugin-prop

Root DSE Backend

- get-root-dse-backend-prop
- set-root-dse-backend-prop

Schema Provider

- create-schema-provider
- delete-schema-provider
- get-schema-provider-prop
- list-schema-providers
- set-schema-provider-prop

Virtual Attribute

- create-virtual-attribute
- delete-virtual-attribute
- get-virtual-attribute-prop
- list-virtual-attributes
- set-virtual-attribute-prop

Work Queue

- get-work-queue-prop
- set-work-queue-prop

Caching and Backends

Backend Index

- create-backend-index
- delete-backend-index
- get-backend-index-prop
- list-backend-indexes
- set-backend-index-prop

Backend VLV Index

- create-backend-vlv-index

- delete-backend-vlv-index
- get-backend-vlv-index-prop
- list-backend-vlv-indexes
- set-backend-vlv-index-prop

Backend

- create-backend
- delete-backend
- get-backend-prop
- list-backends
- set-backend-prop

Entry Cache

- create-entry-cache
- delete-entry-cache
- get-entry-cache-prop
- list-entry-caches
- set-entry-cache-prop

Root DSE Backend

- get-root-dse-backend-prop
- set-root-dse-backend-prop

Logging

Access Log Filtering Criteria

- create-access-log-filtering-criteria
- delete-access-log-filtering-criteria
- get-access-log-filtering-criteria-prop
- list-access-log-filtering-criteria
- set-access-log-filtering-criteria-prop

Debug Target

- create-debug-target
- delete-debug-target

- get-debug-target-prop
- list-debug-targets
- set-debug-target-prop

Log Publisher

- create-log-publisher
- delete-log-publisher
- get-log-publisher-prop
- list-log-publishers
- set-log-publisher-prop

Log Retention Policy

- create-log-retention-policy
- delete-log-retention-policy
- get-log-retention-policy-prop
- list-log-retention-policies
- set-log-retention-policy-prop

Log Rotation Policy

- create-log-rotation-policy
- delete-log-rotation-policy
- get-log-rotation-policy-prop
- list-log-rotation-policies
- set-log-rotation-policy-prop

Directory Proxy

Service Discovery Mechanism

- create-service-discovery-mechanism
- delete-service-discovery-mechanism
- get-service-discovery-mechanism-prop
- list-service-discovery-mechanisms
- set-service-discovery-mechanism-prop

Replication

Replication Domain

- create-replication-domain
- delete-replication-domain
- get-replication-domain-prop
- list-replication-domains
- set-replication-domain-prop

Replication Server

- create-replication-server
- delete-replication-server
- get-replication-server-prop
- list-replication-server
- set-replication-server-prop

Synchronization Provider

- create-synchronization-provider
- delete-synchronization-provider
- get-synchronization-provider-prop
- list-synchronization-providers
- set-synchronization-provider-prop

Authentication and Authorization

Access Control Handler

- create-access-control-handler
- delete-access-control-handler
- get-access-control-handler-prop
- list-access-control-handler
- set-access-control-handler-prop

Certificate Mapper

- create-certificate-mapper

- delete-certificate-mapper
- get-certificate-mapper-prop
- list-certificate-mappers
- set-certificate-mapper-prop

Crypto Manager

- get-crypto-manager-prop
- set-crypto-manager-prop

Global Access Control Policy

- create-global-access-control-policy
- delete-global-access-control-policy
- get-global-access-control-policy-prop
- list-global-access-control-policies
- set-global-access-control-policy-prop

HTTP Authorization Mechanism

- create-http-authorization-mechanism
- delete-http-authorization-mechanism
- get-http-authorization-mechanism-prop
- list-http-authorization-mechanisms
- set-http-authorization-mechanism-prop

Identity Mapper

- create-identity-mapper
- delete-identity-mapper
- get-identity-mapper-prop
- list-identity-mappers
- set-identity-mapper-prop

Key Manager Provider

- create-key-manager-provider
- delete-key-manager-provider
- get-key-manager-provider-prop

- list-key-manager-providers
- set-key-manager-provider-prop

Password Policy

- create-password-policy
- delete-password-policy
- get-password-policy-prop
- list-password-policies
- set-password-policy-prop

SASL Mechanism Handler

- create-sasl-mechanism-handler
- delete-sasl-mechanism-handler
- get-sasl-mechanism-handler-prop
- list-sasl-mechanism-handlers
- set-sasl-mechanism-handler-prop

Trust Manager Provider

- create-trust-manager-provider
- delete-trust-manager-provider
- get-trust-manager-provider-prop
- list-trust-manager-providers
- set-trust-manager-provider-prop

Service Discovery Mechanism

Service Discovery Mechanism

- create-service-discovery-mechanism
- delete-service-discovery-mechanism
- get-service-discovery-mechanism-prop
- list-service-discovery-mechanisms
- set-service-discovery-mechanism-prop

User Management

Account Status Notification Handler

- create-account-status-notification-handler
- delete-account-status-notification-handler
- get-account-status-notification-handler-prop
- list-account-status-notification-handlers
- set-account-status-notification-handler-prop

Certificate Mapper

- create-certificate-mapper
- delete-certificate-mapper
- get-certificate-mapper-prop
- list-certificate-mappers
- set-certificate-mapper-prop

Identity Mapper

- create-identity-mapper
- delete-identity-mapper
- get-identity-mapper-prop
- list-identity-mappers
- set-identity-mapper-prop

Password Generator

- create-password-generator
- delete-password-generator
- get-password-generator-prop
- list-password-generators
- set-password-generator-prop

Password Policy

- create-password-policy
- delete-password-policy
- get-password-policy-prop

- list-password-policies
- set-password-policy-prop

Password Storage Scheme

- create-password-storage-scheme
- delete-password-storage-scheme
- get-password-storage-scheme-prop
- list-password-storage-schemes
- set-password-storage-scheme-prop

Password Validator

- create-password-validator
- delete-password-validator
- get-password-validator-prop
- list-password-validators
- set-password-validator-prop

Help

list-properties

Objects

Core Server

- Administration Connector
- Alert Handler
  - JMX Alert Handler
  - SMTP Alert Handler
- Connection Handler
  - HTTP Connection Handler
  - JMX Connection Handler
  - LDAP Connection Handler
  - LDIF Connection Handler

Caching and Backends

- Backend
  - Local Backend
    - LDIF Backend
    - Memory Backend
    - Monitor Backend
    - Null Backend
    - Pluggable Backend
      - JE Backend
    - Schema Backend
    - Task Backend
  - Proxy Backend
- Backend Index
- Backend VLV Index
- Entry Cache
  - FIFO Entry Cache
  - Soft Reference Entry Cache
- Root DSE Backend

Logging

- Access Log Filtering Criteria
- Debug Target
- Log Publisher
  - Access Log Publisher
    - Common Audit Access Log Publisher
      - CSV File Access Log Publisher
      - External Access Log Publisher
      - JSON File Based Access Log Publisher
    - File Based Access Log Publisher
    - File Based Audit Log Publisher

Authentication and Authorization

- Access Control Handler
  - DSEE Compatible Access Control Handler
  - Policy Based Access Control Handler
- Certificate Mapper
  - Fingerprint Certificate Mapper
  - Subject Attribute To User Attribute Certificate Mapper
  - Subject DN To User Attribute Certificate Mapper
  - Subject Equals DN Certificate Mapper
- Crypto Manager
- Global Access Control Policy
- HTTP Authorization Mechanism
  - HTTP Anonymous Authorization Mechanism
  - HTTP Basic Authorization Mechanism
  - HTTP OAuth2 Authorization Mechanism
    - HTTP OAuth2 CTS Authorization Mechanism
    - HTTP OAuth2 File Based Authorization Mechanism
    - HTTP OAuth2 OpenAM Authorization Mechanism
    - HTTP OAuth2 Token Introspection (RFC 7662) Authorization Mechanism
- Identity Mapper
  - Exact Match Identity Mapper
  - Regular Expression Identity Mapper
- Key Manager Provider
  - File Based Key Manager Provider
  - LDAP Key Manager Provider
  - Pem Key Manager Provider
  - PKCS#11 Key Manager Provider
- SASL Mechanism Handler
  - Anonymous SASL Mechanism Handler
  - CRAM-MD5 SASL Mechanism Handler

Service Discovery Mechanism

User Management

- Identity Mapper
  - Exact Match Identity Mapper
  - Regular Expression Identity Mapper
- Password Generator
  - Random Password Generator
- Password Storage Scheme
  - AES Password Storage Scheme
  - Argon2 Password Storage Scheme
  - Base64 Password Storage Scheme
  - Bcrypt Password Storage Scheme
  - Blowfish Password Storage Scheme
  - Clear Password Storage Scheme
  - Crypt Password Storage Scheme
  - MD5 Password Storage Scheme
  - PBKDF2 Password Storage Scheme
    - PBKDF2-HMAC-SHA256 Password Storage Scheme
    - PBKDF2-HMAC-SHA512 Password Storage Scheme
  - PKCS#5 V2.0 Scheme 2 Password Storage Scheme
  - RC4 Password Storage Scheme
  - Salted MD5 Password Storage Scheme
  - Salted SHA-1 Password Storage Scheme
  - Salted SHA-256 Password Storage Scheme
  - Salted SHA-384 Password Storage Scheme
  - Salted SHA-512 Password Storage Scheme
  - SCRAM-SHA-256 Password Storage Scheme
  - SCRAM-SHA-512 Password Storage Scheme
  - SHA-1 Password Storage Scheme
  - Triple-DES Password Storage Scheme
- Password Validator
  - Attribute Value Password Validator

- Character Set Password Validator
- Dictionary Password Validator
- Length Based Password Validator
- Repeated Characters Password Validator
- Similarity Based Password Validator
- Unique Characters Password Validator

Properties index

A

B

C

changelog-enabled [Replication Server]

changelog-enabled-excluded-domains [Replication Server]

changetime-heartbeat-interval [Replication Synchronization Provider]

character-set [Character Set Password Validator]

character-set-ranges [Character Set Password Validator]

check-references [Referential Integrity Plugin]

check-references-filter-criteria [Referential Integrity Plugin]

check-references-scope-criteria [Referential Integrity Plugin]

check-schema [Global Configuration]

check-substrings [Attribute Value Password Validator]

check-substrings [Dictionary Password Validator]

checksum-algorithm [Entity Tag Virtual Attribute]

cipher-key-length [Crypto Manager]

cipher-key-length [Pluggable Backend]

cipher-key-length [Replication Server]

cipher-transformation [Crypto Manager]

cipher-transformation [Pluggable Backend]

cipher-transformation [Replication Server]

client-id [HTTP OAuth2 Token Introspection (RFC 7662) Authorization Mechanism]

client-secret [HTTP OAuth2 Token Introspection (RFC 7662) Authorization Mechanism]

community [SNMP Connection Handler]

compact-encoding [Pluggable Backend]

confidentiality-enabled [Backend Index]

confidentiality-enabled [Pluggable Backend]

confidentiality-enabled [Replication Server]

config-directory [Rest2LDAP Endpoint]

config-file [External Access Log Publisher]

config-file [External HTTP Access Log Publisher]

conflict-behavior [Collective Attribute Subentries Virtual Attribute]

conflict-behavior [Entity Tag Virtual Attribute]

conflict-behavior [entryDN Virtual Attribute]

conflict-behavior [entryUUID Virtual Attribute]

conflict-behavior [Governing Structure Rule Virtual Attribute]

conflict-behavior [Has Subordinates Virtual Attribute]

conflict-behavior [Is Member Of Virtual Attribute]

conflict-behavior [Member Virtual Attribute]

conflict-behavior [Num Subordinates Virtual Attribute]

conflict-behavior [Password Expiration Time Virtual Attribute]

conflict-behavior [Password Policy Subentry Virtual Attribute]

conflict-behavior [Structural Object Class Virtual Attribute]

conflict-behavior [Subschema Subentry Virtual Attribute]

conflict-behavior [Virtual Attribute]

connection-client-address-equal-to [Access Log Filtering Criteria]

connection-client-address-equal-to [Global Access Control Policy]

connection-client-address-not-equal-to [Access Log Filtering Criteria]

connection-client-address-not-equal-to [Global Access Control Policy]

connection-minimum-ssf [Global Access Control Policy]

connection-pool-idle-timeout [Proxy Backend]

connection-pool-max-size [Proxy Backend]

connection-pool-min-size [Proxy Backend]

connection-port-equal-to [Access Log Filtering Criteria]

connection-port-equal-to [Global Access Control Policy]

connection-protocol-equal-to [Access Log Filtering Criteria]

connection-protocol-equal-to [Global Access Control Policy]

connection-timeout [LDAP Pass Through Authentication Policy]

connection-timeout [Proxy Backend]

connection-timeout [Replication Synchronization Provider]

crypt-password-storage-encryption-algorithm [Crypt Password Storage Scheme]

csv-delimiter-char [CSV File Access Log Publisher]

csv-delimiter-char [CSV File HTTP Access Log Publisher]

D

E

F

G

H

I

J

java-class [Bcrypt Password Storage Scheme]

java-class [Blind Trust Manager Provider]

java-class [Blowfish Password Storage Scheme]

java-class [Cancel Extended Operation Handler]

java-class [Certificate Mapper]

java-class [Change Number Control Plugin]

java-class [Character Set Password Validator]

java-class [Clear Password Storage Scheme]

java-class [Collective Attribute Subentries Virtual Attribute]

java-class [Connection Handler]

java-class [Console Error Log Publisher]

java-class [Core Schema]

java-class [CRAM-MD5 SASL Mechanism Handler]

java-class [Common REST Metrics HTTP Endpoint]

java-class [Crypt Password Storage Scheme]

java-class [CSV File Access Log Publisher]

java-class [CSV File HTTP Access Log Publisher]

java-class [Debug Log Publisher]

java-class [Dictionary Password Validator]

java-class [DIGEST-MD5 SASL Mechanism Handler]

java-class [DSEE Compatible Access Control Handler]

java-class [ETag Plugin]

java-class [Entity Tag Virtual Attribute]

java-class [Entry Cache]

java-class [entryDN Virtual Attribute]

java-class [entryUUID Plugin]

java-class [entryUUID Virtual Attribute]

java-class [Error Log Account Status Notification Handler]

java-class [Error Log Publisher]

java-class [Exact Match Identity Mapper]

java-class [Extended Operation Handler]

java-class [External Access Log Publisher]

java-class [External HTTP Access Log Publisher]

java-class [External SASL Mechanism Handler]

java-class [FIFO Entry Cache]

java-class [File Based Access Log Publisher]

java-class [File Based Audit Log Publisher]

java-class [File Based Debug Log Publisher]

java-class [File Based Error Log Publisher]

java-class [File Based HTTP Access Log Publisher]

java-class [File Based Key Manager Provider]

java-class [File Based Trust Manager Provider]

java-class [File Count Log Retention Policy]

java-class [Fingerprint Certificate Mapper]

java-class [Fixed Time Log Rotation Policy]

java-class [Free Disk Space Log Retention Policy]

java-class [Get Connection ID Extended Operation Handler]

java-class [Get Symmetric Key Extended Operation Handler]

java-class [Governing Structure Rule Virtual Attribute]

java-class [Graphite Monitor Reporter Plugin]

java-class [GSSAPI SASL Mechanism Handler]

java-class [Has Subordinates Virtual Attribute]

java-class [Healthy HTTP endpoint]

java-class [HTTP Access Log Publisher]

java-class [HTTP Anonymous Authorization Mechanism]

java-class [HTTP Authorization Mechanism]

java-class [HTTP Basic Authorization Mechanism]

java-class [HTTP Connection Handler]

java-class [HTTP Endpoint]

java-class [HTTP OAuth2 CTS Authorization Mechanism]

java-class [HTTP OAuth2 File Based Authorization Mechanism]

java-class [HTTP OAuth2 OpenAM Authorization Mechanism]

java-class [HTTP OAuth2 Token Introspection (RFC 7662) Authorization Mechanism]

java-class [Identity Mapper]

java-class [Is Member Of Virtual Attribute]

java-class [JE Backend]

java-class [JMX Alert Handler]

java-class [JMX Connection Handler]

java-class [JSON Equality Matching Rule]

java-class [JSON File Based Access Log Publisher]

java-class [JSON File Based HTTP Access Log Publisher]

java-class [JSON Ordering Matching Rule]

java-class [JSON Query Equality Matching Rule]

java-class [Key Manager Provider]

java-class [Last Mod Plugin]

java-class [LDAP Attribute Description List Plugin]

java-class [LDAP Connection Handler]

java-class [LDAP Key Manager Provider]

java-class [LDAP Pass Through Authentication Policy]

java-class [LDAP Trust Manager Provider]

java-class [LDIF Backend]

java-class [LDIF Connection Handler]

java-class [Length Based Password Validator]

java-class [Log Publisher]

java-class [Log Retention Policy]

java-class [Log Rotation Policy]

java-class [MD5 Password Storage Scheme]

java-class [Member Virtual Attribute]

java-class [Memory Backend]

java-class [Monitor Backend]

java-class [Null Backend]

java-class [Num Subordinates Virtual Attribute]

java-class [Password Expiration Time Virtual Attribute]

java-class [Password Generator]

java-class [Password Modify Extended Operation Handler]

java-class [Password Policy Import Plugin]

java-class [Password Policy State Extended Operation Handler]

java-class [Password Policy Subentry Virtual Attribute]

java-class [Password Policy]

java-class [Password Storage Scheme]

java-class [Password Validator]

java-class [PBKDF2-HMAC-SHA256 Password Storage Scheme]

java-class [PBKDF2-HMAC-SHA512 Password Storage Scheme]

java-class [PBKDF2 Password Storage Scheme]

java-class [Pem Key Manager Provider]

java-class [Pem Trust Manager Provider]

java-class [PKCS#11 Key Manager Provider]

java-class [PKCS#11 Trust Manager Provider]

java-class [PKCS#5 V2.0 Scheme 2 Password Storage Scheme]

java-class [Plain SASL Mechanism Handler]

java-class [Plugin]

java-class [Policy Based Access Control Handler]

java-class [Prometheus HTTP Endpoint]

java-class [Proxy Backend]

java-class [Random Password Generator]

java-class [RC4 Password Storage Scheme]

java-class [Referential Integrity Plugin]

java-class [Regular Expression Identity Mapper]

java-class [Repeated Characters Password Validator]

java-class [Replication Service Discovery Mechanism]

java-class [Replication Synchronization Provider]

java-class [Rest2LDAP Endpoint]

java-class [Salted MD5 Password Storage Scheme]

java-class [Salted SHA-1 Password Storage Scheme]

java-class [Salted SHA-256 Password Storage Scheme]

java-class [Salted SHA-384 Password Storage Scheme]

java-class [Salted SHA-512 Password Storage Scheme]

java-class [Samba Password Plugin]

java-class [SASL Mechanism Handler]

java-class [Schema Backend]

java-class [Schema Provider]

java-class [SCRAM-SHA-256 Password Storage Scheme]

java-class [SCRAM-SHA-256 SASL Mechanism Handler]

java-class [SCRAM-SHA-512 Password Storage Scheme]

java-class [SCRAM-SHA-512 SASL Mechanism Handler]

java-class [Service Discovery Mechanism]

java-class [Seven Bit Clean Plugin]

java-class [SHA-1 Password Storage Scheme]

java-class [Similarity Based Password Validator]

java-class [Size Limit Log Retention Policy]

java-class [Size Limit Log Rotation Policy]

java-class [SMTP Account Status Notification Handler]

java-class [SMTP Alert Handler]

java-class [SNMP Connection Handler]

java-class [Soft Reference Entry Cache]

java-class [StartTLS Extended Operation Handler]

java-class [Static Service Discovery Mechanism]

java-class [Structural Object Class Virtual Attribute]

java-class [Subject Attribute To User Attribute Certificate Mapper]

java-class [Subject DN To User Attribute Certificate Mapper]

K

L

log-field-whitelist [External HTTP Access Log Publisher]

log-field-whitelist [JSON File Based HTTP Access Log Publisher]

log-file [File Based Access Log Publisher]

log-file [File Based Audit Log Publisher]

log-file [File Based Debug Log Publisher]

log-file [File Based Error Log Publisher]

log-file [File Based HTTP Access Log Publisher]

log-file [Referential Integrity Plugin]

log-file-name-prefix [CSV File Access Log Publisher]

log-file-name-prefix [CSV File HTTP Access Log Publisher]

log-file-name-prefix [JSON File Based Access Log Publisher]

log-file-name-prefix [JSON File Based HTTP Access Log Publisher]

log-file-permissions [File Based Access Log Publisher]

log-file-permissions [File Based Audit Log Publisher]

log-file-permissions [File Based Debug Log Publisher]

log-file-permissions [File Based Error Log Publisher]

log-file-permissions [File Based HTTP Access Log Publisher]

log-format [File Based Access Log Publisher]

log-format [File Based HTTP Access Log Publisher]

log-record-time-format [File Based Access Log Publisher]

log-record-time-format [File Based HTTP Access Log Publisher]

log-record-type [Access Log Filtering Criteria]

M

mac-algorithm [Crypto Manager]

mac-key-length [Crypto Manager]

mapped-attribute [LDAP Pass Through Authentication Policy]

mapped-search-base-dn [LDAP Pass Through Authentication Policy]

mapped-search-bind-dn [LDAP Pass Through Authentication Policy]

mapped-search-bind-password [LDAP Pass Through Authentication Policy]

N

O

P

password-attribute [Password Policy]

password-change-requires-current-password [Password Policy]

password-character-set [Random Password Generator]

password-expiration-warning-interval [Password Policy]

password-format [Random Password Generator]

password-generator [Password Policy]

password-history-count [Password Policy]

password-history-duration [Password Policy]

password-validator [Password Policy]

pbkdf2-iterations [PBKDF2 Password Storage Scheme]

pem-directory [Pem Key Manager Provider]

pem-directory [Pem Trust Manager Provider]

permission [Global Access Control Policy]

plugin-order-intermediate-response [Plugin Root]

plugin-order-ldif-import [Plugin Root]

plugin-order-ldif-import-begin [Plugin Root]

plugin-order-ldif-import-end [Plugin Root]

plugin-order-post-connect [Plugin Root]

plugin-order-post-disconnect [Plugin Root]

plugin-order-post-operation-abandon [Plugin Root]

plugin-order-post-operation-add [Plugin Root]

plugin-order-post-operation-bind [Plugin Root]

plugin-order-post-operation-compare [Plugin Root]

plugin-order-post-operation-delete [Plugin Root]

plugin-order-post-operation-extended [Plugin Root]

plugin-order-post-operation-modify [Plugin Root]

plugin-order-post-operation-modify-dn [Plugin Root]

plugin-order-post-operation-search [Plugin Root]

plugin-order-post-operation-unbind [Plugin Root]

plugin-order-post-response-add [Plugin Root]

plugin-order-post-response-bind [Plugin Root]

plugin-order-post-response-compare [Plugin Root]

plugin-order-post-response-delete [Plugin Root]

plugin-order-post-response-extended [Plugin Root]

plugin-order-post-response-modify [Plugin Root]

plugin-order-post-response-modify-dn [Plugin Root]

plugin-order-post-response-search [Plugin Root]

plugin-order-post-synchronization-add [Plugin Root]

plugin-order-post-synchronization-delete [Plugin Root]

plugin-order-post-synchronization-modify [Plugin Root]

plugin-order-post-synchronization-modify-dn [Plugin Root]

plugin-order-pre-operation-add [Plugin Root]

plugin-order-pre-operation-bind [Plugin Root]

plugin-order-pre-operation-compare [Plugin Root]

plugin-order-pre-operation-delete [Plugin Root]

plugin-order-pre-operation-extended [Plugin Root]

plugin-order-pre-operation-modify [Plugin Root]

plugin-order-pre-operation-modify-dn [Plugin Root]

plugin-order-pre-operation-search [Plugin Root]

plugin-order-pre-parse-abandon [Plugin Root]

plugin-order-pre-parse-add [Plugin Root]

plugin-order-pre-parse-bind [Plugin Root]

plugin-order-pre-parse-compare [Plugin Root]

plugin-order-pre-parse-delete [Plugin Root]

plugin-order-pre-parse-extended [Plugin Root]

plugin-order-pre-parse-modify [Plugin Root]

plugin-order-pre-parse-modify-dn [Plugin Root]

plugin-order-pre-parse-search [Plugin Root]

plugin-order-pre-parse-unbind [Plugin Root]

plugin-order-search-result-entry [Plugin Root]

Q

R

request-target-dn-equal-to [Access Log Filtering Criteria]

request-target-dn-equal-to [Global Access Control Policy]

request-target-dn-equal-to-user-dn [Global Access Control Policy]

request-target-dn-not-equal-to [Access Log Filtering Criteria]

request-target-dn-not-equal-to [Global Access Control Policy]

require-change-by-time [Password Policy]

require-secure-authentication [Password Policy]

require-secure-password-changes [Password Policy]

required-scope [HTTP OAuth2 Authorization Mechanism]

response-entry-size-greater-than [Access Log Filtering Criteria]

response-etime-greater-than [Access Log Filtering Criteria]

response-etime-less-than [Access Log Filtering Criteria]

response-result-code-equal-to [Access Log Filtering Criteria]

response-result-code-not-equal-to [Access Log Filtering Criteria]

restricted-client [Administration Connector]

restricted-client [Connection Handler]

restricted-client [Global Configuration]

restricted-client-connection-limit [Administration Connector]

restricted-client-connection-limit [Connection Handler]

restricted-client-connection-limit [Global Configuration]

retention-policy [CSV File Access Log Publisher]

retention-policy [CSV File HTTP Access Log Publisher]

retention-policy [File Based Access Log Publisher]

retention-policy [File Based Audit Log Publisher]

retention-policy [File Based Debug Log Publisher]

retention-policy [File Based Error Log Publisher]

retention-policy [File Based HTTP Access Log Publisher]

retention-policy [JSON File Based Access Log Publisher]

retention-policy [JSON File Based HTTP Access Log Publisher]

return-bind-error-messages [Global Configuration]

S

ssl-cert-nickname [Replication Synchronization Provider]

ssl-cert-nickname [Static Service Discovery Mechanism]

ssl-cipher-suite [Administration Connector]

ssl-cipher-suite [HTTP Connection Handler]

ssl-cipher-suite [HTTP OAuth2 OpenAM Authorization Mechanism]

ssl-cipher-suite [HTTP OAuth2 Token Introspection (RFC 7662) Authorization Mechanism]

ssl-cipher-suite [LDAP Connection Handler]

ssl-cipher-suite [LDAP Pass Through Authentication Policy]

ssl-cipher-suite [Replication Service Discovery Mechanism]

ssl-cipher-suite [Replication Synchronization Provider]

ssl-cipher-suite [Static Service Discovery Mechanism]

ssl-client-auth-policy [HTTP Connection Handler]

ssl-client-auth-policy [LDAP Connection Handler]

ssl-encryption [Replication Synchronization Provider]

ssl-protocol [Administration Connector]

ssl-protocol [HTTP Connection Handler]

ssl-protocol [HTTP OAuth2 OpenAM Authorization Mechanism]

ssl-protocol [HTTP OAuth2 Token Introspection (RFC 7662) Authorization Mechanism]

ssl-protocol [LDAP Connection Handler]

ssl-protocol [LDAP Pass Through Authentication Policy]

ssl-protocol [Replication Service Discovery Mechanism]

ssl-protocol [Replication Synchronization Provider]

ssl-protocol [Static Service Discovery Mechanism]

state-update-failure-policy [Password Policy]

strict-format-boolean [Core Schema]

strict-format-certificates [Core Schema]

strict-format-country-string [Core Schema]

strict-format-jpeg-photos [Core Schema]

strict-format-telephone-numbers [Core Schema]

strip-syntax-min-upper-bound-attribute-type-description [Core Schema]

T

U

V

W

writability-mode [Memory Backend]

writability-mode [Monitor Backend]

writability-mode [Null Backend]

writability-mode [Pluggable Backend]

writability-mode [Schema Backend]

writability-mode [Task Backend]

Duration syntax

Durations are specified with positive integers and unit specifiers.

Unit specifiers include the following:

- `ms` : milliseconds
- `s` : seconds
- `m` : minutes
- `h` : hours
- `d` : days
- `w` : weeks

A duration of 1 week is specified as `1w`. A duration of 1 week, 1 day, 1 hour, 1 minute, and 1 second is specified as `1w1d1h1m1s`.

Whitespace surrounding the value and the unit specifier is not significant. For example, `"5d"` is equivalent to `" 5 d "`.

Not all properties taking a duration allow all unit specifiers. For example, milliseconds are not allowed if durations smaller than one second are not permitted.

Some properties limit minimum or maximum durations.

An unlimited duration is specified using `unlimited` (recommended for readability) or `-1`.

Size syntax

Sizes are specified with non-negative integers and unit specifiers, which are not case-sensitive.

Unit specifiers include the following:

- `b` , `bytes`
- `kb` , `kilobytes` (x1000)
- `kib` , `kibibytes` (x1024)
- `mb` , `megabytes` (x1000x1000)

- `mib`, `mebibytes` (x1024x1024)
- `gb`, `gigabytes` (x1000x1000x1000)
- `gib`, `gibibytes` (x1024x1024x1024)
- `tb`, `terabytes` (x1000x1000x1000x1000)
- `tib`, `tebibytes` (x1024x1024x1024x1024)
- `unlimited`, `-1` (if allowed, explicitly set no upper limit)

For example, you can specify a size of 1,000,000 bytes as `1MB`. To specify a size of 1,048,576 bytes, use `1MiB` or `1 mib`, for example.

Whitespace surrounding the value and the unit specifier is not significant. For example, `"5gb"` is equivalent to `" 5gb "`.

Some properties limit minimum or maximum sizes.

Property value substitution

Property value substitution enables you to achieve the following:

- Define a configuration that is specific to a single instance.

  For example, set the location of the keystore on a particular host.

- Define a configuration whose parameters vary between different environments.

  For example, change hostnames for test, development, and production environments.

Disable certain capabilities on specific servers. For example, disable a database backend and its replication agreement for one set of replicas while enabling it on another set of replicas. This makes it possible to use the same configuration for environments with different data sets.

Property value substitution uses *configuration expressions* to introduce variables into the server configuration. You set configuration expressions as the values of configuration properties. The effective property values can be evaluated in a number of ways.

> **Note**
>
> Scope of configuration expressions
> DS servers only resolve configuration expressions in the `config/config.ldif` file on LDAP attributes whose names start with `ds-cfg-*`. These correspond to configuration properties listed in this reference.
> DS servers do not resolve configuration expressions anywhere else.
> DS servers resolve expressions at startup to determine the configuration. DS commands that read the configuration in offline mode also resolve expressions at startup. When you use expressions in the configuration, you must make their values available before starting the server and also when running such commands.

Configuration expressions share their syntax and underlying implementation with other platform software. Configuration expressions have the following characteristics:

- To distinguish them from static values, expression tokens are preceded by an ampersand and enclosed in braces.

For example: `&{listen.port}`. The expression token in the example is `listen.port`. The `.` serves as the separator character.

- You can use a default value in an expression by including it after a vertical bar following the token.

  For example, the following expression sets the default listen port value to 1389: `&{listen.port|1389}`.

- A configuration property can include a mix of static values and expressions.

  For example, if `hostname` is set to `directory`, `&{hostname}.example.com` evaluates to `directory.example.com`.

- You can define nested properties (that is, a property definition within another property definition).

  For example, if `listen.port` is set to `&{port.prefix}389`, and `port.prefix` is set to `2`, `&{listen.port}` evaluates to `2389`.

- You can read the value of an expression token from a file.

  For example, if the plaintext password is stored in `/path/to/password.txt`, the following expression resolves to the plaintext password: `&{file:/path/to/password.txt}`.

  You specify the file either by its absolute path, or by a path relative to the DS instance directory. In other words, if the DS instance directory is `/path/to/opendj`, then `/path/to/opendj/config/keystore` and `config/keystore` reference the same file.

DS servers define the following expression tokens by default. You can use these in expressions without explicitly setting their values beforehand:

`ds.instance.dir`

The file system directory holding the instance files required to run an instance of a server.

By default, the files are co-located with the product tools, libraries, and configuration files. You can change the location by using the `setup --instancePath` option.

This evaluates to a directory, such as `/path/to/my-instance`.

`ds.install.dir`

The file system directory where the server files are installed.

This evaluates to a directory, such as `/path/to/opendj`.

Expression evaluation and order of precedence

You must define expression values before starting the DS server that uses them. When evaluated, an expression must return the appropriate type for the configuration property. For example, the `listen-port` property takes an integer. If you set it using an expression, the result of the evaluated expression must be an integer. If the type is wrong, the server fails to start due to a syntax error.

If the expression cannot be resolved, and there is no default value in the configuration expression, DS also fails to start.

Expression resolvers evaluate expression tokens to literal values.

Expression resolvers get values from the following sources:

Environment variables

You set an environment variable to hold the value. For example: `export LISTEN_PORT=1389`.

The environment variable name must be composed of uppercase characters and underscores. The name maps to the expression token as follows:

- Uppercase characters are lower cased.
- Underscores, `_`, are replaced with `.` characters.

In other words, the value of `LISTEN_PORT` replaces `&{listen.port}` in the server configuration.

Java system properties

You set a Java system property to hold the value. Java system property names must match expression tokens exactly. In other words, the value of the `listen.port` system property replaces `&{listen.port}` in the server configuration.

Java system properties can be set in a number of ways. One way of setting system properties for DS servers is to pass them through the `OPENDJ_JAVA_ARGS` environment variable. For example: `export OPENDJ_JAVA_ARGS="-Dlisten.port=1389"`.

Expressions files

You set a key in a `.json` or `.properties` file to hold the value.

This optional mechanism is set using the `DS_ENVCONFIG_DIRS` environment variable, or the `ds.envconfig.dirs` Java system property.

Keys in `.properties` files must match expression tokens exactly. In other words, the value of the `listen.port` key replaces `&{listen.port}` in the server configuration. The following example properties file sets the listen port:

```
listen.port=1389
```

JSON expression files can contain nested objects. JSON field names map to expression tokens as follows:

- The JSON path name matches the expression token.
- The `.` character serves as the JSON path separator character.

The following example JSON file sets the listen address and listen port:

```
{
    "listen": {
      "address": "192.168.0.10",
      "port": "1389"
    }
}
```

In other words, the value of the `listen/port` field replaces `&{listen.port}` in the server configuration.

In order to use expression files, set the environment variable, `DS_ENVCONFIG_DIRS`, or the Java system property, `ds.envconfig.dirs`, to a comma-separated list of the directories containing the expression files.

The following constraints apply when using expression files:

- Although DS browses the directories in the specified order, within a directory DS scans the files in a non-deterministic order.
- DS reads all files with `.json` and `.properties` extensions.
- DS does not scan subdirectories.
- Do not define the same configuration token more than once in a file, as you cannot know in advance which value will be used.
- You cannot define the same configuration token in more than one file in a single directory. If the same token occurs in more than one file in a single directory, an error occurs.
- If the same token occurs once in several files which are located in different directories, the first value that DS reads is used.

The preceding list reflects the order of precedence:

- Environment variables override system properties, default token settings, and settings in any expression files.
- System properties override default token settings, and any settings in expression files.
- If `DS_ENVCONFIG_DIRS` or `ds.envconfig.dirs` is set, then the server uses settings found in expression files.
- Default token settings (`ds.config.dir`, `ds.instance.dir`, `ds.install.dir`).

For an embedded DS server, it is possible to change the expression resolvers, in the server configuration.

Use multivalued expressions

A single expression token can evaluate to multiple property values. Such expressions are useful with multivalued properties.

For example, suppose you choose to set a connection handler's `ssl-cipher-suite` property. Instead of listing cipher suites individually, you use an `ssl.cipher.suites` token that takes multiple values.

The following example commands set the token value in the environment, stop the server, use the expression in the LDAP connection handler configuration while the server is offline, and then start the server again:

```
$ export SSL_CIPHER_SUITES=\
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,\
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,\
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,\
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,\
TLS_EMPTY_RENEGOTIATION_INFO_SCSV
$ stop-ds --quiet
$ dsconfig \
 set-connection-handler-prop \
 --offline \
 --handler-name LDAPS \
 --add ssl-protocol:TLSv1.2 \
 --add ssl-cipher-suite:'&{ssl.cipher.suites}' \
 --no-prompt
$ start-ds --quiet
```

Multiple values are separated by commas in environment variables, system properties, and properties files. They are formatted as arrays in JSON files.

Use one of the following alternatives to set the value of the `ssl.cipher.suites` token. In each case, when the server evaluates `&{ssl.cipher.suites}`, the result is the following property values:

```
ssl-cipher-suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
ssl-cipher-suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ssl-cipher-suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ssl-cipher-suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ssl-cipher-suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV
```

Environment variable

```
export SSL_CIPHER_SUITES=\
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,\
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,\
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,\
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,\
TLS_EMPTY_RENEGOTIATION_INFO_SCSV
```

System property

```
export OPENDJ_JAVA_ARGS="-Dssl.cipher.suites=\
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,\
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,\
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,\
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,\
TLS_EMPTY_RENEGOTIATION_INFO_SCSV"
```

Properties file

```
ssl.cipher.suites=\
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,\
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,\
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,\
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,\
TLS_EMPTY_RENEGOTIATION_INFO_SCSV
```

JSON file

```
{
    "ssl.cipher.suites": [
        "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
        "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
        "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
        "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
        "TLS_EMPTY_RENEGOTIATION_INFO_SCSV"
    ]
}
```

Alternative JSON file that sets `ssl.protocol` as well:

```
{
    "ssl": {
        "protocol": "TLSv1.2",
        "cipher.suites": [
            "TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384",
            "TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
            "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
            "TLS_EMPTY_RENEGOTIATION_INFO_SCSV"
        ]
    }
}
```

In order to fully use the settings in this file, you would have to change the example to include the additional expression: `--add ssl-protocol:'&{ssl.protocol}'`.

Debug expressions

You can debug configuration expressions. Create a debug target for `org.forgerock.config.resolvers`. The following example demonstrates the process:

```
$ dsconfig \
 create-debug-target \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --type generic \
 --target-name org.forgerock.config.resolvers \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
$ dsconfig \
 set-log-publisher-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Debug Logger" \
 --set enabled:true \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
$ stop-ds --restart --quiet
```

When the server starts, it logs debugging messages for configuration expressions. Do not leave debug logging enabled in production systems.

LDAP reference

This reference covers LDAP-specific features of DS software.

Standards

Supported standards

Controls

Support LDAP controls

Extended Operations

Supported LDAP extended operations

Localization

Supported locales and languages

ForgeRock Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. For more information, visit https://www.pingidentity.com.

Supported standards

DS software implements the following RFCs, Internet-Drafts, and standards:

RFC 1274: The COSINE and Internet X.500 Schema

X.500 Directory Schema, or Naming Architecture, for use in the COSINE and Internet X.500 pilots.

RFC 1321: The MD5 Message-Digest Algorithm

MD5 message-digest algorithm that takes as input a message of arbitrary length, and produces a 128-bit "fingerprint" or "message digest" of the input.

RFC 1777: Lightweight Directory Access Protocol (LDAPv2)

Provide access to the X.500 Directory while not incurring the resource requirements of the Directory Access Protocol.

Classified as an historic document.

RFC 1778: The String Representation of Standard Attribute Syntaxes

Defines the requirements that must be satisfied by encoding rules, used to render X.500 Directory attribute syntaxes into a form suitable for use in LDAP. Defines the encoding rules for the standard set of attribute syntaxes.

Classified as an historic document.

RFC 1779: A String Representation of Distinguished Names

Defines a string format for representing names, which is designed to give a clean representation of commonly used names, while being able to represent any distinguished name.

Classified as an historic document.

RFC 2079: Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs)

Defines a new attribute type and an auxiliary object class to store URIs, including URLs, in directory entries.

RFC 2222: Simple Authentication and Security Layer (SASL)

Describes a method for adding authentication support to connection-based protocols.

RFC 2246: The TLS Protocol Version 1.0

Specifies Version 1.0 of the Transport Layer Security protocol.

RFC 2247: Using Domains in LDAP/X.500 Distinguished Names

Defines an algorithm by which a name registered with the Internet Domain Name Service can be represented as an LDAP distinguished name.

RFC 2251: Lightweight Directory Access Protocol (v3)

Describes a directory access protocol designed to provide access to directories supporting the X.500 models, while not incurring the resource requirements of the X.500 Directory Access Protocol.

RFC 2252: Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions

Defines a set of syntaxes for LDAPv3, and the rules by which attribute values of these syntaxes are represented as octet strings for transmission in the LDAP protocol.

RFC 2253: Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names

Defines a common UTF-8 format to represent distinguished names unambiguously.

RFC 2254: The String Representation of LDAP Search Filters

Defines the string format for representing names, which is designed to give a clean representation of commonly used distinguished names, while being able to represent any distinguished name.

RFC 2255: The LDAP URL Format

Describes a format for an LDAP URL.

RFC 2256: A Summary of the X.500(96) User Schema for use with LDAPv3

Provides an overview of the attribute types and object classes defined by the ISO and ITU-T committees in the X.500 documents, in particular those intended for use by directory clients.

RFC 2307: An Approach for Using LDAP as a Network Information Service

Describes an experimental mechanism for mapping entities related to TCP/IP and the UNIX system into X.500 entries so that they may be resolved with LDAP.

RFC 2377: Naming Plan for Internet Directory-Enabled Applications

Proposes a new directory naming plan that leverages the strengths of the most popular and successful Internet naming schemes for naming objects in a hierarchical directory.

RFC 2696: LDAP Control Extension for Simple Paged Results Manipulation

Lets a client control the rate at which an LDAP server returns the results of an LDAP search operation.

RFC 2713: Schema for Representing Java(tm) Objects in an LDAP Directory

Defines a common way for applications to store and retrieve Java objects from the directory.

RFC 2714: Schema for Representing CORBA Object References in an LDAP Directory

Define a common way for applications to store and retrieve CORBA object references from the directory.

RFC 2739: Calendar Attributes for vCard and LDAP

Defines a mechanism to locate a user calendar and free/busy time using the LDAP protocol.

RFC 2798: Definition of the inetOrgPerson LDAP Object Class

Defines an object class called inetOrgPerson for use in LDAP and X.500 directory services that extends the X.521 standard organizationalPerson class.

RFC 2829: Authentication Methods for LDAP

Specifies particular combinations of security mechanisms which are required and recommended in LDAP implementations.

RFC 2830: Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security

Defines the Start Transport Layer Security (TLS) operation for LDAP.

RFC 2849: The LDAP Data Interchange Format (LDIF) - Technical Specification

Describes a file format suitable for describing directory information or modifications made to directory information.

RFC 2891: LDAP Control Extension for Server Side Sorting of Search Results

Describes two LDAPv3 control extensions for server-side sorting of search results.

RFC 2926: Conversion of LDAP Schemas to and from SLP Templates

Describes a procedure for mapping between Service Location Protocol service advertisements and LDAP descriptions of services.

RFC 3045: Storing Vendor Information in the LDAP root DSE

Specifies two LDAP attributes, vendorName and vendorVersion that may be included in the root DSA-specific Entry (DSE) to advertise vendor-specific information.

RFC 3062: LDAP Password Modify Extended Operation

Describes an LDAP extended operation to allow modification of user passwords, which does not depend on the authentication identity or the password storage mechanism.

RFC 3112: LDAP Authentication Password Schema

Describes LDAP schema for user/password authentication including the authPassword attribute type. This attribute type holds values derived from the user's password(s) (commonly using cryptographic strength one-way hash).

RFC 3296: Named Subordinate References in Lightweight Directory Access Protocol (LDAP) Directories

Details schema and protocol elements for representing and managing named subordinate references in LDAP directories.

RFC 3377: Lightweight Directory Access Protocol (v3): Technical Specification

Specifies the set of RFCs comprising LDAPv3, and addresses the "IESG Note" attached to RFCs 2251 through 2256.

RFC 3383: Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP)

Provides procedures for registering extensible elements of LDAP.

RFC 3546: Transport Layer Security (TLS) Extensions

Describes extensions that may be used to add functionality to Transport Layer Security.

RFC 3671: Collective Attributes in the Lightweight Directory Access Protocol (LDAP)

Summarizes the X.500 information model for collective attributes and describes use of collective attributes in LDAP.

RFC 3672: Subentries in the Lightweight Directory Access Protocol (LDAP)

Adapts the X.500 subentry mechanisms for use with LDAP.

DS servers extend the subtree specification's `specificationFilter` component to allow any search filter.

RFC 3673: Lightweight Directory Access Protocol version 3 (LDAPv3): All Operational Attributes

Describes an LDAP extension which clients may use to request the return of all operational attributes.

RFC 3674: Feature Discovery in Lightweight Directory Access Protocol (LDAP)

Introduces a general mechanism for discovery of elective features and extensions, which cannot be discovered using existing mechanisms.

RFC 3712: Lightweight Directory Access Protocol (LDAP): Schema for Printer Services

Defines a schema, object classes and attributes, for printers and printer services, for use with LDAP directories.

RFC 3771: Lightweight Directory Access Protocol (LDAP) Intermediate Response Message

Defines and describes the IntermediateResponse message, a general mechanism for defining single-request/multiple-response operations in LDAP.

RFC 3829: Lightweight Directory Access Protocol (LDAP) Authorization Identity Request and Response Controls

Extends the LDAP bind operation with a mechanism for requesting and returning the authorization identity it establishes.

RFC 3876: Returning Matched Values with the Lightweight Directory Access Protocol version 3 (LDAPv3)

Describes a control for LDAPv3 that is used to return a subset of attribute values from an entry.

RFC 3909: Lightweight Directory Access Protocol (LDAP) Cancel Operation

Describes an LDAP extended operation to cancel (or abandon) an outstanding operation, with a response to indicate the outcome of the operation.

RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1

Specifies Version 1.1 of the Transport Layer Security protocol.

RFC 4370: Lightweight Directory Access Protocol (LDAP) Proxied Authorization Control

Defines the Proxy Authorization Control, which lets a client request that an operation be processed under a provided authorization identity instead of under the current authorization identity associated with the connection.

RFC 4403: Lightweight Directory Access Protocol (LDAP) Schema for Universal Description, Discovery, and Integration version 3 (UDDIv3)

Defines the LDAP schema for representing UDDIv3 data types in an LDAP directory.

RFC 4422: Simple Authentication and Security Layer (SASL)

Describes a framework for providing authentication and data security services in connection-oriented protocols via replaceable mechanisms.

RFC 4505: Anonymous Simple Authentication and Security Layer (SASL) Mechanism

Describes a new way to provide anonymous login needed within the context of the SASL framework.

RFC 4510: Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map

Provides a road map of the LDAP Technical Specification.

RFC 4511: Lightweight Directory Access Protocol (LDAP): The Protocol

Describes LDAP protocol elements, and their semantics and encodings.

RFC 4512: Lightweight Directory Access Protocol (LDAP): Directory Information Models

Describes the X.500 Directory Information Models as used in LDAP.

RFC 4513: Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms

Describes LDAP authentication methods and security mechanisms.

RFC 4514: Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names

Defines the string representation used in LDAP to transfer distinguished names.

RFC 4515: Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters

Defines a human-readable string representation of LDAP search filters that is appropriate for use in LDAP URLs and in other applications.

RFC 4516: Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator

Describes a format for an LDAP URL.

RFC 4517: Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules

Defines a base set of syntaxes and matching rules for use in defining attributes for LDAP directories.

RFC 4518: Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation

Defines string preparation algorithms for character-based matching rules defined for use in LDAP.

RFC 4519: Lightweight Directory Access Protocol (LDAP): Schema for User Applications

Provides a technical specification of attribute types and object classes intended for use by LDAP directory clients for many directory services, such as white pages.

RFC 4523: Lightweight Directory Access Protocol (LDAP) Schema Definitions for X.509 Certificates

Describes schema for representing X.509 certificates, X.521 security information, and related elements in LDAP directories.

RFC 4524: COSINE LDAP/X.500 Schema

Provides a collection of LDAP schema elements from the COSINE and Internet X.500 pilot projects.

RFC 4525: Lightweight Directory Access Protocol (LDAP) Modify-Increment Extension

Describes an LDAP extension to the LDAP modify operation that supports an increment capability.

RFC 4526: Lightweight Directory Access Protocol (LDAP) Absolute True and False Filters

Extends LDAP to support absolute True and False filters based upon similar capabilities found in X.500 directory systems.

RFC 4527: Lightweight Directory Access Protocol (LDAP) Read Entry Controls

Specifies an LDAP extension to let the client read the target entry of an update operation.

RFC 4528: Lightweight Directory Access Protocol (LDAP) Assertion Control

Defines the LDAP Assertion Control, which lets a client specify that a directory operation should only be processed if an assertion applied to the target entry of the operation is true.

RFC 4529: Requesting Attributes by Object Class in the Lightweight Directory Access Protocol (LDAP)

Extends LDAP to support a mechanism that lets LDAP clients request the return of all attributes of an object class.

RFC 4530: Lightweight Directory Access Protocol (LDAP) entryUUID Operational Attribute

Describes the LDAP/X.500 entryUUID operational attribute and associated matching rules and syntax.

RFC 4532: Lightweight Directory Access Protocol (LDAP) "Who am I?" Operation

Provides an LDAP mechanism for clients to obtain the authorization identity that the server has associated with the user or application entity.

RFC 4616: The PLAIN Simple Authentication and Security Layer (SASL) Mechanism

Defines a simple plaintext user/password SASL mechanism called the PLAIN mechanism.

RFC 4634: US Secure Hash Algorithms (SHA and HMAC-SHA)

Specifies Secure Hash Algorithms, SHA-256, SHA-384, and SHA-512, for computing a condensed representation of a message or a data file.

RFC 4752: The Kerberos V5 ("GSSAPI") Simple Authentication and Security Layer (SASL) Mechanism

Describes the method for using the GSS-API Kerberos V5 in SASL, called the GSSAPI mechanism.

RFC 4876: A Configuration Profile Schema for Lightweight Directory Access Protocol (LDAP)-Based Agents

Defines a schema for storing a profile for agents that use LDAP.

RFC 5020: The Lightweight Directory Access Protocol (LDAP) entryDN Operational Attribute

Describes the LDAP/X.500 entryDN operational attribute, which provides a copy of the entry's DN for use in attribute value assertions.

RFC 5802: Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms

Describes SASL SCRAM.

RFC 5803: Lightweight Directory Access Protocol (LDAP) Schema for Storing Salted Challenge Response Authentication Mechanism (SCRAM) Secrets

Describes how an LDAP directory server stores passwords for use in SCRAM SASL binds.

RFC 7677: SCRAM-SHA-256 and SCRAM-SHA-256-PLUS Simple Authentication and Security Layer (SASL) Mechanisms

Registers mechanisms for SASL SCRAM, updating RFC 5802.

RFC 9106: Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications

Describes the Argon2 memory-hard function for calculating a password hash.

FIPS 180-1: Secure Hash Standard (SHA-1)

Specifies a Secure Hash Algorithm, SHA-1, for computing a condensed representation of a message or a data file.

FIPS 180-2: Secure Hash Standard (SHA-1, SHA-256, SHA-384, SHA-512)

Specifies four Secure Hash Algorithms for computing a condensed representation of electronic data.

DSMLv2: Directory Service Markup Language

Provides a method for expressing directory queries and updates as XML documents.

JavaScript Object Notation

A data-interchange format that aims to be both "easy for humans to read and write," and "easy for machines to parse and generate."

RFC 7643: System for Cross-domain Identity Management: Core Schema

Platform neutral schema and extension model for representing users and groups in JSON and XML formats. DS supports the JSON format.

The LDAP Relax Rules Control (Internet-Draft)

Experimental LDAP control allowing a directory client application to request temporary relaxation of data and service model rules.

This control relaxes LDAP constraints, allowing operations that are not normally permitted, such as modifying read-only attributes. To prevent misuse, restrict access to this control to limited administrative accounts.

The Proxy Protocol

An HAProxy Technologies protocol that safely transports connection information, such as a client's IP address, through multiple proxy layers.

Supported LDAP controls

LDAP controls provide a mechanism to extend the semantics and arguments of existing LDAP operations. One or more controls may be attached to a single LDAP message. A control only affects the semantics of the message it is attached to. Controls sent by clients are called *request controls*. Controls returned by servers are called *response controls*.

DS software supports the following LDAP controls:

Account Usability Control

Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.8

Sun Microsystems control to determine whether a user account can be used to authenticate to the directory.

Active Directory change notification control

Object Identifier: 1.2.840.113556.1.4.528

Microsoft Active Directory control for a client application to register with the directory to receive change notifications.

Assertion request control

Object Identifier: 1.3.6.1.1.12

RFC: RFC 4528: Lightweight Directory Access Protocol (LDAP) Assertion Control

Authorization Identity request control

Object Identifier: 2.16.840.1.113730.3.4.16

RFC: RFC 3829: Lightweight Directory Access Protocol (LDAP) Authorization Identity Request and Response Controls

Authorization Identity response control

Object Identifier: 2.16.840.1.113730.3.4.15

RFC: RFC 3829: Lightweight Directory Access Protocol (LDAP) Authorization Identity Request and Response Controls

Entry Change Notification response control

Object Identifier: 2.16.840.1.113730.3.4.7

Internet-Draft: draft-ietf-ldapext-psearch: Persistent Search: A Simple LDAP Change Notification Mechanism

Get Effective Rights request control

Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.2

Internet-Draft: draft-ietf-ldapext-acl-model: Access Control Model for LDAPv3

Internal Modifications control

Object Identifier: 1.3.6.1.4.1.36733.2.1.5.3

ForgeRock control that provides additional modifications to a request for internal operations.

Load Balancer Connection Affinity control

Object Identifier: 1.3.6.1.4.1.36733.2.1.5.2

ForgeRock control that provides a value for connection affinity when using a load balancer from the LDAP SDK.

When you use a DS SDK load balancer that does not support connection affinity, attach this control to LDAP operations that require affinity load balancing.

Manage DSAIT request control

Object Identifier: 2.16.840.1.113730.3.4.2

RFC: RFC 3296: Named Subordinate References in Lightweight Directory Access Protocol (LDAP) Directories

Matched Values request control

Object Identifier: 1.2.826.0.1.3344810.2.3

RFC: RFC 3876: Returning Matched Values with the Lightweight Directory Access Protocol version 3 (LDAPv3)

No-Op Control

Object Identifier: 1.3.6.1.4.1.4203.1.10.2

Internet-Draft: draft-zeilenga-ldap-noop: LDAP No-Op Control

Password Expired response control

Object Identifier: 2.16.840.1.113730.3.4.4

Internet-Draft: draft-vchu-ldap-pwd-policy: Password Policy for LDAP Directories

Password Expiring response control

Object Identifier: 2.16.840.1.113730.3.4.5

Internet-Draft: draft-vchu-ldap-pwd-policy: Password Policy for LDAP Directories

Password Policy response control

Object Identifier: 1.3.6.1.4.1.42.2.27.8.5.1

Internet-Draft: draft-behera-ldap-password-policy: Password Policy for LDAP Directories

Password Quality Advice controls

Object Identifier: 1.3.6.1.4.1.36733.2.1.5.5

ForgeRock controls that are used for requesting and returning structured password quality advice. The request and response controls share the same OID.

Interface stability: *Evolving*.

Permissive Modify request control

Object Identifier: 1.2.840.113556.1.4.1413

Microsoft defined this control that, "Allows an LDAP modify to work under less restrictive conditions. Without it, a delete will fail if an attribute does not exist, and an add will fail if an attribute already exists. No data is needed in this control." (source of quote)

Persistent Search request control

Object Identifier: 2.16.840.1.113730.3.4.3

Internet-Draft: draft-ietf-ldapext-psearch: Persistent Search: A Simple LDAP Change Notification Mechanism

Post-Read request control

Object Identifier: 1.3.6.1.1.13.2

RFC: RFC 4527: Lightweight Directory Access Protocol (LDAP) Read Entry Controls

Post-Read response control

Object Identifier: 1.3.6.1.1.13.2

RFC: RFC 4527: Lightweight Directory Access Protocol (LDAP) Read Entry Controls

Pre-Read request control

Object Identifier: 1.3.6.1.1.13.1

RFC: RFC 4527: Lightweight Directory Access Protocol (LDAP) Read Entry Controls

Pre-Read response control

Object Identifier: 1.3.6.1.1.13.1

RFC: RFC 4527: Lightweight Directory Access Protocol (LDAP) Read Entry Controls

Proxied Authorization v1 request control

Object Identifier: 2.16.840.1.113730.3.4.12

Internet-Draft: draft-weltman-ldapv3-proxy-04: LDAP Proxied Authorization Control

Proxied Authorization v2 request control

Object Identifier: 2.16.840.1.113730.3.4.18

RFC: RFC 4370: Lightweight Directory Access Protocol (LDAP) Proxied Authorization Control

Public Changelog Exchange Control

Object Identifier: 1.3.6.1.4.1.26027.1.5.4

DS control for using the bookmark cookie when reading the external change log.

Real Attributes Only Request Control

Object Identifier: 2.16.840.1.113730.3.4.17

Netscape control indicating that the request is only for attributes actually contained in the entry. Do not return virtual attributes even if they are explicitly requested.

The control has no value.

Replication Context control

Object Identifier: 1.3.6.1.4.1.36733.2.1.5.4

ForgeRock control that is used internally to provide some replication-related context to requests. This control may be removed in the future.

Replication repair control

Object Identifier: 1.3.6.1.4.1.26027.1.5.2

DS control that is used to modify the content of a replicated database on a single server without impacting the other servers that are replicated with this server.

Server-Side Sort request control

Object Identifier: 1.2.840.113556.1.4.473

RFC: RFC 2891: LDAP Control Extension for Server Side Sorting of Search Results

Server-Side Sort response control

Object Identifier: 1.2.840.113556.1.4.474

RFC: RFC 2891: LDAP Control Extension for Server Side Sorting of Search Results

Simple Paged Results Control

Object Identifier: 1.2.840.113556.1.4.319

RFC: RFC 2696: LDAP Control Extension for Simple Paged Results Manipulation

Subentries request controls

Object Identifier: 1.3.6.1.4.1.4203.1.10.1

RFC: Subentries in the Lightweight Directory Access Protocol (LDAP)

Object Identifier: 1.3.6.1.4.1.7628.5.101.1

Internet-Draft: draft-ietf-ldup-subentry: LDAP Subentry Schema

Subtree Delete request control

Object Identifier: 1.2.840.113556.1.4.805

Internet-Draft: draft-armijo-ldap-treedelete: Tree Delete Control

Transaction ID control

Object Identifier: 1.3.6.1.4.1.36733.2.1.5.1

ForgeRock control that enables Common Audit to associate an ID with a request. The ID is recorded with audit events, and can be used to correlate and track user interactions as they traverse the components of the ForgeRock platform.

The control's value is the UTF-8 encoding of the transaction ID.

Virtual List View request control

Object Identifier: 2.16.840.1.113730.3.4.9

Internet-Draft: draft-ietf-ldapext-ldapv3-vlv: LDAP Extensions for Scrolling View Browsing of Search Results

Virtual Attributes Only Request Control

Object Identifier: 2.16.840.1.113730.3.4.19

Netscape control indicating that the request is only for virtual attributes. Do not return real attributes contained in the entry even if they are explicitly requested.

The control has no value.

Virtual List View response control

Object Identifier: 2.16.840.1.113730.3.4.10

Internet-Draft: draft-ietf-ldapext-ldapv3-vlv: LDAP Extensions for Scrolling View Browsing of Search Results

Supported LDAP extended operations

LDAP extended operations define additional operations for services not already available in the protocol.

DS software supports the following LDAP extended operations:

Cancel Extended Request

Object Identifier: 1.3.6.1.1.8

RFC: RFC 3909: Lightweight Directory Access Protocol (LDAP) Cancel Operation

Get Connection ID Extended Request

Object Identifier: 1.3.6.1.4.1.26027.1.6.2

DS extended operation to return the connection ID of the associated client connection. This extended operation is intended for DS internal use.

Password Modify Extended Request

Object Identifier: 1.3.6.1.4.1.4203.1.11.1

RFC: RFC 3062: LDAP Password Modify Extended Operation

Password Policy State Extended Operation

Object Identifier: 1.3.6.1.4.1.26027.1.6.1

DS extended operation to query and update password policy state for a given user entry.

Start Transport Layer Security Extended Request

Object Identifier: 1.3.6.1.4.1.1466.20037

RFC: RFC 4511: Lightweight Directory Access Protocol (LDAP): The Protocol

Who am I? Extended Request

Object Identifier: 1.3.6.1.4.1.4203.1.11.3

RFC: RFC 4532: Lightweight Directory Access Protocol (LDAP) "Who am I?" Operation

Support for languages and locales

DS software stores data in UTF-8 format. You can store and search for localized directory data for many locales.

Locales and language subtypes

DS software supports the following locales with their associated language and country codes, and their collation order object identifiers. Locale support depends on the Java Virtual Machine used at run time. The following list reflects all supported locales.

Locales

Afrikaans

Code tag: af
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.1.1

Albanian

Code tag: sq
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.127.1

Amharic

Code tag: am
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.2.1

Arabic

Code tag: ar
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.3.1

Arabic (Algeria)

Code tag: ar-DZ
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.6.1

Arabic (Bahrain)

Code tag: ar-BH
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.5.1

Arabic (Egypt)

Code tag: ar-EG
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.7.1

Arabic (India)

Code tag: ar-IN
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.8.1

Arabic (Iraq)

Code tag: ar-IQ
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.9.1

Arabic (Jordan)

Code tag: ar-JO
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.10.1

Arabic (Kuwait)

Code tag: ar-KW
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.11.1

Arabic (Lebanon)

Code tag: ar-LB
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.12.1

Arabic (Libya)

Code tag: ar-LY
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.13.1

Arabic (Morocco)

Code tag: ar-MA
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.14.1

Arabic (Oman)

Code tag: ar-OM
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.15.1

Arabic (Qatar)

Code tag: ar-QA
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.16.1

Arabic (Saudi Arabia)

Code tag: ar-SA
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.17.1

Arabic (Sudan)

Code tag: ar-SD
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.18.1

Arabic (Syria)

Code tag: ar-SY
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.19.1

Arabic (Tunisia)

Code tag: ar-TN
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.20.1

Arabic (United Arab Emirates)

Code tag: ar-AE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.4.1

Arabic (Yemen)

Code tag: ar-YE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.21.1

Armenian

Code tag: hy
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.89.1

Bangla

Code tag: bn
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.24.1

Basque

Code tag: eu
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.70.1

Belarusian

Code tag: be
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.22.1

Bulgarian

Code tag: bg
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.23.1

Catalan

Code tag: ca
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.25.1

Chinese

Code tag: zh
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.143.1

Chinese (China)

Code tag: zh-CN
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.144.1

Chinese (Hong Kong SAR China)

Code tag: zh-HK
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.145.1

Chinese (Macau SAR China)

Code tag: zh-MO
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.146.1

Chinese (Singapore)

Code tag: zh-SG
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.147.1

Chinese (Taiwan)

Code tag: zh-TW
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.148.1

Cornish

Code tag: kw
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.99.1

Croatian

Code tag: hr
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.87.1

Czech

Code tag: cs
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.26.1

Danish

Code tag: da
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.27.1

Dutch

Code tag: nl
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.105.1

Dutch (Belgium)

Code tag: nl-BE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.106.1

Dutch (Netherlands)

Code tag: nl-NL
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.105.1

English

Code tag: en
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.34.1

English (Australia)

Code tag: en-AU
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.35.1

English (Canada)

Code tag: en-CA
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.36.1

English (Hong Kong SAR China)

Code tag: en-HK
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.38.1

English (India)

Code tag: en-IN
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.40.1

English (Ireland)

Code tag: en-IE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.39.1

English (Malta)

Code tag: en-MT
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.41.1

English (New Zealand)

Code tag: en-NZ
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.42.1

English (Philippines)

Code tag: en-PH
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.43.1

English (Singapore)

Code tag: en-SG
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.44.1

English (South Africa)

Code tag: en-ZA
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.46.1

English (U.S. Virgin Islands)

Code tag: en-VI
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.45.1

English (United Kingdom)

Code tag: en-GB
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.37.1

English (United States)

Code tag: en-US
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.34.1

English (Zimbabwe)

Code tag: en-ZW
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.47.1

Esperanto

Code tag: eo
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.48.1

Estonian

Code tag: et
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.69.1

Faroese

Code tag: fo
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.75.1

Finnish

Code tag: fi
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.74.1

French

Code tag: fr
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.76.1

French (Belgium)

Code tag: fr-BE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.77.1

French (Canada)

Code tag: fr-CA
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.78.1

French (France)

Code tag: fr-FR
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.76.1

French (Luxembourg)

Code tag: fr-LU
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.80.1

French (Switzerland)

Code tag: fr-CH
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.79.1

Galician

Code tag: gl
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.82.1

German

Code tag: de
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.28.1

German (Austria)

Code tag: de-AT
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.29.1

German (Belgium)

Code tag: de-BE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.30.1

German (Germany)

Code tag: de-DE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.28.1

German (Luxembourg)

Code tag: de-LU
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.32.1

German (Switzerland)

Code tag: de-CH
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.31.1

Greek

Code tag: el
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.33.1

Gujarati

Code tag: gu
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.83.1

Hebrew

Code tag: iw
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.85.1

Hindi

Code tag: hi
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.86.1

Hungarian

Code tag: hu
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.88.1

Icelandic

Code tag: is
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.91.1

Indonesian

Code tag: in
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.90.1

Irish

Code tag: ga
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.81.1

Italian

Code tag: it
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.92.1

Italian (Switzerland)

Code tag: it-CH
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.93.1

Japanese

Code tag: ja
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.94.1

Kalaallisut

Code tag: kl
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.95.1

Kannada

Code tag: kn
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.96.1

Konkani

Code tag: kok
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.98.1

Korean

Code tag: ko
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.97.1

Latvian

Code tag: lv
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.101.1

Lithuanian

Code tag: lt
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.100.1

Macedonian

Code tag: mk
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.102.1

Maltese

Code tag: mt
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.104.1

Manx

Code tag: gv
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.84.1

Marathi

Code tag: mr
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.103.1

Norwegian

Code tag: no
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.107.1

Norwegian (Norway)

Code tag: no-NO-NY
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.108.1

Norwegian Bokmål

Code tag: nb
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.110.1

Norwegian Nynorsk

Code tag: nn
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.109.1

Oromo

Code tag: om
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.111.1

Oromo (Ethiopia)

Code tag: om-ET
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.112.1

Oromo (Kenya)

Code tag: om-KE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.113.1

Persian

Code tag: fa
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.71.1

Persian (India)

Code tag: fa-IN
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.72.1

Persian (Iran)

Code tag: fa-IR
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.73.1

Polish

Code tag: pl
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.114.1

Portuguese

Code tag: pt
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.115.1

Portuguese (Brazil)

Code tag: pt-BR
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.116.1

Portuguese (Portugal)

Code tag: pt-PT
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.115.1

Romanian

Code tag: ro
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.117.1

Russian

Code tag: ru
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.118.1

Russian (Russia)

Code tag: ru-RU
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.118.1

Russian (Ukraine)

Code tag: ru-UA
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.119.1

Serbian

Code tag: sr
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.128.1

Serbo-Croatian

Code tag: sh
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.120.1

Slovak

Code tag: sk
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.121.1

Slovenian

Code tag: sl
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.122.1

Somali

Code tag: so
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.123.1

Somali (Djibouti)

Code tag: so-DJ
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.124.1

Somali (Ethiopia)

Code tag: so-ET
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.125.1

Somali (Kenya)

Code tag: so-KE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.126.1

Somali (Somalia)

Code tag: so-SO
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.123.1

Spanish

Code tag: es
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.49.1

Spanish (Argentina)

Code tag: es-AR
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.50.1

Spanish (Bolivia)

Code tag: es-BO
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.51.1

Spanish (Chile)

Code tag: es-CL
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.52.1

Spanish (Colombia)

Code tag: es-CO
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.53.1

Spanish (Costa Rica)

Code tag: es-CR
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.54.1

Spanish (Dominican Republic)

Code tag: es-DO
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.55.1

Spanish (Ecuador)

Code tag: es-EC
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.56.1

Spanish (El Salvador)

Code tag: es-SV
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.65.1

Spanish (Guatemala)

Code tag: es-GT
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.57.1

Spanish (Honduras)

Code tag: es-HN
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.58.1

Spanish (Mexico)

Code tag: es-MX
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.59.1

Spanish (Nicaragua)

Code tag: es-NI
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.60.1

Spanish (Panama)

Code tag: es-PA
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.61.1

Spanish (Paraguay)

Code tag: es-PY
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.64.1

Spanish (Peru)

Code tag: es-PE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.62.1

Spanish (Puerto Rico)

Code tag: es-PR
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.63.1

Spanish (Spain)

Code tag: es-ES
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.49.1

Spanish (United States)

Code tag: es-US
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.66.1

Spanish (Uruguay)

Code tag: es-UY
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.67.1

Spanish (Venezuela)

Code tag: es-VE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.68.1

Swahili

Code tag: sw
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.131.1

Swahili (Kenya)

Code tag: sw-KE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.132.1

Swahili (Tanzania)

Code tag: sw-TZ
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.133.1

Swedish

Code tag: sv
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.129.1

Swedish (Finland)

Code tag: sv-FI
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.130.1

Swedish (Sweden)

Code tag: sv-SE
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.129.1

Tamil

Code tag: ta
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.134.1

Telugu

Code tag: te
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.135.1

Thai

Code tag: th
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.136.1

Tigrinya

Code tag: ti
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.137.1

Tigrinya (Eritrea)

Code tag: ti-ER
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.138.1

Tigrinya (Ethiopia)

Code tag: ti-ET
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.139.1

Turkish

Code tag: tr
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.140.1

Ukrainian

Code tag: uk
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.141.1

Vietnamese

Code tag: vi
Collation order object identifier: 1.3.6.1.4.1.42.2.27.9.4.142.1

Language subtypes

- Afrikaans, af
- Albanian, sq
- Amharic, am
- Arabic, ar
- Armenian, hy
- Bangla, bn
- Basque, eu
- Belarusian, be
- Bulgarian, bg
- Catalan, ca
- Chinese, zh
- Cornish, kw

- Croatian, hr
- Czech, cs
- Danish, da
- Dutch, nl
- English, en
- Esperanto, eo
- Estonian, et
- Faroese, fo
- Finnish, fi
- French, fr
- Galician, gl
- German, de
- Greek, el
- Gujarati, gu
- Hebrew, iw
- Hindi, hi
- Hungarian, hu
- Icelandic, is
- Indonesian, in
- Irish, ga
- Italian, it
- Japanese, ja
- Kalaallisut, kl
- Kannada, kn
- Konkani, kok
- Korean, ko
- Latvian, lv
- Lithuanian, lt
- Macedonian, mk
- Maltese, mt

- Manx, gv
- Marathi, mr
- Norwegian, no
- Norwegian Bokmål, nb
- Norwegian Nynorsk, nn
- Oromo, om
- Persian, fa
- Polish, pl
- Portuguese, pt
- Romanian, ro
- Russian, ru
- Serbian, sr
- Serbo-Croatian, sh
- Slovak, sk
- Slovenian, sl
- Somali, so
- Spanish, es
- Swahili, sw
- Swedish, sv
- Tamil, ta
- Telugu, te
- Thai, th
- Tigrinya, ti
- Turkish, tr
- Ukrainian, uk
- Vietnamese, vi

Localization support

DS software supports localization, but it is not fully localized. Many messages and some tools are available only in English.

Some messages are localized for the following languages:

- Catalan
- French
- German
- Japanese
- Korean
- Polish
- Simplified Chinese
- Spanish
- Traditional Chinese

LDAP result codes

An operation result code as defined in RFC 4511 section 4.1.9 is used to indicate the final status of an operation. If a server detects multiple errors for an operation, only one result code is returned. The server should return the result code that best indicates the nature of the error encountered. Servers may return substituted result codes to prevent unauthorized disclosures.

Result code	Name	Description
-1	Undefined	The result code that should only be used if the actual result code has not yet been determined. Despite not being a standard result code, it is an implementation of the null object design pattern for this type.
0	Success	The result code that indicates that the operation completed successfully.
1	Operations Error	The result code that indicates that the operation is not properly sequenced with relation to other operations (of same or different type). For example, this code is returned if the client attempts to StartTLS [RFC4346] while there are other uncompleted operations or if a TLS layer was already installed.
2	Protocol Error	The result code that indicates that the client sent a malformed or illegal request to the server.

Result code	Name	Description
3	Time Limit Exceeded	The result code that indicates that a time limit was exceeded while attempting to process the request.
4	Size Limit Exceeded	The result code that indicates that a size limit was exceeded while attempting to process the request.
5	Compare False	The result code that indicates that the attribute value assertion included in a compare request did not match the targeted entry.
6	Compare True	The result code that indicates that the attribute value assertion included in a compare request did match the targeted entry.
7	Authentication Method Not Supported	The result code that indicates that the requested authentication attempt failed because it referenced an invalid SASL mechanism.
8	Strong Authentication Required	The result code that indicates that the requested operation could not be processed because it requires that the client has completed a strong form of authentication.
10	Referral	The result code that indicates that a referral was encountered. Strictly speaking this result code should not be exceptional since it is considered as a "success" response. However, referrals should occur rarely in practice and, when they do occur, should not be ignored since the application may believe that a request has succeeded when, in fact, nothing was done.
11	Administrative Limit Exceeded	The result code that indicates that processing on the requested operation could not continue because an administrative limit was exceeded.

Result code	Name	Description
12	Unavailable Critical Extension	The result code that indicates that the requested operation failed because it included a critical extension that is unsupported or inappropriate for that request.
13	Confidentiality Required	The result code that indicates that the requested operation could not be processed because it requires confidentiality for the communication between the client and the server.
14	SASL Bind in Progress	The result code that should be used for intermediate responses in multi-stage SASL bind operations.
16	No Such Attribute	The result code that indicates that the requested operation failed because it targeted an attribute or attribute value that did not exist in the specified entry.
17	Undefined Attribute Type	The result code that indicates that the requested operation failed because it referenced an attribute that is not defined in the server schema.
18	Inappropriate Matching	The result code that indicates that the requested operation failed because it attempted to perform an inappropriate type of matching against an attribute.
19	Constraint Violation	The result code that indicates that the requested operation failed because it would have violated some constraint defined in the server.
20	Attribute or Value Exists	The result code that indicates that the requested operation failed because it would have resulted in a conflict with an existing attribute or attribute value in the target entry.
21	Invalid Attribute Syntax	The result code that indicates that the requested operation failed because it violated the syntax for a specified attribute.

Result code	Name	Description
32	No Such Entry	The result code that indicates that the requested operation failed because it referenced an entry that does not exist.
33	Alias Problem	The result code that indicates that the requested operation failed because it attempted to perform an illegal operation on an alias.
34	Invalid DN Syntax	The result code that indicates that the requested operation failed because it would have resulted in an entry with an invalid or malformed DN.
36	Alias Dereferencing Problem	The result code that indicates that a problem was encountered while attempting to dereference an alias for a search operation.
48	Inappropriate Authentication	The result code that indicates that an authentication attempt failed because the requested type of authentication was not appropriate for the targeted entry.
49	Invalid Credentials	The result code that indicates that an authentication attempt failed because the user did not provide a valid set of credentials.
50	Insufficient Access Rights	The result code that indicates that the client does not have sufficient permission to perform the requested operation.
51	Busy	The result code that indicates that the server is too busy to process the requested operation. This is a transient error which means the operation can safely be retried.
52	Unavailable	The result code that indicates that either the entire server or one or more required resources were not available for use in processing the request. This is a transient error which means the operation can safely be retried.

Result code	Name	Description
53	Unwilling to Perform	The result code that indicates that the server is unwilling to perform the requested operation.
54	Loop Detected	The result code that indicates that a referral or chaining loop was detected while processing the request.
60	Sort Control Missing	The result code that indicates that a search request included a VLV request control without a server-side sort control.
61	Offset Range Error	The result code that indicates that a search request included a VLV request control with an invalid offset.
64	Naming Violation	The result code that indicates that the requested operation failed because it would have violated the server's naming configuration.
65	Object Class Violation	The result code that indicates that the requested operation failed because it would have resulted in an entry that violated the server schema.
66	Not Allowed on Non-Leaf	The result code that indicates that the requested operation is not allowed for non-leaf entries.
67	Not Allowed on RDN	The result code that indicates that the requested operation is not allowed on an RDN attribute.
68	Entry Already Exists	The result code that indicates that the requested operation failed because it would have resulted in an entry that conflicts with an entry that already exists.
69	Object Class Modifications Prohibited	The result code that indicates that the operation could not be processed because it would have modified the objectclasses associated with an entry in an illegal manner.

Result code	Name	Description
71	Affects Multiple DSAs	The result code that indicates that the operation could not be processed because it would impact multiple DSAs or other repositories.
76	Virtual List View Error	The result code that indicates that the operation could not be processed because there was an error while processing the virtual list view control.
80	Other	The result code that should be used if no other result code is appropriate.
81	Server Connection Closed	The client-side result code that indicates that the server is down. This is for client-side use only and should never be transferred over protocol. This is a transient error which means the operation can be retried.
82	Local Error	The client-side result code that indicates that a local error occurred that had nothing to do with interaction with the server. This is for client-side use only and should never be transferred over protocol.
83	Encoding Error	The client-side result code that indicates that an error occurred while encoding a request to send to the server. This is for client-side use only and should never be transferred over protocol.
84	Decoding Error	The client-side result code that indicates that an error occurred while decoding a response from the server. This is for client-side use only and should never be transferred over protocol.

Result code	Name	Description
85	Client-Side Timeout	The client-side result code that indicates that the client did not receive an expected response in a timely manner. This is for client-side use only and should never be transferred over protocol. This is a transient error which means the operation can be retried.
86	Unknown Authentication Mechanism	The client-side result code that indicates that the user requested an unknown or unsupported authentication mechanism. This is for client-side use only and should never be transferred over protocol.
87	Filter Error	The client-side result code that indicates that the filter provided by the user was malformed and could not be parsed. This is for client-side use only and should never be transferred over protocol.
88	Cancelled by User	The client-side result code that indicates that the user cancelled an operation. This is for client-side use only and should never be transferred over protocol.
89	Parameter Error	The client-side result code that indicates that there was a problem with one or more of the parameters provided by the user. This is for client-side use only and should never be transferred over protocol.
90	Out of Memory	The client-side result code that indicates that the client application was not able to allocate enough memory for the requested operation. This is for client-side use only and should never be transferred over protocol.

Result code	Name	Description
91	Connect Error	The client-side result code that indicates that the client was not able to establish a connection to the server. This is for client-side use only and should never be transferred over protocol. This is a transient error which means the operation can be retried.
92	Operation Not Supported	The client-side result code that indicates that the user requested an operation that is not supported. This is for client-side use only and should never be transferred over protocol.
93	Control Not Found	The client-side result code that indicates that the client expected a control to be present in the response from the server but it was not included. This is for client-side use only and should never be transferred over protocol.
94	No Results Returned	The client-side result code that indicates that the requested single entry search operation or read operation failed because the Directory Server did not return any matching entries. This is for client-side use only and should never be transferred over protocol.
95	Unexpected Results Returned	The client-side result code that the requested single entry search operation or read operation failed because the Directory Server returned multiple matching entries (or search references) when only a single matching entry was expected. This is for client-side use only and should never be transferred over protocol.

Result code	Name	Description
96	Referral Loop Detected	The client-side result code that indicates that the client detected a referral loop caused by servers referencing each other in a circular manner. This is for client-side use only and should never be transferred over protocol.
97	Referral Hop Limit Exceeded	The client-side result code that indicates that the client reached the maximum number of hops allowed when attempting to follow a referral (i.e., following one referral resulted in another referral which resulted in another referral and so on). This is for client-side use only and should never be transferred over protocol.
118	Canceled	The result code that indicates that a request has been cancelled by a cancel request.
119	No Such Operation	The result code that indicates that a cancel request was unsuccessful because the targeted operation did not exist or had already completed.
120	Too Late	The result code that indicates that a cancel request was unsuccessful because processing on the targeted operation had already reached a point at which it could not be canceled.
121	Cannot Cancel	The result code that indicates that a cancel request was unsuccessful because the targeted operation was one that could not be canceled.
122	Assertion Failed	The result code that indicates that the filter contained in an assertion control failed to match the target entry.
123	Authorization Denied	The result code that should be used if the server will not allow the client to use the requested authorization.

Result code	Name	Description
16,654	No Operation	The result code that should be used if the server did not actually complete processing on the associated operation because the request included the LDAP No-Op control.

LDAP schema reference

About This Reference

This reference describes the default directory schema. Each schema definition has its own section, with links to related sections. Reference pages for the most commonly used elements may include additional descriptions and examples that are not present in the directory schema definitions.

This reference does not include directory configuration attributes and object classes, collation matching rules.

LDAP directory schema defines how data can be stored in the directory. When a directory server receives a request to update directory data, it can check the data changes against the directory schema, refusing any request that would result in a violation of the directory schema and directory data corruption.

Schema checking prevents errors such as the following:

- Adding inappropriate attributes to an entry
- Removing required attributes from an entry
- Using an attribute value that has the wrong syntax
- Adding the wrong type of subordinate object

LDAP directory schema consists of definitions for the following:

Attribute types

Define attributes of directory entries, including their syntaxes and matching rules

Directory Information Tree (DIT) content rules

Define the content of entries with a given structural object class

DIT structure rules

Define the names entries may have, and how entries may be related to each other

Matching rules

Define how values of attributes are matched and compared

Matching rule uses

List attributes that can be used with an extensibleMatch search filter

Name forms

Define naming relations for structural object classes

Object classes

Define the types of objects that an entry represents, and the required and optional attributes for entries of those types

Syntaxes

Define the encodings used in LDAP

For a technical description of LDAP directory schema, read *Directory Schema* in *Lightweight Directory Access Protocol (LDAP): Directory Information Models* (RFC 4512).

LDAP directory servers allow client applications to access directory schema while the server is running. This enables applications to validate their changes against the schema before sending an update request to the server. As a result, LDAP schema definitions are optimized for applications, not humans. The reader must resolve relationships between schema definitions, and must find most documentation elsewhere.

Attribute types

This part covers schema definitions for attribute types:

- aci
- aclRights
- aclRightsInfo
- administratorsAddress
- aliasedObjectName
- alive
- altServer
- aRecord
- assignedDashboard
- associatedDomain
- associatedName
- attributeMap
- attributeTypes
- audio
- authenticationMethod
- authorityRevocationList
- authPassword
- automountInformation

- automountKey
- automountMapName
- bindTimeLimit
- blockInheritance
- bootFile
- bootParameter
- buildingName
- businessCategory
- c-FacsimileTelephoneNumber
- c-InternationalISDNNumber
- c-l
- c-o
- c-ou
- c-PhysicalDeliveryOfficeName
- c-PostalAddress
- c-PostalCode
- c-PostOfficeBox
- c-st
- c-street
- c-TelephoneNumber
- c-TelexNumber
- c
- cACertificate
- calCalAdrURI
- calCalURI
- calCAPURI
- calFBURL
- calOtherCalAdrURIs
- calOtherCalURIs
- calOtherCAPURIs

- calOtherFBURLs
- carLicense
- certificateRevocationList
- changeInitiatorsName
- changelog
- changeLogCookie
- changeNumber
- changes
- changeTime
- changeType
- cn
- cNAMERecord
- co
- collectiveAttributeSubentries
- collectiveConflictBehavior
- collectiveExclusions
- corbaIor
- corbaRepositoryId
- coreTokenDate01
- coreTokenDate02
- coreTokenDate03
- coreTokenDate04
- coreTokenDate05
- coreTokenExpirationDate
- coreTokenId
- coreTokenInteger01
- coreTokenInteger02
- coreTokenInteger03
- coreTokenInteger04
- coreTokenInteger05

- coreTokenInteger06
- coreTokenInteger07
- coreTokenInteger08
- coreTokenInteger09
- coreTokenInteger10
- coreTokenMultiString01
- coreTokenMultiString02
- coreTokenMultiString03
- coreTokenObject
- coreTokenString01
- coreTokenString02
- coreTokenString03
- coreTokenString04
- coreTokenString05
- coreTokenString06
- coreTokenString07
- coreTokenString08
- coreTokenString09
- coreTokenString10
- coreTokenString11
- coreTokenString12
- coreTokenString13
- coreTokenString14
- coreTokenString15
- coreTokenTtlDate
- coreTokenType
- coreTokenUserId
- createTimestamp
- creatorsName
- credentialLevel

- crossCertificatePair
- dc
- defaultSearchBase
- defaultSearchScope
- defaultServerList
- deleteOldRDN
- deltaRevocationList
- departmentNumber
- dereferenceAliases
- description
- destinationIndicator
- devicePrintProfiles
- deviceProfiles
- displayName
- distinguishedName
- dITContentRules
- dITRedirect
- dITStructureRules
- dmdName
- dnQualifier
- documentAuthor
- documentIdentifier
- documentLocation
- documentPublisher
- documentTitle
- documentVersion
- drink
- ds-certificate-fingerprint
- ds-certificate-issuer-dn
- ds-certificate-subject-dn

- ds-last-login-time
- ds-mon-abandoned-requests
- ds-mon-active-connections-count
- ds-mon-active-persistent-searches
- ds-mon-admin-hostport
- ds-mon-alias
- ds-mon-alive-errors
- ds-mon-alive
- ds-mon-backend-degraded-index-count
- ds-mon-backend-degraded-index
- ds-mon-backend-entry-count
- ds-mon-backend-entry-size-read
- ds-mon-backend-entry-size-written
- ds-mon-backend-filter-use-indexed
- ds-mon-backend-filter-use-start-time
- ds-mon-backend-filter-use-unindexed
- ds-mon-backend-filter-use
- ds-mon-backend-is-private
- ds-mon-backend-proxy-base-dn
- ds-mon-backend-proxy-shard
- ds-mon-backend-ttl-entries-deleted
- ds-mon-backend-ttl-is-running
- ds-mon-backend-ttl-last-run-time
- ds-mon-backend-ttl-queue-size
- ds-mon-backend-ttl-thread-count
- ds-mon-backend-writability-mode
- ds-mon-base-dn-entry-count
- ds-mon-base-dn
- ds-mon-build-number
- ds-mon-build-time

- ds-mon-bytes-read
- ds-mon-bytes-written
- ds-mon-cache-entry-count
- ds-mon-cache-max-entry-count
- ds-mon-cache-max-size-bytes
- ds-mon-cache-misses
- ds-mon-cache-total-tries
- ds-mon-certificate-expires-at
- ds-mon-certificate-issuer-dn
- ds-mon-certificate-serial-number
- ds-mon-certificate-subject-dn
- ds-mon-changelog-hostport
- ds-mon-changelog-id
- ds-mon-changelog-purge-delay
- ds-mon-compact-version
- ds-mon-config-dn
- ds-mon-connected-to-server-hostport
- ds-mon-connected-to-server-id
- ds-mon-connection
- ds-mon-connections
- ds-mon-current-connections
- ds-mon-current-receive-window
- ds-mon-current-time
- ds-mon-db-cache-evict-internal-nodes-count
- ds-mon-db-cache-evict-leaf-nodes-count
- ds-mon-db-cache-leaf-nodes
- ds-mon-db-cache-misses-internal-nodes
- ds-mon-db-cache-misses-leaf-nodes
- ds-mon-db-cache-size-active
- ds-mon-db-cache-size-total

- ds-mon-db-cache-total-tries-internal-nodes
- ds-mon-db-cache-total-tries-leaf-nodes
- ds-mon-db-checkpoint-count
- ds-mon-db-log-cleaner-file-deletion-count
- ds-mon-db-log-files-open
- ds-mon-db-log-files-opened
- ds-mon-db-log-size-active
- ds-mon-db-log-size-total
- ds-mon-db-log-utilization-max
- ds-mon-db-log-utilization-min
- ds-mon-db-version
- ds-mon-disk-dir
- ds-mon-disk-free
- ds-mon-disk-full-threshold
- ds-mon-disk-low-threshold
- ds-mon-disk-root
- ds-mon-disk-state
- ds-mon-domain-generation-id
- ds-mon-domain-name
- ds-mon-entries-awaiting-updates-count
- ds-mon-fix-ids
- ds-mon-full-version
- ds-mon-group-id
- ds-mon-healthy-errors
- ds-mon-healthy
- ds-mon-index-uses
- ds-mon-index
- ds-mon-install-path
- ds-mon-instance-path
- ds-mon-jvm-architecture

- ds-mon-jvm-arguments
- ds-mon-jvm-available-cpus
- ds-mon-jvm-class-path
- ds-mon-jvm-classes-loaded
- ds-mon-jvm-classes-unloaded
- ds-mon-jvm-java-home
- ds-mon-jvm-java-vendor
- ds-mon-jvm-java-version
- ds-mon-jvm-memory-heap-init
- ds-mon-jvm-memory-heap-max
- ds-mon-jvm-memory-heap-reserved
- ds-mon-jvm-memory-heap-used
- ds-mon-jvm-memory-init
- ds-mon-jvm-memory-max
- ds-mon-jvm-memory-non-heap-init
- ds-mon-jvm-memory-non-heap-max
- ds-mon-jvm-memory-non-heap-reserved
- ds-mon-jvm-memory-non-heap-used
- ds-mon-jvm-memory-reserved
- ds-mon-jvm-memory-used
- ds-mon-jvm-supported-tls-ciphers
- ds-mon-jvm-supported-tls-protocols
- ds-mon-jvm-threads-blocked-count
- ds-mon-jvm-threads-count
- ds-mon-jvm-threads-daemon-count
- ds-mon-jvm-threads-deadlock-count
- ds-mon-jvm-threads-deadlocks
- ds-mon-jvm-threads-new-count
- ds-mon-jvm-threads-runnable-count
- ds-mon-jvm-threads-terminated-count

- ds-mon-jvm-threads-timed-waiting-count
- ds-mon-jvm-threads-waiting-count
- ds-mon-jvm-vendor
- ds-mon-jvm-version
- ds-mon-last-seen
- ds-mon-ldap-hostport
- ds-mon-ldap-starttls-hostport
- ds-mon-ldaps-hostport
- ds-mon-listen-address
- ds-mon-lost-connections
- ds-mon-major-version
- ds-mon-max-connections
- ds-mon-minor-version
- ds-mon-newest-change-number
- ds-mon-newest-csn-timestamp
- ds-mon-newest-csn
- ds-mon-oldest-change-number
- ds-mon-oldest-csn-timestamp
- ds-mon-oldest-csn
- ds-mon-os-architecture
- ds-mon-os-name
- ds-mon-os-version
- ds-mon-point-version
- ds-mon-process-id
- ds-mon-product-name
- ds-mon-protocol
- ds-mon-receive-delay
- ds-mon-replay-delay
- ds-mon-replayed-internal-updates
- ds-mon-replayed-updates-conflicts-resolved

- ds-mon-replayed-updates-conflicts-unresolved
- ds-mon-replayed-updates
- ds-mon-replication-domain
- ds-mon-replication-protocol-version
- ds-mon-requests-abandon
- ds-mon-requests-add
- ds-mon-requests-bind
- ds-mon-requests-compare
- ds-mon-requests-delete
- ds-mon-requests-extended
- ds-mon-requests-failure-client-invalid-request
- ds-mon-requests-failure-client-redirect
- ds-mon-requests-failure-client-referral
- ds-mon-requests-failure-client-resource-limit
- ds-mon-requests-failure-client-security
- ds-mon-requests-failure-server
- ds-mon-requests-failure-uncategorized
- ds-mon-requests-get
- ds-mon-requests-in-queue
- ds-mon-requests-modify-dn
- ds-mon-requests-modify
- ds-mon-requests-patch
- ds-mon-requests-post
- ds-mon-requests-put
- ds-mon-requests-search-base
- ds-mon-requests-search-one
- ds-mon-requests-search-sub
- ds-mon-requests-submitted
- ds-mon-requests-unbind
- ds-mon-requests-uncategorized

- ds-mon-revision
- ds-mon-sent-updates
- ds-mon-server-id
- ds-mon-server-is-local
- ds-mon-server-state
- ds-mon-short-name
- ds-mon-ssl-encryption
- ds-mon-start-time
- ds-mon-status-last-changed
- ds-mon-status
- ds-mon-system-name
- ds-mon-total-connections
- ds-mon-total-update-entry-count
- ds-mon-total-update-entry-left
- ds-mon-total-update
- ds-mon-updates-already-in-progress
- ds-mon-updates-inbound-queue
- ds-mon-updates-outbound-queue
- ds-mon-updates-totals-per-replay-thread
- ds-mon-vendor-name
- ds-mon-version-qualifier
- ds-mon-working-directory
- ds-private-naming-contexts
- ds-privilege-name
- ds-pwp-account-disabled
- ds-pwp-account-expiration-time
- ds-pwp-account-status-notification-handler
- ds-pwp-allow-expired-password-changes
- ds-pwp-allow-multiple-password-values
- ds-pwp-allow-pre-encoded-passwords

- ds-pwp-allow-user-password-changes
- ds-pwp-attribute-value-check-substrings
- ds-pwp-attribute-value-match-attribute
- ds-pwp-attribute-value-min-substring-length
- ds-pwp-attribute-value-test-reversed-password
- ds-pwp-character-set-allow-unclassified-characters
- ds-pwp-character-set-character-set-ranges
- ds-pwp-character-set-character-set
- ds-pwp-character-set-min-character-sets
- ds-pwp-default-password-storage-scheme
- ds-pwp-deprecated-password-storage-scheme
- ds-pwp-dictionary-case-sensitive-validation
- ds-pwp-dictionary-check-substrings
- ds-pwp-dictionary-data
- ds-pwp-dictionary-min-substring-length
- ds-pwp-dictionary-test-reversed-password
- ds-pwp-expire-passwords-without-warning
- ds-pwp-force-change-on-add
- ds-pwp-force-change-on-reset
- ds-pwp-grace-login-count
- ds-pwp-idle-lockout-interval
- ds-pwp-last-login-time-attribute
- ds-pwp-last-login-time-format
- ds-pwp-last-login-time
- ds-pwp-length-based-max-password-length
- ds-pwp-length-based-min-password-length
- ds-pwp-lockout-duration
- ds-pwp-lockout-failure-count
- ds-pwp-lockout-failure-expiration-interval
- ds-pwp-max-password-age

- ds-pwp-max-password-reset-age
- ds-pwp-min-password-age
- ds-pwp-password-attribute
- ds-pwp-password-change-requires-current-password
- ds-pwp-password-changed-by-required-time
- ds-pwp-password-expiration-time
- ds-pwp-password-expiration-warning-interval
- ds-pwp-password-history-count
- ds-pwp-password-history-duration
- ds-pwp-password-policy-dn
- ds-pwp-previous-last-login-time-format
- ds-pwp-random-password-character-set
- ds-pwp-random-password-format
- ds-pwp-repeated-characters-case-sensitive-validation
- ds-pwp-repeated-characters-max-consecutive-length
- ds-pwp-require-change-by-time
- ds-pwp-require-secure-authentication
- ds-pwp-require-secure-password-changes
- ds-pwp-reset-time
- ds-pwp-similarity-based-min-password-difference
- ds-pwp-skip-validation-for-administrators
- ds-pwp-state-update-failure-policy
- ds-pwp-unique-characters-case-sensitive-validation
- ds-pwp-unique-characters-min-unique-characters
- ds-pwp-warned-time
- ds-rlim-idle-time-limit
- ds-rlim-lookthrough-limit
- ds-rlim-max-candidate-set-size
- ds-rlim-size-limit
- ds-rlim-time-limit

- ds-sync-conflict
- ds-sync-delay
- ds-sync-fractional-exclude
- ds-sync-fractional-include
- ds-sync-generation-id
- ds-sync-hist
- ds-sync-is-available
- ds-sync-state
- ds-target-group-dn
- dSAQuality
- emailAddress
- employeeNumber
- employeeType
- enhancedSearchGuide
- entryDN
- entryUUID
- etag
- facsimileTelephoneNumber
- firstChangeNumber
- followReferrals
- fr-idm-accountStatus
- fr-idm-cluster-json
- fr-idm-condition
- fr-idm-consentedMapping
- fr-idm-custom-attrs
- fr-idm-effectiveAssignment
- fr-idm-effectiveGroup
- fr-idm-effectiveRole
- fr-idm-internal-role-authzmembers-internal-user
- fr-idm-internal-role-authzmembers-managed-user

- fr-idm-internal-user-authzroles-internal-role
- fr-idm-internal-user-authzroles-managed-role
- fr-idm-json
- fr-idm-kbaInfo
- fr-idm-lastSync
- fr-idm-link-firstid-constraint
- fr-idm-link-firstid
- fr-idm-link-qualifier
- fr-idm-link-secondid-constraint
- fr-idm-link-secondid
- fr-idm-link-type
- fr-idm-lock-nodeid
- fr-idm-managed-assignment-json
- fr-idm-managed-group-condition
- fr-idm-managed-group-json
- fr-idm-managed-organization-admin-roleid
- fr-idm-managed-organization-admin
- fr-idm-managed-organization-child
- fr-idm-managed-organization-json
- fr-idm-managed-organization-member
- fr-idm-managed-organization-name
- fr-idm-managed-organization-owner-roleid
- fr-idm-managed-organization-owner
- fr-idm-managed-organization-parent
- fr-idm-managed-role-assignments
- fr-idm-managed-role-json
- fr-idm-managed-user-activate-account
- fr-idm-managed-user-active-date
- fr-idm-managed-user-authzroles-internal-role
- fr-idm-managed-user-authzroles-managed-role

- fr-idm-managed-user-custom-attrs
- fr-idm-managed-user-expire-account
- fr-idm-managed-user-groups
- fr-idm-managed-user-inactive-date
- fr-idm-managed-user-json
- fr-idm-managed-user-manager
- fr-idm-managed-user-memberoforgid
- fr-idm-managed-user-meta
- fr-idm-managed-user-notifications
- fr-idm-managed-user-roles
- fr-idm-name
- fr-idm-notification-json
- fr-idm-password
- fr-idm-preferences
- fr-idm-privilege
- fr-idm-recon-id
- fr-idm-recon-targetIds
- fr-idm-reconassoc-finishtime
- fr-idm-reconassoc-isanalysis
- fr-idm-reconassoc-mapping
- fr-idm-reconassoc-reconid
- fr-idm-reconassoc-sourceresourcecollection
- fr-idm-reconassoc-targetresourcecollection
- fr-idm-reconassocentry-action
- fr-idm-reconassocentry-ambiguoustargetobjectids
- fr-idm-reconassocentry-exception
- fr-idm-reconassocentry-linkqualifier
- fr-idm-reconassocentry-message
- fr-idm-reconassocentry-messagedetail
- fr-idm-reconassocentry-phase

- fr-idm-reconassocentry-reconid
- fr-idm-reconassocentry-situation
- fr-idm-reconassocentry-sourceobjectid
- fr-idm-reconassocentry-status
- fr-idm-reconassocentry-targetobjectid
- fr-idm-relationship-json
- fr-idm-role
- fr-idm-syncqueue-context
- fr-idm-syncqueue-createdate
- fr-idm-syncqueue-mapping
- fr-idm-syncqueue-newobject
- fr-idm-syncqueue-nodeid
- fr-idm-syncqueue-objectrev
- fr-idm-syncqueue-oldobject
- fr-idm-syncqueue-remainingretries
- fr-idm-syncqueue-resourcecollection
- fr-idm-syncqueue-resourceid
- fr-idm-syncqueue-state
- fr-idm-syncqueue-syncaction
- fr-idm-temporal-constraints
- fr-idm-uuid
- fullVendorVersion
- gecos
- generationQualifier
- gidNumber
- givenName
- governingStructureRule
- hasSubordinates
- healthy
- homeDirectory

- homePhone
- homePostalAddress
- host
- houseIdentifier
- includedAttributes
- inetUserHttpURL
- inetUserStatus
- info
- inheritable
- inheritAttribute
- inheritFromBaseRDN
- inheritFromDNAttribute
- inheritFromDNParent
- inheritFromRDNAttribute
- inheritFromRDNType
- initials
- internationaliSDNNumber
- ipHostNumber
- iplanet-am-auth-configuration
- iplanet-am-auth-login-failure-url
- iplanet-am-auth-login-success-url
- iplanet-am-auth-post-login-process-class
- iplanet-am-session-destroy-sessions
- iplanet-am-session-get-valid-sessions
- iplanet-am-session-max-caching-time
- iplanet-am-session-max-idle-time
- iplanet-am-session-max-session-time
- iplanet-am-session-quota-limit
- iplanet-am-session-service-status
- iplanet-am-user-account-life

- iplanet-am-user-admin-start-dn
- iplanet-am-user-alias-list
- iplanet-am-user-auth-config
- iplanet-am-user-auth-modules
- iplanet-am-user-failure-url
- iplanet-am-user-login-status
- iplanet-am-user-password-reset-force-reset
- iplanet-am-user-password-reset-options
- iplanet-am-user-password-reset-question-answer
- iplanet-am-user-service-status
- iplanet-am-user-success-url
- ipNetmaskNumber
- ipNetworkNumber
- ipProtocolNumber
- ipServicePort
- ipServiceProtocol
- ipTnetNumber
- ipTnetTemplateName
- isMemberOf
- janetMailbox
- javaClassName
- javaClassNames
- javaCodebase
- javaDoc
- javaFactory
- javaReferenceAddress
- javaSerializedData
- jpegPhoto
- kbaActiveIndex
- kbaInfo

- kbaInfoAttempts
- knowledgeInformation
- l
- labeledURI
- labeledURL
- lastChangeNumber
- lastEmailSent
- lastExternalChangelogCookie
- lastModifiedBy
- lastModifiedTime
- ldapSyntaxes
- loginShell
- macAddress
- mail
- mailPreferenceOption
- manager
- matchingRules
- matchingRuleUse
- mDRecord
- member
- memberGid
- memberNisNetgroup
- memberof
- memberUid
- memberURL
- mgrpRFC822MailMember
- mobile
- modifiersName
- modifyTimestamp
- mxRecord

- name
- nameForms
- namingContexts
- newRDN
- newSuperior
- nisDomain
- nisMapEntry
- nisMapName
- nisNetgroupTriple
- nisNetIdGroup
- nisNetIdHost
- nisNetIdUser
- nisplusTimeZone
- nisPublicKey
- nisSecretKey
- nsds50ruv
- nSRecord
- nsUniqueId
- numSubordinates
- o
- oath2faEnabled
- oathDeviceProfiles
- objectClass
- objectClasses
- objectclassMap
- oncRpcNumber
- organizationalStatus
- otherMailbox
- ou
- owner

- pager
- personalSignature
- personalTitle
- photo
- physicalDeliveryOfficeName
- postalAddress
- postalCode
- postOfficeBox
- preferredDeliveryMethod
- preferredLanguage
- preferredLocale
- preferredServerList
- preferredTimeZone
- presentationAddress
- printer-aliases
- printer-charset-configured
- printer-charset-supported
- printer-color-supported
- printer-compression-supported
- printer-copies-supported
- printer-current-operator
- printer-delivery-orientation-supported
- printer-document-format-supported
- printer-finishings-supported
- printer-generated-natural-language-supported
- printer-info
- printer-ipp-versions-supported
- printer-job-k-octets-supported
- printer-job-priority-supported
- printer-location

- printer-make-and-model
- printer-media-local-supported
- printer-media-supported
- printer-more-info
- printer-multiple-document-jobs-supported
- printer-name
- printer-natural-language-configured
- printer-number-up-supported
- printer-output-features-supported
- printer-pages-per-minute-color
- printer-pages-per-minute
- printer-print-quality-supported
- printer-resolution-supported
- printer-service-person
- printer-sides-supported
- printer-stacking-order-supported
- printer-uri
- printer-xri-supported
- profileTTL
- protocolInformation
- push2faEnabled
- pushDeviceProfiles
- pwdAccountLockedTime
- pwdAllowUserChange
- pwdAttribute
- pwdChangedTime
- pwdCheckQuality
- pwdExpireWarning
- pwdFailureCountInterval
- pwdFailureTime

- pwdGraceAuthNLimit
- pwdGraceUseTime
- pwdHistory
- pwdInHistory
- pwdLockout
- pwdLockoutDuration
- pwdMaxAge
- pwdMaxFailure
- pwdMinAge
- pwdMinLength
- pwdMustChange
- pwdPolicySubentry
- pwdReset
- pwdSafeModify
- ref
- registeredAddress
- replicaIdentifier
- replicationCSN
- retryLimitNodeCount
- rfc822mailMember
- roleOccupant
- roomNumber
- sambaAcctFlags
- sambaAlgorithmicRidBase
- sambaBadPasswordCount
- sambaBadPasswordTime
- sambaBoolOption
- sambaDomainName
- sambaForceLogoff
- sambaGroupType

- sambaHomeDrive
- sambaHomePath
- sambaIntegerOption
- sambaKickoffTime
- sambaLMPassword
- sambaLockoutDuration
- sambaLockoutObservationWindow
- sambaLockoutThreshold
- sambaLogoffTime
- sambaLogonHours
- sambaLogonScript
- sambaLogonTime
- sambaLogonToChgPwd
- sambaMaxPwdAge
- sambaMinPwdAge
- sambaMinPwdLength
- sambaMungedDial
- sambaNextGroupRid
- sambaNextRid
- sambaNextUserRid
- sambaNTPassword
- sambaOptionName
- sambaPasswordHistory
- sambaPrimaryGroupSID
- sambaPrivilegeList
- sambaProfilePath
- sambaPwdCanChange
- sambaPwdHistoryLength
- sambaPwdLastSet
- sambaPwdMustChange

- sambaRefuseMachinePwdChange
- sambaShareName
- sambaSID
- sambaSIDList
- sambaStringListOption
- sambaStringOption
- sambaTrustFlags
- sambaUserWorkstations
- searchGuide
- searchTimeLimit
- secretary
- seeAlso
- serialNumber
- service-advert-attribute-authenticator
- service-advert-scopes
- service-advert-service-type
- service-advert-url-authenticator
- serviceAuthenticationMethod
- serviceCredentialLevel
- serviceSearchDescriptor
- shadowExpire
- shadowFlag
- shadowInactive
- shadowLastChange
- shadowMax
- shadowMin
- shadowWarning
- singleLevelQuality
- sn
- sOARecord

- SolarisAttrKeyValue
- SolarisAttrLongDesc
- SolarisAttrReserved1
- SolarisAttrReserved2
- SolarisAttrShortDesc
- SolarisAuditAlways
- SolarisAuditNever
- SolarisAuthMethod
- SolarisBindDN
- SolarisBindPassword
- SolarisBindTimeLimit
- SolarisCacheTTL
- SolarisCertificatePassword
- SolarisCertificatePath
- SolarisDataSearchDN
- SolarisKernelSecurityPolicy
- SolarisLDAPServers
- SolarisPreferredServer
- SolarisPreferredServerOnly
- SolarisProfileId
- SolarisProfileType
- SolarisProjectAttr
- SolarisProjectID
- SolarisProjectName
- SolarisSearchBaseDN
- SolarisSearchReferral
- SolarisSearchScope
- SolarisSearchTimeLimit
- SolarisTransportSecurity
- SolarisUserQualifier

- st
- street
- structuralObjectClass
- subschemaSubentry
- subtreeMaximumQuality
- subtreeMinimumQuality
- subtreeSpecification
- sun-fm-saml2-nameid-info
- sun-fm-saml2-nameid-infokey
- sun-printer-bsdaddr
- sun-printer-kvp
- sunAMAuthInvalidAttemptsData
- sunIdentityMSISDNNumber
- sunKeyValue
- sunPluginSchema
- sunserviceID
- sunServiceSchema
- sunsmspriority
- sunxmlKeyValue
- supportedAlgorithms
- supportedApplicationContext
- supportedAuthPasswordSchemes
- supportedControl
- supportedExtension
- supportedFeatures
- supportedLDAPVersion
- supportedSASLMechanisms
- supportedTLSCiphers
- supportedTLSProtocols
- targetDN

- targetEntryUUID
- telephoneNumber
- teletexTerminalIdentifier
- telexNumber
- template-major-version-number
- template-minor-version-number
- template-url-syntax
- textEncodedORAddress
- title
- uddiAccessPoint
- uddiAddressLine
- uddiAuthorizedName
- uddiBindingKey
- uddiBusinessKey
- uddiCategoryBag
- uddiDescription
- uddiDiscoveryURLs
- uddiEMail
- uddiFromKey
- uddiHostingRedirector
- uddiIdentifierBag
- uddiInstanceDescription
- uddiInstanceParms
- uddiIsHidden
- uddiIsProjection
- uddiKeyedReference
- uddiLang
- uddiName
- uddiOperator
- uddiOverviewDescription

- uddiOverviewURL
- uddiPersonName
- uddiPhone
- uddiServiceKey
- uddiSortCode
- uddiTModelKey
- uddiToKey
- uddiUseType
- uddiUUID
- uddiv3BindingKey
- uddiv3BriefResponse
- uddiv3BusinessKey
- uddiv3DigitalSignature
- uddiv3EntityCreationTime
- uddiv3EntityDeletionTime
- uddiv3EntityKey
- uddiv3EntityModificationTime
- uddiv3ExpiresAfter
- uddiv3MaxEntities
- uddiv3NodeId
- uddiv3NotificationInterval
- uddiv3ServiceKey
- uddiv3SubscriptionFilter
- uddiv3SubscriptionKey
- uddiv3TModelKey
- uid
- uidNumber
- uniqueIdentifier
- uniqueMember
- userCertificate

- userClass
- userPassword
- userPKCS12
- userSMIMECertificate
- vendorName
- vendorVersion
- webauthnDeviceProfiles
- winAccountName
- x121Address
- x500UniqueIdentifier

DIT content rules

No elements of this type are defined in the default LDAP schema.

DIT structure rules

This part covers schema definitions for DIT structure rules:

- uddiAddressStructureRule
- uddiBindingTemplateStructureRule
- uddiBusinessEntityStructureRule
- uddiBusinessServiceStructureRule
- uddiContactStructureRule
- uddiPublisherAssertionStructureRule
- uddiTModelInstanceInfoStructureRule
- uddiTModelStructureRule
- uddiv3EntityObituaryStructureRule
- uddiv3SubscriptionStructureRule

Matching rule uses

No elements of this type are defined in the default LDAP schema.

Matching rules

This part covers schema definitions for matching rules:

- directoryStringFirstComponentMatch
- distinguishedNameMatch
- distinguishedNamePatternMatch
- ds-mr-double-metaphone-approx
- ds-mr-user-password-equality
- ds-mr-user-password-exact
- generalizedTimeMatch
- generalizedTimeOrderingMatch
- historicalCsnOrderingMatch
- historicalCsnRangeMatch
- integerFirstComponentMatch
- integerMatch
- integerOrderingMatch
- jsonFirstComponentCaseExactJsonQueryMatch
- jsonFirstComponentCaseIgnoreJsonQueryMatch
- keywordMatch
- nameAndJsonCaseExactJsonIdEqualityMatch
- nameAndJsonCaseIgnoreJsonIdEqualityMatch
- nameAndJsonCaseIgnoreJsonQueryFilterMatch
- nameAndJsonEqualityMatchingRule
- numericStringMatch
- numericStringOrderingMatch
- numericStringSubstringsMatch
- objectIdentifierFirstComponentMatch
- objectIdentifierMatch
- octetStringMatch
- octetStringOrderingMatch
- octetStringSubstringsMatch
- partialDateAndTimeMatchingRule
- presentationAddressMatch

- protocolInformationMatch
- relativeTimeGTOrderingMatch
- relativeTimeLTOrderingMatch
- telephoneNumberMatch
- telephoneNumberSubstringsMatch
- uniqueMemberMatch
- uuidMatch
- uuidOrderingMatch
- wordMatch

Name forms

This part covers schema definitions for name forms:

- uddiAddressNameForm
- uddiBindingTemplateNameForm
- uddiBusinessEntityNameForm
- uddiBusinessServiceNameForm
- uddiContactNameForm
- uddiPublisherAssertionNameForm
- uddiTModelInstanceInfoNameForm
- uddiTModelNameForm
- uddiv3EntityObituaryNameForm
- uddiv3SubscriptionNameForm

Object classes

This part covers schema definitions for object classes:

- account
- alias
- applicationEntity
- applicationProcess
- authPasswordObject

- automount
- automountMap
- bootableDevice
- calEntry
- certificationAuthority-V2
- certificationAuthority
- changeLogEntry
- collectiveAttributeSubentry
- container
- corbaContainer
- corbaObject
- corbaObjectReference
- country
- cRLDistributionPoint
- dcObject
- deltaCRL
- device
- devicePrintProfilesContainer
- deviceProfilesContainer
- dmd
- dNSDomain
- document
- documentSeries
- domain
- domainRelatedObject
- ds-certificate-user
- ds-monitor-backend-db
- ds-monitor-backend-index
- ds-monitor-backend-pluggable
- ds-monitor-backend-proxy

- ds-monitor-backend
- ds-monitor-base-dn
- ds-monitor-branch
- ds-monitor-certificate
- ds-monitor-changelog-domain
- ds-monitor-changelog
- ds-monitor-connected-changelog
- ds-monitor-connected-replica
- ds-monitor-connection-handler
- ds-monitor-disk-space
- ds-monitor-entry-cache
- ds-monitor-health-status
- ds-monitor-http-connection-handler
- ds-monitor-je-database
- ds-monitor-jvm
- ds-monitor-ldap-connection-handler
- ds-monitor-raw-je-database-statistics
- ds-monitor-remote-replica
- ds-monitor-replica-db
- ds-monitor-replica
- ds-monitor-server
- ds-monitor-topology-server
- ds-monitor-work-queue
- ds-monitor
- ds-pwp-attribute-value-validator
- ds-pwp-character-set-validator
- ds-pwp-dictionary-validator
- ds-pwp-length-based-validator
- ds-pwp-password-policy
- ds-pwp-random-generator

- ds-pwp-repeated-characters-validator
- ds-pwp-similarity-based-validator
- ds-pwp-unique-characters-validator
- ds-pwp-validator
- ds-root-dse
- ds-virtual-static-group
- dSA
- DUAConfigProfile
- extensibleObject
- forgerock-am-dashboard-service
- fr-idm-cluster-obj
- fr-idm-generic-obj
- fr-idm-hybrid-obj
- fr-idm-internal-role
- fr-idm-internal-user
- fr-idm-link
- fr-idm-lock
- fr-idm-managed-assignment
- fr-idm-managed-group
- fr-idm-managed-organization
- fr-idm-managed-role
- fr-idm-managed-user-explicit
- fr-idm-managed-user-hybrid-obj
- fr-idm-managed-user
- fr-idm-notification
- fr-idm-recon-clusteredTargetIds
- fr-idm-reconassoc
- fr-idm-reconassocentry
- fr-idm-relationship
- fr-idm-syncqueue

- frCoreToken
- friendlyCountry
- glue
- groupOfEntries
- groupOfMembers
- groupOfNames
- groupOfUniqueNames
- groupOfURLs
- ieee802Device
- inetOrgPerson
- inetuser
- inheritableLDAPSubEntry
- inheritedCollectiveAttributeSubentry
- inheritedFromDNCollectiveAttributeSubentry
- inheritedFromRDNCollectiveAttributeSubentry
- ipHost
- iplanet-am-auth-configuration-service
- iplanet-am-managed-person
- iplanet-am-session-service
- iplanet-am-user-service
- iPlanetPreferences
- ipNetwork
- ipProtocol
- ipService
- ipTnetHost
- ipTnetTemplate
- javaContainer
- javaMarshalledObject
- javaNamingReference
- javaObject

- javaSerializedObject
- kbaInfoContainer
- labeledURIObject
- ldapSubEntry
- locality
- mailGroup
- namedObject
- nisDomainObject
- nisKeyObject
- nisMailAlias
- nisMap
- nisNetgroup
- nisNetId
- nisObject
- nisplusTimeZoneData
- oathDeviceProfilesContainer
- oncRpc
- organization
- organizationalPerson
- organizationalRole
- organizationalUnit
- person
- pilotDSA
- pilotObject
- pilotOrganization
- pilotPerson
- pkiCA
- pkiUser
- posixAccount
- posixGroup

- printerAbstract
- printerIPP
- printerLPR
- printerService
- printerServiceAuxClass
- pushDeviceProfilesContainer
- pwdPolicy
- pwdValidatorPolicy
- qualityLabelledData
- referral
- residentialPerson
- rFC822LocalPart
- room
- sambaConfig
- sambaConfigOption
- sambaDomain
- sambaGroupMapping
- sambaIdmapEntry
- sambaPrivilege
- sambaSamAccount
- sambaShare
- sambaSidEntry
- sambaTrustPassword
- sambaUnixIdPool
- shadowAccount
- simpleSecurityObject
- slpService
- slpServicePrinter
- SolarisAuditUser
- SolarisAuthAttr

- SolarisExecAttr
- SolarisNamingProfile
- SolarisProfAttr
- SolarisProject
- SolarisUserAttr
- strongAuthenticationUser
- subentry
- subschema
- sunAMAuthAccountLockout
- sunFMSAML2NameIdentifier
- sunPrinter
- sunRealmService
- sunservice
- sunservicecomponent
- top
- uddiAddress
- uddiBindingTemplate
- uddiBusinessEntity
- uddiBusinessService
- uddiContact
- uddiPublisherAssertion
- uddiTModel
- uddiTModelInstanceInfo
- uddiv3EntityObituary
- uddiv3Subscription
- uidObject
- untypedObject
- userSecurityInformation
- webauthnDeviceProfilesContainer

Syntaxes

This part covers schema definitions for syntaxes:

- Attribute Type Description
- Authentication Password Syntax
- Binary
- Bit String
- Boolean
- Certificate
- Certificate List
- Certificate Pair
- Collective Conflict Behavior
- Counter metric
- Country String
- CSN (Change Sequence Number)
- Delivery Method
- Directory String
- Distinguished Name Pattern Match Assertion
- DIT Content Rule Description
- DIT Structure Rule Description
- DN
- Duration in milli-seconds
- Enhanced Guide
- Expression syntax for Boolean
- Expression syntax for Certificate
- Expression syntax for Directory String
- Expression syntax for DN
- Expression syntax for Generalized Time
- Expression syntax for IA5 String
- Expression syntax for Integer

- Expression syntax for Numeric String
- Expression syntax for Octet String
- Expression syntax for OID
- Expression syntax for Sun-defined Access Control Information
- Expression syntax for User Password
- Facsimile Telephone Number
- Fax
- Filesystem path
- Generalized Time
- Guide
- Host port
- IA5 String
- Integer
- JPEG
- Json
- Json Query
- LDAP Syntax Description
- Matching Rule Description
- Matching Rule Use Description
- Name and JSON query filter assertion
- Name and JSON with id
- Name and Optional UID
- Name Form Description
- Numeric String
- Object Class Description
- Octet String
- OID
- Other Mailbox
- Postal Address
- Presentation Address

- Printable String
- Protocol Information
- Size in bytes
- Substring Assertion
- Subtree Specification
- Summary metric
- Sun-defined Access Control Information
- Supported Algorithm
- Telephone Number
- Teletex Terminal Identifier
- Telex Number
- Timer metric
- User Password
- UTC Time
- UUID
- X.509 Certificate Exact Assertion

Log message reference

This reference lists server error messages by category.

Category: Access control

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 104

Severity: ERROR
Message: The global access control policy defined in "%s" could not be parsed because it contains an invalid target DN pattern "%s".

ID: 105

Severity: ERROR
Message: The global access control policy defined in "%s" could not be parsed because it contains an invalid user DN pattern "%s".

ID: 107

Severity: ERROR
Message: The global access control policy defined in "%s" could not be parsed because it contains an unrecognized control alias "%s".

ID: 108

Severity: ERROR
Message: The global access control policy defined in "%s" could not be parsed because it contains an unrecognized extended operation alias "%s".

Category: Administration tools

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: N/A

Severity: ERROR
Message: Could not find the "%s" command in "%s".

Category: Configuration management

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 7

Severity: ERROR
Message: Unable to set the value for Boolean configuration attribute %s because the provided value %s was not either 'true' or 'false'.

ID: 11

Severity: ERROR
Message: Unable to set the value for integer configuration attribute %s because the provided value %s cannot be interpreted as an integer value: %s.

ID: 12

Severity: ERROR
Message: Unable to set the value for configuration attribute %s because the provided value %d is less than the lowest allowed value of %d.

ID: 13

Severity: ERROR
Message: Unable to set the value for configuration attribute %s because the provided value %d is greater than the largest allowed value of %d.

ID: 26

Severity: ERROR
Message: The specified configuration file %s does not exist or is not readable.

ID: 28

Severity: ERROR
Message: An error occurred while attempting to open the configuration file %s for reading: %s.

ID: 33

Severity: ERROR
Message: The first entry read from LDIF configuration file %s had a DN of "%s" rather than the expected "%s" which should be used as the Directory Server configuration root.

ID: 34

Severity: ERROR
Message: An unexpected error occurred while attempting to process the Directory Server configuration file %s: %s.

ID: 38

Severity: ERROR
Message: Unable to determine the Directory Server instance root from either an environment variable or based on the location of the configuration file. Please set an environment variable named %s with a value containing the absolute path to the server installation root.

ID: 39

Severity: ERROR
Message: An unexpected error occurred while trying to write configuration entry %s to LDIF: %s.

ID: 41

Severity: ERROR
Message: The Directory Server configuration may not be altered by importing a new configuration from LDIF.

ID: 49

Severity: ERROR
Message: An error occurred while attempting to create a Directory Server logger from the information in configuration entry %s: %s.

ID: 50

Severity: ERROR
Message: Configuration entry %s does not contain a valid objectclass for a Directory Server access, error, or debug logger definition.

ID: 54

Severity: ERROR
Message: Class %s specified in attribute ds-cfg-java-class of configuration entry %s cannot be instantiated as a Directory Server access logger: %s.

ID: 55

Severity: ERROR
Message: Class %s specified in attribute ds-cfg-java-class of configuration entry %s cannot be instantiated as a Directory Server error logger: %s.

ID: 56

Severity: ERROR
Message: Class %s specified in attribute ds-cfg-java-class of configuration entry %s cannot be instantiated as a Directory Server debug logger: %s.

ID: 74

Severity: ERROR
Message: Configuration entry %s does not contain attribute %s (or that attribute exists but is not accessible using JMX).

ID: 78

Severity: ERROR
Message: There is no method %s for any invokable component registered with configuration entry %s.

ID: 83

Severity: ERROR
Message: The Directory Server could not register a JMX MBean for the component associated with configuration entry %s: %s.

ID: 84

Severity: ERROR
Message: An unexpected error occurred while trying to export the Directory Server configuration to LDIF: %s.

ID: 94

Severity: ERROR
Message: Worker thread "%s" has experienced too many repeated failures while attempting to retrieve the next operation from the work queue (%d failures experienced, maximum of %d failures allowed). This worker thread will be destroyed.

ID: 95

Severity: ERROR
Message: A problem occurred while trying to create and start an instance of class %s to use as a monitor provider for the Directory Server work queue: %s. No monitor information will be available for the work queue.

ID: 103

Severity: ERROR
Message: An unexpected error occurred while trying to register the configuration handler base DN "%s" as a private suffix with the Directory Server: %s.

ID: 105

Severity: ERROR
Message: The entry cn=Backends,cn=config does not appear to exist in the Directory Server configuration. This is a required entry.

ID: 107

Severity: ERROR
Message: An unexpected error occurred while interacting with backend configuration entry %s: %s.

ID: 112

Severity: ERROR
Message: An unexpected error occurred while attempting to determine whether the backend associated with configuration entry %s should be enabled or disabled: %s. It will be disabled.

ID: 115

Severity: ERROR
Message: The Directory Server was unable to load class %s and use it to create a backend instance as defined in configuration entry %s. The error that occurred was: %s. This backend will be disabled.

ID: 116

Severity: ERROR
Message: An error occurred while trying to initialize a backend loaded from class %s with the information in configuration entry %s: %s. This backend will be disabled.

ID: 154

Severity: ERROR
Message: An error occurred while trying to initialize a connection handler loaded from class %s with the information in configuration entry %s: %s. This connection handler will be disabled.

ID: 188

Severity: ERROR
Message: Unable to read the Directory Server schema definitions because the schema directory %s does not exist.

ID: 189

Severity: ERROR
Message: Unable to read the Directory Server schema definitions because the schema directory %s exists but is not a directory.

ID: 190

Severity: ERROR
Message: Unable to read the Directory Server schema definitions from directory %s because an unexpected error occurred while trying to list the files in that directory: %s.

ID: 200

Severity: ERROR
Message: An unexpected error occurred that prevented the server from installing its default entry cache framework: %s.

ID: 202

Severity: ERROR
Message: An error occurred while attempting to initialize an instance of class %s for use as the Directory Server entry cache: %s. As a result, the entry cache will be disabled.

ID: 203

Severity: ERROR
Message: The configuration for the entry cache defined in configuration entry %s was not acceptable: %s.

ID: 204

Severity: ERROR
Message: The configuration for the entry cache defined in configuration entry %s was not acceptable: the entry cache level %d is already in use.

ID: 245

Severity: ERROR
Message: An error occurred while attempting to initialize an instance of class %s as a Directory Server plugin using the information in configuration entry %s: %s. This plugin will be disabled.

ID: 256

Severity: ERROR
Message: Class %s specified in configuration entry %s does not contain a valid extended operation handler implementation: %s.

ID: 261

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as an extended operation handler as defined in configuration entry %s: %s.

ID: 277

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as a SASL mechanism handler as defined in configuration entry %s: %s.

ID: 278

Severity: ERROR
Message: Entry %s cannot be removed from the Directory Server configuration because that DN does not have a parent.

ID: 280

Severity: ERROR
Message: Entry %s cannot be added to the Directory Server configuration because another configuration entry already exists with that DN.

ID: 281

Severity: ERROR
Message: Entry %s cannot be added to the Directory Server configuration because that DN does not have a parent.

ID: 282

Severity: ERROR
Message: Entry %s cannot be added to the Directory Server configuration because its parent entry %s does not exist.

ID: 283

Severity: ERROR
Message: The Directory Server is unwilling to add configuration entry %s because one of the add listeners registered with the parent entry %s rejected this change with the message: %s.

ID: 284

Severity: ERROR
Message: An unexpected error occurred while attempting to add configuration entry %s as a child of entry %s: %s.

ID: 285

Severity: ERROR
Message: Entry %s cannot be removed from the Directory Server configuration because the specified entry does not exist.

ID: 286

Severity: ERROR
Message: Entry %s cannot be removed from the Directory Server configuration because the specified entry has one or more subordinate entries.

ID: 287

Severity: ERROR
Message: Entry %s cannot be removed from the Directory Server configuration because the entry does not have a parent and removing the configuration root entry is not allowed.

ID: 288

Severity: ERROR
Message: Entry %s cannot be removed from the Directory Server configuration because one of the delete listeners registered with the parent entry %s rejected this change with the message: %s.

ID: 289

Severity: ERROR
Message: An unexpected error occurred while attempting to remove configuration entry %s as a child of entry %s: %s.

ID: 290

Severity: ERROR
Message: Entry %s cannot be modified because the specified entry does not exist.

ID: 291

Severity: ERROR
Message: Entry %s cannot be modified because one of the configuration change listeners registered for that entry rejected the change: %s.

ID: 292

Severity: ERROR
Message: An unexpected error occurred while attempting to modify configuration entry %s as a child of entry %s: %s.

ID: 300

Severity: ERROR
Message: An error occurred while attempting to export the new Directory Server configuration to file %s: %s.

ID: 301

Severity: ERROR
Message: An error occurred while attempting to rename the new Directory Server configuration from file %s to %s: %s.

ID: 302

Severity: ERROR
Message: Modify DN operations are not allowed in the Directory Server configuration.

ID: 376

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as a password storage scheme as defined in configuration entry %s: %s.

ID: 377

Severity: ERROR
Message: Unable to add a new password storage scheme entry with DN %s because there is already a storage scheme registered with that DN.

ID: 422

Severity: ERROR
Message: The Directory Server was unable to acquire a shared lock for backend %s: %s. This generally means that the backend is in use by a process that requires an exclusive lock (e.g., importing from LDIF or restoring a backup). This backend will be disabled.

ID: 442

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as an identity mapper as defined in configuration entry %s: %s.

ID: 464

Severity: ERROR
Message: An error occurred while attempting to instantiate class %s referenced in synchronization provider configuration entry %s: %s.

ID: 465

Severity: ERROR
Message: An error occurred while attempting to initialize the Directory Server synchronization provider referenced in configuration entry %s: %s.

ID: 489

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as a password validator as defined in configuration entry %s: %s.

ID: 505

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as a password generator as defined in configuration entry %s: %s.

ID: 514

Severity: ERROR
Message: No password policies have been defined below the cn=Password Policies,cn=config entry in the Directory Server configuration. At least one password policy configuration must be defined.

ID: 515

Severity: ERROR
Message: The password policy defined in configuration entry %s is invalid: %s.

ID: 516

Severity: ERROR
Message: The Directory Server default password policy is defined as %s, but that entry does not exist or is not below the password policy configuration base cn=Password Policies,cn=config.

ID: 533

Severity: ERROR
Message: An error occurred while attempting to instantiate class %s referenced in the access control configuration entry %s: %s.

ID: 558

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as an account status notification handler as defined in configuration entry %s: %s.

ID: 559

Severity: ERROR
Message: Unable to add a new account status notification handler entry with DN %s because there is already a notification handler registered with that DN.

ID: 563

Severity: ERROR
Message: An error occurred while attempting to apply the changes contained in file %s to the server configuration at startup: %s.

ID: 565

Severity: ERROR
Message: One or more errors occurred while applying changes on server startup: %s.

ID: 567

Severity: ERROR
Message: Configuration entry %s does not contain a valid value for configuration attribute ds-cfg-db-directory-permissions (It should be an UNIX permission mode in three-digit octal notation.).

ID: 568

Severity: ERROR
Message: Invalid UNIX file permissions %s does not allow read and write access to the backend database directory by the backend.

ID: 571

Severity: ERROR
Message: No default password policy is configured for the Directory Server. The default password policy must be specified by the ds-cfg-default-password-policy attribute in the cn=config entry.

ID: 574

Severity: ERROR
Message: An error occurred while trying to create the configuration archive directory %s: %s.

ID: 575

Severity: ERROR
Message: An error occurred while trying to write the current configuration to the configuration archive: %s.

ID: 598

Severity: ERROR
Message: You do not have sufficient privileges to perform add operations in the Directory Server configuration.

ID: 599

Severity: ERROR
Message: You do not have sufficient privileges to perform delete operations in the Directory Server configuration.

ID: 600

Severity: ERROR
Message: You do not have sufficient privileges to perform modify operations in the Directory Server configuration.

ID: 601

Severity: ERROR
Message: You do not have sufficient privileges to perform modify DN operations in the Directory Server configuration.

ID: 602

Severity: ERROR
Message: You do not have sufficient privileges to perform search operations in the Directory Server configuration.

ID: 603

Severity: ERROR
Message: You do not have sufficient privileges to change the set of default root privileges.

ID: 614

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as a certificate mapper as defined in configuration entry %s: %s.

ID: 627

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as a key manager provider as defined in configuration entry %s: %s.

ID: 640

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as a trust manager provider as defined in configuration entry %s: %s.

ID: 643

Severity: ERROR
Message: Unable to retrieve JMX attribute %s associated with configuration entry %s: %s.

ID: 645

Severity: ERROR
Message: %s.%s returned a result of null for entry %s.

ID: 646

Severity: ERROR
Message: %s.%s failed for entry %s: result code=%s, admin action required=%b, messages="%s".

ID: 649

Severity: ERROR
Message: Unable to parse value "%s" from config entry "%s" as a valid search filter: %s.

ID: 650

Severity: ERROR
Message: An error occurred while trying to load an instance of class %s referenced in configuration entry %s as a virtual attribute provider: %s.

ID: 651

Severity: ERROR
Message: The virtual attribute configuration in entry "%s" is not valid because attribute type %s is single-valued but provider %s may generate multiple values.

ID: 652

Severity: ERROR
Message: The virtual attribute configuration in entry "%s" is not valid because attribute type %s is single-valued but the conflict behavior is configured to merge real and virtual values.

ID: 653

Severity: ERROR
Message: Configuration entry %s cannot be modified because the change would alter its structural object class.

ID: 654

Severity: ERROR
Message: An error occurred while attempting to calculate a SHA-1 digest of file %s: %s.

ID: 656

Severity: ERROR
Message: The Directory Server encountered an error while attempting to determine whether the configuration file %s has been externally edited with the server online, and/or trying to preserve such changes: %s. Any manual changes made to that file may have been lost.

ID: 657

Severity: ERROR
Message: Class %s specified in attribute ds-cfg-java-class of configuration entry %s cannot be instantiated as a Directory Server log rotation policy: %s.

ID: 658

Severity: ERROR
Message: Class %s specified in attribute ds-cfg-java-class of configuration entry %s cannot be instantiated as a Directory Server log retention policy: %s.

ID: 659

Severity: ERROR
Message: An error occurred while attempting to create a Directory Server log rotation policy from the information in configuration entry %s: %s.

ID: 660

Severity: ERROR
Message: An error occurred while attempting to create a Directory Server log retention policy from the information in configuration entry %s: %s.

ID: 661

Severity: ERROR
Message: An error occurred while attempting to create a text writer for a Directory Server logger from the information in configuration entry %s: %s.

ID: 674

Severity: ERROR
Message: Unable to initialize an instance of class %s as a work queue as specified in configuration entry %s: %s.

ID: 676

Severity: ERROR
Message: The attempt to apply the configuration add failed. The preliminary checks were all successful and the entry was added to the server configuration, but at least one of the configuration add listeners reported an error when attempting to apply the change: %s.

ID: 677

Severity: ERROR
Message: The attempt to apply the configuration delete failed. The preliminary checks were all successful and the entry was removed from the server configuration, but at least one of the configuration delete listeners reported an error when attempting to apply the change: %s.

ID: 678

Severity: ERROR
Message: The attempt to apply the configuration modification failed. The preliminary checks were all successful and the modified entry was written to the server configuration, but at least one of the configuration change listeners reported an error when attempting to apply the change: %s.

ID: 679

Severity: ERROR
Message: The configuration for the key manager provider defined in configuration entry %s was not acceptable: %s.

ID: 680

Severity: ERROR
Message: The configuration for the trust manager provider defined in configuration entry %s was not acceptable: %s.

ID: 681

Severity: ERROR
Message: The configuration for the trust manager provider defined in configuration entry %s was not acceptable: %s.

ID: 682

Severity: ERROR
Message: The configuration for the account status notification handler defined in configuration entry %s was not acceptable: %s.

ID: 684

Severity: ERROR
Message: The configuration for the certificate mapper defined in configuration entry %s was not acceptable: %s.

ID: 687

Severity: ERROR
Message: The configuration for the identity mapper defined in configuration entry %s was not acceptable: %s.

ID: 689

Severity: ERROR
Message: The configuration for the password generator defined in configuration entry %s was not acceptable: %s.

ID: 690

Severity: ERROR
Message: The configuration for the password storage scheme defined in configuration entry %s was not acceptable: %s.

ID: 691

Severity: ERROR
Message: The configuration for the password validator defined in configuration entry %s was not acceptable: %s.

ID: 692

Severity: ERROR
Message: The configuration for the plugin defined in configuration entry %s was not acceptable: %s.

ID: 693

Severity: ERROR
Message: The configuration for the SASL mechanism handler defined in configuration entry %s was not acceptable: %s.

ID: 694

Severity: ERROR
Message: The configuration for the virtual attribute provider defined in configuration entry %s was not acceptable: %s.

ID: 695

Severity: ERROR
Message: The configuration for the alert handler defined in configuration entry %s was not acceptable: %s.

ID: 696

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as an alert handler as defined in configuration entry %s: %s.

ID: 698

Severity: ERROR
Message: An error occurred while attempting to open the current configuration file %s for reading in order to copy it to the ".startok" file: %s.

ID: 700

Severity: ERROR
Message: An error occurred while attempting to copy the current configuration from file %s into temporary file %s for use as the ".startok" configuration file: %s.

ID: 701

Severity: ERROR
Message: An error occurred while attempting to rename file %s to %s for use as the ".startok" configuration file: %s.

ID: 704

Severity: ERROR
Message: An error occurred while trying to parse and validate Berkeley DB JE property %s: %s.

ID: 705

Severity: ERROR
Message: An error occurred while trying to parse and validate Berkeley DB JE property %s: the property does not follow a singular property=value form.

ID: 706

Severity: ERROR
Message: An error occurred while trying to parse and validate Berkeley DB JE property %s: the property shadows configuration attribute %s.

ID: 707

Severity: ERROR
Message: An error occurred while trying to parse and validate Berkeley DB JE property %s: the property is already defined for this component.

ID: 709

Severity: ERROR
Message: An error occurred while attempting to open the configured log file %s for logger %s: %s.

ID: 715

Severity: ERROR
Message: Invalid UNIX file permissions %s does not allow write access to the log file by the log publisher.

ID: 716

Severity: ERROR
Message: Invalid UNIX file permissions %s: %s.

ID: 726

Severity: ERROR
Message: The configuration entry '%s' is currently defined to be the default password policy, however it is not a password policy.

ID: 727

Severity: ERROR
Message: The default password policy value '%s' is invalid because it refers to an authentication policy which is not a password policy.

ID: 728

Severity: ERROR
Message: The timestamp format string "%s" is not a valid format string. The format string should conform to the syntax described in the documentation for the "java.text.SimpleDateFormat" class.

ID: 729

Severity: ERROR
Message: The access log filtering criteria defined in "%s" could not be parsed because it contains an invalid user DN pattern "%s".

ID: 730

Severity: ERROR
Message: The access log filtering criteria defined in "%s" could not be parsed because it contains an invalid target DN pattern "%s".

ID: 732

Severity: ERROR
Message: Class %s specified in attribute ds-cfg-java-class of configuration entry %s cannot be instantiated as a Directory Server HTTP access logger: %s.

ID: 735

Severity: ERROR
Message: An error occurred while attempting to update a Directory Server logger from the information in configuration entry %s: %s.

ID: 737

Severity: ERROR
Message: Cannot configure java.util.logging root logger level: %s. java.util.logging support is now disabled.

ID: 738

Severity: ERROR
Message: An error occurred while trying to initialize an instance of class %s as an HTTP endpoint as defined in configuration entry %s: %s.

ID: 739

Severity: ERROR
Message: An error occurred while starting the HTTP endpoint as defined in configuration entry %s: %s.

ID: 741

Severity: ERROR
Message: The HTTP endpoint configuration defined in %s is invalid: %s.

ID: 743

Severity: ERROR
Message: Cannot initialize the configuration framework: %s.

ID: 744

Severity: ERROR
Message: Unable to retrieve children of configuration entry with dn: %s.

ID: 745

Severity: ERROR
Message: Unable to load the configuration-enabled schema: %s.

ID: 746

Severity: ERROR
Message: Backend config error when trying to delete an entry: %s.

ID: 747

Severity: ERROR
Message: The HTTP endpoint configuration defined in %s is referencing a non existing authorization DN %s.

ID: 748

Severity: ERROR
Message: The HTTP endpoint configuration defined in %s is referencing mutually exclusive authorization DNs %s and %s.

ID: 749

Severity: ERROR
Message: Unable to read the configuration from %s in the REST2LDAP endpoint configuration entry %s: %s.

ID: 750

Severity: ERROR
Message: Invalid JSON element %s from %s in the REST2LDAP endpoint configuration entry %s: %s.

ID: 751

Severity: ERROR
Message: Invalid configuration element from %s in the REST2LDAP endpoint configuration entry %s: %s.

ID: 752

Severity: ERROR
Message: The OAuth2 authorization mechanism defined in %s contains an invalid JSON Pointer %s: %s.

ID: 753

Severity: ERROR
Message: The authorization mechanism defined in %s is referencing a non-existing or non-readable directory: %s.

ID: 754

Severity: ERROR
Message: The authorization mechanism defined in %s is referencing a non existing DN: %s.

ID: 755

Severity: ERROR
Message: The authorization mechanism defined in %s is referencing an invalid URL %s: %s.

ID: 756

Severity: ERROR
Message: Unable to configure the authorization mechanism defined in %s: %s.

ID: 757

Severity: ERROR
Message: The requested admin API version '%s' is unsupported. This endpoint only supports the following admin API version(s): %s.

ID: 758

Severity: ERROR
Message: The configuration of schema provider '%s' is not acceptable for the following reasons: %s.

ID: 759

Severity: ERROR
Message: The schema provider class '%s' could not be instantiated or initialized with the configuration '%s' : %s.

ID: 760

Severity: ERROR
Message: The core schema provider defined by '%s' has been disabled. The core schema must always be enabled.

ID: 763

Severity: ERROR
Message: Unable to configure the backend '%s' because one of its base DNs is the empty DN.

ID: 764

Severity: ERROR
Message: Cannot configure new SSL protocols because the following protocols are not supported: %s. Look for supported protocols in 'cn=jvm,cn=monitor'.

ID: 765

Severity: ERROR
Message: Cannot configure new SSL cipher suites because the following cipher suites are not supported: %s. Look for supported cipher suites in 'cn=jvm,cn=monitor'.

ID: 766

Severity: ERROR
Message: The metric name pattern to exclude '%s' cannot be parsed as a valid regular expression due to the following error: '%s'.

ID: 767

Severity: ERROR
Message: The metric name pattern to include '%s' cannot be parsed as a valid regular expression due to the following error: '%s'.

ID: 772

Severity: ERROR
Message: The list of keys defined for the JSON matching rule contains an invalid JSON pointer : %s.

ID: 774

Severity: ERROR
Message: Cannot create the property resolver due to the following error: '%s'.

ID: 775

Severity: ERROR
Message: Error creating SSL socket factory: %s.

ID: 776

Severity: ERROR
Message: The smtp-server value '%s' is invalid: %s.

ID: 777

Severity: ERROR
Message: Unable to resolve the host for the listen address '%s' of the LDAP connection handler %s.

Category: Connections and protocols

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 159

Severity: ERROR
Message: The server attempted to send a response to the %s operation (conn=%d, op=%d), but the operation did not have a result code. This could indicate that the operation did not complete properly or that it is one that is not allowed to have a response. Using a generic 'Operations Error' response.

ID: 160

Severity: ERROR
Message: The server attempted to send a response to the %s operation (conn=%d, op=%d), but this type of operation is not allowed to have responses. Backtrace: %s.

ID: 180

Severity: ERROR
Message: The connection attempt from client %s to %s has been rejected because the client was included in one of the denied address ranges.

ID: 181

Severity: ERROR
Message: The connection attempt from client %s to %s has been rejected because the client was not included in one of the allowed address ranges.

ID: 202

Severity: ERROR
Message: Terminating this connection because the client sent an invalid message of type %s (LDAP message ID %d) that is not allowed for request messages.

ID: 203

Severity: ERROR
Message: An unexpected failure occurred while trying to process a request of type %s (LDAP message ID %d): %s. The client connection will be terminated.

ID: 205

Severity: ERROR
Message: This client connection is being terminated because a protocol error occurred while trying to process a bind request. The LDAP message ID was %d and the error message for the bind response was %s.

ID: 206

Severity: ERROR
Message: An extended response message would have been sent to an LDAPv2 client (connection ID=%d, operation ID=%d): %s. LDAPv2 does not allow extended operations, so this response will not be sent.

ID: 207

Severity: ERROR
Message: A search performed by an LDAPv2 client (connection ID=%d, operation ID=%d) would have included a search result reference %s. Referrals are not allowed for LDAPv2 clients, so this search reference will not be sent.

ID: 208

Severity: ERROR
Message: The original result code for this message was 10 but this result is not allowed for LDAPv2 clients.

ID: 209

Severity: ERROR
Message: The response included one or more referrals, which are not allowed for LDAPv2 clients. The referrals included were: %s.

ID: 210

Severity: ERROR
Message: The Directory Server has been configured to deny access to LDAPv2 clients. This connection will be closed.

ID: 211

Severity: ERROR
Message: The client with connection ID %d authenticated to the Directory Server using LDAPv2, but attempted to send an extended operation request (LDAP message ID %d), which is not allowed for LDAPv2 clients. The connection will be terminated.

ID: 214

Severity: ERROR
Message: The attempt to register this connection with the Directory Server was rejected. This indicates that the server already has the maximum allowed number of concurrent connections established.

ID: 272

Severity: ERROR
Message: StartTLS cannot be enabled on this LDAP client connection because the corresponding LDAP connection handler is configured to reject StartTLS requests. The use of StartTLS can be enabled using the ds-cfg-allow-start-tls configuration attribute.

ID: 296

Severity: ERROR
Message: User %s specified in the proxied authorization V1 control does not exist in the Directory Server.

ID: 299

Severity: ERROR
Message: Unable to process proxied authorization V2 control because it contains an authorization ID based on a username and no proxied authorization identity mapper is configured in the Directory Server.

ID: 300

Severity: ERROR
Message: The authorization ID "%s" contained in the proxied authorization V2 control is invalid because it does not start with "dn:" to indicate a user DN or "u:" to indicate a username.

ID: 301

Severity: ERROR
Message: User %s specified in the proxied authorization V2 control does not exist in the Directory Server.

ID: 372

Severity: ERROR
Message: Use of the proxied authorization V1 control for user %s is not allowed by the password policy configuration.

ID: 431

Severity: ERROR
Message: LDAPv2 clients are not allowed to use request controls.

ID: 438

Severity: ERROR
Message: You do not have sufficient privileges to perform search operations through JMX.

ID: 439

Severity: ERROR
Message: You do not have sufficient privileges to establish the connection through JMX. At least JMX_READ privilege is required.

ID: 440

Severity: ERROR
Message: User %s does not exist in the directory.

ID: 447

Severity: ERROR
Message: An error occurred while trying to read a change record from the LDIF file: %s. This change will be skipped but processing on the LDIF file will continue.

ID: 448

Severity: ERROR
Message: An error occurred while trying to read a change record from the LDIF file: %s. No further processing on this LDIF file can be performed.

ID: 454

Severity: ERROR
Message: An I/O error occurred while the LDIF connection handler was processing LDIF file %s: %s.

ID: 455

Severity: ERROR
Message: An error occurred while the LDIF connection handler was attempting to rename partially-processed file from %s to %s: %s.

ID: 456

Severity: ERROR
Message: An error occurred while the LDIF connection handler was attempting to delete processed file %s: %s.

ID: 1462

Severity: ERROR
Message: No Configuration was defined for this connection handler. The configuration parameters ds-cfg-listen-port and ds-cfg-trap-port are required by the connection handler to start.

ID: 1463

Severity: ERROR
Message: Traps Destination %s is an unknown host. Traps will not be sent to this destination.

ID: 1464

Severity: ERROR
Message: You do not have the appropriate OpenDMK jar files to enable the SNMP Connection Handler. The jdmkrt.jar file must be installed into the 'extlib' directory. The SNMP connection Handler could not be started.

ID: 1465

Severity: ERROR
Message: Cannot initialize the SNMP Connection Handler. Please check the configuration attributes.

ID: 1466

Severity: ERROR
Message: No valid trap destinations has been found. No trap will be sent.

ID: 1504

Severity: ERROR
Message: An error occurred while attempting to initialize the SSL context for use in the LDAP Connection Handler: %s.

ID: 1505

Severity: ERROR
Message: The Directory Server does not support LDAP protocol version %d. This connection will be closed.

ID: 1507

Severity: ERROR
Message: The required OpenDMK classes could not be loaded using jar file '%s'. Verify that the jar file is not corrupted.

ID: 1508

Severity: ERROR
Message: Cannot decode the provided control %s because an error occurred while attempting to decode the control value: %s.

ID: 1509

Severity: ERROR
Message: Unable to process the provided internal modifications request control because it did not contain an origin.

ID: 1510

Severity: ERROR
Message: Cannot decode the provided entry changelog notification control because an error occurred while attempting to decode the control value: %s.

ID: 1512

Severity: ERROR
Message: Unable to process the provided replication context request control because it did not contain a CSN.

ID: 1513

Severity: ERROR
Message: Unable to process the provided replication context request control because it did not contain an entry UUID.

ID: 1516

Severity: ERROR
Message: An error occurred during multi-stage authentication: '%s'.

ID: 1518

Severity: ERROR
Message: Unable to process request '%s' received for internal client connection: %s.

ID: 1520

Severity: ERROR
Message: No result received after completion for request '%s' received for internal client connection.

ID: 1522

Severity: ERROR
Message: Unable to process request '%s' received for JMX client because this type of request is not supported for JMX.

ID: 1524

Severity: ERROR
Message: Unable to process response received for JMX client connection for request '%s' because the response '%s' is not of any of the expected types.

ID: 1525

Severity: ERROR
Message: Authorization as '%s' specified in the proxied authorization control is not permitted.

ID: 1526

Severity: ERROR
Message: The key with alias '%s' used by '%s' could not be found, which may cause subsequent SSL connections to fail. Verify that the underlying keystore is properly configured.

ID: 1527

Severity: ERROR
Message: No usable key was found for '%s', which may cause subsequent SSL connections to fail. Verify that the underlying keystore is properly configured.

ID: 1529

Severity: ERROR
Message: Could not write data to the client for %s.

ID: 1533

Severity: ERROR
Message: Use of the proxied authorization V2 control for user %s is not allowed: the account is disabled.

ID: 1534

Severity: ERROR
Message: Use of the proxied authorization V2 control for user %s is not allowed: the account is expired.

ID: 1535

Severity: ERROR
Message: Use of the proxied authorization V2 control for user %s is not allowed: the account is locked.

ID: 1536

Severity: ERROR
Message: Use of the proxied authorization V2 control for user %s is not allowed: the account's password is expired.

ID: 1537

Severity: ERROR
Message: The connection attempt from client %s to %s has been rejected because there are too many open connections from this client.

ID: 1538

Severity: ERROR
Message: Unable to process the provided server-side sort request control: %s.

ID: 1540

Severity: ERROR
Message: Unable to process HTTP request '%s': %s.

ID: 1541

Severity: ERROR
Message: Unable to write HTTP response to the client '%s': %s.

ID: 1543

Severity: ERROR
Message: Error while starting the HTTP application: %s.

ID: 1545

Severity: ERROR
Message: cancel() invoked.

ID: 1546

Severity: ERROR
Message: JMX connection %s with JMX connection ID '%s' has been disconnected because it was finalized by GC.

ID: 1547

Severity: ERROR
Message: JMX connection %s with JMX connection ID '%s' has been disconnected because it received a '%s' notification.

ID: 1548

Severity: ERROR
Message: The key with alias '%s' used by '%s' is not valid yet. The key will be used, but SSL connections may fail depending on the validation performed by the peer. Verify that the underlying keystore is properly configured with a valid key.

ID: 1549

Severity: ERROR
Message: The key with alias '%s' used by '%s' has expired. The key will be used, but SSL connections may fail depending on the validation performed by the peer. Verify that the underlying keystore is properly configured with a valid key.

Category: Core server

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 1

Severity: ERROR
Message: Abandon requests cannot be canceled.

ID: 2

Severity: ERROR
Message: Bind requests cannot be canceled.

ID: 3

Severity: ERROR
Message: Unbind requests cannot be canceled.

ID: 108

Severity: ERROR
Message: %s encountered an uncaught exception while processing operation %s: %s.

ID: 118

Severity: ERROR
Message: The Directory Server is currently running. The configuration may not be bootstrapped while the server is online.

ID: 122

Severity: ERROR
Message: The Directory Server may not be started before the configuration has been bootstrapped.

ID: 123

Severity: ERROR
Message: The Directory Server may not be started while it is already running. Please stop the running instance before attempting to start it again.

ID: 138

Severity: ERROR
Message: An error occurred while attempting to create the JMX MBean server that will be used for monitoring, notification, and configuration interaction within the Directory Server: %s.

ID: 140

Severity: ERROR
Message: An uncaught exception during processing for thread "%s" has caused it to terminate abnormally. The stack trace for that exception is: %s.

ID: 142

Severity: ERROR
Message: The Directory Server shutdown hook detected that the JVM is shutting down. This generally indicates that JVM received an external request to stop (e.g., through a kill signal).

ID: 183

Severity: ERROR
Message: An error occurred while trying to retrieve the root DSE configuration entry (cn=Root DSE,cn=config) from the Directory Server configuration: %s.

ID: 218

Severity: ERROR
Message: Unable to bind to the Directory Server because no such user exists in the server.

ID: 220

Severity: ERROR
Message: A fatal error occurred when executing one of the Directory Server startup plugins: %s (error ID %d). The Directory Server startup process has been aborted.

ID: 221

Severity: ERROR
Message: Unable to bind to the Directory Server using simple authentication because that user does not have a password.

ID: 222

Severity: ERROR
Message: Unable to process the bind request because it attempted to use an unknown SASL mechanism %s that is not available in the Directory Server.

ID: 228

Severity: ERROR
Message: The specified entry %s does not exist in the Directory Server.

ID: 230

Severity: ERROR
Message: The provided entry cannot be added because it contains a null DN. This DN is reserved for the root DSE, and that entry may not be added over protocol.

ID: 231

Severity: ERROR
Message: The provided entry %s cannot be added because it does not have a parent and is not defined as one of the suffixes within the Directory Server.

ID: 233

Severity: ERROR
Message: Entry %s cannot be added because its parent entry %s does not exist in the server.

ID: 234

Severity: ERROR
Message: Entry %s cannot be added because the server failed to obtain a write lock for this entry after multiple attempts.

ID: 235

Severity: ERROR
Message: Entry %s cannot be removed because the server failed to obtain a write lock for this entry after multiple attempts.

ID: 238

Severity: ERROR
Message: The maximum time limit of %d seconds for processing this search operation has expired.

ID: 239

Severity: ERROR
Message: This search operation has sent the maximum of %d entries to the client.

ID: 240

Severity: ERROR
Message: The entry %s specified as the search base does not exist in the Directory Server.

ID: 241

Severity: ERROR
Message: Entry %s does not exist in the Directory Server.

ID: 242

Severity: ERROR
Message: Entry %s cannot be removed because the backend that should contain that entry has a subordinate backend with a base DN of %s that is below the target DN.

ID: 243

Severity: ERROR
Message: A modify DN operation cannot be performed on entry %s because the new RDN would not have a parent DN.

ID: 244

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because no backend is registered to handle that DN.

ID: 245

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because no backend is registered to handle the new DN %s.

ID: 246

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because the backend holding the current entry is different from the backend used to handle the new DN %s. Modify DN operations may not span multiple backends.

ID: 247

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because the server was unable to obtain a write lock for that DN.

ID: 249

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because the server was unable to obtain a write lock for the new DN %s.

ID: 250

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because that entry does not exist in the server.

ID: 251

Severity: ERROR
Message: Entry %s cannot be modified because the server failed to obtain a write lock for this entry after multiple attempts.

ID: 252

Severity: ERROR
Message: Entry %s cannot be modified because no such entry exists in the server.

ID: 253

Severity: ERROR
Message: Entry %s cannot be modified because the modification contained an add component for attribute %s but no values were provided.

ID: 254

Severity: ERROR
Message: When attempting to modify entry %s to add one or more values for attribute %s, value "%s" was found to be invalid according to the associated syntax: %s.

ID: 255

Severity: ERROR
Message: Entry %s cannot be modified because it would have resulted in one or more duplicate values for attribute %s: %s.

ID: 256

Severity: ERROR
Message: Entry %s cannot be modified because the change to attribute %s would have removed a value used in the RDN.

ID: 257

Severity: ERROR
Message: Entry %s cannot be modified because the attempt to update attribute %s would have removed one or more values from the attribute that were not present: %s.

ID: 258

Severity: ERROR
Message: Entry %s cannot be modified because an attempt was made to remove one or more values from attribute %s but this attribute is not present in the entry.

ID: 259

Severity: ERROR
Message: When attempting to modify entry %s to replace the set of values for attribute %s, value "%s" was found to be invalid according to the associated syntax: %s.

ID: 260

Severity: ERROR
Message: Entry %s cannot be modified because an attempt was made to increment the value of attribute %s which is used as an RDN attribute for the entry.

ID: 261

Severity: ERROR
Message: Entry %s cannot be modified because an attempt was made to increment the value of attribute %s but the request did not include a value for that attribute specifying the amount by which to increment the value.

ID: 262

Severity: ERROR
Message: Entry %s cannot be modified because an attempt was made to increment the value of attribute %s but the request contained multiple values, where only a single integer value is allowed.

ID: 264

Severity: ERROR
Message: Entry %s cannot be modified because an attempt was made to increment the value of attribute %s but that attribute did not have any values in the target entry.

ID: 265

Severity: ERROR
Message: Entry %s cannot be modified because an attempt was made to increment the value of attribute %s but the value "%s" could not be parsed as an integer.

ID: 266

Severity: ERROR
Message: Entry %s cannot be modified because the resulting entry would have violated the server schema: %s.

ID: 268

Severity: ERROR
Message: There is no extended operation handler registered with the Directory Server for handling extended operations with a request OID of %s.

ID: 270

Severity: ERROR
Message: An unexpected error was encountered while processing a search in one of the Directory Server backends: %s.

ID: 271

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because the change would have violated the server schema: %s.

ID: 276

Severity: ERROR
Message: Object class %s cannot be added to entry %s because that class is not defined in the Directory Server schema.

ID: 279

Severity: ERROR
Message: The password provided by the user did not match any password(s) stored in the user's entry.

ID: 289

Severity: ERROR
Message: An error occurred while attempting to parse the provided set of command line arguments: %s.

ID: 290

Severity: ERROR
Message: An error occurred while attempting to bootstrap the Directory Server: %s.

ID: 291

Severity: ERROR
Message: An error occurred while trying to start the Directory Server: %s.

ID: 311

Severity: ERROR
Message: The attempt to obtain a shared lock on file %s was rejected because an exclusive lock was already held on that file.

ID: 312

Severity: ERROR
Message: The attempt to obtain a shared lock on file %s was rejected because the attempt to create the lock file failed: %s.

ID: 313

Severity: ERROR
Message: The attempt to obtain a shared lock on file %s was rejected because the attempt to open the lock file failed: %s.

ID: 314

Severity: ERROR
Message: The attempt to obtain a shared lock on file %s was rejected because an error occurred while attempting to acquire the lock: %s.

ID: 315

Severity: ERROR
Message: The shared lock requested for file %s was not granted, which indicates that another process already holds an exclusive lock on that file.

ID: 316

Severity: ERROR
Message: The attempt to obtain an exclusive lock on file %s was rejected because an exclusive lock was already held on that file.

ID: 317

Severity: ERROR
Message: The attempt to obtain an exclusive lock on file %s was rejected because a shared lock was already held on that file.

ID: 318

Severity: ERROR
Message: The attempt to obtain an exclusive lock on file %s was rejected because the attempt to create the lock file failed: %s.

ID: 319

Severity: ERROR
Message: The attempt to obtain an exclusive lock on file %s was rejected because the attempt to open the lock file failed: %s.

ID: 320

Severity: ERROR
Message: The attempt to obtain an exclusive lock on file %s was rejected because an error occurred while attempting to acquire the lock: %s.

ID: 321

Severity: ERROR
Message: The exclusive lock requested for file %s was not granted, which indicates that another process already holds a shared or exclusive lock on that file.

ID: 322

Severity: ERROR
Message: The attempt to release the exclusive lock held on %s failed: %s.

ID: 323

Severity: ERROR
Message: The attempt to release the shared lock held on %s failed: %s.

ID: 324

Severity: ERROR
Message: The attempt to release the lock held on %s failed because no record of a lock on that file was found.

ID: 343

Severity: ERROR
Message: The Directory Server could not acquire an exclusive lock on file %s: %s. This generally means that another instance of this server is already running.

ID: 346

Severity: ERROR
Message: Entry %s cannot be modified because the modification attempted to update attribute %s which is defined as NO-USER-MODIFICATION in the server schema.

ID: 347

Severity: ERROR
Message: Entry %s cannot be added because it includes attribute %s which is defined as NO-USER-MODIFICATION in the server schema.

ID: 348

Severity: ERROR
Message: Entry %s cannot be renamed because the current DN includes attribute %s which is defined as NO-USER-MODIFICATION in the server schema and the deleteOldRDN flag was set in the modify DN request.

ID: 349

Severity: ERROR
Message: Entry %s cannot be renamed because the new RDN includes attribute %s which is defined as NO-USER-MODIFICATION in the server schema, and the target value for that attribute is not already included in the entry.

ID: 356

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because a pre-operation plugin modified the entry in a way that caused it to violate the server schema: %s.

ID: 357

Severity: ERROR
Message: Entry %s cannot be modified because the request contained an LDAP assertion control and the associated filter did not match the contents of the entry.

ID: 359

Severity: ERROR
Message: Entry %s cannot be modified because the request contained a critical control with OID %s that is not supported by the Directory Server for this type of operation.

ID: 362

Severity: ERROR
Message: Entry %s cannot be removed because the request contained an LDAP assertion control and the associated filter did not match the contents of the entry.

ID: 364

Severity: ERROR
Message: Entry %s cannot be removed because the request contained a critical control with OID %s that is not supported by the Directory Server for this type of operation.

ID: 365

Severity: ERROR
Message: Entry %s cannot be renamed because the request contained an LDAP assertion control and the associated filter did not match the contents of the entry.

ID: 367

Severity: ERROR
Message: Entry %s cannot be renamed because the request contained a critical control with OID %s that is not supported by the Directory Server for this type of operation.

ID: 368

Severity: ERROR
Message: Entry %s cannot be added because the request contained an LDAP assertion control and the associated filter did not match the contents of the provided entry.

ID: 370

Severity: ERROR
Message: Entry %s cannot be added because the request contained a critical control with OID %s that is not supported by the Directory Server for this type of operation.

ID: 371

Severity: ERROR
Message: The search request cannot be processed because it contains an LDAP assertion control and an error occurred while trying to retrieve the base entry to compare it against the assertion filter: %s.

ID: 372

Severity: ERROR
Message: The search request cannot be processed because it contains an LDAP assertion control but the search base entry does not exist.

ID: 373

Severity: ERROR
Message: The search request cannot be processed because it contains an LDAP assertion control and the assertion filter did not match the contents of the base entry.

ID: 375

Severity: ERROR
Message: The search request cannot be processed because it contains a critical control with OID %s that is not supported by the Directory Server for this type of operation.

ID: 376

Severity: ERROR
Message: Cannot perform the compare operation on entry %s because the request contained an LDAP assertion control and the associated filter did not match the contents of the entry.

ID: 378

Severity: ERROR
Message: Cannot perform the compare operation on entry %s because the request contained a critical control with OID %s that is not supported by the Directory Server for this type of operation.

ID: 385

Severity: ERROR
Message: Entry %s cannot be added because it is missing attribute %s that is contained in the entry's RDN. All attributes used in the RDN must also be provided in the attribute list for the entry.

ID: 394

Severity: ERROR
Message: Unable to process the bind request because it contained a control with OID %s that was marked critical but this control is not supported for the bind operation.

ID: 400

Severity: ERROR
Message: The entry %s cannot be added because an entry with that name already exists.

ID: 401

Severity: ERROR
Message: An error occurred during preoperation synchronization processing for the add operation with connection ID %d and operation ID %d: %s.

ID: 402

Severity: ERROR
Message: An error occurred during postoperation synchronization processing for the add operation with connection ID %d and operation ID %d: %s.

ID: 403

Severity: ERROR
Message: An error occurred during preoperation synchronization processing for the delete operation with connection ID %d and operation ID %d: %s.

ID: 404

Severity: ERROR
Message: An error occurred during postoperation synchronization processing for the delete operation with connection ID %d and operation ID %d: %s.

ID: 405

Severity: ERROR
Message: An error occurred during preoperation synchronization processing for the modify operation with connection ID %d and operation ID %d: %s.

ID: 406

Severity: ERROR
Message: An error occurred during postoperation synchronization processing for the modify operation with connection ID %d and operation ID %d: %s.

ID: 407

Severity: ERROR
Message: An error occurred during preoperation synchronization processing for the modify DN operation with connection ID %d and operation ID %d: %s.

ID: 408

Severity: ERROR
Message: An error occurred during postoperation synchronization processing for the modify DN operation with connection ID %d and operation ID %d: %s.

ID: 409

Severity: ERROR
Message: An error occurred during conflict resolution synchronization processing for the add operation with connection ID %d and operation ID %d: %s.

ID: 410

Severity: ERROR
Message: An error occurred during conflict resolution synchronization processing for the delete operation with connection ID %d and operation ID %d: %s.

ID: 411

Severity: ERROR
Message: An error occurred during conflict resolution synchronization processing for the modify operation with connection ID %d and operation ID %d: %s.

ID: 412

Severity: ERROR
Message: An error occurred during conflict resolution synchronization processing for the modify DN operation with connection ID %d and operation ID %d: %s.

ID: 413

Severity: ERROR
Message: Unable to add entry %s because the Directory Server is configured in read-only mode.

ID: 414

Severity: ERROR
Message: Unable to add entry %s because the backend that should hold that entry is configured in read-only mode.

ID: 415

Severity: ERROR
Message: Unable to delete entry %s because the Directory Server is configured in read-only mode.

ID: 416

Severity: ERROR
Message: Unable to delete entry %s because the backend that holds that entry is configured in read-only mode.

ID: 417

Severity: ERROR
Message: Unable to modify entry %s because the Directory Server is configured in read-only mode.

ID: 418

Severity: ERROR
Message: Unable to modify entry %s because the backend that holds that entry is configured in read-only mode.

ID: 419

Severity: ERROR
Message: Unable to rename entry %s because the Directory Server is configured in read-only mode.

ID: 420

Severity: ERROR
Message: Unable to rename entry %s because the backend that holds that entry is configured in read-only mode.

ID: 421

Severity: ERROR
Message: Unable to process the simple bind request because it contained a bind DN but no password, which is forbidden by the server configuration.

ID: 425

Severity: ERROR
Message: The password policy definition contained in configuration entry "%s" is invalid because the specified password attribute "%s" is not defined in the server schema.

ID: 426

Severity: ERROR
Message: The password policy definition contained in configuration entry "%s" is invalid because the specified password attribute "%s" has a syntax OID of %s. The password attribute must have a syntax OID of either 1.3.6.1.4.1.26027.1.3.1 (for the user password syntax) or 1.3.6.1.4.1.4203.1.1.2 (for the authentication password syntax).

ID: 477

Severity: ERROR
Message: An error occurred while attempting to determine the value for attribute ds-cfg-require-change-by-time in configuration entry %s: %s.

ID: 482

Severity: ERROR
Message: The password policy definition contained in configuration entry "%s" is invalid because the specified last login time format "%s" is not a valid format string The last login time format string should conform to the syntax described in the API documentation for the <CODE>java.text.SimpleDateFormat</CODE> class.

ID: 485

Severity: ERROR
Message: The password policy definition contained in configuration entry "%s" is invalid because the specified previous last login time format "%s" is not a valid format string The previous last login time format strings should conform to the syntax described in the API documentation for the <CODE>java.text.SimpleDateFormat</CODE> class.

ID: 496

Severity: ERROR
Message: Attribute options are not allowed for the password attribute %s.

ID: 497

Severity: ERROR
Message: Only a single value may be provided for the password attribute %s.

ID: 498

Severity: ERROR
Message: Pre-encoded passwords are not allowed for the password attribute %s.

ID: 499

Severity: ERROR
Message: The password value for attribute %s was found to be unacceptable: %s.

ID: 500

Severity: ERROR
Message: The password policy defined in configuration entry %s is configured to always send at least one warning notification before the password is expired, but no warning interval has been set. If configuration attribute ds-cfg-expire-passwords-without-warning is set to "false", then configuration attribute ds-cfg-password-expiration-warning-interval must have a positive value.

ID: 501

Severity: ERROR
Message: A bind operation is currently in progress on the associated client connection. No other requests may be made on this client connection until the bind processing has completed.

ID: 502

Severity: ERROR
Message: %s must change their password before it will be allowed to request any other operations.

ID: 504

Severity: ERROR
Message: An error occurred while attempting to decode the ds-pwp-password-policy-dn value "%s" in user entry "%s" as a DN: %s.

ID: 505

Severity: ERROR
Message: User entry %s is configured to use a password policy subentry of %s but no such password policy has been defined in the server configuration.

ID: 506

Severity: ERROR
Message: An error occurred while attempting to decode value "%s" for attribute %s in user entry %s in accordance with the generalized time format: %s.

ID: 507

Severity: ERROR
Message: Unable to decode value "%s" for attribute %s in user entry %s as a Boolean value.

ID: 508

Severity: ERROR
Message: CryptoManager failed to verify the wrapped key entry signature: %s.

ID: 509

Severity: ERROR
Message: The user cannot bind due to insufficient access rights.

ID: 510

Severity: ERROR
Message: The entry %s cannot be compared due to insufficient access rights.

ID: 511

Severity: ERROR
Message: The entry %s cannot be deleted due to insufficient access rights.

ID: 512

Severity: ERROR
Message: The extended operation %s cannot be performed due to insufficient access rights.

ID: 513

Severity: ERROR
Message: The entry %s cannot be renamed due to insufficient access rights.

ID: 514

Severity: ERROR
Message: The entry %s cannot be modified due to insufficient access rights.

ID: 515

Severity: ERROR
Message: The entry %s cannot be searched due to insufficient access rights.

ID: 516

Severity: ERROR
Message: Rejecting a simple bind request because the password policy requires secure authentication.

ID: 517

Severity: ERROR
Message: Rejecting a bind request because the account has been administratively disabled.

ID: 518

Severity: ERROR
Message: Rejecting a bind request because the account has been locked due to too many failed authentication attempts.

ID: 519

Severity: ERROR
Message: Rejecting a bind request because the account has been locked after the user's password was not changed in a timely manner after an administrative reset.

ID: 520

Severity: ERROR
Message: Rejecting a bind request because the account has been locked after remaining idle for too long.

ID: 521

Severity: ERROR
Message: Rejecting a bind request because that user's password is expired.

ID: 522

Severity: ERROR
Message: An error occurred while attempting to update password policy state information for user %s: %s.

ID: 523

Severity: ERROR
Message: Rejecting a SASL %s bind request for user %s because the password policy requires secure authentication.

ID: 531

Severity: ERROR
Message: Rejecting a bind request because the account has expired.

ID: 532

Severity: ERROR
Message: Attributes used to hold user passwords are not allowed to have any attribute options.

ID: 533

Severity: ERROR
Message: Users are not allowed to change their own passwords.

ID: 534

Severity: ERROR
Message: Password changes must be performed over a secure authentication channel.

ID: 535

Severity: ERROR
Message: The password cannot be changed because it has not been long enough since the last password change.

ID: 536

Severity: ERROR
Message: Multiple password values are not allowed in user entries.

ID: 537

Severity: ERROR
Message: User passwords may not be provided in pre-encoded form.

ID: 538

Severity: ERROR
Message: Invalid modification type %s attempted on password attribute %s.

ID: 539

Severity: ERROR
Message: The user entry does not have any existing passwords to remove.

ID: 541

Severity: ERROR
Message: The provided user password does not match any password in the user's entry.

ID: 542

Severity: ERROR
Message: The password policy requires that user password changes include the current password in the request.

ID: 543

Severity: ERROR
Message: The password change would result in multiple password values in the user entry, which is not allowed.

ID: 544

Severity: ERROR
Message: The provided password value was rejected by a password validator: %s.

ID: 545

Severity: ERROR
Message: %s must change their password before it will be allowed to perform any other operations.

ID: 548

Severity: ERROR
Message: The account has been locked as a result of too many failed authentication attempts (time to unlock: %s).

ID: 549

Severity: ERROR
Message: The account has been locked as a result of too many failed authentication attempts. It may only be unlocked by an administrator.

ID: 556

Severity: ERROR
Message: The specified password value already exists in the user entry.

ID: 566

Severity: ERROR
Message: Entry %s cannot be updated because the request did not contain any modifications.

ID: 570

Severity: ERROR
Message: Unable to process the request for extended operation %s because it contained an unsupported critical control with OID %s.

ID: 571

Severity: ERROR
Message: Unable to register backend %s with the Directory Server because another backend with the same backend ID is already registered.

ID: 572

Severity: ERROR
Message: Unable to register base DN %s with the Directory Server for backend %s because that base DN is already registered for backend %s.

ID: 573

Severity: ERROR
Message: Unable to register base DN %s with the Directory Server for backend %s because that backend already contains another base DN %s that is within the same hierarchical path.

ID: 574

Severity: ERROR
Message: Unable to register base DN %s with the Directory Server for backend %s because that backend already contains another base DN %s that is not subordinate to the same base DN in the parent backend.

ID: 575

Severity: ERROR
Message: Unable to register base DN %s with the Directory Server for backend %s because that backend already contains one or more other base DNs that are subordinate to backend %s but the new base DN is not.

ID: 577

Severity: ERROR
Message: Unable to de-register base DN %s with the Directory Server because that base DN is not registered for any active backend.

ID: 580

Severity: ERROR
Message: Rejecting the requested operation because the connection has not been authenticated.

ID: 583

Severity: ERROR
Message: Entry %s cannot be modified because the modification attempted to set one or more new values for attribute %s which is marked OBSOLETE in the server schema.

ID: 584

Severity: ERROR
Message: Object class %s added to entry %s is marked OBSOLETE in the server schema.

ID: 585

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because the new RDN includes attribute type %s which is declared OBSOLETE in the server schema.

ID: 589

Severity: ERROR
Message: You do not have sufficient privileges to reset user passwords.

ID: 590

Severity: ERROR
Message: You do not have sufficient privileges to access the server configuration.

ID: 591

Severity: ERROR
Message: You do not have sufficient privileges to add entries that include privileges.

ID: 592

Severity: ERROR
Message: You do not have sufficient privileges to modify the set of privileges contained in an entry.

ID: 595

Severity: ERROR
Message: You do not have sufficient privileges to use the proxied authorization control.

ID: 598

Severity: ERROR
Message: OpenDJ is configured to run as a Windows service and it cannot run in no-detach mode.

ID: 600

Severity: ERROR
Message: Unable to decode an entry because it had an unsupported entry version byte value of %s.

ID: 601

Severity: ERROR
Message: The wrapped key entry signature does not match the entry content.

ID: 611

Severity: ERROR
Message: The request control with Object Identifier (OID) "%s" cannot be used due to insufficient access rights.

ID: 612

Severity: ERROR
Message: The connection handler %s is trying to use the listener %s which is already in use by another connection handler.

ID: 614

Severity: ERROR
Message: No enabled connection handler available.

ID: 615

Severity: ERROR
Message: Could not start connection handlers.

ID: 616

Severity: ERROR
Message: Unable to process the non-root bind because the server is in lockdown mode.

ID: 620

Severity: ERROR
Message: Unable to decode the provided attribute because it used an undefined attribute description token %s.

ID: 621

Severity: ERROR
Message: Unable to decode the provided object class set because it used an undefined token %s.

ID: 623

Severity: ERROR
Message: Unable to decode the provided entry encode configuration element because it has an invalid length.

ID: 628

Severity: ERROR
Message: Rejecting a bind request for user %s because either the entire server or the user's backend has a writability mode of 'disabled' and password policy state updates would not be allowed.

ID: 629

Severity: ERROR
Message: The provided new password was found in the password history for the user.

ID: 633

Severity: ERROR
Message: The password policy configuration entry "%s" is invalid because if a maximum password age is configured, then the password expiration warning interval must be shorter than the maximum password age.

ID: 634

Severity: ERROR
Message: The password policy configuration entry "%s" is invalid because if both a minimum password age and a maximum password age are configured, then the sum of the minimum password age and the password expiration warning interval must be shorter than the maximum password age.

ID: 638

Severity: ERROR
Message: An error occurred while attempting to disconnect client connection %d: %s.

ID: 639

Severity: ERROR
Message: An unexpected error occurred in the idle time limit thread: %s.

ID: 640

Severity: ERROR
Message: The Directory Server is currently running. Environment configuration changes are not allowed with the server running.

ID: 641

Severity: ERROR
Message: The specified server root directory '%s' is invalid. The specified path must exist and must be a directory.

ID: 642

Severity: ERROR
Message: The specified config file path '%s' is invalid. The specified path must exist and must be a file.

ID: 644

Severity: ERROR
Message: The specified schema configuration directory '%s' is invalid. The specified path must exist and must be a directory.

ID: 648

Severity: ERROR
Message: The Directory Server is currently running. The environment configuration can not be altered while the server is online.

ID: 649

Severity: ERROR
Message: An error occurred while attempting to initialize a SSL context for server to server communication: %s.

ID: 654

Severity: ERROR
Message: An error occurred in the trust store synchronization thread: %s.

ID: 657

Severity: ERROR
Message: The password storage scheme defined in configuration entry %s does not support the auth password syntax, which is used by password attribute %s.

ID: 659

Severity: ERROR
Message: Password policy configuration entry %s references deprecated password storage scheme DN %s which does not support the auth password syntax.

ID: 661

Severity: ERROR
Message: CryptoManager cannot get the requested digest %s: %s.

ID: 662

Severity: ERROR
Message: CryptoManager cannot get the requested MAC engine %s: %s.

ID: 663

Severity: ERROR
Message: CryptoManager cannot get the requested encryption cipher %s: %s.

ID: 664

Severity: ERROR
Message: CryptoManager cannot get the preferred key wrapping cipher: %s.

ID: 670

Severity: ERROR
Message: CryptoManager failed to retrieve the collection of instance-key-pair public-key-certificates from ADS container "%s": %s.

ID: 671

Severity: ERROR
Message: CryptoManager failed to encode symmetric key attribute value: %s.

ID: 672

Severity: ERROR
Message: CryptoManager symmetric key attribute value "%s" syntax is invalid: incorrect number of fields.

ID: 673

Severity: ERROR
Message: CryptoManager symmetric key attribute value "%s" syntax is invalid. Parsing failed in field "%s" at offset %d.

ID: 675

Severity: ERROR
Message: CryptoManager failed to decipher the wrapped secret-key value: %s.

ID: 676

Severity: ERROR
Message: CryptoManager cannot find the public-key-certificate (identifier "%s") requested for symmetric key re-encoding.

ID: 677

Severity: ERROR
Message: CryptoManager failed to decode the key entry identifier "%s": %s.

ID: 678

Severity: ERROR
Message: CrytpoManager passed invalid MAC algorithm "%s": %s.

ID: 679

Severity: ERROR
Message: CryptoManager failed to initialize MAC engine: %s.

ID: 680

Severity: ERROR
Message: CryptoManager passed invalid Cipher transformation "%s": %s.

ID: 681

Severity: ERROR
Message: CryptoManager cannot initialize Cipher: %s.

ID: 682

Severity: ERROR
Message: CryptoManager failed to write the stream prologue: %s.

ID: 683

Severity: ERROR
Message: CryptoManager failed to decrypt the supplied data because it could not read the symmetric key identifier in the data prologue: %s.

ID: 684

Severity: ERROR
Message: CryptoManager failed to decrypt the supplied data because the symmetric key identifier in the data prologue does not match any known key entries.

ID: 685

Severity: ERROR
Message: CryptoManager failed to decrypt the supplied data because it could not read the cipher initialization vector in the data prologue.

ID: 686

Severity: ERROR
Message: CryptoManager failed to decrypt the supplied data because there was an error reading from the input stream: %s.

ID: 687

Severity: ERROR
Message: CryptoManager failed to import the symmetric key entry "%s" because it could not obtain a symmetric key attribute value that can be decoded by this instance.

ID: 688

Severity: ERROR
Message: CryptoManager detected a field mismatch between the key entry to be imported and an entry in the key cache that share the key identifier "%s".

ID: 689

Severity: ERROR
Message: CryptoManager failed to import the symmetric key entry "%s": %s.

ID: 690

Severity: ERROR
Message: CryptoManager failed to import the symmetric key entry "%s" because it could not add a symmetric key attribute value that can be decoded by this instance.

ID: 691

Severity: ERROR
Message: CryptoManager failed to instantiate a KeyGenerator for algorithm "%s": %s.

ID: 692

Severity: ERROR
Message: CryptoManager failed to add locally produced symmetric key entry "%s": %s.

ID: 693

Severity: ERROR
Message: CryptoManager cipher transformation specification "%s" is invalid: it must be of the form "algorithm/mode/padding".

ID: 694

Severity: ERROR
Message: CryptoManager cipher transformation specification "%s" is invalid: it must be of the form "algorithm/mode/padding".

ID: 695

Severity: ERROR
Message: CryptoManager failed to decrypt the supplied data because it could not read the version number in the data prologue: %s.

ID: 696

Severity: ERROR
Message: CryptoManager failed to decrypt the supplied data because the version "%d" in the data prologue is unknown.

ID: 697

Severity: ERROR
Message: The provided entry %s cannot be added because its suffix is not defined as one of the suffixes within the Directory Server.

ID: 700

Severity: ERROR
Message: Start TLS extended operations cannot be canceled.

ID: 701

Severity: ERROR
Message: Cancel extended operations can not be canceled.

ID: 702

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because the new superior entry %s is equal to or a subordinate of the entry to be moved.

ID: 715

Severity: ERROR
Message: Entry %s can not be added because BER encoding of %s attribute is not supported.

ID: 723

Severity: ERROR
Message: In no-detach mode, the 'timeout' option cannot be used.

ID: 726

Severity: ERROR
Message: The subentry %s must have either the pwdPolicy or ds-pwp-password-policy objectclasses, which is required for the Directory Server password policy.

ID: 728

Severity: ERROR
Message: CryptoManager failed to initialize because the specified cipher key length "%d" is beyond the allowed cryptography strength "%d" in jurisdiction policy files.

ID: 729

Severity: ERROR
Message: Failed to update free disk space for directory %s: %s.

ID: 730

Severity: ERROR
Message: The directory server is not accepting a new persistent search request because the server has already reached its limit.

ID: 739

Severity: ERROR
Message: This operation involves LDAP subentries which you do not have sufficient privileges to administer.

ID: 743

Severity: ERROR
Message: When attempting to modify entry %s, one value for attribute %s was found to be invalid according to the associated syntax: %s.

ID: 744

Severity: ERROR
Message: When attempting to modify entry %s to replace the set of values for attribute %s, one value was found to be invalid according to the associated syntax: %s.

ID: 745

Severity: ERROR
Message: The password policy definition contained in configuration entry "%s" is invalid because the password validator "%s" specified in attribute "%s" cannot be found.

ID: 746

Severity: ERROR
Message: The password could not be validated because of misconfiguration. Please contact the administrator.

ID: 747

Severity: ERROR
Message: The password for user %s could not be validated because the password policy subentry %s is referring to an unknown password validator (%s). Please make sure the password policy subentry only refers to validators that exist on all replicas.

ID: 748

Severity: ERROR
Message: Could not get filesystem for directory %s: %s.

ID: 749

Severity: ERROR
Message: The free space (%s) on the disk containing directory "%s" is between low and full threshold for the following subsystems: %s. Write operations are only permitted by a user with the BYPASS_LOCKDOWN privilege until the free space rises above the threshold. Replication updates are still allowed.

ID: 750

Severity: ERROR
Message: The free space (%s) on the disk containing directory "%s" is below full threshold for the following subsystems: %s. Write operations to the backend, replication updates included, will fail until the free space rises above the threshold.

ID: 752

Severity: ERROR
Message: A StartTLS operation is currently in progress on the associated client connection. No other requests may be made on this client connection until the StartTLS processing has completed.

ID: 753

Severity: ERROR
Message: A SASL bind operation is currently in progress on the associated client connection. No other requests may be made on this client connection until the SASL bind processing has completed.

ID: 754

Severity: ERROR
Message: Cannot properly use SHA-1 using the java provider. Verify java.security is properly configured.

ID: 755

Severity: ERROR
Message: Cannot complete initialization of server's backends because the root and administrative backends have not been initialized yet.

ID: 756

Severity: ERROR
Message: An error occurred while adding Service Discovery Mechanism '%s': %s.

ID: 757

Severity: ERROR
Message: Registering Service Discovery Manager's listener failed : %s.

ID: 758

Severity: ERROR
Message: Discovery mechanism '%s' initialization failed : %s.

ID: 761

Severity: ERROR
Message: Error occurred while creating an SSL context for service discovery mechanism '%s' : %s.

ID: 762

Severity: ERROR
Message: Could not retrieve the list of replicas from replication server '%s' for replication server group '%s'. Exception : %s.

ID: 763

Severity: ERROR
Message: Could not retrieve auto-configuration data from directory server '%s' for replication server group '%s'. Exception : %s".

ID: 764

Severity: ERROR
Message: Service discovery mechanism '%s' failed to refresh the partition information. Exception : %s",.

ID: 770

Severity: ERROR
Message: Service discovery mechanism '%s' failed to refresh the connection options. Exception : %s",.

ID: 779

Severity: ERROR
Message: "%s" (low=%s, full=%s).

ID: 780

Severity: ERROR
Message: You do not have sufficient privileges to read directory server monitoring information.

ID: 783

Severity: ERROR
Message: Entry %s cannot be added because its parent entry %s is a subentry.

ID: 786

Severity: ERROR
Message: The master key with alias '%s' does not exist in the '%s' key manager. Please check that the correct key manager has been configured and that it contains the specified master keys.

ID: 787

Severity: ERROR
Message: The CryptoManager could not encode a symmetric because the master key with alias '%s' does not exist in the '%s' key manager. Please check that the correct key manager has been configured and that it contains the specified master keys.

ID: 788

Severity: ERROR
Message: Cipher %s is not yet supported.

ID: 794

Severity: ERROR
Message: The CryptoManager could was not able to obtain the deployment's pepper. Please check that the CryptoManager has a correctly configured key manager and preferred digest mechanism.

ID: 795

Severity: ERROR
Message: No enabled password storage schemes in '%s' in subentry '%s'.

ID: 796

Severity: ERROR
Message: Cannot use both pwdValidatorPolicy and ds-pwp-validator in subentry '%s'.

ID: 799

Severity: ERROR
Message: The dictionary data could not be decompressed: %s.

ID: 800

Severity: ERROR
Message: CryptoManager failed to find the master key pair with ID '%s', make sure cryptoManager has access to the master key pair that was used at the time of wrapping the key.

ID: 802

Severity: ERROR
Message: Requested cipher for a non existing cipher key: cryptographic services were not properly initialized, programming error.

ID: 803

Severity: ERROR
Message: Type %d is not a valid secret key type. The Valid type is '0' for a cipher key. Secret key initialization cannot continue, check the data source and re-initialize if needed.

ID: 804

Severity: ERROR
Message: The subentry %s cannot use both %s and %s objectclasses.

ID: 805

Severity: ERROR
Message: The subentry %s using the %s objectclass cannot define validators using the old %s objectclass.

ID: 806

Severity: ERROR
Message: The subentry %s using the %s objectclass cannot define validators using the new %s objectclass.

ID: 807

Severity: ERROR
Message: The value for the '%s' attribute is not a valid duration.

ID: 808

Severity: ERROR
Message: The value for the '%s' attribute is not a valid integer.

ID: 809

Severity: ERROR
Message: The value for the '%s' attribute is not a valid boolean.

ID: 810

Severity: ERROR
Message: The value for the '%s' attribute is not a valid time.

ID: 811

Severity: ERROR
Message: The value for the '%s' attribute is not a valid string.

ID: 812

Severity: ERROR
Message: The value for the '%s' attribute is not a valid attribute.

ID: 813

Severity: ERROR
Message: The values for the '%s' attribute are not valid strings.

ID: 814

Severity: ERROR
Message: The value for the '%s' attribute is not a valid state update failure policy.

ID: 815

Severity: ERROR
Message: A values for the '%s' attribute is not a valid attribute name.

ID: 816

Severity: ERROR
Message: Could not start connection handler %s with listen addresses "%s". The error was: %s.

Category: Generic backends

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 2

Severity: ERROR
Message: An attempt was made to configure the root DSE backend without providing a configuration entry. This is not allowed.

ID: 9

Severity: ERROR
Message: Unwilling to update entry "%s" because modify operations are not supported in the root DSE backend. If you wish to alter the contents of the root DSE itself, then it may be possible to do so by modifying the "%s" entry in the configuration.

ID: 11

Severity: ERROR
Message: Unwilling to perform a search (connection ID %d, operation ID %d) with a base DN of "%s" in the root DSE backend. The base DN for searches in this backend must be the DN of the root DSE itself.

ID: 13

Severity: ERROR
Message: Unable to process the search with connection ID %d and operation ID %d because it had an invalid scope of %s.

ID: 14

Severity: ERROR
Message: An unexpected error occurred while trying to open the LDIF writer for the root DSE backend: %s.

ID: 15

Severity: ERROR
Message: An unexpected error occurred while trying to export the root DSE entry to the specified LDIF target: %s.

ID: 21

Severity: ERROR
Message: An attempt was made to configure the monitor backend without providing a configuration entry. This is not allowed, and no monitor information will be available over protocol.

ID: 23

Severity: ERROR
Message: Unwilling to add entry "%s" because add operations are not supported in the "%s" backend.

ID: 24

Severity: ERROR
Message: Unwilling to remove entry "%s" because delete operations are not supported in the "%s" backend.

ID: 25

Severity: ERROR
Message: Unwilling to update entry "%s" because modify operations are not supported in the monitor backend. If you wish to alter the contents of the base monitor entry itself, then it may be possible to do so by modifying the "%s" entry in the configuration.

ID: 26

Severity: ERROR
Message: Unwilling to rename entry "%s" because modify DN operations are not supported in the "%s" backend.

ID: 27

Severity: ERROR
Message: An error occurred while attempting to export the base monitor entry: %s.

ID: 28

Severity: ERROR
Message: An error occurred while attempting to export the monitor entry for monitor provider %s: %s.

ID: 29

Severity: ERROR
Message: The "%s" backend does not support LDIF import operations.

ID: 32

Severity: ERROR
Message: Unable to retrieve the requested entry from the "%s" backend because the provided DN was null.

ID: 34

Severity: ERROR
Message: Unable to retrieve the requested entry %s from the monitor backend because the DN is not below the monitor base of %s.

ID: 38

Severity: ERROR
Message: An attempt was made to configure the schema backend without providing a configuration entry. This is not allowed, and no schema information will be available over protocol.

ID: 45

Severity: ERROR
Message: An error occurred while attempting to export the base schema entry: %s.

ID: 48

Severity: ERROR
Message: Unable to retrieve the requested entry %s from the schema backend because the DN is equal to one of the schema entry DNs.

ID: 49

Severity: ERROR
Message: An unexpected error occurred while trying to open the LDIF writer for the schema backend: %s.

ID: 55

Severity: ERROR
Message: The Directory Server was unable to obtain a lock on entry %s after multiple attempts. This could mean that the entry is already locked by a long-running operation or that the entry has previously been locked but was not properly unlocked.

ID: 91

Severity: ERROR
Message: The task defined in entry %s is invalid because it has an invalid state %s.

ID: 92

Severity: ERROR
Message: An error occurred while trying to parse the scheduled start time value %s from task entry %s.

ID: 93

Severity: ERROR
Message: An error occurred while trying to parse the actual start time value %s from task entry %s.

ID: 94

Severity: ERROR
Message: An error occurred while trying to parse the completion time value %s from task entry %s.

ID: 95

Severity: ERROR
Message: Task entry %s is missing required attribute %s.

ID: 96

Severity: ERROR
Message: There are multiple instances of attribute %s in task entry %s.

ID: 98

Severity: ERROR
Message: There are multiple values for attribute %s in task entry %s.

ID: 99

Severity: ERROR
Message: An error occurred while executing the task defined in entry %s: %s.

ID: 100

Severity: ERROR
Message: The provided recurring task entry does not contain attribute %s which is needed to hold the recurring task ID.

ID: 101

Severity: ERROR
Message: The provided recurring task entry contains multiple attributes with type %s, which is used to hold the recurring task ID, but only a single instance is allowed.

ID: 102

Severity: ERROR
Message: The provided recurring task entry does not contain any values for the %s attribute, which is used to specify the recurring task ID.

ID: 103

Severity: ERROR
Message: The provided recurring task entry contains multiple values for the %s attribute, which is used to specify the recurring task ID, but only a single value is allowed.

ID: 104

Severity: ERROR
Message: The provided recurring task entry does not contain attribute %s which is needed to specify recurring task schedule.

ID: 105

Severity: ERROR
Message: The provided recurring task entry contains multiple attributes with type %s, which is used to hold recurring task schedule, but only a single instance is allowed.

ID: 106

Severity: ERROR
Message: The provided recurring task entry does not contain any values for the %s attribute, which is used to specify recurring task schedule.

ID: 107

Severity: ERROR
Message: The provided recurring task entry contains multiple values for the %s attribute, which is used to specify recurring task schedule, but only a single value is allowed.

ID: 108

Severity: ERROR
Message: An error occurred while attempting to load class %s specified in attribute %s of the provided recurring task entry: %s. Does this class exist in the Directory Server classpath?.

ID: 109

Severity: ERROR
Message: An error occurred while trying to create an instance of class %s as a Directory Server task. Is this class a subclass of %s?.

ID: 110

Severity: ERROR
Message: An error occurred while attempting to perform internal initialization on an instance of class %s with the information contained in the provided entry: %s.

ID: 121

Severity: ERROR
Message: The specified task data backing file %s already exists and the Directory Server will not attempt to overwrite it. Please delete or rename the existing file before attempting to use that path for the new backing file, or choose a new path.

ID: 122

Severity: ERROR
Message: The specified path %s for the new task data backing file appears to be an invalid path. Please choose a new path for the task data backing file.

ID: 123

Severity: ERROR
Message: The parent directory %s for the new task data backing file %s does not exist. Please create this directory before attempting to use this path for the new backing file or choose a new path.

ID: 124

Severity: ERROR
Message: The parent directory %s for the new task data backing file %s exists but is not a directory. Please choose a new path for the task data backing file.

ID: 125

Severity: ERROR
Message: An error occurred while attempting to determine the new path to the task data backing file: %s.

ID: 130

Severity: ERROR
Message: New entries in the task backend may only be added immediately below %s for scheduled tasks or immediately below %s for recurring tasks.

ID: 133

Severity: ERROR
Message: Unable to add recurring task %s to the task scheduler because another recurring task already exists with the same ID.

ID: 134

Severity: ERROR
Message: Unable to schedule task %s because another task already exists with the same ID.

ID: 136

Severity: ERROR
Message: An error occurred while attempting to schedule the next iteration of recurring task %s: %s.

ID: 137

Severity: ERROR
Message: An error occurred while attempting to read an entry from the tasks backing file %s on or near line %d: %s. This is not a fatal error, so the task scheduler will attempt to continue parsing the file and schedule any additional tasks that it contains.

ID: 138

Severity: ERROR
Message: An error occurred while attempting to read an entry from the tasks backing file %s on or near line %d: %s. This is an unrecoverable error, and parsing cannot continue.

ID: 139

Severity: ERROR
Message: Entry %s read from the tasks backing file is invalid because it has no parent and does not match the task root DN of %s.

ID: 140

Severity: ERROR
Message: An error occurred while attempting to parse entry %s as a recurring task and add it to the scheduler: %s.

ID: 141

Severity: ERROR
Message: An error occurred while attempting to parse entry %s as a task and add it to the scheduler: %s.

ID: 142

Severity: ERROR
Message: Entry %s read from the tasks backing file %s has a DN which is not valid for a task or recurring task definition and will be ignored.

ID: 143

Severity: ERROR
Message: An error occurred while attempting to read from the tasks data backing file %s: %s.

ID: 144

Severity: ERROR
Message: An error occurred while attempting to create a new tasks backing file %s for use with the task scheduler: %s.

ID: 145

Severity: ERROR
Message: The provided task entry does not contain attribute %s which is needed to specify the fully-qualified name of the class providing the task logic.

ID: 146

Severity: ERROR
Message: The provided task entry contains multiple attributes with type %s, which is used to hold the task class name, but only a single instance is allowed.

ID: 147

Severity: ERROR
Message: The provided task entry does not contain any values for the %s attribute, which is used to specify the fully-qualified name of the class providing the task logic.

ID: 148

Severity: ERROR
Message: The provided task entry contains multiple values for the %s attribute, which is used to specify the task class name, but only a single value is allowed.

ID: 149

Severity: ERROR
Message: An error occurred while attempting to load class %s specified in attribute %s of the provided task entry: %s. Does this class exist in the Directory Server classpath?.

ID: 150

Severity: ERROR
Message: An error occurred while trying to create an instance of class %s as a Directory Server task. Is this class a subclass of %s?.

ID: 151

Severity: ERROR
Message: An error occurred while attempting to perform internal initialization on an instance of class %s with the information contained in the provided entry: %s.

ID: 153

Severity: ERROR
Message: An error occurred while attempting to rename the new tasks backing file from %s to %s: %s. If the Directory Server is restarted, then the task scheduler may not work as expected.

ID: 154

Severity: ERROR
Message: An error occurred while attempting to write the new tasks data backing file %s: %s. Configuration information reflecting the latest update may be lost.

ID: 161

Severity: ERROR
Message: Unable to remove pending task %s because no such task exists.

ID: 162

Severity: ERROR
Message: Unable to remove pending task %s because the task is no longer pending.

ID: 163

Severity: ERROR
Message: Unable to remove completed task %s because no such task exists in the list of completed tasks.

ID: 164

Severity: ERROR
Message: Unable to remove entry %s from the task backend because its DN is either not appropriate for that backend or it is not below the scheduled or recurring tasks base entry.

ID: 165

Severity: ERROR
Message: Unable to remove entry %s from the task backend because there is no scheduled task associated with that entry DN.

ID: 166

Severity: ERROR
Message: Unable to delete entry %s from the task backend because the associated task is currently running.

ID: 167

Severity: ERROR
Message: Unable to remove entry %s from the task backend because there is no recurring task associated with that entry DN.

ID: 168

Severity: ERROR
Message: Unable to process the search operation in the task backend because the provided base DN %s is not valid for entries in the task backend.

ID: 169

Severity: ERROR
Message: Unable to process the search operation in the task backend because there is no scheduled task associated with the provided search base entry %s.

ID: 170

Severity: ERROR
Message: Unable to process the search operation in the task backend because there is no recurring task associated with the provided search base entry %s.

ID: 186

Severity: ERROR
Message: Unwilling to update entry "%s" because modify operations are not supported in the "%s" backend.

ID: 192

Severity: ERROR
Message: Exactly one base DN must be provided for use with the memory-based backend.

ID: 193

Severity: ERROR
Message: Entry %s already exists in the memory-based backend.

ID: 194

Severity: ERROR
Message: Entry %s does not belong in the memory-based backend.

ID: 195

Severity: ERROR
Message: Unable to add entry %s because its parent entry %s does not exist in the memory-based backend.

ID: 196

Severity: ERROR
Message: Entry %s does not exist in the "%s" backend.

ID: 197

Severity: ERROR
Message: Cannot delete entry %s because it has one or more subordinate entries.

ID: 199

Severity: ERROR
Message: Unable to create an LDIF writer: %s.

ID: 200

Severity: ERROR
Message: Cannot write entry %s to LDIF: %s.

ID: 201

Severity: ERROR
Message: Unable to create an LDIF reader: %s.

ID: 202

Severity: ERROR
Message: An unrecoverable error occurred while reading from LDIF: %s.

ID: 203

Severity: ERROR
Message: An unexpected error occurred while processing the import: %s.

ID: 205

Severity: ERROR
Message: Cannot rename entry %s because it has one or more subordinate entries.

ID: 206

Severity: ERROR
Message: Cannot rename entry %s because the target entry is in a different backend.

ID: 207

Severity: ERROR
Message: Cannot rename entry %s because the new parent entry %s doesn't exist.

ID: 210

Severity: ERROR
Message: An error occurred while attempting to register the base DNs %s in the Directory Server: %s.

ID: 212

Severity: ERROR
Message: The schema backend does not support the %s modification type.

ID: 213

Severity: ERROR
Message: The schema backend does not support the modification of the %s attribute type. Only attribute types, object classes, ldap syntaxes, name forms, DIT content rules, DIT structure rules, and matching rule uses may be modified.

ID: 222

Severity: ERROR
Message: An error occurred while attempting to write the updated schema: %s.

ID: 227

Severity: ERROR
Message: The server will not allow removing all values for the %s attribute type in the server schema.

ID: 248

Severity: ERROR
Message: Circular reference detected for attribute type %s in which the superior type chain references the attribute type itself.

ID: 249

Severity: ERROR
Message: Circular reference detected for objectclass %s in which the superior class chain references the objectclass itself.

ID: 250

Severity: ERROR
Message: Circular reference detected for DIT structure rule %s in which the superior rule chain references the DIT structure rule itself.

ID: 251

Severity: ERROR
Message: An error occurred while attempting to create copies of the existing schema files before applying the updates: %s. The server was able to restore the original schema configuration, so no additional cleanup should be required.

ID: 252

Severity: ERROR
Message: An error occurred while attempting to create copies of the existing schema files before applying the updates: %s. A problem also occurred when attempting to restore the original schema configuration, so the server may be left in an inconsistent state and could require manual cleanup.

ID: 253

Severity: ERROR
Message: An error occurred while attempting to write new versions of the server schema files: %s. The server was able to restore the original schema configuration, so no additional cleanup should be required.

ID: 254

Severity: ERROR
Message: An error occurred while attempting to write new versions of the server schema files: %s. A problem also occurred when attempting to restore the original schema configuration, so the server may be left in an inconsistent state and could require manual cleanup.

ID: 255

Severity: ERROR
Message: Unable to remove attribute type %s from the server schema because no such attribute type is defined.

ID: 261

Severity: ERROR
Message: Unable to remove objectclass %s from the server schema because no such objectclass is defined.

ID: 265

Severity: ERROR
Message: Unable to remove name form %s from the server schema because no such name form is defined.

ID: 267

Severity: ERROR
Message: Unable to remove DIT content rule %s from the server schema because no such DIT content rule is defined.

ID: 268

Severity: ERROR
Message: Unable to remove DIT structure rule %s from the server schema because no such DIT structure rule is defined.

ID: 270

Severity: ERROR
Message: Unable to remove matching rule use %s from the server schema because no such matching rule use is defined.

ID: 293

Severity: ERROR
Message: You do not have sufficient privileges to modify the Directory Server schema.

ID: 294

Severity: ERROR
Message: Unable to find a file containing concatenated schema element definitions in order to determine if any schema changes were made with the server offline. The file was expected in the %s directory and should have been named either %s or %s.

ID: 295

Severity: ERROR
Message: An error occurred while attempting to determine whether any schema changes had been made by directly editing the schema files with the server offline: %s.

ID: 296

Severity: ERROR
Message: An error occurred while attempting to write file %s containing a concatenated list of all server schema elements: %s. The server may not be able to accurately identify any schema changes made with the server offline.

ID: 298

Severity: ERROR
Message: The Directory Server is not configured to allow task %s to be invoked.

ID: 305

Severity: ERROR
Message: Indexes are not supported in the "%s" backend.

ID: 307

Severity: ERROR
Message: LDIF import and export operations are not supported in the "%s" backend.

ID: 329

Severity: ERROR
Message: The root container for backend %s has not been initialized preventing this backend from processing the requested operation.

ID: 330

Severity: ERROR
Message: Unable to obtain a write lock on entry %s.

ID: 331

Severity: ERROR
Message: Entry %s cannot be modified because it does not represent a task entry. Only task entries may be modified in the task backend.

ID: 332

Severity: ERROR
Message: Entry %s cannot be modified because it does not represent a valid task in the server.

ID: 333

Severity: ERROR
Message: Entry %s cannot be modified because the associated task has completed running. Completed tasks cannot be modified.

ID: 334

Severity: ERROR
Message: Entry %s cannot be modified because the server does not currently support modifying recurring task entries.

ID: 335

Severity: ERROR
Message: The task associated with entry %s is currently running. The only modification allowed for running tasks is to replace the value of the ds-task-state attribute with "cancel".

ID: 339

Severity: ERROR
Message: The LDIF backend defined in configuration entry %s only supports a single base DN, but was configured for use with multiple base DNs.

ID: 342

Severity: ERROR
Message: LDIF file %s configured for use with the LDIF backend defined in configuration entry %s has multiple entries with a DN of %s.

ID: 343

Severity: ERROR
Message: LDIF file %s configured for use with the LDIF backend defined in configuration entry %s includes entry %s which is not below the base DN defined for that backend.

ID: 344

Severity: ERROR
Message: LDIF file %s configured for use with the LDIF backend defined in configuration entry %s contains entry %s but its parent entry has not yet been read.

ID: 345

Severity: ERROR
Message: An error occurred while trying to create file %s to write an updated version of the data for the LDIF backend defined in configuration entry %s: %s.

ID: 346

Severity: ERROR
Message: An error occurred while trying to write updated data to file %s for the LDIF backend defined in configuration entry %s: %s.

ID: 347

Severity: ERROR
Message: An error occurred while attempting to rename file %s to %s while writing updated data for the LDIF backend defined in configuration entry %s: %s.

ID: 348

Severity: ERROR
Message: Entry %s already exists in the LDIF backend.

ID: 349

Severity: ERROR
Message: The parent for entry %s does not exist.

ID: 350

Severity: ERROR
Message: Entry %s does not exist.

ID: 351

Severity: ERROR
Message: Entry %s has one or more subordinate entries and cannot be deleted until all of its subordinate entries are removed first.

ID: 352

Severity: ERROR
Message: Entry %s does not exist.

ID: 353

Severity: ERROR
Message: Source entry %s does not exist.

ID: 354

Severity: ERROR
Message: Target entry %s already exists.

ID: 355

Severity: ERROR
Message: The new parent DN %s does not exist.

ID: 356

Severity: ERROR
Message: Entry %s specified as the search base DN does not exist.

ID: 357

Severity: ERROR
Message: An error occurred while trying to create the writer for the LDIF export operation: %s.

ID: 358

Severity: ERROR
Message: An error occurred while trying to write entry %s during the LDIF export: %s.

ID: 359

Severity: ERROR
Message: An error occurred while trying to create the reader for the LDIF import operation: %s.

ID: 360

Severity: ERROR
Message: An unrecoverable error occurred while attempting to read data from the import file: %s. The LDIF import cannot continue.

ID: 365

Severity: ERROR
Message: The target entry %s does not exist.

ID: 366

Severity: ERROR
Message: The target entry %s does not exist.

ID: 369

Severity: ERROR
Message: This backend does not provide support for the numSubordinates operational attribute.

ID: 371

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid number of tokens.

ID: 372

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid minute token.

ID: 373

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid hour token.

ID: 374

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid day of the month token.

ID: 375

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid month of the year token.

ID: 376

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid day of the week token.

ID: 377

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid tokens combination yielding a nonexistent calendar date.

ID: 378

Severity: ERROR
Message: An error occurred while attempting to export task backend data: %s.

ID: 412

Severity: ERROR
Message: Unable to schedule task %s because its dependency task %s is missing.

ID: 416

Severity: ERROR
Message: Unable to remove ldap syntax description %s from the server schema because no such ldap syntax description is defined.

ID: 418

Severity: ERROR
Message: An error occurred while attempting to decode the ldapsyntax description "%s": %s.

ID: 419

Severity: ERROR
Message: The provided recurring task schedule value has an invalid number of tokens.

ID: 420

Severity: ERROR
Message: The provided recurring task schedule value has an invalid minute token.

ID: 421

Severity: ERROR
Message: The provided recurring task schedule value has an invalid hour token.

ID: 422

Severity: ERROR
Message: The provided recurring task schedule value has an invalid day of the month token.

ID: 423

Severity: ERROR
Message: The provided recurring task schedule value has an invalid month of the year token.

ID: 424

Severity: ERROR
Message: The provided recurring task schedule value has an invalid day of the week token.

ID: 425

Severity: ERROR
Message: The schema backend does not support the Replace modification type for the %s attribute type.

ID: 426

Severity: ERROR
Message: An error occurred while trying to close file %s for the LDIF backend defined in configuration entry %s: %s.

ID: 427

Severity: ERROR
Message: The file %s written for the LDIF backend defined in configuration entry %s is 0 bytes long and unusable.

ID: 428

Severity: ERROR
Message: Configuration attribute ds-cfg-db-cache-size has a value of %d but the JVM has only %d available. Consider using ds-cfg-db-cache-percent.

ID: 429

Severity: ERROR
Message: Configuration attribute ds-cfg-db-cache-percent has a value of %d%% but the JVM has only %d%% available.

ID: 430

Severity: ERROR
Message: Unable to process the virtual list view request because the target assertion could not be decoded as a valid value for the '%s' attribute type.

ID: 433

Severity: ERROR
Message: An error occurred while trying to list the files to backup for backend '%s': %s.

ID: 438

Severity: ERROR
Message: Insufficient free memory (%d bytes) to perform import. At least %d bytes of free memory is required.

ID: 440

Severity: ERROR
Message: Index for attribute '%s' cannot be created for matching rule '%s' because it cannot be found in the schema. Fix the matching rule name in the config or add the matching rule to the schema.

ID: 441

Severity: ERROR
Message: Unable to process the virtual list view request because the target start position was before the beginning of the result set.

ID: 443

Severity: ERROR
Message: The entry database does not contain a record for ID %s.

ID: 445

Severity: ERROR
Message: Execution error during backend operation: %s.

ID: 447

Severity: ERROR
Message: The backend database directory could not be created: %s.

ID: 451

Severity: ERROR
Message: The backend database directory '%s' is not a valid directory.

ID: 453

Severity: ERROR
Message: The entry '%s' cannot be added because an entry with that name already exists.

ID: 454

Severity: ERROR
Message: The entry '%s' cannot be added because its parent entry does not exist.

ID: 455

Severity: ERROR
Message: There is no index configured for attribute type '%s'.

ID: 457

Severity: ERROR
Message: An error occurred while attempting to decode an attribute description token from the compressed schema definitions: %s.

ID: 458

Severity: ERROR
Message: An error occurred while attempting to decode an object class set token from the compressed schema definitions: %s.

ID: 459

Severity: ERROR
Message: An error occurred while attempting to store compressed schema information in the database: %s.

ID: 460

Severity: ERROR
Message: An error occurred while parsing the search filter %s defined for VLV index %s: %s.

ID: 461

Severity: ERROR
Message: Sort attribute %s for VLV index %s is not defined in the server schema.

ID: 462

Severity: ERROR
Message: Database exception: %s.

ID: 463

Severity: ERROR
Message: A plugin caused the delete operation to be aborted while deleting a subordinate entry %s.

ID: 464

Severity: ERROR
Message: The entry '%s' cannot be removed because it has subordinate entries.

ID: 465

Severity: ERROR
Message: The entry '%s' cannot be removed because it does not exist.

ID: 466

Severity: ERROR
Message: An entry container named '%s' is already registered for base DN '%s'.

ID: 467

Severity: ERROR
Message: The entry database does not contain a valid record for ID %s.

ID: 468

Severity: ERROR
Message: I/O error occurred while exporting entry: %s.

ID: 469

Severity: ERROR
Message: The backend must be disabled before the import process can start.

ID: 471

Severity: ERROR
Message: Unable to create the temporary directory %s.

ID: 481

Severity: ERROR
Message: The import has been aborted because the entry '%s' does not have a parent entry.

ID: 482

Severity: ERROR
Message: Entry record is not compatible with this version of the backend database. Entry version: %x.

ID: 483

Severity: ERROR
Message: An error occurred while reading from index %s. The index seems to be corrupt and is now operating in a degraded state. The index must be rebuilt before it can return to normal operation.

ID: 484

Severity: ERROR
Message: The following paged results control cookie value was not recognized: %s.

ID: 487

Severity: ERROR
Message: A plugin caused the modify DN operation to be aborted while moving and/or renaming an entry from %s to %s.

ID: 489

Severity: ERROR
Message: The entry cannot be renamed to '%s' because an entry with that name already exists.

ID: 490

Severity: ERROR
Message: The entry '%s' cannot be renamed because it does not exist.

ID: 491

Severity: ERROR
Message: The entry '%s' cannot be modified because it does not exist.

ID: 492

Severity: ERROR
Message: The entry cannot be moved because the new parent entry '%s' does not exist.

ID: 493

Severity: ERROR
Message: The database environment could not be opened: %s.

ID: 494

Severity: ERROR
Message: Rebuilding system index(es) must be done with the backend containing the base DN disabled.

ID: 495

Severity: ERROR
Message: The backend database files could not be removed: %s.

ID: 496

Severity: ERROR
Message: The requested search operation included both the simple paged results control and the virtual list view control. These controls are mutually exclusive and cannot be used together.

ID: 497

Severity: ERROR
Message: The search results cannot be sorted because the given search request is not indexed.

ID: 498

Severity: ERROR
Message: The search base entry '%s' does not exist.

ID: 499

Severity: ERROR
Message: You do not have sufficient privileges to perform an unindexed search.

ID: 500

Severity: ERROR
Message: Unchecked exception during database transaction: %s.

ID: 501

Severity: ERROR
Message: There is no VLV index configured with name '%s'.

ID: 561

Severity: ERROR
Message: The database logging level string '%s' provided for configuration entry '%s' is invalid. The value must be one of OFF, SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST, or ALL. Note that these values are case sensitive.

ID: 569

Severity: ERROR
Message: Configuration attribute ds-cfg-db-cache-size has a value of %d which is less than the minimum: %d.

ID: 579

Severity: ERROR
Message: The backend must be disabled before verification process can start.

ID: 583

Severity: ERROR
Message: Missing ID %d%n%s.

ID: 585

Severity: ERROR
Message: Reference to unknown entry ID %s%n%s.

ID: 586

Severity: ERROR
Message: The entry with ID %s is associated with the wrong key%n%s.

ID: 587

Severity: ERROR
Message: Empty ID set: %n%s.

ID: 588

Severity: ERROR
Message: Duplicate reference to ID %d%n%s.

ID: 589

Severity: ERROR
Message: Reference to unknown ID %d%n%s.

ID: 590

Severity: ERROR
Message: Reference to entry <%s> which does not match the value%n%s.

ID: 591

Severity: ERROR
Message: File dn2id is missing key %s.

ID: 592

Severity: ERROR
Message: File dn2id has ID %d instead of %d for key %s.

ID: 593

Severity: ERROR
Message: File dn2id has DN <%s> referencing unknown ID %d.

ID: 594

Severity: ERROR
Message: File dn2id has DN <%s> referencing entry with wrong DN <%s>.

ID: 595

Severity: ERROR
Message: The stored entry count in id2entry (%d) does not agree with the actual number of entry records found (%d).

ID: 596

Severity: ERROR
Message: File id2childrenCount has wrong number of children for DN <%s> (got %d, expecting %d).

ID: 597

Severity: ERROR
Message: File id2ChildrenCount references non-existing EntryID <%d>.

ID: 600

Severity: ERROR
Message: Ignoring schema definition '%s' because the following error occurred while it was being parsed: %s.

ID: 601

Severity: ERROR
Message: Schema definition could not be parsed as valid attribute value.

ID: 602

Severity: ERROR
Message: Attribute %s is set as confidential on a backend whose entries are still cleartext. Enable confidentiality on the backend first.

ID: 603

Severity: ERROR
Message: The attribute '%s' cannot enable confidentiality for keys and values at the same time.

ID: 604

Severity: ERROR
Message: Cannot encode entry for writing on storage: %s.

ID: 605

Severity: ERROR
Message: Input stream ended unexpectedly while decoding entry.

ID: 606

Severity: ERROR
Message: Confidentiality cannot be disabled on suffix '%s' because the following indexes have confidentiality still enabled: %s.

ID: 608

Severity: ERROR
Message: Error while enabling confidentiality with cipher %s, %d bits: %s.

ID: 609

Severity: ERROR
Message: The import has been aborted because the data to be imported contains duplicate copies of entry '%s'.

ID: 611

Severity: ERROR
Message: Proxy backend '%s' could not discover remote servers capabilities: %s.

ID: 615

Severity: ERROR
Message: Proxy backend '%s' is non functional because it could not find any primary nor secondary servers via the service discovery mechanism '%s'.

ID: 616

Severity: ERROR
Message: Proxy backend '%s' cannot find the configured service discovery mechanism '%s'.

ID: 622

Severity: ERROR
Message: No backend is associated with the base DN '%s'.

ID: 624

Severity: ERROR
Message: Proxy backend '%s' cannot register itself against base DN %s because this base DN is already registered against backend '%s'.

ID: 625

Severity: ERROR
Message: Proxy backend '%s' is being deregistered from base DN %s because local backend '%s' is registering against it. Local backends take precedence over proxy backends.

ID: 639

Severity: ERROR
Message: The partition base DN '%s' should be subordinate to one of the base DNs %s of proxy backend '%s'.

ID: 640

Severity: ERROR
Message: Backend database cache preload for backend '%s' is not supported in this release.

ID: 644

Severity: ERROR
Message: There are insufficient resources to perform the operation.

ID: 645

Severity: ERROR
Message: The time-to-live (TTL) feature can only be enabled for generalized time ordering indexes.

ID: 646

Severity: ERROR
Message: An unexpected error occurred while purging expired entries: %s.

ID: 650

Severity: ERROR
Message: The partition base DN '%s' shouldn't be subordinate to one of the other partition base DNs %s of proxy backend '%s'.

ID: 656

Severity: ERROR
Message: An error occurred while trying to retrieve the key managers from the key manager provider `%s`.

ID: 658

Severity: ERROR
Message: Could not stop export-ldif threads after 30 seconds. Now forcing stop by interrupting them.

ID: 660

Severity: ERROR
Message: The index(es) cannot be rebuilt because the server failed to obtain a write lock for the entry '%s' after multiple attempts.

ID: 661

Severity: ERROR
Message: VLV index '%s' must be configured with at least one sort attribute.

ID: 663

Severity: ERROR
Message: Missing entry %s in index %s.

ID: 664

Severity: ERROR
Message: Big index for attribute '%s' cannot be created for matching rule '%s' because it is not an equality matching rule.

ID: 666

Severity: ERROR
Message: A Server Side Sort control must be specified whenever a Virtual List View control is present.

ID: 667

Severity: ERROR
Message: Counter of %s reports wrong number of entries for key <%s> (got %d, expecting %d).

ID: 668

Severity: ERROR
Message: An error occurred while attempting to send an email for the completion of %s task: Task ID: %s, Task State: %s, Scheduled Start Time: %s, Actual Start Time: %s, Completion Time: %s. The error was: %s.

ID: 669

Severity: ERROR
Message: Index for attribute '%s' cannot be created because the configuration contains an included attribute value '%s' which appears to be invalid according to the schema: %s.

ID: 672

Severity: ERROR
Message: An internal error occurred when accessing backend '%s': %s.

ID: 673

Severity: ERROR
Message: An internal error was detected when accessing backend '%s'.

Category: LDAP schema

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 26

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because the last non-space character was a comma or semicolon.

ID: 28

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because character '%c' at position %d is not allowed in an attribute name.

ID: 30

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because the hyphen character is not allowed as the first character of an attribute name.

ID: 33

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because it contained an RDN containing an empty attribute name.

ID: 34

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because the parsed attribute name %s included a period but that name did not appear to be a valid OID.

ID: 35

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because the last non-space character was part of the attribute name '%s'.

ID: 36

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because the next non-space character after attribute name "%s" should have been an equal sign but instead was '%c'.

ID: 37

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because character '%c' at position %d is not valid.

ID: 38

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because an attribute value started with an octothorpe (#) but was not followed by a positive multiple of two hexadecimal digits.

ID: 39

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because an attribute value started with an octothorpe (#) but contained a character %c that was not a valid hexadecimal digit.

ID: 40

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because an unexpected failure occurred while attempting to parse an attribute value from one of the RDN components: "%s".

ID: 41

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because one of the RDN components included a quoted value that did not have a corresponding closing quotation mark.

ID: 42

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid distinguished name because one of the RDN components included a value with an escaped hexadecimal digit that was not followed by a second hexadecimal digit.

ID: 257

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid RFC 3672 subtree specification.

ID: 269

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid subtree specification.

ID: 341

Severity: ERROR
Message: There should be no warnings on the schema, but instead got %d warnings: %s.

ID: 344

Severity: ERROR
Message: Unable to parse the OID from the provided definition of objectclass: '%s'.

ID: 345

Severity: ERROR
Message: Unable to parse the OID from the provided definition of attribute type: '%s'.

ID: 346

Severity: ERROR
Message: Unable to parse the OID from the provided definition of ldap syntax: '%s'.

ID: 347

Severity: ERROR
Message: Unable to parse the OID from the provided definition of matching rule use: '%s'.

ID: 350

Severity: ERROR
Message: Unable to parse the OID from the provided definition of name form: '%s'.

ID: 351

Severity: ERROR
Message: Unable to parse the OID from the provided definition of DIT content rule: '%s'.

ID: 352

Severity: ERROR
Message: Unable to parse the rule ID from the provided definition of DIT structure rule: '%s'.

Category: Native logging

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 1

Severity: ERROR
Message: Error occurred while writing log record for logger %s: %s. Any further write errors will be ignored.

ID: 2

Severity: ERROR
Message: Error occurred while opening log file %s for logger %s: %s.

ID: 3

Severity: ERROR
Message: Error occurred while closing log file for logger %s: %s.

ID: 4

Severity: ERROR
Message: Error occurred while flushing writer buffer for logger %s: %s.

ID: 10

Severity: ERROR
Message: Error occurred while listing log files named by policy with initial file name %s.

ID: 11

Severity: ERROR
Message: Error occurred while obtaining free disk space in the partition containing log file %s: %s.

ID: 12

Severity: ERROR
Message: Error occurred while enforcing retention policy %s for logger %s: %s.

ID: 14

Severity: ERROR
Message: Error while creating or updating common audit log publisher %s: %s.

ID: 16

Severity: ERROR
Message: Error while adding common audit log publisher %s, the publisher has an unsupported handler type.

ID: 17

Severity: ERROR
Message: Error while reading JSON configuration file %s while creating common audit external log publisher %s: %s.

ID: 18

Severity: ERROR
Message: Error while creating common audit external log publisher %s: %s.

ID: 19

Severity: ERROR
Message: Error while creating common audit log publisher %s: %s.

ID: 20

Severity: ERROR
Message: Error while adding common audit log publisher %s, the publisher defines an unsupported log rotation policy %s.

ID: 21

Severity: ERROR
Message: Error while adding common audit log publisher %s, the publisher defines an unsupported log retention policy %s.

ID: 22

Severity: ERROR
Message: Error while processing common audit log publisher %s, this type of log publisher is unsupported.

ID: 23

Severity: ERROR
Message: Error while processing common audit log publisher %s, delimiter char '%s' should not contains more than one character.

ID: 24

Severity: ERROR
Message: Error while processing common audit log publisher %s, quote char '%s' should not contains more than one character.

ID: 25

Severity: ERROR
Message: Error while processing common audit log publisher %s, time of the day value '%s' for fixed time log rotation policy is not valid, it should use a 24-hour format "HHmm" : %s.

ID: 27

Severity: ERROR
Message: Error while processing a log event for common audit: %s.

ID: 28

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore pin file %s is missing.

ID: 29

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore pin file %s could not be read: %s.

ID: 30

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore pin file %s contains an empty pin.

ID: 31

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore file %s is missing.

ID: 32

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore file %s could not be read: %s.

ID: 33

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore file %s is empty.

ID: 34

Severity: ERROR
Message: The log file %s unexpectedly disappeared. It looks like an external system is also trying to manage the log files retention (either deleting or moving files away). This system configuration is incorrect: either ForgeRock DS manages the log file retention OR the external system does. Pick one only.

ID: 35

Severity: ERROR
Message: The message with ID %d has been generated %d times in the last second: %s.

Category: Replication

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 1

Severity: ERROR
Message: The configured DN is already used by another domain.

ID: 6

Severity: ERROR
Message: Replication Server failed to start : could not bind to the listen port : %d. Error : %s.

ID: 7

Severity: ERROR
Message: Unknown operation type : %s.

ID: 9

Severity: ERROR
Message: Internal Error : Operation %s change number %s was not found in local change list.

ID: 10

Severity: ERROR
Message: Internal Error : Operation %s change number %s was not found in remote change list.

ID: 11

Severity: ERROR
Message: The replication server failed to start because the database %s could not be read : %s.

ID: 12

Severity: ERROR
Message: An Exception was caught while replaying operation %s %s (%s) : %s.

ID: 15

Severity: ERROR
Message: Error %s when searching for server state %s : %s base dn : %s.

ID: 20

Severity: ERROR
Message: Caught IOException while sending topology info (for update) on domain %s for %s server %s : %s.

ID: 21

Severity: ERROR
Message: Error when searching old changes from the database for base DN %s: %s.

ID: 25

Severity: ERROR
Message: Error trying to replay %s, operation could not be decoded: %s.

ID: 29

Severity: ERROR
Message: Error during the Replication Server database trimming or flush process. The Changelog service is going to shutdown: %s.

ID: 32

Severity: ERROR
Message: An unexpected error happened handling connection with %s. This connection is going to be closed.

ID: 35

Severity: ERROR
Message: A loop was detected while replaying operation: %s %s (%s) error %s.

ID: 36

Severity: ERROR
Message: An Exception was caught while testing existence or trying to create the directory for the Replication Server database : %s %s.

ID: 44

Severity: ERROR
Message: The current request is rejected due to an import or an export already in progress for the same data.

ID: 45

Severity: ERROR
Message: On domain %s, initialization of server with serverId:%s has been requested from a server with an invalid serverId:%s. %s.

ID: 46

Severity: ERROR
Message: Invalid target for the export.

ID: 47

Severity: ERROR
Message: Domain %s: the server with serverId=%s is unreachable.

ID: 48

Severity: ERROR
Message: No domain matches the provided base DN '%s'.

ID: 49

Severity: ERROR
Message: Multiple domains match the base DN provided.

ID: 50

Severity: ERROR
Message: The provider class does not allow the operation requested.

ID: 54

Severity: ERROR
Message: Duplicate server IDs found: replica '%s' for domain '%s' tried to connect from '%s', but replica '%s' is already connected from '%s'. Make sure the two replicas are configured with different server IDs.

ID: 55

Severity: ERROR
Message: Duplicate server IDs found: replication server '%s' tried to connect from '%s', but replication server '%s' is already connected from '%s'. Make sure the two replication servers are configured with different server IDs.

ID: 56

Severity: ERROR
Message: Entry %s was containing some unknown historical information, This may cause some inconsistency for this entry.

ID: 57

Severity: ERROR
Message: A conflict was detected but the conflict information could not be added to entry %s. Conflict with %s %s, Result: %s.

ID: 58

Severity: ERROR
Message: An error happened trying to rename a conflicting entry. DN: %s, Operation: %s, Result: %s.

ID: 61

Severity: ERROR
Message: The Replication is configured for suffix %s but was not able to connect to any Replication Server.

ID: 65

Severity: ERROR
Message: An unexpected error occurred while sending an Error Message to %s. This connection is going to be closed and reopened.

ID: 66

Severity: ERROR
Message: An unexpected error occurred while sending a Message to %s. This connection is going to be closed and reopened.

ID: 67

Severity: ERROR
Message: Could not replay operation %s %s with ChangeNumber %s error %s %s.

ID: 68

Severity: ERROR
Message: The entry %s has historical information for attribute %s which is not defined in the schema. This information will be ignored.

ID: 70

Severity: ERROR
Message: The Replication Server socket could not be closed : %s.

ID: 71

Severity: ERROR
Message: The thread listening on the replication server port could not be stopped : %s.

ID: 73

Severity: ERROR
Message: An unexpected error occurred when searching for generation id for domain "%s": %s.

ID: 74

Severity: ERROR
Message: An unexpected error occurred when looking for the replicated backend : %s. It may be not configured or disabled.

ID: 75

Severity: ERROR
Message: An unexpected error occurred when searching in %s for the generation ID : %s.

ID: 76

Severity: ERROR
Message: An unexpected error occurred when updating generation ID for domain "%s": %s.

ID: 79

Severity: ERROR
Message: The following error has been received : %s.

ID: 82

Severity: ERROR
Message: Initialization cannot be done because import is not supported by the backend %s.

ID: 83

Severity: ERROR
Message: Initialization cannot be done because export is not supported by the backend %s.

ID: 84

Severity: ERROR
Message: Initialization cannot be done because the following error occurred while locking the backend %s : %s.

ID: 86

Severity: ERROR
Message: Replication server caught exception while listening for client connections: %s.

ID: 87

Severity: ERROR
Message: While clearing the database %s , the following error happened: %s.

ID: 89

Severity: ERROR
Message: An unexpected error occurred when testing existence or creating the replication backend : %s.

ID: 93

Severity: ERROR
Message: An error occurred when searching for %s : %s.

ID: 95

Severity: ERROR
Message: The base DN %s is not stored by any of the Directory Server backend.

ID: 109

Severity: ERROR
Message: An Exception was caught while replaying replication message : %s.

ID: 114

Severity: ERROR
Message: Caught exception publishing fake operations for domain %s : %s.

ID: 115

Severity: ERROR
Message: Caught exception computing fake operations for domain %s for replication server %s : %s.

ID: 118

Severity: ERROR
Message: For replicated domain %s, in server with serverId=%s, the generation ID could not be set to value %s in the rest of the topology because this server is NOT connected to any replication server. You should check in the configuration that the domain is enabled and that there is one replication server up and running.

ID: 119

Severity: ERROR
Message: Directory server DS(%s) encountered an unexpected error while connecting to replication server %s for domain "%s": %s.

ID: 121

Severity: ERROR
Message: DN sent by remote replication server: %s does not match local replication server one: %s.

ID: 122

Severity: ERROR
Message: DN sent by replication server: %s does not match local directory server one: %s.

ID: 123

Severity: ERROR
Message: Caught IOException while forwarding ResetGenerationIdMsg to peer replication servers for domain %s : %s.

ID: 124

Severity: ERROR
Message: Computed invalid initial status: %s in DS replication domain %s with server id %s.

ID: 125

Severity: ERROR
Message: Replication server received invalid initial status: %s for replication domain %s from server id %s.

ID: 126

Severity: ERROR
Message: Received invalid requested status %s in DS replication domain %s with server id %s.

ID: 127

Severity: ERROR
Message: Could not compute new status in RS replication domain %s for server id %s. Was in %s status and received %s event.

ID: 128

Severity: ERROR
Message: Could not compute new status in DS replication domain %s with server id %s. Was in %s status and received %s event.

ID: 129

Severity: ERROR
Message: Caught IOException while changing status for domain %s and serverId: %s after reset for generation id: %s.

ID: 130

Severity: ERROR
Message: Received change status message does not come from a directory server (dn: %s, server id: %s, msg: %s).

ID: 132

Severity: ERROR
Message: Received invalid new status %s in RS for replication domain %s and directory server id %s.

ID: 134

Severity: ERROR
Message: Replication broker with dn %s and server id %s failed to signal status change because of: %s.

ID: 139

Severity: ERROR
Message: Caught IOException while changing status for domain %s and serverId: %s from status analyzer: %s.

ID: 152

Severity: ERROR
Message: The generation ID could not be reset for domain %s.

ID: 154

Severity: ERROR
Message: The Replication was not started on base-dn %s : %s.

ID: 157

Severity: ERROR
Message: Replication protocol error. Bad message type. %s received, %s required.

ID: 168

Severity: ERROR
Message: The fractional replication ldif import plugin is configured with invalid plugin type %s. Only the ldifImport plugin type is allowed.

ID: 173

Severity: ERROR
Message: An error occurred when accessing the change number database : %s.

ID: 174

Severity: ERROR
Message: The initialization failed because the domain %s is not connected to a replication server.

ID: 175

Severity: ERROR
Message: Could not retrieve the configuration for a replication domain matching the entry %s.

ID: 181

Severity: ERROR
Message: The connection from this replication server RS(%s) to replication server RS(%s) at %s for domain "%s" has failed.

ID: 185

Severity: ERROR
Message: Full resync required. Reason: The provided cookie contains unknown replicated domain %s. Current starting cookie <%s>.

ID: 186

Severity: ERROR
Message: Full resync required. Reason: The provided cookie is older than the start of historical in the server for the replicated domain : %s.

ID: 187

Severity: ERROR
Message: Invalid syntax for the provided cookie '%s'.

ID: 189

Severity: ERROR
Message: Domain %s (server id: %s) : remote exporter server disconnection (server id: %s) detected during initialization.

ID: 190

Severity: ERROR
Message: During initialization from a remote server, the following error occurred : %s.

ID: 191

Severity: ERROR
Message: Connection failure with Replication Server %s during import.

ID: 192

Severity: ERROR
Message: Bad msg id sequence during import. Expected:%s Actual:%s.

ID: 193

Severity: ERROR
Message: The following servers did not acknowledge initialization in the expected time for domain %s. They are potentially down or too slow. Servers list: %s.

ID: 194

Severity: ERROR
Message: The following servers did not end initialization being connected with the right generation (%s). They are potentially stopped or too slow. Servers list: %s.

ID: 195

Severity: ERROR
Message: When initializing remote server(s), connection to Replication Server with serverId=%s is lost.

ID: 196

Severity: ERROR
Message: When initializing remote server(s), the initialized server with serverId=%s is potentially stopped or too slow.

ID: 197

Severity: ERROR
Message: When sending a new initialization request for an initialization from a remote server, the following error occurred %s. The initial error was : %s.

ID: 202

Severity: ERROR
Message: Error while trying to solve conflict with DN : %s ERROR : %s.

ID: 211

Severity: ERROR
Message: The connection from this replication server RS(%s) to directory server DS(%s) at %s for domain "%s" has failed.

ID: 235

Severity: ERROR
Message: Could not create replica database because the changelog database is shutting down.

ID: 240

Severity: ERROR
Message: Could not add change %s to replicaDB %s %s because flushing thread is shutting down.

ID: 243

Severity: ERROR
Message: Error when retrieving changelog state from root path '%s' : IO error on domain directory '%s' when retrieving list of server ids.

ID: 244

Severity: ERROR
Message: Could not get or create replica DB for base DN '%s', serverId '%s', generationId '%d': %s.

ID: 245

Severity: ERROR
Message: Could not get or create change number index DB in root path '%s', using path '%s'.

ID: 246

Severity: ERROR
Message: Could not delete generation id file '%s' for DN '%s': %s.

ID: 247

Severity: ERROR
Message: Could not create directory '%s' for server id %s: %s.

ID: 248

Severity: ERROR
Message: Could not create generation id file '%s': %s.

ID: 250

Severity: ERROR
Message: Could not read server id filename because it uses a wrong format, expecting '[id].server' where [id] is numeric but got '%s'.

ID: 251

Severity: ERROR
Message: Could not read generation id because it uses a wrong format, expecting a number but got '%s'.

ID: 252

Severity: ERROR
Message: Could not open log file '%s' for write.

ID: 253

Severity: ERROR
Message: Could not open a reader on log file '%s': %s.

ID: 254

Severity: ERROR
Message: Could not decode a record from data read in log file '%s'.

ID: 255

Severity: ERROR
Message: Could not delete log files: %s.

ID: 256

Severity: ERROR
Message: Could not create log file '%s'.

ID: 258

Severity: ERROR
Message: Could not add record '%s' in log file '%s'.

ID: 259

Severity: ERROR
Message: Could not synchronize written records to file system for log file '%s'.

ID: 260

Severity: ERROR
Message: Could not seek to position %d for reader on log file '%s'.

ID: 261

Severity: ERROR
Message: Could not create root directory '%s' for log file: %s.

ID: 262

Severity: ERROR
Message: Could not decode DN from domain state file '%s', from line '%s'.

ID: 263

Severity: ERROR
Message: Could not read domain state file '%s'. The replication server cannot continue, it should be restored from a backup. Cause was : %s.

ID: 264

Severity: ERROR
Message: There is a mismatch between domain state file and actual domain directories found in file system. Expected domain ids : %s. Actual domain ids found in file system: %s.

ID: 265

Severity: ERROR
Message: Could not create a new domain id %s for domain DN %s and save it in the domain state file. Replication will continue, but if the domain state file cannot be written when stopping the server, it should be restored from backup. Cause was : %s.

ID: 267

Severity: ERROR
Message: Could not decode the key from string [%s].

ID: 270

Severity: ERROR
Message: Could not initialize the log '%s' : %s.

ID: 271

Severity: ERROR
Message: Could not retrieve key bounds from log file '%s'.

ID: 272

Severity: ERROR
Message: Could not retrieve read-only log files from log '%s'.

ID: 273

Severity: ERROR
Message: While purging log, could not delete log file(s): '%s'.

ID: 274

Severity: ERROR
Message: The following log '%s' must be released but it is not referenced.

ID: 279

Severity: ERROR
Message: Could not read replica offline state file '%s' for domain %s, it should contain exactly one line corresponding to the offline CSN.

ID: 280

Severity: ERROR
Message: Could not read content of replica offline state file '%s' for domain %s.

ID: 282

Severity: ERROR
Message: Could not retrieve file length of log file '%s'.

ID: 283

Severity: ERROR
Message: An error occurred while recovering the replication change log file '%s'. The recovery has been aborted and this replication server will be removed from the replication topology. The change log file system may be read-only, full, or corrupt and must be fixed before this replication server can be used. The underlying error was: %s.

ID: 286

Severity: ERROR
Message: An error occurred when searching base DN '%s' with filter '%s' in changelog backend : %s.

ID: 287

Severity: ERROR
Message: An error occurred when retrieving attribute value for attribute '%s' for entry DN '%s' in changelog backend : %s.

ID: 288

Severity: ERROR
Message: Could not create file '%s' to store last log rotation time %d.

ID: 289

Severity: ERROR
Message: Could not delete file '%s' that stored the previous last log rotation time.

ID: 290

Severity: ERROR
Message: Cursor on log '%s' has been aborted after a purge.

ID: 291

Severity: ERROR
Message: Could not position and read newest record from log file '%s'.

ID: 293

Severity: ERROR
Message: The change number index could not be reset to start with %d in base DN '%s' because starting CSN '%s' does not exist in the change log.

ID: 294

Severity: ERROR
Message: The change number could not be reset to %d because the associated change with CSN '%s' has already been purged from the change log. Try resetting to a more recent change.

ID: 295

Severity: ERROR
Message: Change number indexing is disabled for replication domain '%s'.

ID: 297

Severity: ERROR
Message: Cannot decode change-log record with version %x.

ID: 298

Severity: ERROR
Message: Cannot start total update in domain "%s" from this directory server DS(%s): no remote directory servers found.

ID: 299

Severity: ERROR
Message: Cannot start total update in domain "%s" from this directory server DS(%s): cannot find remote directory server DS(%s).

ID: 300

Severity: ERROR
Message: New replication connection from %s started with unexpected message %s and is being closed.

ID: 301

Severity: ERROR
Message: The directory server %s can no longer keep up with changes coming from replication server %s for base DN %s. Some missing changes have been purged by this replication server and the connection will be terminated. The directory server may fail-over to another replication server that has not purged the changes that it needs. If there is no replication server containing the missing changes then it will fail to connect to any replication server and will need to be reinitialized. (Underlying error is: %s).

ID: 302

Severity: ERROR
Message: The replication server %s can no longer keep up with changes coming from replication server %s for base DN %s. Some missing changes have been purged by this replication server and the connection will be terminated. The directory servers connected to this replication server may fail-over to another replication server that has not purged the changes that it needs. If there is no replication server containing the missing changes then the directory servers will fail to connect to any replication server and will need to be reinitialized. (Underlying error is: %s).

ID: 305

Severity: ERROR
Message: Invalid operator '%s' specified in historicalCsnRangeMatch extensible matching rule assertion.

ID: 306

Severity: ERROR
Message: Specified assertion '%s' for historicalCsnRangeMatch extensible matching rule does not conform to expected syntax. The assertion must specify a CSN range.

ID: 307

Severity: ERROR
Message: Specified CSNs '%s' and '%s' have two different server ids. The historicalCsnRangeMatch extensible matching rule requires CSNs to have the same server id.

ID: 308

Severity: ERROR
Message: Specified operators '%s' and '%s' do not specify a range for historicalCsnRangeMatch extensible matching rule.

ID: 309

Severity: ERROR
Message: Could not restart the Replication Server, bind to listen port %d failed : %s.

ID: 310

Severity: ERROR
Message: The replication server has detected that the file system containing the changelog is full. In order to prevent further problems, the replication server will disconnect from the replication topology and wait for sufficient disk space to be recovered, at which point it will reconnect.

ID: 315

Severity: ERROR
Message: The replication server connector thread could not be stopped : %s.

ID: 317

Severity: ERROR
Message: Unable to position reader to key '%s' using strategy '%s' on log '%s'. Changelog may be corrupted. Directory servers connected to this replication server may need to be reinitialized.

ID: 318

Severity: ERROR
Message: Assured replication is not supported anymore, it should be disabled on the topology.

ID: 320

Severity: ERROR
Message: Replication Server '%s' expected to negotiate with another Replication Server but got information for Directory Server '%s' instead. The connection will be closed.

ID: 324

Severity: ERROR
Message: Detected one or more corrupted records in log file '%s', this replication server will be removed from the replication topology. Recover the server from a valid filesystem backup if available or re-create it.

ID: 325

Severity: ERROR
Message: An error occurred while verifying integrity of log file '%s' : %s.

ID: 327

Severity: ERROR
Message: Cannot enable replication to server '%s' as this server's ID '%s' is not a number between 1 and 32767.

ID: 329

Severity: ERROR
Message: An error occurred in session '%s' when trying to send a message to the socket: %s.

ID: 331

Severity: ERROR
Message: Could not convert value '%s' to long.

ID: 332

Severity: ERROR
Message: Could not find replica update message matching the index record. No more replica update messages with a csn newer than %s exist.

ID: 333

Severity: ERROR
Message: An exception was encountered while trying to encode a replication %s message for entry "%s" into an External Change Log entry: %s.

ID: 334

Severity: ERROR
Message: Unexpected message type when trying to create changelog entry for dn %s : %s".

ID: 335

Severity: ERROR
Message: %s in Replication Server=%s, an initialization message of type %s cannot be sent to its destination server. Details: routing table is empty.

ID: 336

Severity: ERROR
Message: %s message of type %s cannot be routed. Details: %s.

ID: 338

Severity: ERROR
Message: Error when trying to publish a message in '%s'. The connection is going to be closed and reopened.

ID: 339

Severity: ERROR
Message: Error when trying to publish a message in '%s'. The connection is going to be closed and reopened. Error: %s.

ID: 340

Severity: ERROR
Message: Error when initializing publisher of messages in '%s'.

ID: 341

Severity: ERROR
Message: Error when initializing publisher of messages in '%s'.

ID: 342

Severity: ERROR
Message: Directory server %s was attempting to connect to replication server %s but an error occurred in handshake phase. Error: %s.

ID: 343

Severity: ERROR
Message: Replication server %s was attempting to connect to replication server %s but an error occurred in handshake phase. Error: %s.

ID: 344

Severity: ERROR
Message: Error when enabling external changelog: %s.

ID: 347

Severity: ERROR
Message: The domain state file cannot be written. The server will shutdown but it should be restored from backup. Cause was : %s.

ID: 348

Severity: ERROR
Message: Some data in the domain state file '%s' in not of the form <domainId>[csn:csn]:<domainDN>, the replication server cannot start. Restore the server from backup.

ID: 351

Severity: ERROR
Message: Error when purging historical information for entry %s: %s.

ID: 352

Severity: ERROR
Message: Replication server failed to start because setting socket timeout on port %d caused error %s.

ID: 354

Severity: ERROR
Message: %s for domain %s cannot route message of type %s to all the replicas in the topology because none are reachable.

ID: 355

Severity: ERROR
Message: "%s for domain %s cannot route message of type %s to replica %s because it is unreachable. Reachable replicas: %s.

ID: 356

Severity: ERROR
Message: Server %s should be initialized but is not connected to the topology.

ID: 358

Severity: ERROR
Message: Changelog file %s should be version %d, but found version %d. Likely causes for this error: the server binaries were accidentally downgraded, data was restored from a file system backup taken on a more recent version of the server, or the changelog file was corrupted. If the server was downgraded, upgrade it. If the data was restored from a file system backup, restore it again from a recent, valid file system backup taken on the same version of the server. Otherwise, as last resort, remove the changelog by running the 'dsrepl clear-changelog' command and restart the server.

ID: 366

Severity: ERROR
Message: Peer '%s' has sent an update, but it is not allowed to do so by the configuration of this replication server. This update will be discarded. Check the configuration of both this replication server and its peer to determine if they have to be adjusted.

ID: 367

Severity: ERROR
Message: Replication peer certificate verification failed with error: %s.

ID: 368

Severity: ERROR
Message: Digest algorithm '%s' of fingerprint '%s' is not supported by the JVM.

ID: 369

Severity: ERROR
Message: The following replication server listener threads were unexpectedly stopped: %s. A server restart may be needed.

ID: 370

Severity: ERROR
Message: Cursor on log '%s' has been aborted after a clear.

ID: 371

Severity: ERROR
Message: Initialization of domain '%s' interrupted: no data received from the remote exporter '%s' for %s.

ID: 376

Severity: ERROR
Message: Publish of a replication message to server '%s' was interrupted.

ID: 377

Severity: ERROR
Message: The changelog no longer contains changes for domain '%s' which are required by replica '%s'. The replica will no longer receive replicated changes and must be re-initialized.

Category: Server extensions

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 1

Severity: ERROR
Message: An error occurred while attempting to initialize the message digest generator for the %s algorithm: %s.

ID: 2

Severity: ERROR
Message: An error occurred while attempting to base64-decode the password value %s: %s.

ID: 3

Severity: ERROR
Message: The %s password storage scheme is not reversible, so it is impossible to recover the plaintext version of an encoded password.

ID: 4

Severity: ERROR
Message: An error occurred while trying to register the JMX alert handler with the MBean server: %s.

ID: 5

Severity: ERROR
Message: An unexpected error occurred while attempting to encode a password using the storage scheme defined in class %s: %s.

ID: 6

Severity: ERROR
Message: The ds-cfg-include-filter attribute of configuration entry %s, which specifies a set of search filters that may be used to control which entries are included in the cache, has an invalid value of "%s": %s.

ID: 7

Severity: ERROR
Message: The ds-cfg-exclude-filter attribute of configuration entry %s, which specifies a set of search filters that may be used to control which entries are excluded from the cache, has an invalid value of "%s": %s.

ID: 8

Severity: ERROR
Message: A fatal error occurred while trying to initialize fifo entry cache: %s.

ID: 9

Severity: ERROR
Message: A fatal error occurred while trying to initialize soft reference entry cache: %s.

ID: 33

Severity: ERROR
Message: An unexpected error occurred while attempting to decode the password modify extended request sequence: %s.

ID: 34

Severity: ERROR
Message: The password modify extended request cannot be processed because it does not contain an authorization ID and the underlying connection is not authenticated.

ID: 35

Severity: ERROR
Message: The password modify extended request cannot be processed because the server was unable to obtain a write lock on user entry %s after multiple attempts.

ID: 36

Severity: ERROR
Message: The password modify extended request cannot be processed because the server cannot decode "%s" as a valid DN for use in the authorization ID for the operation.

ID: 37

Severity: ERROR
Message: The password modify extended request cannot be processed because it contained an invalid userIdentity field. The provided userIdentity string was "%s".

ID: 38

Severity: ERROR
Message: The password modify extended request cannot be processed because it was not possible to identify the user entry to update based on the authorization DN of "%s".

ID: 41

Severity: ERROR
Message: The password modify extended operation cannot be processed because the current password provided for the user is invalid.

ID: 45

Severity: ERROR
Message: The keystore file %s specified in attribute ds-cfg-key-store-file of configuration entry %s does not exist.

ID: 62

Severity: ERROR
Message: An error occurred while trying to load the keystore contents from file %s: %s.

ID: 63

Severity: ERROR
Message: The keystore type %s specified in attribute ds-cfg-key-store-type of configuration entry %s is not valid: %s.

ID: 81

Severity: ERROR
Message: An error occurred while trying to access the PKCS#11 key manager: %s.

ID: 83

Severity: ERROR
Message: An error occurred while trying to create a key manager factory to access the contents of keystore file %s: %s.

ID: 84

Severity: ERROR
Message: An error occurred while trying to create a key manager factory to access the contents of the PKCS#11 keystore: %s.

ID: 87

Severity: ERROR
Message: The trust store file %s specified in attribute ds-cfg-trust-store-file of configuration entry %s does not exist.

ID: 104

Severity: ERROR
Message: An error occurred while trying to load the trust store contents from file %s: %s.

ID: 105

Severity: ERROR
Message: An error occurred while trying to create a trust manager factory to access the contents of trust store file %s: %s.

ID: 106

Severity: ERROR
Message: The trust store type %s specified in attribute ds-cfg-trust-store-type of configuration entry %s is not valid: %s.

ID: 118

Severity: ERROR
Message: Could not map the provided certificate chain to a user entry because no peer certificate was available.

ID: 119

Severity: ERROR
Message: Could not map the provided certificate chain to a user because the peer certificate was not an X.509 certificate (peer certificate format was %s).

ID: 120

Severity: ERROR
Message: Could not map the provided certificate chain to a user because the peer certificate subject "%s" could not be decoded as an LDAP DN: %s.

ID: 121

Severity: ERROR
Message: Could not map the provided certificate chain to a user because an error occurred while attempting to retrieve the user entry with DN "%s": %s.

ID: 122

Severity: ERROR
Message: Could not map the provided certificate chain to a user because no user entry exists with a DN of %s.

ID: 123

Severity: ERROR
Message: The SASL EXTERNAL bind request could not be processed because the associated bind request does not have a reference to the client connection.

ID: 124

Severity: ERROR
Message: The SASL EXTERNAL bind request could not be processed because the associated client connection instance is not an instance of LDAPClientConnection.

ID: 126

Severity: ERROR
Message: The SASL EXTERNAL bind request could not be processed because the client did not present a certificate chain during SSL/TLS negotiation.

ID: 127

Severity: ERROR
Message: The SASL EXTERNAL bind request failed because the certificate chain presented by the client during SSL/TLS negotiation could not be mapped to a user entry in the Directory Server.

ID: 128

Severity: ERROR
Message: StartTLS cannot be used on this connection because the underlying client connection is not available.

ID: 129

Severity: ERROR
Message: StartTLS cannot be used on this client connection because this connection type is not capable of using StartTLS to protect its communication.

ID: 137

Severity: ERROR
Message: Unable to authenticate via SASL EXTERNAL because the mapped user entry %s does not have any certificates with which to verify the presented peer certificate.

ID: 138

Severity: ERROR
Message: Unable to authenticate via SASL EXTERNAL because the mapped user entry %s did not contain the peer certificate presented by the client.

ID: 139

Severity: ERROR
Message: An error occurred while attempting to validate the peer certificate presented by the client with a certificate from the user's entry %s: %s.

ID: 147

Severity: ERROR
Message: SASL PLAIN authentication requires that SASL credentials be provided but none were included in the bind request.

ID: 148

Severity: ERROR
Message: The SASL PLAIN bind request did not include any NULL characters. NULL characters are required as delimiters between the authorization ID and authentication ID, and also between the authentication ID and the password.

ID: 149

Severity: ERROR
Message: The SASL PLAIN bind request did not include a second NULL character in the credentials, which is required as a delimiter between the authentication ID and the password.

ID: 150

Severity: ERROR
Message: The authentication ID contained in the SASL PLAIN bind request had a length of zero characters, which is not allowed. SASL PLAIN authentication does not allow an empty string for use as the authentication ID.

ID: 151

Severity: ERROR
Message: The password contained in the SASL PLAIN bind request had a length of zero characters, which is not allowed. SASL PLAIN authentication does not allow an empty string for use as the password.

ID: 152

Severity: ERROR
Message: An error occurred while attempting to decode the SASL PLAIN authentication ID "%s" because it appeared to contain a DN but DN decoding failed: %s.

ID: 153

Severity: ERROR
Message: The authentication ID in the SASL PLAIN bind request appears to be an empty DN. This is not allowed.

ID: 154

Severity: ERROR
Message: An error occurred while attempting to retrieve user entry %s as specified in the DN-based authentication ID of a SASL PLAIN bind request: %s.

ID: 157

Severity: ERROR
Message: The server was not able to find any user entries for the provided authentication ID of %s.

ID: 160

Severity: ERROR
Message: The provided password is invalid.

ID: 194

Severity: ERROR
Message: An unexpected error occurred while attempting to determine the value of the ds-cfg-server-fqdn attribute in configuration entry %s: %s.

ID: 195

Severity: ERROR
Message: An unexpected error occurred while trying to create an %s context: %s.

ID: 196

Severity: ERROR
Message: An error occurred while attempting to decode the SASL %s username "%s" because it appeared to contain a DN but DN decoding failed: %s.

ID: 197

Severity: ERROR
Message: The username in the SASL %s bind request appears to be an empty DN. This is not allowed.

ID: 199

Severity: ERROR
Message: An error occurred while attempting to retrieve user entry %s as specified in the DN-based username of a SASL %s bind request: %s.

ID: 200

Severity: ERROR
Message: The username contained in the SASL %s bind request had a length of zero characters, which is not allowed. %s authentication does not allow an empty string for use as the username.

ID: 201

Severity: ERROR
Message: The server was not able to find any user entries for the provided username of %s.

ID: 202

Severity: ERROR
Message: The provided authorization ID %s contained an invalid DN: %s.

ID: 203

Severity: ERROR
Message: The entry %s specified as the authorization identity does not exist.

ID: 204

Severity: ERROR
Message: The entry %s specified as the authorization identity could not be retrieved: %s.

ID: 205

Severity: ERROR
Message: The server was unable to find any entry corresponding to authorization ID %s.

ID: 207

Severity: ERROR
Message: An error occurred while attempting to retrieve the clear-text password(s) for user %s in order to perform SASL %s authentication: %s.

ID: 208

Severity: ERROR
Message: SASL %s authentication is not possible for user %s because none of the passwords in the user entry are stored in a reversible form.

ID: 209

Severity: ERROR
Message: SASL %s protocol error: %s.

ID: 210

Severity: ERROR
Message: The authenticating user %s does not have sufficient privileges to assume a different authorization identity.

ID: 211

Severity: ERROR
Message: The authenticating user %s does not have sufficient access to assume a different authorization identity.

ID: 212

Severity: ERROR
Message: The server was unable to find any entry corresponding to authentication ID %s.

ID: 213

Severity: ERROR
Message: The server was unable to because both the ds-cfg-kdc-address and ds-cfg-realm attributes must be defined or neither defined.

ID: 214

Severity: ERROR
Message: An error occurred while attempting to map authorization ID %s to a user entry: %s.

ID: 215

Severity: ERROR
Message: An error occurred while attempting to write a temporary JAAS configuration file for use during GSSAPI processing: %s.

ID: 216

Severity: ERROR
Message: An error occurred while attempting to create the JAAS login context for GSSAPI authentication: %s.

ID: 277

Severity: ERROR
Message: You do not have sufficient privileges to use the proxied authorization control.

ID: 306

Severity: ERROR
Message: ID string %s could not be mapped to exactly one user. It maps at least to both '%s' and '%s'.

ID: 307

Severity: ERROR
Message: The internal search based on ID string %s could not be processed efficiently: %s. Check the server configuration to ensure that all associated backends are properly configured for these types of searches.

ID: 308

Severity: ERROR
Message: An internal failure occurred while attempting to resolve ID string %s to a user entry: %s.

ID: 309

Severity: ERROR
Message: ID string %s mapped to multiple users.

ID: 319

Severity: ERROR
Message: An error occurred while attempting to map username %s to a Directory Server entry: %s.

ID: 325

Severity: ERROR
Message: An error occurred while attempting to map username %s to a Directory Server entry: %s.

ID: 327

Severity: ERROR
Message: Unable to process the cancel request because the extended operation did not include a request value.

ID: 328

Severity: ERROR
Message: An error occurred while attempting to decode the value of the cancel extended request: %s.

ID: 330

Severity: ERROR
Message: Password storage scheme %s does not support use with the authentication password attribute syntax.

ID: 335

Severity: ERROR
Message: The configured minimum password length of %d characters is greater than the configured maximum password length of %d.

ID: 336

Severity: ERROR
Message: The provided password is shorter than the minimum required length of %d characters.

ID: 337

Severity: ERROR
Message: The provided password is longer than the maximum allowed length of %d characters.

ID: 341

Severity: ERROR
Message: Configuration entry "%s" does not contain attribute ds-cfg-password-character-set which specifies the sets of characters that should be used when generating the password. This is a required attribute.

ID: 342

Severity: ERROR
Message: Configuration entry "%s" contains multiple definitions for the %s character set.

ID: 343

Severity: ERROR
Message: An error occurred while attempting to decode the value(s) of the configuration attribute ds-cfg-password-character-set, which is used to hold the character set(s) for use in generating the password: %s.

ID: 346

Severity: ERROR
Message: The password format string "%s" references an undefined character set "%s".

ID: 347

Severity: ERROR
Message: The password format string "%s" contains an invalid syntax. This value should be a comma-delimited sequence of elements, where each element is the name of a character set followed by a colon and the number of characters to choose at random from that character set.

ID: 348

Severity: ERROR
Message: An error occurred while attempting to decode the value for configuration attribute ds-cfg-password-format, which is used to specify the format for the generated passwords: %s.

ID: 354

Severity: ERROR
Message: An error occurred while attempting to get the password policy for user %s: %s.

ID: 355

Severity: ERROR
Message: The current password must be provided for self password changes.

ID: 356

Severity: ERROR
Message: Password modify operations that supply the user's current password must be performed over a secure communication channel.

ID: 357

Severity: ERROR
Message: End users are not allowed to change their passwords.

ID: 358

Severity: ERROR
Message: Password changes must be performed over a secure communication channel.

ID: 359

Severity: ERROR
Message: The password cannot be changed because the previous password change was too recent.

ID: 360

Severity: ERROR
Message: The password cannot be changed because it is expired.

ID: 361

Severity: ERROR
Message: No new password was provided, and no password generator has been defined that may be used to automatically create a new password.

ID: 362

Severity: ERROR
Message: An error occurred while attempting to create a new password using the password generator: %s.

ID: 363

Severity: ERROR
Message: The password policy does not allow users to supply pre-encoded passwords.

ID: 364

Severity: ERROR
Message: The provided new password failed the validation checks defined in the server: %s.

ID: 365

Severity: ERROR
Message: Unable to encode the provided password using the default scheme(s): %s.

ID: 369

Severity: ERROR
Message: An error occurred while attempting to determine the identity mapper to use in conjunction with the password modify extended operation defined in configuration entry %s: %s. The password modify extended operation will not be enabled for use in the server.

ID: 370

Severity: ERROR
Message: The provided authorization ID string "%s" could not be mapped to any user in the directory.

ID: 371

Severity: ERROR
Message: An error occurred while attempting to map authorization ID string "%s" to a user entry: %s.

ID: 378

Severity: ERROR
Message: An error occurred while attempting to verify the password for user %s during SASL PLAIN authentication: %s.

ID: 381

Severity: ERROR
Message: The user account has been administratively disabled.

ID: 382

Severity: ERROR
Message: The user account is locked.

ID: 386

Severity: ERROR
Message: Entry %s cannot be parsed as a valid static group because it does not contain the groupOfEntries, groupOfNames or groupOfUniqueNames object classes.

ID: 387

Severity: ERROR
Message: Value %s for attribute %s in entry %s cannot be parsed as a valid DN: %s. It will be excluded from the set of group members.

ID: 392

Severity: ERROR
Message: You do not have sufficient privileges to perform password reset operations.

ID: 393

Severity: ERROR
Message: The provided authorization ID was empty, which is not allowed for DIGEST-MD5 authentication.

ID: 400

Severity: ERROR
Message: The provided authorization ID %s contained an invalid DN: %s.

ID: 401

Severity: ERROR
Message: The authenticating user %s does not have sufficient privileges to specify an alternate authorization ID.

ID: 402

Severity: ERROR
Message: The entry corresponding to authorization DN %s does not exist in the Directory Server.

ID: 403

Severity: ERROR
Message: An error occurred while attempting to retrieve entry %s specified as the authorization ID: %s.

ID: 404

Severity: ERROR
Message: No entry corresponding to authorization ID %s was found in the server.

ID: 405

Severity: ERROR
Message: An error occurred while attempting to map authorization ID %s to a user entry: %s.

ID: 417

Severity: ERROR
Message: Could not map the provided certificate chain to a user entry because no peer certificate was available.

ID: 418

Severity: ERROR
Message: Could not map the provided certificate chain to a user because the peer certificate was not an X.509 certificate (peer certificate format was %s).

ID: 419

Severity: ERROR
Message: An unexpected error occurred while attempting to read the instance key '%s' in "cn=admin data": %s.

ID: 422

Severity: ERROR
Message: Configuration entry %s has value '%s' which violates the format required for attribute mappings. The expected format is 'certattr:userattr'.

ID: 423

Severity: ERROR
Message: Configuration entry %s contains multiple mappings for certificate attribute %s.

ID: 424

Severity: ERROR
Message: Mapping %s in configuration entry %s references attribute %s which is not defined in the server schema.

ID: 425

Severity: ERROR
Message: Configuration entry %s contains multiple mappings for user attribute %s.

ID: 429

Severity: ERROR
Message: Could not map the provided certificate chain to a user entry because no peer certificate was available.

ID: 430

Severity: ERROR
Message: Could not map the provided certificate chain to a user because the peer certificate was not an X.509 certificate (peer certificate format was %s).

ID: 431

Severity: ERROR
Message: Unable to decode peer certificate subject %s as a DN: %s.

ID: 432

Severity: ERROR
Message: Peer certificate subject %s does not contain any attributes for which a mapping has been established.

ID: 433

Severity: ERROR
Message: The certificate with subject %s could not be mapped to exactly one user. It maps at least to both '%s' and '%s'.

ID: 443

Severity: ERROR
Message: Could not map the provided certificate chain to a user entry because no peer certificate was available.

ID: 444

Severity: ERROR
Message: Could not map the provided certificate chain to a user because the peer certificate was not an X.509 certificate (peer certificate format was %s).

ID: 445

Severity: ERROR
Message: An error occurred while attempting to calculate the fingerprint for the peer certificate with subject %s: %s.

ID: 446

Severity: ERROR
Message: The certificate with fingerprint '%s' could not be mapped to exactly one user. It maps at least to both '%s' and '%s'.

ID: 447

Severity: ERROR
Message: Unable to decode value "%s" in entry "%s" as an LDAP URL: %s.

ID: 449

Severity: ERROR
Message: Dynamic groups do not support explicitly altering their membership.

ID: 451

Severity: ERROR
Message: An error occurred while attempting perform an internal search with base DN %s and filter %s to resolve the member list for dynamic group %s: result code %s, error message %s.

ID: 456

Severity: ERROR
Message: The provided password differs less than the minimum required difference of %d characters.

ID: 457

Severity: ERROR
Message: The provided password contained too many instances of the same character appearing consecutively. The maximum number of times the same character may appear consecutively in a password is %d.

ID: 458

Severity: ERROR
Message: The provided password does not contain enough unique characters. The minimum number of unique characters that may appear in a user password is %d.

ID: 459

Severity: ERROR
Message: The %s attribute is not searchable and should not be included in otherwise unindexed search filters.

ID: 460

Severity: ERROR
Message: The provided password contained a word from the server's dictionary.

ID: 461

Severity: ERROR
Message: The specified dictionary file %s does not exist.

ID: 462

Severity: ERROR
Message: An error occurred while attempting to load the dictionary from file %s: %s.

ID: 463

Severity: ERROR
Message: The provided password was found in another attribute in the user entry.

ID: 464

Severity: ERROR
Message: The provided password contained character '%s' which is not allowed for use in passwords.

ID: 465

Severity: ERROR
Message: The provided password did not contain enough characters from the character set '%s'. The minimum number of characters from that set that must be present in user passwords is %d.

ID: 466

Severity: ERROR
Message: The provided character set definition '%s' is invalid because it does not contain a colon to separate the minimum count from the character set.

ID: 467

Severity: ERROR
Message: The provided character set definition '%s' is invalid because the provided character set is empty.

ID: 468

Severity: ERROR
Message: The provided character set definition '%s' is invalid because the value before the colon must be an integer greater or equal to zero.

ID: 469

Severity: ERROR
Message: The provided character set definition '%s' is invalid because it contains character '%s' which has already been used.

ID: 470

Severity: ERROR
Message: The virtual static group defined in entry %s contains multiple target group DNs, but only one is allowed.

ID: 471

Severity: ERROR
Message: Unable to decode "%s" as the target DN for group %s: %s.

ID: 472

Severity: ERROR
Message: The virtual static group defined in entry %s does not contain a target group definition.

ID: 474

Severity: ERROR
Message: Target group %s referenced by virtual static group %s does not exist.

ID: 475

Severity: ERROR
Message: Altering membership for virtual static group %s is not allowed.

ID: 476

Severity: ERROR
Message: Virtual static group %s references target group %s which is itself a virtual static group. One virtual static group is not allowed to reference another as its target group.

ID: 502

Severity: ERROR
Message: You do not have sufficient privileges to use the password policy state extended operation.

ID: 503

Severity: ERROR
Message: The provided password policy state extended request did not include a request value.

ID: 504

Severity: ERROR
Message: An unexpected error occurred while attempting to decode password policy state extended request value: %s.

ID: 506

Severity: ERROR
Message: An unexpected error occurred while attempting to decode an operation from the password policy state extended request: %s.

ID: 507

Severity: ERROR
Message: No value was provided for the password policy state operation intended to set the disabled state for the user. Exactly one value (either 'true' or 'false') must be given.

ID: 508

Severity: ERROR
Message: Multiple values were provided for the password policy state operation intended to set the disabled state for the user. Exactly one value (either 'true' or 'false') must be given.

ID: 509

Severity: ERROR
Message: The value provided for the password policy state operation intended to set the disabled state for the user was invalid. The value must be either 'true' or 'false'.

ID: 510

Severity: ERROR
Message: Multiple values were provided for the password policy state operation intended to set the account expiration time for the user. Exactly one value must be given.

ID: 511

Severity: ERROR
Message: The value %s provided for the password policy state operation used to set the account expiration time was invalid: %s. The value should be specified using the generalized time format.

ID: 512

Severity: ERROR
Message: Multiple values were provided for the password policy state operation intended to set the password changed time for the user. Exactly one value must be given.

ID: 513

Severity: ERROR
Message: The value %s provided for the password policy state operation used to set the password changed time was invalid: %s. The value should be specified using the generalized time format.

ID: 514

Severity: ERROR
Message: Multiple values were provided for the password policy state operation intended to set the password warned time for the user. Exactly one value must be given.

ID: 515

Severity: ERROR
Message: The value %s provided for the password policy state operation used to set the password warned time was invalid: %s. The value should be specified using the generalized time format.

ID: 516

Severity: ERROR
Message: Multiple values were provided for the password policy state operation intended to add an authentication failure time for the user. Exactly one value must be given.

ID: 517

Severity: ERROR
Message: The value %s provided for the password policy state operation used to update the authentication failure times was invalid: %s. The value should be specified using the generalized time format.

ID: 518

Severity: ERROR
Message: Multiple values were provided for the password policy state operation intended to set the last login time for the user. Exactly one value must be given.

ID: 519

Severity: ERROR
Message: The value %s provided for the password policy state operation used to set the last login time was invalid: %s. The value should be specified using the generalized time format.

ID: 520

Severity: ERROR
Message: No value was provided for the password policy state operation intended to set the reset state for the user. Exactly one value (either 'true' or 'false') must be given.

ID: 521

Severity: ERROR
Message: Multiple values were provided for the password policy state operation intended to set the reset state for the user. Exactly one value (either 'true' or 'false') must be given.

ID: 522

Severity: ERROR
Message: The value provided for the password policy state operation intended to set the reset state for the user was invalid. The value must be either 'true' or 'false'.

ID: 523

Severity: ERROR
Message: Multiple values were provided for the password policy state operation intended to add a grace login use time for the user. Exactly one value must be given.

ID: 524

Severity: ERROR
Message: The value %s provided for the password policy state operation used to update the grace login use times was invalid: %s. The value should be specified using the generalized time format.

ID: 525

Severity: ERROR
Message: Multiple values were provided for the password policy state operation intended to set the required change time for the user. Exactly one value must be given.

ID: 526

Severity: ERROR
Message: The value %s provided for the password policy state operation used to set the required change time was invalid: %s. The value should be specified using the generalized time format.

ID: 527

Severity: ERROR
Message: The password policy state extended request included an operation with an invalid or unsupported operation type of %s.

ID: 530

Severity: ERROR
Message: The provided new password was already contained in the password history.

ID: 531

Severity: ERROR
Message: The Directory Server is not configured with any SMTP servers. The SMTP alert handler cannot be used unless the Directory Server is configured with information about at least one SMTP server.

ID: 533

Severity: ERROR
Message: The provided match pattern "%s" could not be parsed as a regular expression: %s.

ID: 535

Severity: ERROR
Message: The processed ID string %s could not be mapped to exactly one user. It maps at least to both '%s' and '%s'.

ID: 536

Severity: ERROR
Message: The internal search based on processed ID string %s could not be processed efficiently: %s. Check the server configuration to ensure that all associated backends are properly configured for these types of searches.

ID: 537

Severity: ERROR
Message: An internal failure occurred while attempting to resolve processed ID string %s to a user entry: %s.

ID: 538

Severity: ERROR
Message: The processed ID string %s mapped to multiple users.

ID: 540

Severity: ERROR
Message: Group instance with DN %s has been deleted and is no longer valid.

ID: 543

Severity: ERROR
Message: The SMTP account status notification handler defined in configuration entry %s cannot be enabled unless the Directory Server is with information about one or more SMTP servers.

ID: 544

Severity: ERROR
Message: SMTP account status notification handler configuration entry '%s' does not include any email address attribute types or recipient addresses. At least one of these must be provided.

ID: 545

Severity: ERROR
Message: Unable to parse message subject value '%s' from configuration entry '%s' because the value does not contain a colon to separate the notification type from the subject.

ID: 546

Severity: ERROR
Message: Unable to parse message subject value '%s' from configuration entry '%s' because '%s' is not a valid account status notification type.

ID: 547

Severity: ERROR
Message: The message subject definitions contained in configuration entry '%s' have multiple subjects defined for notification type %s.

ID: 548

Severity: ERROR
Message: Unable to parse message template file path value '%s' from configuration entry '%s' because the value does not contain a colon to separate the notification type from the template file path.

ID: 549

Severity: ERROR
Message: Unable to parse message template file path value '%s' from configuration entry '%s' because '%s' is not a valid account status notification type.

ID: 550

Severity: ERROR
Message: The message template file path definitions contained in configuration entry '%s' have multiple template file paths defined for notification type %s.

ID: 551

Severity: ERROR
Message: The message template file '%s' referenced in configuration entry '%s' does not exist.

ID: 552

Severity: ERROR
Message: An unclosed token was found starting at column %d of line %d.

ID: 553

Severity: ERROR
Message: The notification-user-attr token starting at column %d of line %d references undefined attribute type %s.

ID: 554

Severity: ERROR
Message: The notification-property token starting at column %d of line %d references undefined notification property %s.

ID: 555

Severity: ERROR
Message: An unrecognized token %s was found at column %d of line %d.

ID: 556

Severity: ERROR
Message: An error occurred while attempting to parse message template file '%s' referenced in configuration entry '%s': %s.

ID: 558

Severity: ERROR
Message: An error occurred while attempting to send an account status notification message for notification type %s for user entry %s: %s.

ID: 559

Severity: ERROR
Message: An error occurred while trying to encrypt a value using password storage scheme %s: %s.

ID: 560

Severity: ERROR
Message: An error occurred while trying to decrypt a value using password storage scheme %s: %s.

ID: 561

Severity: ERROR
Message: Cannot decode the provided symmetric key extended operation because it does not have a value.

ID: 563

Severity: ERROR
Message: Cannot decode the provided symmetric key extended request: %s.

ID: 564

Severity: ERROR
Message: An unexpected error occurred while attempting to decode the symmetric key extended request sequence: %s.

ID: 572

Severity: ERROR
Message: Failed to create a SASL server for SASL mechanism %s.

ID: 573

Severity: ERROR
Message: GSSAPI SASL mechanism handler initalization failed because the keytab file %s does not exist.

ID: 578

Severity: ERROR
Message: The password value %s has been base64-decoded but is too short to be valid.

ID: 579

Severity: ERROR
Message: The provided minimum required number of character sets '%d' is invalid because it must at least include all mandatory character sets.

ID: 580

Severity: ERROR
Message: The provided minimum required number of character sets '%d' is invalid because it is greater than the total number of defined character sets.

ID: 581

Severity: ERROR
Message: The provided password did not contain characters from at least %d of the following character sets or ranges: %s.

ID: 583

Severity: ERROR
Message: SASL %s authentication is not supported for user %s because the account is not managed locally.

ID: 584

Severity: ERROR
Message: Password modification is not supported for user %s because the account is not managed locally.

ID: 585

Severity: ERROR
Message: The password policy state extended operation is not supported for user %s because the account is not managed locally.

ID: 586

Severity: ERROR
Message: The user "%s" could not be authenticated using LDAP PTA policy "%s" because the following mapping attributes were not found in the user's entry: %s.

ID: 587

Severity: ERROR
Message: The user "%s" could not be authenticated using LDAP PTA policy "%s" because the search of base DN "%s" returned more than one entry matching the filter "%s".

ID: 588

Severity: ERROR
Message: The user "%s" could not be authenticated using LDAP PTA policy "%s" because the search did not return any entries matching the filter "%s".

ID: 589

Severity: ERROR
Message: The user "%s" could not be authenticated using LDAP PTA policy "%s" because the search failed unexpectedly for the following reason: %s.

ID: 590

Severity: ERROR
Message: The user "%s" could not be authenticated using LDAP PTA policy "%s" because the bind failed unexpectedly for the following reason: %s.

ID: 613

Severity: ERROR
Message: The configuration of LDAP PTA policy "%s" is invalid because it does not specify the a means for obtaining the mapped search bind password.

ID: 616

Severity: ERROR
Message: The certificate with subject %s mapped to multiple users.

ID: 617

Severity: ERROR
Message: The internal search based on the certificate with subject %s could not be processed efficiently: %s. Check the server configuration to ensure that all associated backends are properly configured for these types of searches.

ID: 618

Severity: ERROR
Message: An internal failure occurred while attempting to map the certificate with subject %s to a user entry: %s.

ID: 619

Severity: ERROR
Message: The certificate with subject %s mapped to multiple users.

ID: 620

Severity: ERROR
Message: The internal search based on the certificate with subject %s could not be processed efficiently: %s. Check the server configuration to ensure that all associated backends are properly configured for these types of searches.

ID: 621

Severity: ERROR
Message: An internal failure occurred while attempting to map the certificate with subject %s to a user entry: %s.

ID: 622

Severity: ERROR
Message: The certificate with fingerprint %s mapped to multiple users.

ID: 623

Severity: ERROR
Message: The internal search based on the certificate with fingerprint %s could not be processed efficiently: %s. Check the server configuration to ensure that all associated backends are properly configured for these types of searches.

ID: 624

Severity: ERROR
Message: An internal failure occurred while attempting to map the certificate with fingerprint %s to a user entry: %s.

ID: 629

Severity: ERROR
Message: The provided password did not contain enough characters from the character range '%s'. The minimum number of characters from that range that must be present in user passwords is %d.

ID: 630

Severity: ERROR
Message: The provided character range definition '%s' is invalid because it does not contain a colon to separate the minimum count from the character range.

ID: 631

Severity: ERROR
Message: The provided character range definition '%s' is invalid because it does not contain a colon to separate the minimum count from the character range.

ID: 632

Severity: ERROR
Message: The provided character range definition '%s' is invalid because the value before the colon must be an integer greater or equal to zero.

ID: 633

Severity: ERROR
Message: The provided character range definition '%s' is invalid because the range '%s' is reversed.

ID: 634

Severity: ERROR
Message: The provided character range definition '%s' is invalid because the range '%s' is missing the minus.

ID: 635

Severity: ERROR
Message: The provided character range definition '%s' is invalid because the range '%s' is too short.

ID: 636

Severity: ERROR
Message: There is no private key entry in keystore '%s' used by file based key manager provider '%s'. TLS connections which rely on this key manager provider may fail. Ensure that keystore file contains at least one private key.

ID: 638

Severity: ERROR
Message: An error occurred while attempting to match a bcrypt hashed password value: %s.

ID: 639

Severity: ERROR
Message: The mapped search filter template "%s" could not be parsed as a valid LDAP filter.

ID: 640

Severity: ERROR
Message: An error occurred while trying to create a key manager factory to access the contents of LDAP keystore with base DN '%s': %s.

ID: 641

Severity: ERROR
Message: An error occurred while trying to create a trust manager factory to access the certificates in "cn=admin data": %s.

ID: 645

Severity: ERROR
Message: '%s' cannot list the secret files in directory '%s', all the secrets will be ignored: %s.

ID: 647

Severity: ERROR
Message: The file '%s' exceeds max size '%s'.

ID: 648

Severity: ERROR
Message: '%s' cannot decode the secret file '%s': %s.

ID: 651

Severity: ERROR
Message: Invalid excluded file name pattern: %s.

ID: 657

Severity: ERROR
Message: An error occurred while trying to create a trust manager factory to access the contents of the PKCS#11 keystore: %s.

ID: 658

Severity: ERROR
Message: An error occurred while trying to access the PKCS#11 trust manager: %s.

ID: 659

Severity: ERROR
Message: Unable to load JVM default keystore from system properties: %s.

ID: 660

Severity: ERROR
Message: An error occurred while reading information contained within key manager provider from configuration: "%s".

ID: 661

Severity: ERROR
Message: Unable to get the JVM default truststore: %s.

ID: 662

Severity: ERROR
Message: Could not map the provided certificate chain to a user becausethe peer certificate issuer "%s" could not be decoded as an LDAP DN: %s.

ID: 663

Severity: ERROR
Message: Could not map the provided certificate chain to a user because the matching user entry with DN '%s' does not contain an issuer DN matching the certificate issuer DN '%s'.

ID: 664

Severity: ERROR
Message: The baseDN '%s' specified as match base DN in the exact match identity mapper defined in configuration entry '%s', does not belong to a local backend.

ID: 665

Severity: ERROR
Message: The baseDN '%s' specified as match base DN in the regular expression identity mapper defined in configuration entry '%s', does not belong to a local backend.

ID: 666

Severity: ERROR
Message: The processed ID string %s is mapped to multiple users.

ID: 667

Severity: ERROR
Message: File based key manager provider '%s' failed to load content from file '%s'. TLS connections which rely on this key manager provider may fail. Ensure that keystore file contains at least one private key compatible with the security providers. Security providers available in the running JVM are '%s'. The security provider used for loading key manager will be the first in the list which is compatible with the algorithm '%s'.

ID: 669

Severity: ERROR
Message: File based key manager provider '%s' cannot load an X509 extended key manager from keystore file '%s'. TLS connections which rely on this key manager provider may fail. Security providers available in the running JVM are %s. The security provider used for loading key manager will be the first in the list which is compatible with the algorithm '%s'.

ID: 670

Severity: ERROR
Message: File based key manager provider '%s' cannot load content from keystore file '%s'. TLS connections which rely on this key manager provider may fail. Restarting the server or the impacted connection handler may resolve this problem. Error detail: %s.

ID: 673

Severity: ERROR
Message: File based trust manager provider '%s' cannot load content from truststore file '%s'. TLS connections which rely on this trust manager provider may fail. Restarting the server or the impacted connection handler may resolve this problem. Error detail: %s.

ID: 674

Severity: ERROR
Message: File based trust manager provider '%s' failed to load content from file '%s'. TLS connections which rely on this trust manager provider may fail. Ensure that truststore file contains at least one private key compatible with the security providers. Security providers available in the running JVM are '%s'. The security provider used for loading trust manager will be the first in the list which is compatible with the algorithm '%s'.

ID: 676

Severity: ERROR
Message: File based trust manager provider '%s' cannot load an X509 extended trust manager from truststore file '%s'. TLS connections which rely on this trust manager provider may fail. Security providers available in the running JVM are %s. The security provider used for loading trust manager will be the first in the list which is compatible with the algorithm '%s'.

ID: 677

Severity: ERROR
Message: The %s SCRAM password storage scheme could not be initialized because the algorithm is not supported by the JVM.

ID: 678

Severity: ERROR
Message: An error occurred while attempting to decode the SCRAM credential value %s: %s.

ID: 679

Severity: ERROR
Message: SASL %s authentication is not possible for user '%s' because the user entry does not contain any SCRAM credentials.

ID: 680

Severity: ERROR
Message: An error occurred while attempting to retrieve the SCRAM credentials for user '%s' in order to perform SASL %s authentication: %s.

ID: 681

Severity: ERROR
Message: The %s SCRAM SASL mechanism handler could not be initialized because the algorithm is not supported by the JVM.

ID: 682

Severity: ERROR
Message: Error loading dictionary: %s.

Category: Server plugins

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 3

Severity: ERROR
Message: The LDAP attribute description list plugin instance defined in configuration entry %s does not list any plugin types. This plugin must be configured to operate as a pre-parse search plugin.

ID: 4

Severity: ERROR
Message: The LDAP attribute description list plugin instance defined in configuration entry %s lists an invalid plugin type %s. This plugin can only be used as a pre-parse search plugin.

ID: 30

Severity: ERROR
Message: The startup plugin defined in configuration entry %s threw an exception when it was invoked during the Directory Server startup process: %s. The server startup process has been aborted.

ID: 31

Severity: ERROR
Message: The startup plugin defined in configuration entry %s returned a null value when it was invoked during the Directory Server startup process. This is an illegal return value, and the server startup process has been aborted.

ID: 33

Severity: ERROR
Message: The startup plugin defined in configuration entry %s encountered an error when it was invoked during the Directory Server startup process: %s (error ID %d). The server startup process has been aborted.

ID: 34

Severity: ERROR
Message: The shutdown plugin defined in configuration entry %s threw an exception when it was invoked during the Directory Server shutdown process: %s.

ID: 35

Severity: ERROR
Message: The post-connect plugin defined in configuration entry %s threw an exception when it was invoked for connection %d from %s: %s. The connection will be terminated.

ID: 36

Severity: ERROR
Message: The post-connect plugin defined in configuration entry %s returned null when invoked for connection %d from %s. This is an illegal response, and the connection will be terminated.

ID: 37

Severity: ERROR
Message: The post-disconnect plugin defined in configuration entry %s threw an exception when it was invoked for connection %d from %s: %s.

ID: 38

Severity: ERROR
Message: The post-disconnect plugin defined in configuration entry %s returned null when invoked for connection %d from %s. This is an illegal response.

ID: 39

Severity: ERROR
Message: The pre-parse %s plugin defined in configuration entry %s threw an exception when it was invoked for connection %d operation %d: %s. Processing on this operation will be terminated.

ID: 40

Severity: ERROR
Message: The pre-parse %s plugin defined in configuration entry %s returned null when invoked for connection %d operation %d. This is an illegal response, and processing on this operation will be terminated.

ID: 41

Severity: ERROR
Message: The pre-operation %s plugin defined in configuration entry %s threw an exception when it was invoked for connection %d operation %d: %s. Processing on this operation will be terminated.

ID: 42

Severity: ERROR
Message: The pre-operation %s plugin defined in configuration entry %s returned null when invoked for connection %d operation %d. This is an illegal response, and processing on this operation will be terminated.

ID: 43

Severity: ERROR
Message: The post-operation %s plugin defined in configuration entry %s threw an exception when it was invoked for connection %d operation %d: %s. Processing on this operation will be terminated.

ID: 44

Severity: ERROR
Message: The post-operation %s plugin defined in configuration entry %s returned null when invoked for connection %d operation %d. This is an illegal response, and processing on this operation will be terminated.

ID: 45

Severity: ERROR
Message: The post-response %s plugin defined in configuration entry %s threw an exception when it was invoked for connection %d operation %d: %s. Processing on this operation will be terminated.

ID: 46

Severity: ERROR
Message: The post-response %s plugin defined in configuration entry %s returned null when invoked for connection %d operation %d. This is an illegal response, and processing on this operation will be terminated.

ID: 47

Severity: ERROR
Message: The search result entry plugin defined in configuration entry %s threw an exception when it was invoked for connection %d operation %d with entry %s: %s. Processing on this search operation will be terminated.

ID: 48

Severity: ERROR
Message: The search result entry plugin defined in configuration entry %s returned null when invoked for connection %d operation %d with entry %s. This is an illegal response, and processing on this search operation will be terminated.

ID: 49

Severity: ERROR
Message: The search result reference plugin defined in configuration entry %s threw an exception when it was invoked for connection %d operation %d with referral URL(s) %s: %s. Processing on this search operation will be terminated.

ID: 50

Severity: ERROR
Message: The search result reference plugin defined in configuration entry %s returned null when invoked for connection %d operation %d with referral URL(s) %s. This is an illegal response, and processing on this search operation will be terminated.

ID: 51

Severity: ERROR
Message: An attempt was made to register the LastMod plugin to be invoked as a %s plugin. This plugin type is not allowed for this plugin.

ID: 58

Severity: ERROR
Message: The LDIF import plugin defined in configuration entry %s threw an exception when it was invoked on entry %s: %s.

ID: 59

Severity: ERROR
Message: The LDIF import plugin defined in configuration entry %s returned null when invoked on entry %s. This is an illegal response.

ID: 62

Severity: ERROR
Message: An attempt was made to register the EntryUUID plugin to be invoked as a %s plugin. This plugin type is not allowed for this plugin.

ID: 65

Severity: ERROR
Message: An attempt was made to register the password policy import plugin to be invoked as a %s plugin. This plugin type is not allowed for this plugin.

ID: 66

Severity: ERROR
Message: An error occurred while attempting to encode a password value stored in attribute %s of user entry %s: %s. Password values for this user will not be encoded.

ID: 67

Severity: ERROR
Message: The plugin defined in configuration entry %s does not support the %s plugin type.

ID: 69

Severity: ERROR
Message: The password policy import plugin is not configured any default auth password schemes, and the server does not support the %s auth password scheme.

ID: 70

Severity: ERROR
Message: Auth password storage scheme %s referenced by the password policy import plugin is not configured for use in the server.

ID: 71

Severity: ERROR
Message: The password policy import plugin is not configured any default user password schemes, and the server does not support the %s auth password scheme.

ID: 72

Severity: ERROR
Message: User password storage scheme %s referenced by the password policy import plugin is not configured for use in the server.

ID: 75

Severity: ERROR
Message: The subordinate modify DN plugin defined in configuration entry %s threw an exception when it was invoked for connection %d operation %d: %s. Processing on this operation will be terminated.

ID: 76

Severity: ERROR
Message: The subordinate modify DN plugin defined in configuration entry %s returned null when invoked for connection %d operation %s. This is an illegal response, and processing on this operation will be terminated.

ID: 77

Severity: ERROR
Message: An attempt was made to register the Unique Attribute plugin to be invoked as a %s plugin. This plugin type is not allowed for this plugin.

ID: 81

Severity: ERROR
Message: An attempt was made to register the Referential Integrity plugin to be invoked as a %s plugin. This plugin type is not allowed for this plugin.

ID: 82

Severity: ERROR
Message: An error occurred during Referential Integity plugin initialization because log file creation failed: %s.

ID: 83

Severity: ERROR
Message: An error occurred closing the Referential Integrity plugin update log file: %s.

ID: 84

Severity: ERROR
Message: An error occurred replacing the Referential Integrity plugin update log file: %s.

ID: 89

Severity: ERROR
Message: The Referential Integrity plugin failed when performaing an internal search: %s.

ID: 90

Severity: ERROR
Message: The Referential Integrity plugin failed when performing an internal modify on entry %s: %s.

ID: 91

Severity: ERROR
Message: The Referential Integrity plugin failed to decode a entry DN from the update log: %s.

ID: 93

Severity: ERROR
Message: An error occurred in the Referential Integrity plugin while attempting to configure the attribute type %s which has a syntax OID of %s. A Referential Integrity attribute type must have a syntax OID of either 1.3.6.1.4.1.1466.115.121.1.12 (for the distinguished name syntax) or 1.3.6.1.4.1.1466.115.121.1.34 (for the name and optional uid syntax) or 1.3.6.1.4.1.36733.2.1.3.12 (for the name and json syntax).

ID: 96

Severity: ERROR
Message: The 7-bit clean plugin is configured with invalid plugin type %s. Only the ldifImport, preOperationAdd, preOperationModify, and preOperationModifyDN plugin types are allowed.

ID: 102

Severity: ERROR
Message: The modify DN operation would have resulted in a value for attribute %s that was not 7-bit clean.

ID: 103

Severity: ERROR
Message: The entry included a value for attribute %s that was not 7-bit clean.

ID: 104

Severity: ERROR
Message: The password policy import plugin references default auth password storage scheme %s which is not available for use in the server.

ID: 105

Severity: ERROR
Message: The post-synchronization %s plugin defined in configuration entry %s threw an exception when it was invoked for connection %d operation %d: %s.

ID: 106

Severity: ERROR
Message: A unique attribute conflict was detected for attribute %s: value %s already exists in entry %s.

ID: 107

Severity: ERROR
Message: A unique attribute conflict was detected for attribute %s during synchronization (connID=%d, opID=%d): value %s in entry %s conflicts with an existing value in entry %s. Manual interaction is required to eliminate the conflict.

ID: 108

Severity: ERROR
Message: An internal error occurred while attempting to determine whether the operation would have resulted in a unique attribute conflict (result %s, message %s).

ID: 109

Severity: ERROR
Message: An internal error occurred while attempting to determine whether the synchronization operation (connID=%d, opID=%d) for entry %s would have resulted in a unique attribute conflict (result %s, message %s).

ID: 110

Severity: ERROR
Message: The referential integrity plugin defined in configuration entry %s is configured to operate on attribute %s but there is no equality index defined for this attribute in backend %s.

ID: 111

Severity: ERROR
Message: The unique attribute plugin defined in configuration entry %s is configured to operate on attribute %s but there is no equality index defined for this attribute in backend %s.

ID: 113

Severity: ERROR
Message: An attempt was made to register the Change Number Control plugin to be invoked as a %s plugin. This plugin type is not allowed for this plugin.

ID: 114

Severity: ERROR
Message: An attempt was made to register the Change Number Control plugin with the following plugin types : %s. However this plugin must be configured with all of the following plugin types : %s.

ID: 115

Severity: ERROR
Message: The subordinate delete plugin defined in configuration entry %s threw an exception when it was invoked for connection %d operation %d: %s. Processing on this operation will be terminated.

ID: 116

Severity: ERROR
Message: The subordinate delete plugin defined in configuration entry %s returned null when invoked for connection %d operation %s. This is an illegal response, and processing on this operation will be terminated.

ID: 117

Severity: ERROR
Message: An attempt was made to register the Samba password synchronization plugin to be invoked as a %s plugin. This plugin type is not allowed for this plugin.

ID: 118

Severity: ERROR
Message: The Samba password synchronization plugin could not encode a password for the following reasons: %s.

ID: 119

Severity: ERROR
Message: The Samba password synchronization plugin could not process a modification for the following reason: %s.

ID: 120

Severity: ERROR
Message: Invalid plugin type '%s' for the Attribute Cleanup plugin.

ID: 121

Severity: ERROR
Message: Attribute '%s' is not defined in the directory schema.

ID: 122

Severity: ERROR
Message: The attribute '%s' has already been defined in the configuration.

ID: 123

Severity: ERROR
Message: The mapping '%s:%s' maps the attribute to itself.

ID: 124

Severity: ERROR
Message: The property 'check-references-filter-criteria' specifies filtering criteria for attribute '%s', but this attribute is not listed in the 'attribute-type' property.

ID: 125

Severity: ERROR
Message: The filtering criteria '%s' specified in property 'check-references-filter-criteria' is invalid because the filter could not be decoded: '%s'.

ID: 126

Severity: ERROR
Message: The entry referenced by the value '%s' of the attribute '%s' in the entry '%s' does not exist in any of the configured naming contexts.

ID: 127

Severity: ERROR
Message: The entry referenced by the value '%s' of the attribute '%s' in the entry '%s' does not match the filter '%s'.

ID: 128

Severity: ERROR
Message: The entry referenced by the value '%s' of the attribute '%s' in the entry '%s' does not belong to any of the configured naming contexts.

ID: 129

Severity: ERROR
Message: The opration could not be processed due to an unexpected exception: '%s'.

ID: 130

Severity: ERROR
Message: An attempt was made to register the Graphite Monitor Reporter Plugin plugin to be invoked as a %s plugin. This plugin type is not allowed for this plugin.

ID: 131

Severity: ERROR
Message: Unable to report metrics to Graphite server '%s' because the Graphite server hostname resolution has failed. Ensure that the plugin configuration is correct and that the Graphite server is reachable. The Graphite plugin will be disabled until a change is performed in its configuration or the server restart.

ID: 132

Severity: ERROR
Message: The referential integrity plugin defined in configuration entry %s is configured to operate on attribute %s but there is no %s extensible matching rule index defined for this attribute in backend %s.

ID: 133

Severity: ERROR
Message: An attempt was made to register the Entity Tag plugin to be invoked as a %s plugin. This plugin type is not allowed for this plugin.

Category: Server tools

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 3

Severity: ERROR
Message: The value %s cannot be decoded as a hexadecimal string because it does not have a length that is a multiple of two bytes.

ID: 4

Severity: ERROR
Message: The value %s cannot be decoded as a hexadecimal string because it contains an illegal character %c that is not a valid hexadecimal digit.

ID: 5

Severity: ERROR
Message: Unable to parse line %d ("%s") from the LDIF source because the line started with a space but there were no previous lines in the entry to which this line could be appended.

ID: 6

Severity: ERROR
Message: Unable to parse LDIF entry starting at line %d because the line "%s" does not include an attribute name.

ID: 7

Severity: ERROR
Message: Unable to parse LDIF entry starting at line %d because the first line does not contain a DN (the first line was "%s".

ID: 9

Severity: ERROR
Message: Unable to parse LDIF entry starting at line %d because an error occurred while trying to parse the value of line "%s" as a distinguished name: %s.

ID: 11

Severity: ERROR
Message: Unable to parse LDIF entry starting at line %d because it was not possible to base64-decode the DN on line "%s": %s.

ID: 12

Severity: ERROR
Message: Unable to parse LDIF entry %s starting at line %d because it was not possible to base64-decode the attribute on line "%s": %s.

ID: 15

Severity: ERROR
Message: Entry %s starting at line %d includes multiple values for single-valued attribute %s.

ID: 17

Severity: ERROR
Message: Entry %s read from LDIF starting at line %d is not valid because it violates the server's schema configuration: %s.

ID: 18

Severity: ERROR
Message: The specified LDIF file %s already exists and the export configuration indicates that no attempt should be made to append to or replace the file.

ID: 19

Severity: ERROR
Message: Unable to parse LDIF entry %s starting at line %d because the value of attribute %s was to be read from a URL but the URL was invalid: %s.

ID: 20

Severity: ERROR
Message: Unable to parse LDIF entry %s starting at line %d because the value of attribute %s was to be read from URL %s but an error occurred while trying to read that content: %s.

ID: 21

Severity: ERROR
Message: The specified reject file %s already exists and the import configuration indicates that no attempt should be made to append to or replace the file.

ID: 22

Severity: ERROR
Message: An error occurred while attempting to determine whether LDIF entry "%s" starting at line %d should be imported as a result of the include and exclude filter configuration: %s.

ID: 76

Severity: ERROR
Message: The provided sender address %s is invalid: %s.

ID: 77

Severity: ERROR
Message: The provided recipient address %s is invalid: %s.

ID: 78

Severity: ERROR
Message: The specified e-mail message could not be sent using any of the configured mail servers.

ID: 110

Severity: ERROR
Message: The provided string "%s" cannot be decoded as an LDAP URL because it does not contain the necessary :// component to separate the scheme from the rest of the URL.

ID: 111

Severity: ERROR
Message: The provided string "%s" cannot be decoded as an LDAP URL because it does not contain a protocol scheme.

ID: 112

Severity: ERROR
Message: The provided string "%s" cannot be decoded as an LDAP URL because it does not contain a host before the colon to specify the port number.

ID: 113

Severity: ERROR
Message: The provided string "%s" cannot be decoded as an LDAP URL because it does not contain a port number after the colon following the host.

ID: 114

Severity: ERROR
Message: The provided string "%s" cannot be decoded as an LDAP URL because the port number portion %s cannot be decoded as an integer.

ID: 115

Severity: ERROR
Message: The provided string "%s" cannot be decoded as an LDAP URL because the provided port number %d is not within the valid range between 1 and 65535.

ID: 116

Severity: ERROR
Message: The provided string "%s" cannot be decoded as an LDAP URL because the scope string %s was not one of the allowed values of base, one, sub, or subordinate.

ID: 117

Severity: ERROR
Message: The provided URL component "%s" could not be decoded because the percent character at byte %d was not followed by two hexadecimal digits.

ID: 118

Severity: ERROR
Message: The provided URL component "%s" could not be decoded because the character at byte %d was not a valid hexadecimal digit.

ID: 120

Severity: ERROR
Message: Cannot decode value "%s" as a named character set because it does not contain a colon to separate the name from the set of characters.

ID: 121

Severity: ERROR
Message: The named character set is invalid because it does not contain a name.

ID: 122

Severity: ERROR
Message: The named character set is invalid because the provide name "%s" has an invalid character at position %d. Only ASCII alphabetic characters are allowed in the name.

ID: 123

Severity: ERROR
Message: Cannot decode value "%s" as a named character set because it does not contain a name to use for the character set.

ID: 124

Severity: ERROR
Message: Cannot decode value "%s" as a named character set because there are no characters to include in the set.

ID: 141

Severity: ERROR
Message: Unable to set permissions for file %s because it does not exist.

ID: 143

Severity: ERROR
Message: One or more exceptions were thrown in the process of updating the file permissions for %s. Some of the permissions for the file may have been altered.

ID: 146

Severity: ERROR
Message: The provided string %s does not represent a valid UNIX file mode. UNIX file modes must be a three-character string in which each character is a numeric digit between zero and seven.

ID: 164

Severity: ERROR
Message: The specified skip file %s already exists and the import configuration indicates that no attempt should be made to append to or replace the file.

ID: 165

Severity: ERROR
Message: Skipping entry %s because the DN is not one that should be included based on the include and exclude branches/filters.

ID: 167

Severity: ERROR
Message: The embedded server with server root '%s' cannot be started because it is already running.

ID: 201

Severity: ERROR
Message: Skipping entry %s because the DN is excluded by the exclude branch "%s".

ID: 202

Severity: ERROR
Message: Skipping entry %s because the DN is excluded by the exclude filter "%s".

ID: 203

Severity: ERROR
Message: Skipping entry %s because the DN is not included by any include branches.

ID: 204

Severity: ERROR
Message: Skipping entry %s because the DN is not included by any include filters.

ID: 224

Severity: ERROR
Message: Rejecting entry %s because it was rejected by a plugin.

ID: 225

Severity: ERROR
Message: Rejecting entry %s because it was rejected by a plugin: %s.

ID: 271

Severity: ERROR
Message: Unable to parse LDIF entry %s starting at line %d because it has an invalid binary option for attribute %s.

ID: 301

Severity: ERROR
Message: Skipping entry %s because the following error was received when reading its attributes: %s.

ID: 310

Severity: ERROR
Message: An error occurred while attempting to obtain a list of the files in directory %s to include in the backup: %s.

ID: 333

Severity: ERROR
Message: An error occurred while attempting to extract server archive '%s' before setup of embedded server with server root '%s': %s.

ID: 334

Severity: ERROR
Message: An error occurred while attempting to rebuild index of embedded server with server root '%s': %s.

ID: 336

Severity: ERROR
Message: An error occurred while attempting to start the embedded server with server root '%s' : %s.

ID: 337

Severity: ERROR
Message: An error occurred while attempting to upgrade the embedded server with server root '%s' : %s.

ID: 338

Severity: ERROR
Message: An error occurred while attempting to retrieve an internal connection to the server with the user DN '%s'.

ID: 339

Severity: ERROR
Message: The setup from an archive can only be done with a server root directory named after the root directory contained in the archive: '%s'. The provided server root was: '%s'.

ID: 342

Severity: ERROR
Message: An error occurred while attempting to initialize the configuration framework or to read the configuration file '%s'.

ID: 345

Severity: ERROR
Message: An error occurred while attempting to import LDIF file '%s' into embedded server with server root '%s'. Import LDIF task state was '%s'. You can look at the task logs printed on the embedded server output stream for more details.

ID: 346

Severity: ERROR
Message: An error occurred while attempting to import LDIF file '%s' into embedded server with server root '%s': '%s'.

ID: 347

Severity: ERROR
Message: An error occurred while attempting to rebuild index of embedded server with server root '%s'. Rebuild task state was '%s'. You can look at the task logs printed on the embedded server output stream for more details.

ID: 348

Severity: ERROR
Message: An error occurred while attempting to retrieve the configuration version of the directory server: '%s'.

ID: 349

Severity: ERROR
Message: An error occurred while attempting to retrieve the data version of the directory server: '%s'.

ID: 350

Severity: ERROR
Message: An error occurred while initializing configuration of embedded server with server root '%s': %s.

ID: 351

Severity: ERROR
Message: The directory to move %s does not exist.

ID: 352

Severity: ERROR
Message: The directory to move %s exists but is a file.

ID: 353

Severity: ERROR
Message: The target directory %s already exists.

ID: 355

Severity: ERROR
Message: Configuration error: an LDAP port or an LDAPS port must be configured before finishing configuring the embedded server.

Category: setup command

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: N/A

Severity: ERROR
Message: An error occurred while attempting to write the monitor user entry: %s.

Category: Tasks

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 5

Severity: ERROR
Message: Unable to add one or more files to the server schema because no schema file names were provided in attribute %s of task entry %s.

ID: 6

Severity: ERROR
Message: Unable to add one or more files to the server schema because the specified schema file %s does not exist in schema directory %s.

ID: 7

Severity: ERROR
Message: Unable to add one or more files to the server schema because an error occurred while attempting to determine whether file %s exists in schema directory %s: %s.

ID: 8

Severity: ERROR
Message: An error occurred while attempting to load the contents of schema file %s into the server schema: %s.

ID: 9

Severity: ERROR
Message: Unable to add one or more files to the server schema because the server was unable to obtain a write lock on the schema entry %s after multiple attempts.

ID: 10

Severity: ERROR
Message: You do not have sufficient privileges to modify the server schema.

ID: 11

Severity: ERROR
Message: You do not have sufficient privileges to initiate a Directory Server backup or backup purge.

ID: 12

Severity: ERROR
Message: You do not have sufficient privileges to initiate a Directory Server restore.

ID: 13

Severity: ERROR
Message: You do not have sufficient privileges to initiate an LDIF import.

ID: 14

Severity: ERROR
Message: You do not have sufficient privileges to initiate an LDIF export.

ID: 15

Severity: ERROR
Message: You do not have sufficient privileges to initiate a Directory Server restart.

ID: 16

Severity: ERROR
Message: You do not have sufficient privileges to initiate a Directory Server shutdown.

ID: 17

Severity: ERROR
Message: An error occurred while attempting to notify a synchronization provider of type %s about the schema changes made by the add schema file task: %s.

ID: 18

Severity: ERROR
Message: You do not have sufficient privileges to initiate an index rebuild.

ID: 20

Severity: ERROR
Message: Invalid DN provided to the Initialize task: %s.

ID: 21

Severity: ERROR
Message: Only users with the SERVER_LOCKDOWN privilege may place the server in lockdown mode.

ID: 22

Severity: ERROR
Message: Only users with the SERVER_LOCKDOWN privilege connected from a loopback address may place the server in lockdown mode.

ID: 23

Severity: ERROR
Message: Only users with the SERVER_LOCKDOWN privilege may cause the server to leave lockdown mode.

ID: 24

Severity: ERROR
Message: Only users with the SERVER_LOCKDOWN privilege connected from a loopback address may cause the server to leave lockdown mode.

ID: 25

Severity: ERROR
Message: You do not have sufficient privileges to terminate client connections.

ID: 26

Severity: ERROR
Message: Unable to decode value %s as an integer connection ID.

ID: 27

Severity: ERROR
Message: Attribute %s must be provided to specify the connection ID for the client to disconnect.

ID: 28

Severity: ERROR
Message: Unable to decode value %s as an indication of whether to notify the client before disconnecting it. The provided value should be either 'true' or 'false'.

ID: 30

Severity: ERROR
Message: There is no client connection with connection ID %s.

ID: 103

Severity: ERROR
Message: Invalid generation ID provided with the Initialize task. Error was: %s.

ID: 108

Severity: ERROR
Message: Index option cannot be specified when the rebuildAll or rebuildDegraded option is used.

ID: 110

Severity: ERROR
Message: Attribute %s has an invalid value. Reason: %s.

ID: 112

Severity: ERROR
Message: No changelog database was found for baseDN '%s'. Either the baseDN is not replicated or its changelog has not been enabled in this server.

ID: 113

Severity: ERROR
Message: The change number index cannot be reset because this OpenDJ instance does not appear to be a replication server.

ID: 114

Severity: ERROR
Message: Invalid change number (%d) specified, it must be greater than zero.

ID: 115

Severity: ERROR
Message: Unable to reset the change number index: %s.

ID: 116

Severity: ERROR
Message: The changes made by the add schema file task failed schema validation: %s.

ID: 127

Severity: ERROR
Message: Invalid DN provided to the Initialize Target task: %s.

ID: 128

Severity: ERROR
Message: Invalid DN provided to the Purge Conflicts Historical task: %s.

ID: 129

Severity: ERROR
Message: Invalid DN provided to the Reset Generation ID task: %s.

Category: Tools

Access and audit logs concern client operations rather than the server and tools, and so are not listed here. Instead, this document covers severe and fatal error messages for the server and its tools, such as those logged in logs/errors, and logs/replication.

ID: 17

Severity: ERROR
Message: An error occurred while parsing the command-line arguments: %s.

ID: 18

Severity: ERROR
Message: No clear-text password was specified. Use --%s, --%s or --%s to specify the password to encode.

ID: 19

Severity: ERROR
Message: No password storage scheme was specified. Use the --%s argument to specify the storage scheme.

ID: 20

Severity: ERROR
Message: An unexpected error occurred while attempting to bootstrap the Directory Server client-side code: %s.

ID: 21

Severity: ERROR
Message: An error occurred while trying to load the Directory Server configuration: %s.

ID: 22

Severity: ERROR
Message: An error occurred while trying to load the Directory Server schema: %s.

ID: 23

Severity: ERROR
Message: An error occurred while trying to initialize the core Directory Server configuration: %s.

ID: 24

Severity: ERROR
Message: An error occurred while trying to initialize the Directory Server password storage schemes: %s.

ID: 25

Severity: ERROR
Message: No password storage schemes have been configured for use in the Directory Server.

ID: 26

Severity: ERROR
Message: Password storage scheme "%s" is not configured for use in the Directory Server.

ID: 30

Severity: ERROR
Message: An error occurred while attempting to encode the clear-text password: %s.

ID: 52

Severity: ERROR
Message: Unable to decode exclude filter string "%s" as a valid search filter: %s.

ID: 53

Severity: ERROR
Message: Unable to decode include filter string "%s" as a valid search filter: %s.

ID: 54

Severity: ERROR
Message: Unable to decode base DN string "%s" as a valid distinguished name: %s.

ID: 55

Severity: ERROR
Message: Multiple Directory Server backends are configured with the requested backend ID "%s".

ID: 56

Severity: ERROR
Message: None of the Directory Server backends are configured with the requested backend ID "%s".

ID: 57

Severity: ERROR
Message: Unable to decode exclude branch string "%s" as a valid distinguished name: %s.

ID: 58

Severity: ERROR
Message: Unable to decode wrap column value "%s" as an integer.

ID: 59

Severity: ERROR
Message: An error occurred while attempting to process the LDIF export: %s.

ID: 63

Severity: ERROR
Message: Unable to load class %s referenced in configuration entry %s for use as a Directory Server backend: %s.

ID: 64

Severity: ERROR
Message: Unable to create an instance of class %s referenced in configuration entry %s as a Directory Server backend: %s.

ID: 89

Severity: ERROR
Message: Unable to decode exclude filter string "%s" as a valid search filter: %s.

ID: 90

Severity: ERROR
Message: Unable to decode include filter string "%s" as a valid search filter: %s.

ID: 92

Severity: ERROR
Message: Imported branches or backend IDs can not span across multiple Directory Server backends.

ID: 93

Severity: ERROR
Message: None of the Directory Server backends are configured with the requested backend ID or base DNs that include the specified branches.

ID: 94

Severity: ERROR
Message: Unable to decode exclude branch string "%s" as a valid distinguished name: %s.

ID: 95

Severity: ERROR
Message: An error occurred while trying to open the rejects file %s for writing: %s.

ID: 96

Severity: ERROR
Message: An error occurred while attempting to process the LDIF import: %s.

ID: 210

Severity: ERROR
Message: An error occurred while attempting to perform index verification: %s.

ID: 211

Severity: ERROR
Message: Only one index at a time may be verified for cleanliness.

ID: 212

Severity: ERROR
Message: The backend does not support indexing.

ID: 213

Severity: ERROR
Message: The Directory Server backend with backend ID "%s" does not provide a mechanism for performing LDIF exports.

ID: 214

Severity: ERROR
Message: The Directory Server backend with backend ID %s does not provide a mechanism for performing LDIF imports.

ID: 218

Severity: ERROR
Message: Unable to decode include branch string "%s" as a valid distinguished name: %s.

ID: 219

Severity: ERROR
Message: Provided include base DN "%s" is not handled by the backend with backend ID %s.

ID: 230

Severity: ERROR
Message: Multiple Directory Server backends are configured to support base DN "%s".

ID: 231

Severity: ERROR
Message: None of the Directory Server backends are configured to support the requested base DN "%s".

ID: 242

Severity: ERROR
Message: Provided include base DN "%s" is not handled by the backend with backend ID %s.

ID: 284

Severity: ERROR
Message: An error occurred while attempting to initialize the crypto manager: %s.

ID: 285

Severity: ERROR
Message: An error occurred while attempting to initialize the subentry manager: %s.

ID: 332

Severity: ERROR
Message: An error occurred while attempting to acquire an exclusive lock for backend %s: %s. This generally means some other process is still using this backend (e.g., it is in use by the Directory Server or a backup or LDIF export is in progress). The LDIF import cannot continue.

ID: 334

Severity: ERROR
Message: An error occurred while attempting to acquire a shared lock for backend %s: %s. This generally means that some other process has an exclusive lock on this backend (e.g., an LDIF import or a restore). The LDIF export cannot continue.

ID: 336

Severity: ERROR
Message: An error occurred while attempting to acquire a shared lock for backend %s: %s. This generally means that some other process has an exclusive lock on this backend (e.g., an LDIF import or a restore). The index verification cannot continue.

ID: 371

Severity: ERROR
Message: Authentication password storage scheme "%s" is not configured for use in the Directory Server.

ID: 372

Severity: ERROR
Message: An error occurred while copying OpenDMK jar file '%s' to '%s': %s.

ID: 373

Severity: ERROR
Message: An error occurred while attempting to initialize the password policy components: %s.

ID: 396

Severity: ERROR
Message: ERROR: Unable to decode the provided stop time. It should be in the form YYYYMMDDhhmmssZ for UTC time or YYYYMMDDhhmmss for local time.

ID: 403

Severity: ERROR
Message: ERROR: An I/O error occurred while attempting to communicate with the Directory Server: %s.

ID: 605

Severity: ERROR
Message: Neither the %s or the %s argument was provided. One of these arguments must be given to specify the source for the LDIF data to be imported.

ID: 606

Severity: ERROR
Message: Unable to parse the specified file %s as a MakeLDIF template file: %s.

ID: 748

Severity: ERROR
Message: The provided password is not a valid encoded user password value: %s.

ID: 813

Severity: ERROR
Message: Could not find the service name for the server.

ID: 814

Severity: ERROR
Message: An unexpected error occurred starting the server as a windows service.

ID: 815

Severity: ERROR
Message: An unexpected error occurred stopping the server windows service.

ID: 823

Severity: ERROR
Message: You can only provide one of the following arguments: enableService, disableService, serviceState or cleanupService.

ID: 824

Severity: ERROR
Message: You must provide at least one of the following arguments: enableService, disableService or serviceState or cleanupService.

ID: 829

Severity: ERROR
Message: The server could not be enabled to run as a Windows service. The service name is already in use.

ID: 830

Severity: ERROR
Message: An unexpected error occurred trying to enable the server as a Windows service.%nCheck that you have administrator rights (only Administrators can enable the server to run as a Windows Service).

ID: 834

Severity: ERROR
Message: An unexpected error occurred trying to disable the server as a Windows service%nCheck that you have administrator rights (only Administrators can disable the server as a Windows Service).

ID: 837

Severity: ERROR
Message: An unexpected error occurred trying to retrieve the state of the server as a Windows service.

ID: 846

Severity: ERROR
Message: Could not find the service with name %s.

ID: 848

Severity: ERROR
Message: An unexpected error occurred cleaning up the service %s.

ID: 852

Severity: ERROR
Message: An error occurred while attempting to perform index rebuild: %s.

ID: 853

Severity: ERROR
Message: The backend does not support rebuilding of indexes.

ID: 854

Severity: ERROR
Message: At least one index must be specified for the rebuild process.

ID: 855

Severity: ERROR
Message: An error occurred while attempting to acquire a exclusive lock for backend %s: %s. This generally means that some other process has an lock on this backend or the server is running with this backend online. The rebuild process cannot continue.

ID: 857

Severity: ERROR
Message: An error occurred while attempting to acquire a shared lock for backend %s: %s. This generally means that some other process has an exclusive lock on this backend (e.g., an LDIF import or a restore). The rebuild process cannot continue.

ID: 887

Severity: ERROR
Message: The specified LDIF file %s cannot be read.

ID: 1198

Severity: ERROR
Message: An error occurred while trying to open the skip file %s for writing: %s.

ID: 1252

Severity: ERROR
Message: The file '%s' could not be renamed to '%s': %s.

ID: 1321

Severity: ERROR
Message: This tool may only be used on UNIX-based systems.

ID: 1324

Severity: ERROR
Message: Unable to determine the path to the server root directory. Please ensure that the %s system property or the %s environment variable is set to the path of the server root directory.

ID: 1325

Severity: ERROR
Message: An error occurred while attempting to generate the RC script: %s.

ID: 1347

Severity: ERROR
Message: None of the Directory Server JE backends are configured with the requested backend ID %s.

ID: 1348

Severity: ERROR
Message: None of the entry containers are configured with the requested base DN %s in backend %s.

ID: 1352

Severity: ERROR
Message: Unable to decode base DN string "%s" as a valid distinguished name: %s.

ID: 1363

Severity: ERROR
Message: An error occurred while attempting to acquire a shared lock for backend %s: %s. This generally means that some other process has exclusive access to this backend (e.g., a restore or an LDIF import).

ID: 1374

Severity: ERROR
Message: A sub-command must be specified.

ID: 1378

Severity: ERROR
Message: The directory %s specified as the OPENDJ_JAVA_HOME path does not exist or is not a directory.

ID: 1411

Severity: ERROR
Message: The argument '%s' is incompatible with '%s'.

ID: 1422

Severity: ERROR
Message: Invalid menu item or task number '%s'.

ID: 1437

Severity: ERROR
Message: Error retrieving task entry %s: %s.

ID: 1438

Severity: ERROR
Message: There are no tasks with ID %s.

ID: 1448

Severity: ERROR
Message: Error canceling task '%s': %s.

ID: 1449

Severity: ERROR
Message: Error accessing logs for task '%s': %s.

ID: 1450

Severity: ERROR
Message: Task at index %d is not cancelable.

ID: 1453

Severity: ERROR
Message: There are no tasks defined with ID '%s'.

ID: 1454

Severity: ERROR
Message: Task '%s' has finished and cannot be canceled.

ID: 1455

Severity: ERROR
Message: State for task '%s' cannot be determined.

ID: 1457

Severity: ERROR
Message: The start date/time must in YYYYMMDDhhmmssZ format for UTC time or YYYYMMDDhhmmss for local time.

ID: 1459

Severity: ERROR
Message: You have provided options for scheduling this operation as a task but options provided for connecting to the server's tasks backend resulted in the following error: '%s'.

ID: 1473

Severity: ERROR
Message: The option %s is only applicable when scheduling this operation as a task.

ID: 1474

Severity: ERROR
Message: The value %s for option %s is not a valid email address.

ID: 1475

Severity: ERROR
Message: The failed dependency action value %s is invalid. The value must be one of %s.

ID: 1476

Severity: ERROR
Message: The failed dependency action option is to be used in conjunction with one or more dependencies.

ID: 1477

Severity: ERROR
Message: Error: task %s is not in a cancelable state.

ID: 1652

Severity: ERROR
Message: An error occurred while attempting to initialize server components to run the tool: %s.

ID: 1653

Severity: ERROR
Message: The %s argument is not supported for online imports.

ID: 1667

Severity: ERROR
Message: The specified start time '%s' has already passed.

ID: 1669

Severity: ERROR
Message: The specified stop time '%s' has already passed.

ID: 1680

Severity: ERROR
Message: The timeout of '%d' seconds has been reached. You can use the argument '--%s' to increase this timeout.

ID: 1688

Severity: ERROR
Message: The value %s for threadCount cannot be parsed: %s.

ID: 1693

Severity: ERROR
Message: Provided passwords don’t matched.

ID: 1694

Severity: ERROR
Message: Cannot read password from the input: %s.

ID: 1709

Severity: ERROR
Message: The Windows Service was successfully configured but there was an error starting it. Error code starting Windows Service: %d.

ID: 1718

Severity: ERROR
Message: The provided schedule value has an invalid format. The schedule must be expressed using a crontab(5) format. Error details: %s.

ID: 1737

Severity: ERROR
Message: The version of the installed OpenDJ could not be determined because the version file '%s' could not be found. Restore it from backup before continuing.

ID: 1738

Severity: ERROR
Message: The version of the installed OpenDJ could not be determined because the version file '%s' exists but contains invalid data. Restore it from backup before continuing.

ID: 1739

Severity: ERROR
Message: The OpenDJ binary version '%s' does not match the installed configuration version '%s'. Please run upgrade before continuing.

ID: 1800

Severity: ERROR
Message: The upgrade failed to complete for the following reason: %s.

ID: 1805

Severity: ERROR
Message: OpenDJ cannot be upgraded because the server is currently running. Please stop the server and try again.

ID: 1806

Severity: ERROR
Message: OpenDJ has already been upgraded to version %s.

ID: 1807

Severity: ERROR
Message: An unexpected error occurred while attempting to display a notification: %s.

ID: 1808

Severity: ERROR
Message: An error occurred when trying to read the configuration version in %s : %s.

ID: 1812

Severity: ERROR
Message: An error occurred while performing an upgrade task: %s.

ID: 1816

Severity: ERROR
Message: No %s with OID %s exists in the schema.

ID: 1817

Severity: ERROR
Message: An error occurred when trying to upgrade the config/upgrade folder: %s.

ID: 1827

Severity: ERROR
Message: The upgrade failed because %d errors were encountered. Please check log for further details.

ID: 1828

Severity: ERROR
Message: An error occurred while copying the schema file '%s': %s.

ID: 1829

Severity: ERROR
Message: An error occurred while adding one or more attributes to the schema file '%s': %s.

ID: 1835

Severity: ERROR
Message: An error occurred while adding configuration file '%s': %s.

ID: 1843

Severity: ERROR
Message: An error occurred during post upgrade task. Process aborted. Please check log for further details.

ID: 1846

Severity: ERROR
Message: Invalid log file %s.

ID: 1850

Severity: ERROR
Message: '%s' is missing or empty, it is probably corrupted.

ID: 1863

Severity: ERROR
Message: An error occurred while listing the base DNs: %s.

ID: 1864

Severity: ERROR
Message: An error occurred while listing indexes: %s.

ID: 1865

Severity: ERROR
Message: An unexpected error occurred while attempting to initialize the backend '%s': %s.

ID: 1866

Severity: ERROR
Message: An unexpected error occurred while attempting to read and/or decode records from an index: %s.

ID: 1868

Severity: ERROR
Message: No index exists with the requested name '%s' in base DN '%s' and backend '%s'.

ID: 1869

Severity: ERROR
Message: Cannot specify a minimum key both as a string and as an hexadecimal string.

ID: 1870

Severity: ERROR
Message: Cannot specify a maximum key both as a string and as an hexadecimal string.

ID: 1871

Severity: ERROR
Message: An error occurred while processing arguments: %s.

ID: 1872

Severity: ERROR
Message: An error occurred while trying to execute %s: %s.

ID: 1881

Severity: ERROR
Message: Cannot configure backend %s: %s.

ID: 1887

Severity: ERROR
Message: At key number %d, %s:.

ID: 1890

Severity: ERROR
Message: Data decoder for printing is not available, should use hex dump.

ID: 1891

Severity: ERROR
Message: No storage index exists with the requested name %s in backend %s.

ID: 1897

Severity: ERROR
Message: An error occurred while initializing server backends: %s.

ID: 1898

Severity: ERROR
Message: An error occurred while initializing plugins: %s.

ID: 1899

Severity: ERROR
Message: Subsystem %s should be initialized first.

ID: 1907

Severity: ERROR
Message: OpenDJ data has already been upgraded to version %s.

ID: 1910

Severity: ERROR
Message: The OpenDJ binary version '%s' does not match the installed data version '%s'. Please run 'upgrade --dataOnly' before continuing.

ID: 1928

Severity: ERROR
Message: An error occurred when reading the replication server configuration entry '%s': %s.

ID: 1930

Severity: ERROR
Message: The replication domain '%s' is not found. Make sure the domain is replicated.

ID: 1932

Severity: ERROR
Message: Replica DB folder '%s' not found.

ID: 1934

Severity: ERROR
Message: The provided change number '%s' is not valid.

ID: 1935

Severity: ERROR
Message: The provided CSN '%s' is not valid.

ID: 1936

Severity: ERROR
Message: The output folder '%s' cannot be created.

ID: 1937

Severity: ERROR
Message: The provided output folder '%s' doesn’t exist.

ID: 1938

Severity: ERROR
Message: The provided DN '%s' is invalid.

ID: 1939

Severity: ERROR
Message: Cannot list the changelog files in '%s': %s.

ID: 1940

Severity: ERROR
Message: Cannot convert the filename '%s' to a valid change number.

ID: 1941

Severity: ERROR
Message: Cannot convert the filename '%s' to a valid CSN.

ID: 1942

Severity: ERROR
Message: Error while decoding the changelog file '%s' : %s.

ID: 1944

Severity: ERROR
Message: The OpenDJ binary version '%s' is older than the configuration version '%s', it usually means that an older version has been unzipped over the previous binaries. Unzip a more recent version than the configuration version and run upgrade again.

ID: 1945

Severity: ERROR
Message: The OpenDJ binary version '%s' is older than the data version '%s', it usually means that an older version has been unzipped over the previous binaries. Unzip a more recent version than the data version and run upgrade again.

ID: 10055

Severity: ERROR
Message: Unable to access the LDIF file %s to import. Please check that the file is local to the server and the path correct.

ID: 10082

Severity: ERROR
Message: Could not find a server ID to set for the server. Verify the configuration references a valid server ID for domain cn=admin data.

ID: 10095

Severity: ERROR
Message: Crypto manager configuration entry not found.

ID: 20013

Severity: ERROR
Message: The server has not been configured. Please run the 'setup' command first.

ID: 20019

Severity: ERROR
Message: Rebuild index aborted: an error has occurred while rebuilding indexes for base DN '%s'.

ID: 20022

Severity: ERROR
Message: An error occurred while attempting to generate the systemd service file: %s.

Tools reference

About this reference

This reference covers Directory Services tools, which are bundled with the software. For the `dsconfig` command, also see the Configuration reference.

addrate

`addrate` — measure add and delete throughput and response time

Synopsis

```
addrate {options} template-file-path
```

Description

This utility can be used to measure add and optionally delete throughput and response time of a directory server using user-defined entries. The {template-file-path} argument identifies a template file that has the same form as a template file for the makeldif command.

Examples:

This example adds entries and randomly deletes them while the number of entries added is greater than 10,000:

addrate -p 1636 -Z -X -D uid=admin -w password -f -c 10 -C random -s 10000 addrate.template

This example adds entries and starts to delete them in the same order if their age is greater than a certain time:

addrate -p 1636 -Z -X -D uid=admin -w password -f -c 10 -C fifo -a 2 addrate.template

For details about the template file, see the documentation.

When you do not use the `-f` option to keep connections open and rebind on the connections, the tool can exhaust its available ports, causing the tool to crash. You can work around this problem on test systems by changing TCP settings on the system.

For example, on Linux systems, set the following parameters in the `/etc/sysctl.conf` file:

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
```

The parameter `net.ipv4.tcp_fin_timeout` sets the length of time in seconds to wait for a final FIN packet before forcing a close of the socket. The default is 60 (seconds).

The parameter `net.ipv4.tcp_tw_recycle` enables fast recycling of TIME_WAIT sockets. The default is 0 (false). Enabling this can cause Network Address Translation (NAT) issues.

The parameter `net.ipv4.tcp_tw_reuse` enables reuse of TIME_WAIT sockets for new connections. The default is 0 (false).

These settings are recommended only for testing, and *not for production systems* .

After making the changes to `/etc/sysctl.conf` , reload the configuration with the `sysctl` command:

```
# sysctl -p
```

Options

The `addrate` command takes the following options:

Command options:

`-a | --deleteAgeThreshold {seconds}`

Specifies the age at which added entries will become candidates for deletion.

`-B | --warmUpDuration {warmUpDuration}`

Warm up duration in seconds. Default: 0

`-c | --numConnections {numConnections}`

Number of connections. Default: 1

`-C | --deleteMode {fifo | random | off}`

The algorithm used for selecting entries to be deleted which must be one of "fifo", "random", or "off". Default: FIFO

`-d | --maxDuration {maxDuration}`

Maximum duration in seconds, 0 for unlimited. Default: 0

`-e | --percentile {percentile}`

Calculate max response time for a percentile of operations.

-f | --keepConnectionsOpen

Keep connections open. Default: false

-F | --noRebind

Keep connections open and do not rebind. Default: false

-g | --constant {name=value}

A constant that overrides the value set in the template file.

-i | --statInterval {statInterval}

Display results each specified number of seconds. Default: 5

-m | --maxIterations {maxIterations}

Max iterations, 0 for unlimited. Default: 0

-M | --targetThroughput {targetThroughput}

Target average throughput to achieve. Default: 0

-n | --noPurge

Disable the purge phase when the tool stops. Default: false

-r | --resourcePath {path}

Path to look for template resources (e.g. data files). The utility looks for resources in the following locations in this order:

1. The current directory where the command is run.
2. The resource path directory.
3. The built-in files.

-R | --randomSeed {seed}

The seed to use for initializing the random number generator. To always generate the same data with the same command, use the same non-zero seed value. A value of zero (the default) results in different data each time the tool is run. Default: 0

-s | --deleteSizeThreshold {count}

Specifies the number of entries to be added before deletion begins. Default: 10000

-S | --scriptFriendly

Use script-friendly mode. Default: false

-t | --numConcurrentRequests {numConcurrentRequests}

Number of concurrent requests per connection. Default: 1

-Y | --proxyAs {authzID}

Use the proxied authorization control with the given authorization ID.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default:

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server port number.

-q | --useStartTls

Use StartTLS to secure communication with the server. Default: false

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

-Z | --useSsl

Use SSL for secure communication with the server. Default: false

Utility input/output options:

--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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-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.

80

The command could not complete due to an input/output error.

89

An error occurred while parsing the command-line arguments.

Examples

The following example adds entries, and then randomly deletes them when more than 10,000 entries have been added:

```
$ addrate \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --bindPassword password \
 --numConnections 10 \
 --keepConnectionsOpen \
 --deleteMode random \
 --deleteSizeThreshold 10,000 \
 --maxDuration 30 \
 /path/to/opendj/config/MakeLDIF/addrate.template
--------------------------------------------------------------------------------------
|     Throughput    |                 Response Time                |    Additional   |
|    (ops/second)   |                (milliseconds)                |    Statistics   |
|   recent  average |   recent  average    99.9%   99.99%  99.999% |  err/sec   Add% |
--------------------------------------------------------------------------------------
|    499.7    499.7 |   13.666   13.666   141.56   212.86   212.86 |       0.0 100.00 |
|   1114.4    807.0 |    6.340    8.608    98.04   167.77   212.86 |       0.0 100.00 |
|   1441.8   1018.6 |    4.946    6.880    72.35   167.77   212.86 |       0.0  63.36 |
```

```
|   1554.5   1152.6 |    4.615    6.116    53.74   167.77   212.86 |      0.0  49.98 |
|   1708.2   1263.7 |    4.176    5.592    49.55   141.56   212.86 |      0.0  49.96 |
|   1112.6   1238.5 |    6.455    5.721    51.38   203.42   212.86 |      0.0  50.02 |
|    611.1   1238.2 |    9.125    5.722    51.38   203.42   212.86 |      0.0   0.00 |
Purge phase...
9999 entries have been successfully purged
```

The following example also adds entries, and then deletes them in the order they were added after they are 10 seconds old:

```
$ addrate \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --bindPassword password \
 --numConnections 10 \
 --keepConnectionsOpen \
 --deleteMode fifo \
 --deleteAgeThreshold 10 \
 --maxDuration 30 \
 /path/to/opendj/config/MakeLDIF/addrate.template
-------------------------------------------------------------------------------------
|     Throughput    |                 Response Time                |    Additional   |
|    (ops/second)   |                (milliseconds)                |    Statistics   |
|   recent  average |   recent  average    99.9%   99.99%  99.999% |  err/sec   Add% |
-------------------------------------------------------------------------------------
|   1489.6   1489.6 |    4.585    4.585    28.70    31.20    51.64 |      0.0 100.00 |
|   1262.8   1376.2 |    5.698    5.096    41.68    52.69    55.31 |      0.0 100.00 |
|   1596.2   1449.5 |    4.430    4.851    36.18    52.43    55.31 |      0.0  50.71 |
|   1237.8   1396.6 |    5.859    5.075    44.56   115.34   119.01 |      0.0  50.00 |
|   1156.0   1348.5 |    6.195    5.267    44.83   115.34   119.01 |      0.0  49.96 |
|   1373.3   1352.6 |    5.226    5.260    46.40   114.82   119.01 |      0.0  49.99 |
Purge phase...
Purge in progress:  8195/13885 entries deleted (1638.2 ops/sec). ETA 00:00:03
```

These examples use the following options:

`--hostname localhost`, `--port 1636`, `--useSsl`, `--usePkcs12TrustStore /path/to/opendj/config/keystore`, `--trustStorePassword:file /path/to/opendj/config/keystore.pin`

Access the server running on the local system over a secure LDAPS connection to port 1636.

`--bindDn uid=admin`, `--bindPassword password`

Authenticate as the directory root user `uid=admin` with the bind password that is literally `password`. This user is not subject to access control, so rates may be higher than what you observe with a regular user.

`--numConnections 10`

Open 10 connections to the server.

`--keepConnectionsOpen`

Keep the connections open to reuse them during the operation.

`--deleteMode (random | fifo)`

After adding entries, delete them in random order, or in first-in-first-out order.

`--deleteSizeThreshold 10,000`

Add 10,000 entries before starting to delete them.

`--deleteAgeThreshold 10`

Begin to delete entries when they are 10 seconds old.

`/path/to/opendj/config/MakeLDIF/addrate.template`

When building entries to add, use this file as the template.

`--maxDuration 30`

Run for a maximum of 30 seconds.

Notice the following characteristics of the output:

- The first two columns show the throughput in operations completed per second. The recent column shows the average rate for operations reflected in this row of output. The average column shows the average rate since the beginning of the run.
- The response time columns indicate characteristics of response latency in milliseconds. The recent column shows the average latency for operations reflected in this row of output. The average column shows the average latency since the beginning of the run. The "99.9%" column shows the latency after which 99.9% of operations have completed. Only 1 operation in 1000 took longer than this. The "99.99%" column shows the latency after which 99.99% of operations have completed. Only 1 operation in 10,000 took longer than this. The "99.999%" column shows the latency after which 99.999% of operations have completed. Only 1 operation in 100,000 took longer than this.
- The additional statistics columns show information about what is happening during the run. The "err/sec" column shows the rate of error results per second for this row of output. Unless you have intentionally set up the command to generate errors, this column should indicate `0.0` . Check that this column matches your expectations before looking at any other columns. The "Add%" column shows the percentage of operations performed that were adds. The rest are delete operations. Notice that the percentage of add operations drops as the command begins to delete entries.

authrate

`authrate` — measure bind throughput and response time

Synopsis

```
authrate {options} [filter template string] [attributes …]
```

Description

This utility can be used to measure bind throughput and response time of a directory service using user-defined bind or search-then-bind operations.

Template strings may be used in the bind DN option as well as the authid and authzid SASL bind options. A search operation may be used to retrieve the bind DN by specifying the base DN and a filter. The retrieved entry DN will be appended as the last argument in the argument list when evaluating template strings.

Example (bind only):

authrate -p 1636 -Z -X -D 'uid=user.{},ou=people,dc=example,dc=com' \

-w password -f -c 10 -g 'rand(0,2000)'

Example (search then bind):

authrate -p 1636 -Z -X -D '{2}' -w password -f -c 10 \

-b 'ou=people,dc=example,dc=com' -s one -g 'rand(0,2000)' '(uid=user.{1})'

Before trying the example, import 2000 randomly generated users.

When you do not use the `-f` option to keep connections open and rebind on the connections, the tool can exhaust its available ports, causing the tool to crash. You can work around this problem on test systems by changing TCP settings on the system.

For example, on Linux systems, set the following parameters in the `/etc/sysctl.conf` file:

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
```

The parameter `net.ipv4.tcp_fin_timeout` sets the length of time in seconds to wait for a final FIN packet before forcing a close of the socket. The default is 60 (seconds).

The parameter `net.ipv4.tcp_tw_recycle` enables fast recycling of TIME_WAIT sockets. The default is 0 (false). Enabling this can cause Network Address Translation (NAT) issues.

The parameter `net.ipv4.tcp_tw_reuse` enables reuse of TIME_WAIT sockets for new connections. The default is 0 (false).

These settings are recommended only for testing, and *not for production systems* .

After making the changes to `/etc/sysctl.conf` , reload the configuration with the `sysctl` command:

```
# sysctl -p
```

Options

The `authrate` command takes the following options:

Command options:

`-a | --dereferencePolicy {dereferencePolicy}`

Alias dereference policy ('never', 'always', 'search', or 'find'). Default: never

`-b | --baseDn {baseDN}`

Base DN template string.

-B | --warmUpDuration {warmUpDuration}

Warm up duration in seconds. Default: 0

-c | --numConnections {numConnections}

Number of connections. Default: 1

-d | --maxDuration {maxDuration}

Maximum duration in seconds, 0 for unlimited. Default: 0

-e | --percentile {percentile}

Calculate max response time for a percentile of operations.

-f | --keepConnectionsOpen

Keep connections open. Default: false

-g | --argument {generator function or static string}

Argument used to evaluate the template strings in program parameters (ie. Base DN, Search Filter). The set of all arguments provided form the argument list in order. Besides static string arguments, they can be generated per iteration with the following functions:

"inc({filename})" Consecutive, incremental line from file

"inc({min},{max})" Consecutive, incremental number

"rand({filename})" Random line from file

"rand({min},{max})" Random number

"randstr({length},*charSet*)" Random string of specified length and optionally from characters in the charSet string. A range of character can be specified with [start-end] charSet notation. If no charSet is specified, the default charSet of [A-Z][a-z][0-9] will be used.

These functions do not support formatted integers with comma due to the ambiguity between a comma used to separate function arguments and a comma used to separate digits in a formatted integer.

-i | --statInterval {statInterval}

Display results each specified number of seconds. Default: 5

-I | --invalidPassword {invalidPassword}

Percentage of requests that will send an invalid password (between 0 and 100). Default: 0

-m | --maxIterations {maxIterations}

Max iterations, 0 for unlimited. Default: 0

-M | --targetThroughput {targetThroughput}

Target average throughput to achieve. Default: 0

-s | --searchScope {searchScope}

Search scope ('base', 'one', 'sub', or 'subordinates'). Note: 'subordinates' is an LDAP extension that might not work with all LDAP servers. Default: sub

-S | --scriptFriendly

Use script-friendly mode. Default: false

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default:

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server port number.

-q | --useStartTls

Use StartTLS to secure communication with the server. Default: false

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

-Z | --useSsl

Use SSL for secure communication with the server. 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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-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.

89

An error occurred while parsing the command-line arguments.

Examples

The following example demonstrates measuring simple bind performance:

```
$ authrate \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --argument "rand(0,2000)" --bindDn "uid=user.{},ou=people,dc=example,dc=com" \
 --bindPassword password \
 --numConnections 10 \
 --maxDuration 30 \
 --keepConnectionsOpen
--------------------------------------------------------------------------------
|     Throughput    |                 Response Time                |          |
|    (ops/second)   |                (milliseconds)                |          |
|   recent  average |   recent  average    99.9%   99.99%  99.999% |  err/sec |
--------------------------------------------------------------------------------
|  20306.0  20306.0 |    0.469    0.469    11.40    38.01    55.05 |      0.0 |
|  27672.6  23989.3 |    0.352    0.401     8.52    24.12    50.33 |      0.0 |
|  27410.0  25129.5 |    0.355    0.385     7.60    18.35    43.78 |      0.0 |
|  27082.2  25617.7 |    0.359    0.378     7.21    17.43    43.25 |      0.0 |
|  28027.4  26099.6 |    0.347    0.371     6.62    17.17    42.99 |      0.0 |
|  26805.7  26217.2 |    0.361    0.370     6.32    16.65    42.99 |      0.0 |
```

This example uses the following options:

`--hostname localhost`, `--port 1636`, `--useSsl`, `--usePkcs12TrustStore /path/to/opendj/config/keystore`, `--trustStorePassword:file /path/to/opendj/config/keystore.pin`

Access the server running on the local system over a secure LDAPS connection to port 1636.

`--argument "rand(0,2000)" --bindDn "uid=user.{},ou=people,dc=example,dc=com"`

Authenticate as a user with bind DN `uid=user.number,ou=people,dc=example,dc=com`, where *number* is a random number between 0 and 2000, inclusive.

`--bindPassword password`

Authenticate with the bind password that is literally `password`.

`--numConnections 10`

Open 10 connections to the server.

`--maxDuration 30`

Run for a maximum of 30 seconds.

`--keepConnectionsOpen`

Keep the connections open to reuse them during the operation.

Notice the following characteristics of the output:

- The first two columns show the throughput in operations completed per second. The recent column shows the average rate for operations reflected in this row of output. The average column shows the average rate since the beginning of the run.
- The response time columns indicate characteristics of response latency in milliseconds. The recent column shows the average latency for operations reflected in this row of output. The average column shows the average latency since the beginning of the run. The "99.9%" column shows the latency after which 99.9% of operations have completed. Only 1 operation in 1000 took longer than this. The "99.99%" column shows the latency after which 99.99% of operations have completed. Only 1 operation in 10,000 took longer than this. The "99.999%" column shows the latency after which 99.999% of operations have completed. Only 1 operation in 100,000 took longer than this.
- The "err/sec" column show the rate of error results per second for this row of output. Unless you have intentionally set up the command to generate errors, this column should indicate `0.0`. Check that this column matches your expectations before looking at any other columns.

backendstat

`backendstat` — gather OpenDJ backend debugging information

Synopsis

```
backendstat {subcommand} {options}
```

Description

This utility can be used to debug a backend.

Options

The `backendstat` command takes the following options:

`-V | --version`

Display Directory Server version information. Default: false

`-H | --help`

Display this usage information. Default: false

Subcommands

The `backendstat` command supports the following subcommands:

backendstat dump-index

```
backendstat dump-index {options}
```

Dump records from an index, decoding keys and values. Depending on index size, this subcommand can generate lots of output.

Options

In addition to the global `backendstat` options, the `backendstat dump-index` subcommand takes the following options:

`-b | --baseDn {baseDN}`

The base DN within the backend.

`-i | --indexName {indexName}`

The name of the index.

`-k | --minKeyValue {minKeyValue}`

Only show records with keys that should be ordered after the provided value using the comparator for the database container.

`-K | --maxKeyValue {maxKeyValue}`

Only show records with keys that should be ordered before the provided value using the comparator for the database container.

`-n | --backendId {backendName}`

The backend ID of the backend.

-p | --skipDecode

Do not try to decode backend data to their appropriate types. Default: false

-q | --statsOnly

Do not display backend data, just statistics. Default: false

-s | --minDataSize {minDataSize}

Only show records whose data is no smaller than the provided value. Default: -1

-S | --maxDataSize {maxDataSize}

Only show records whose data is no larger than the provided value. Default: -1

-x | --minHexKeyValue {minKeyValue}

Only show records with keys that should be ordered after the provided value using the comparator for the database container.

-X | --maxHexKeyValue {maxKeyValue}

Only show records with keys that should be ordered before the provided value using the comparator for the database container.

backendstat dump-raw-db

`backendstat dump-raw-db {options}`

Dump the raw records in hexadecimal format for a low-level database within the pluggable backend's storage engine. Depending on index size, this subcommand can generate lots of output.

Options

In addition to the global `backendstat` options, the `backendstat dump-raw-db` subcommand takes the following options:

-d | --dbName {databaseName}

The raw database name.

-k | --minKeyValue {minKeyValue}

Only show records with keys that should be ordered after the provided value using the comparator for the database container.

-K | --maxKeyValue {maxKeyValue}

Only show records with keys that should be ordered before the provided value using the comparator for the database container.

-l | --singleLine

Write hexadecimal data on a single line instead of pretty format. Default: false

-n | --backendId {backendName}

The backend ID of the backend.

-q | --statsOnly

Do not display backend data, just statistics. Default: false

-s | --minDataSize {minDataSize}

Only show records whose data is no smaller than the provided value. Default: -1

-S | --maxDataSize {maxDataSize}

Only show records whose data is no larger than the provided value. Default: -1

-x | --minHexKeyValue {minKeyValue}

Only show records with keys that should be ordered after the provided value using the comparator for the database container.

-X | --maxHexKeyValue {maxKeyValue}

Only show records with keys that should be ordered before the provided value using the comparator for the database container.

backendstat list-backends

```
backendstat list-backends
```

List the pluggable backends.

backendstat list-base-dns

```
backendstat list-base-dns {options}
```

List the base DNs in a backend.

Options

In addition to the global `backendstat` options, the `backendstat list-base-dns` subcommand takes the following options:

-n | --backendId {backendName}

The backend ID of the backend.

backendstat list-indexes

```
backendstat list-indexes {options}
```

List the indexes associated with a pluggable backend. This subcommand may take a long time to complete depending on the size of the backend.

Options

In addition to the global `backendstat` options, the `backendstat list-indexes` subcommand takes the following options:

`-b | --baseDn {baseDN}`

The base DN within the backend.

`-n | --backendId {backendName}`

The backend ID of the backend.

backendstat list-raw-dbs

```
backendstat list-raw-dbs {options}
```

List the low-level databases within a pluggable backend's storage engine. This subcommand may take a long time to complete depending on the size of the backend.

Options

In addition to the global `backendstat` options, the `backendstat list-raw-dbs` subcommand takes the following options:

`-n | --backendId {backendName}`

The backend ID of the backend.

`-u | --useSiUnits`

Uses SI Units for printing sizes. Default: false

backendstat show-index-status

```
backendstat show-index-status {options}
```

Shows the status of indexes for a backend base DN. This subcommand can take a long time to complete, as it reads all indexes for all backends.

When you run the `show-index-status` subcommand, the result is a table, followed by a "Total", which is the total number of indexes, followed by a list of indexes with "Over index-entry-limit keys" to show the values for which the number of entries exceeded the index entry limit.

The table has the following columns:

(No label)

If the index needs rebuilding, its row starts with `!` . Otherwise, its row starts with a space.

Index Name

Name of the index, where the format depends on the index. For example, `givenName.caseIgnoreSubstringsMatch:6` :

- Attribute indexes: *attr.type* . *type*
- Big indexes: *attr.type* .big. *type*

- VLV indexes: vlv. *type*

Secure

`+` means confidentiality is enabled for the index. `-` means confidentiality is disabled.

Size

The size on disk.

Key Count

Number of indexed keys. Use the `backendstat dump-tree` command to see how many entry IDs correspond to each key.

Over

Number of keys for which there are too many values to maintain an index, based on the `index-entry-limit` . This is recorded as `-` for VLV indexes. In other words, with the default index entry limit of 4000, if every user in your large directory has an email address ending in `@example.com` , and a substring index with default substring length of 6 is maintained for `mail` , then the directory server does not maintain indexes for keys corresponding to substrings in `@example.com` . As a result, an LDAP search with the filter `"(mail=*@example.com)"` becomes an unindexed search even though a substring index exists for the mail attribute. By default, the directory server does not allow unindexed searches except by privileged users. This is usually exactly the behavior you want in order to prevent client applications from sending searches that return every user in the directory for example. Clients should refine their search filters instead.

Entry Limit

The `index-entry-limit` setting that applies to this index. Default: `4000`

Mean

Average number of values per key for this index.

Median

Median number of values per key for this index.

80th, 95th, 99th

Percentage of keys having at most the specified number of values. This is a measure of how full the entry ID lists are.

Options

In addition to the global `backendstat` options, the `backendstat show-index-status` subcommand takes the following options:

`-b | --baseDn {baseDN}`

The base DN within the backend.

`-n | --backendId {backendName}`

The backend ID of the backend.

Exit codes

0

The command completed successfully.

> 0

An error occurred.

base64

`base64` — encode and decode base64 strings

Synopsis

```
base64 {subcommand} {options}
```

Description

This utility can be used to encode and decode information using base64.

Options

The `base64` command takes the following options:

`-V | --version`

Display Directory Server version information. Default: false

`-H | --help`

Display this usage information. Default: false

Subcommands

The `base64` command supports the following subcommands:

base64 decode

```
base64 decode {options}
```

Decode base64-encoded information into raw data. When no options are specified, this subcommand reads from standard input and writes to standard output.

Options

In addition to the global `base64` options, the `base64 decode` subcommand takes the following options:

-d | --encodedData {data}

The base64-encoded data to be decoded.

-f | --encodedDataFile {path}

The path to a file containing the base64-encoded data to be decoded.

-o | --toRawFile {path}

The path to a file to which the raw base64-decoded data should be written.

base64 encode

`base64 encode {options}`

Encode raw data using base64. When no options are specified, this subcommand reads from standard input and writes to standard output.

Options

In addition to the global `base64` options, the `base64 encode` subcommand takes the following options:

-d | --rawData {data}

The raw data to be base64 encoded.

-f | --rawDataFile {path}

The path to a file containing the raw data to be base64 encoded.

-o | --toEncodedFile {path}

The path to a file to which the base64-encoded data should be written.

Exit codes

0

The command completed successfully.

> 0

An error occurred.

changelogstat

`changelogstat` — debug changelog and changenumber files

Synopsis

```
changelogstat {subcommand} {options}
```

Description

This utility can be used to debug changelog and changenumber files.

Options

The `changelogstat` command takes the following options:

`-V | --version`

Display Directory Server version information. Default: false

`-H | --help`

Display this usage information. Default: false

Subcommands

The `changelogstat` command supports the following subcommands:

changelogstat dump-change-number-db

```
changelogstat dump-change-number-db {options}
```

Dump the change number DB.

Options

In addition to the global `changelogstat` options, the `changelogstat dump-change-number-db` subcommand takes the following options:

`--from {change number}`

The lower bound of the range of change numbers to dump.

`--outputDir {directory}`

The output directory for the dump files.

`--to {change number}`

The upper bound of the range of change numbers to dump.

changelogstat dump-replica-db

```
changelogstat dump-replica-db {options} baseDN replicaID
```

Dump the replica DB for a given domain and replica.

Options

In addition to the global `changelogstat` options, the `changelogstat dump-replica-db` subcommand takes the following options:

`--from {csn}`

The lower bound of the range of changes to dump.

`--outputDir {directory}`

The output directory for the dump files.

`--to {csn}`

The upper bound of the range of changes to dump.

changelogstat dump-replica-db-file

```
changelogstat dump-replica-db-file {options} file
```

Dump a replica DB file.

Options

In addition to the global `changelogstat` options, the `changelogstat dump-replica-db-file` subcommand takes the following options:

`--baseDn {base dn}`

The base-dn of the changes contained in the provided replica DB file. Default:

`--from {csn}`

The lower bound of the range of changes to dump.

`--to {csn}`

The upper bound of the range of changes to dump.

Exit codes

0

The command completed successfully.

1

An error occurred.

Examples

To dump the records for change numbers 10 to 15:

```
$ changelogstat dump-change-number-db --from 10 --to 15
{ "changeNumber": 10, "baseDn": "dc=example,dc=com", "csn": { "value": "010f017f87c4135d00025abdevaluation-
only", "serverId": "evaluation-only", "timestamp": "Mon Mar 14 10:30:48 CET 2022", "seqnum": 154301 } }
{ "changeNumber": 11, "baseDn": "dc=example,dc=com", "csn": { "value": "010f017f87c4135d00025abeevaluation-
only", "serverId": "evaluation-only", "timestamp": "Mon Mar 14 10:30:48 CET 2022", "seqnum": 154302 } }
{ "changeNumber": 12, "baseDn": "dc=example,dc=com", "csn": { "value": "010f017f87c4135d00025abfevaluation-
only", "serverId": "evaluation-only", "timestamp": "Mon Mar 14 10:30:48 CET 2022", "seqnum": 154303 } }
{ "changeNumber": 13, "baseDn": "dc=example,dc=com", "csn": { "value": "010f017f87c4135d00025ac0evaluation-
only", "serverId": "evaluation-only", "timestamp": "Mon Mar 14 10:30:48 CET 2022", "seqnum": 154304 } }
{ "changeNumber": 14, "baseDn": "dc=example,dc=com", "csn": { "value": "010f017f87c4135d00025ac1evaluation-
only", "serverId": "evaluation-only", "timestamp": "Mon Mar 14 10:30:48 CET 2022", "seqnum": 154305 } }
{ "changeNumber": 15, "baseDn": "dc=example,dc=com", "csn": { "value": "010f017f87c4135d00025ac2evaluation-
only", "serverId": "evaluation-only", "timestamp": "Mon Mar 14 10:30:48 CET 2022", "seqnum": 154306 } }
```

To dump the replica DB for the domain `dc=example,dc=com` and the server with ID `ds-1` :

```
$ changelogstat dump-replica-db --outputDir myOutputDir dc=example,dc=com ds-1
```

To dump a specific replica DB file:

```
$ changelogstat dump-replica-db-file changelogDb/2.dom/1.server/01010166aaf2a3e3000002bd1.cdb

{ "msgType": "ModifyMsg", "dn": "uid=user.84614,ou=people", "csn": "010f017f87c42baa0002f04fevaluation-only",
"uniqueId": "83719220-5de4-3271-a2a1-49f719778533" }
{ "msgType": "ModifyMsg", "dn": "uid=user.67749,ou=people", "csn": "010f017f87c42baa0002f050evaluation-only",
"uniqueId": "981f226e-5dff-35b3-b95f-6cfd582633ab" }
{ "msgType": "ModifyMsg", "dn": "uid=user.15128,ou=people", "csn": "010f017f87c42baa0002f051evaluation-only",
"uniqueId": "d0146ad4-ae04-3c93-b0e1-92c627f0bdae" }
{ "msgType": "ModifyMsg", "dn": "uid=user.56721,ou=people", "csn": "010f017f87c42baa0002f052evaluation-only",
"uniqueId": "3a578584-5e9d-3835-a7d4-1f5c78e41325" }
...
{ "msgType": "ModifyMsg", "dn": "uid=user.58621,ou=people", "csn": "010f017f87c439c900035566evaluation-only",
"uniqueId": "0281f279-b441-3018-9036-f6f97bf3903a" }
{ "msgType": "ModifyMsg", "dn": "uid=user.6745,ou=people", "csn": "010f017f87c439c900035567evaluation-only",
"uniqueId": "90853018-3abb-3e88-9fb2-0477919c067d" }
{ "msgType": "ModifyMsg", "dn": "uid=user.28215,ou=people", "csn": "010f017f87c439c900035568evaluation-only",
"uniqueId": "abfe1a55-5c64-36e8-8714-7d6e1f6d67aa" }
{ "msgType": "ModifyMsg", "dn": "uid=user.86811,ou=people", "csn": "010f017f87c439c900035569evaluation-only",
"uniqueId": "0810f7af-94ea-3f34-a455-c22432ad9429" }
```

create-rc-script

`create-rc-script` — script to manage OpenDJ as a service on UNIX

Synopsis

```
create-rc-script {options}
```

Description

Create an RC script that may be used to start, stop, and restart the Directory Server on UNIX-based systems.

Options

The `create-rc-script` command takes the following options:

Command options:

`-f | --outputFile {path}`

The path to the output file to create.

`-j | --javaHome {path}`

The path to the Java installation that should be used to run the server.

`-J | --javaArgs {args}`

A set of arguments that should be passed to the JVM when running the server.

`-s | --systemdService {path}`

The path to the systemd service file to create.

`-u | --userName {userName}`

The name of the user account under which the server should run.

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.

> 0

An error occurred.

dsbackup

`dsbackup` — Backup and restore backends

Synopsis

```
dsbackup {subcommand} {options}
```

Description

Backup and restore backends, manage backup files.

Options

The `dsbackup` command takes the following options:

Utility input/output options:

`--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

`--noPropertiesFile`

No properties file will be used to get default command line argument values. Default: false

`--propertiesFilePath {propertiesFilePath}`

Path to the file containing default property values used for command line arguments.

General options:

`-V | --version`

Display Directory Server version information. Default: false

`-H | --help`

Display this usage information. Default: false

Subcommands

The `dsbackup` command supports the following subcommands:

dsbackup create

```
dsbackup create {options}
```

Take encrypted and signed backups of individual backends and send them to the desired location.

Options

In addition to the global `dsbackup` options, the `dsbackup create` subcommand takes the following options:

SubCommand Options:

-d | --backupLocation {backup location}

Backup file-system path or URI for alternative storage mechanisms. File-system paths may be expressed as absolute or relative paths and are resolved relative to the current working directory when the tool is run in offline mode, or relative to the server instance directory when the tool is run in task mode. Read the documentation for further information regarding alternative backup storage mechanisms.

-n | --backendName {backendName}

The name of the backend to back up. Specify this option multiple times to backup multiple backends or skip this option to backup all the enabled backends that support backups.

--offline

Indicates that the command will operate independently of the server process. It will run regardless of whether the server is started or stopped. When using this option with the restore sub-command, the server must be stopped; also as the command will write to server files, you should run the command as a user having the same filesystem permissions as the user running the server. Using this option with the create sub-command when the server is running is possible and supported. With JE Backends, the integrity of the backup is ensured by the process. With LDIF backends, avoid simultaneous changes to the backends. Default: false

--storageProperty {PROP:VALUE}

Assigns a value to a storage property where PROP is the name of the property and VALUE is the single value to be assigned. Specify the same property multiple times in order to assign more than one value to it.

Task Scheduling Options

--completionNotify {emailAddress}

Email address of a recipient to be notified when the task completes. This option may be specified more than once.

--dependency {taskID}

ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution.

--description {description}

Gives a description to the task.

--errorNotify {emailAddress}

Email address of a recipient to be notified if an error occurs when this task executes. This option may be specified more than once.

--failedDependencyAction {action}

Action this task will take should one if its dependent tasks fail. The value must be one of PROCESS, CANCEL, DISABLE. If not specified defaults to CANCEL.

--recurringTask {schedulePattern}

Indicates the task is recurring and will be scheduled according to the value argument expressed in crontab(5) compatible time/date pattern. The schedule pattern for a recurring task supports only the following `crontab` features:

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names)
day of week	0-7 (0 or 7 is Sunday, or use names)

A field can contain an asterisk, `*` . An asterisk stands for `first-last` .

Fields can include ranges of numbers. A range is two numbers separated by a hyphen, and is inclusive. For example, `8-10` for an "hour" field means execution at hours 8, 9, and 10.

Fields can include lists. A list is a set of numbers or ranges separated by commas. For example, `4,8-10` for an "hour" field means execution at hours 4, 8, 9, and 10.

When using names for in "month" or "day of week" fields, use the first three letters of the particular month or day of the week. Case does not matter. Ranges and lists of names are not supported.

-t | --start {startTime}

Indicates the date/time at which this operation will start when scheduled as a server task expressed in YYYYMMDDhhmmssZ format for UTC time or YYYYMMDDhhmmss for local time. A value of '0' will cause the task to be scheduled for immediate execution. When this option is specified the operation will be scheduled to start at the specified time after which this utility will exit immediately.

--taskId {taskID}

Gives an ID to the task.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default: uid=admin

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

dsbackup list

`dsbackup list {options}`

List the backups at the specified location.

Options

In addition to the global `dsbackup` options, the `dsbackup list` subcommand takes the following options:

-d | --backupLocation {backup location}

Location containing backups: file-system path or URI for alternative storage mechanisms. File-system paths may be expressed as absolute or relative paths and are resolved relative to the current working directory when the tool is run in offline mode, or relative to the server instance directory when the tool is run in task mode. Read the documentation for further information regarding alternative backup storage mechanisms.

--last

Show only the last backup for each backend. Default: false

-n | --backendName {backendName}

Show only backups taken from the provided backend.

--serverId {server ID}

Show only backups taken from the provided server.

--storageProperty {PROP:VALUE}

Assigns a value to a storage property where PROP is the name of the property and VALUE is the single value to be assigned. Specify the same property multiple times in order to assign more than one value to it.

--verify

Verify backups completeness, integrity and whether they can be decrypted. Default: false

dsbackup purge

`dsbackup purge {options}`

Delete one or more backups.

Options

In addition to the global `dsbackup` options, the `dsbackup purge` subcommand takes the following options:

SubCommand Options:

`--backupId {backup ID}`

The ID of the backup that should be deleted. Specify this option multiple times to purge multiple backups.

`-d | --backupLocation {backup location}`

Location containing backups: file-system path or URI for alternative storage mechanisms. File-system paths may be expressed as absolute or relative paths and are resolved relative to the current working directory when the tool is run in offline mode, or relative to the server instance directory when the tool is run in task mode. Read the documentation for further information regarding alternative backup storage mechanisms.

`--force`

Must be used with the '--olderThan' option, indicates that the last backup of each backend can be deleted if older than the provided duration. Default: false

`--keepCount {number of backups}`

The number of backups to keep per backend. Use this option to keep the n latest backups of each backend and delete the others. If n=0, all the backups will be removed.

`-n | --backendName {backend name}`

Purge only backups of the specified backend. Specify this option multiple times to allow purging backups of different backends. Skip this option to allow purging backups of all backends. This can only be used with options '--keepCount' or '--olderThan'.

`--offline`

Indicates that the command will operate independently of the server process. It will run regardless of whether the server is started or stopped. When using this option with the restore sub-command, the server must be stopped; also as the command will write to server files, you should run the command as a user having the same filesystem permissions as the user running the server. Using this option with the create sub-command when the server is running is possible and supported. With JE Backends, the integrity of the backup is ensured by the process. With LDIF backends, avoid simultaneous changes to the backends. Default: false

`--olderThan {duration}`

Delete backups that are older than the provided duration. The latest backup of each backend will always be kept unless the '--force' option is also provided. Duration examples: '12 hours', '3 days', '1y'.

`--storageProperty {PROP:VALUE}`

Assigns a value to a storage property where PROP is the name of the property and VALUE is the single value to be assigned. Specify the same property multiple times in order to assign more than one value to it.

Task Scheduling Options

`--completionNotify {emailAddress}`

Email address of a recipient to be notified when the task completes. This option may be specified more than once.

`--dependency {taskID}`

ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution.

`--description {description}`

Gives a description to the task.

`--errorNotify {emailAddress}`

Email address of a recipient to be notified if an error occurs when this task executes. This option may be specified more than once.

`--failedDependencyAction {action}`

Action this task will take should one if its dependent tasks fail. The value must be one of PROCESS, CANCEL, DISABLE. If not specified defaults to CANCEL.

`--recurringTask {schedulePattern}`

Indicates the task is recurring and will be scheduled according to the value argument expressed in crontab(5) compatible time/date pattern. The schedule pattern for a recurring task supports only the following `crontab` features:

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names)
day of week	0-7 (0 or 7 is Sunday, or use names)

A field can contain an asterisk, `*` . An asterisk stands for `first-last` .

Fields can include ranges of numbers. A range is two numbers separated by a hyphen, and is inclusive. For example, `8-10` for an "hour" field means execution at hours 8, 9, and 10.

Fields can include lists. A list is a set of numbers or ranges separated by commas. For example, `4,8-10` for an "hour" field means execution at hours 4, 8, 9, and 10.

When using names for in "month" or "day of week" fields, use the first three letters of the particular month or day of the week. Case does not matter. Ranges and lists of names are not supported.

-t | --start {startTime}

Indicates the date/time at which this operation will start when scheduled as a server task expressed in YYYYMMDDhhmmssZ format for UTC time or YYYYMMDDhhmmss for local time. A value of '0' will cause the task to be scheduled for immediate execution. When this option is specified the operation will be scheduled to start at the specified time after which this utility will exit immediately.

--taskId {taskID}

Gives an ID to the task.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default: uid=admin

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

dsbackup restore

`dsbackup restore {options}`

Restore one or more backends. In order to decrypt and verify signatures on backup files, the server must have access to the master key pair used to encrypt and sign the files when they were created.

Options

In addition to the global `dsbackup` options, the `dsbackup restore` subcommand takes the following options:

SubCommand Options:

--backupId {backup ID}

Restore the backup having the provided ID. Specify this option multiple times to restore multiple backends.

-d | --backupLocation {backup location}

Location containing backups: file-system path or URI for alternative storage mechanisms. File-system paths may be expressed as absolute or relative paths and are resolved relative to the current working directory when the tool is run in offline mode, or relative to the server instance directory when the tool is run in task mode. Read the documentation for further information regarding alternative backup storage mechanisms.

-n | --backendName {backendName}

Restore the last backup of the provided backend. Specify this option multiple times to restore multiple backends.

--offline

Indicates that the command will operate independently of the server process. It will run regardless of whether the server is started or stopped. When using this option with the restore sub-command, the server must be stopped; also as the command will write to server files, you should run the command as a user having the same filesystem permissions as the user running the server. Using this option with the create sub-command when the server is running is possible and supported. With JE Backends, the integrity of the backup is ensured by the process. With LDIF backends, avoid simultaneous changes to the backends. Default: false

--storageProperty {PROP:VALUE}

Assigns a value to a storage property where PROP is the name of the property and VALUE is the single value to be assigned. Specify the same property multiple times in order to assign more than one value to it.

Task Scheduling Options

--completionNotify {emailAddress}

Email address of a recipient to be notified when the task completes. This option may be specified more than once.

--dependency {taskID}

ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution.

--description {description}

Gives a description to the task.

--errorNotify {emailAddress}

Email address of a recipient to be notified if an error occurs when this task executes. This option may be specified more than once.

--failedDependencyAction {action}

Action this task will take should one if its dependent tasks fail. The value must be one of PROCESS, CANCEL, DISABLE. If not specified defaults to CANCEL.

--recurringTask {schedulePattern}

Indicates the task is recurring and will be scheduled according to the value argument expressed in crontab(5) compatible time/date pattern. The schedule pattern for a recurring task supports only the following `crontab` features:

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names)
day of week	0-7 (0 or 7 is Sunday, or use names)

A field can contain an asterisk, `*` . An asterisk stands for `first-last` .

Fields can include ranges of numbers. A range is two numbers separated by a hyphen, and is inclusive. For example, `8-10` for an "hour" field means execution at hours 8, 9, and 10.

Fields can include lists. A list is a set of numbers or ranges separated by commas. For example, `4,8-10` for an "hour" field means execution at hours 4, 8, 9, and 10.

When using names for in "month" or "day of week" fields, use the first three letters of the particular month or day of the week. Case does not matter. Ranges and lists of names are not supported.

-t | --start {startTime}

Indicates the date/time at which this operation will start when scheduled as a server task expressed in YYYYMMDDhhmmssZ format for UTC time or YYYYMMDDhhmmss for local time. A value of '0' will cause the task to be scheduled for immediate execution. When this option is specified the operation will be scheduled to start at the specified time after which this utility will exit immediately.

--taskId {taskID}

Gives an ID to the task.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default: uid=admin

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

Exit codes

0

The command completed successfully.

> 0

An error occurred.

dsconfig

`dsconfig` — manage OpenDJ server configuration

Synopsis

```
dsconfig {subcommand} {options}
```

Description

This utility can be used to define a base configuration for the Directory Server.

The `dsconfig` command is the primary command-line tool for viewing and editing the server configuration. When started without arguments, `dsconfig` prompts you for administration connection information, including the host name, administration port number, administrator bind DN and administrator password. The `dsconfig` command then connects securely to the directory server over the administration port. Once connected it presents you with a menu-driven interface to the server configuration.

When you pass connection information, subcommands, and additional options to `dsconfig` , the command runs in script mode and so is not interactive, though it can prompt you to ask whether to apply changes and whether to trust certificates (unless you use the `--no-prompt` and `--trustAll` options, respectively).

You can prepare `dsconfig` batch scripts by running the tool with the `--commandFilePath` option in interactive mode, then reading from the batch file with the `--batchFilePath` option in script mode. Batch files can be useful when you have many `dsconfig` commands to run and want to avoid starting the JVM for each command. Alternatively, you can read commands from standard input by using the `--batch` option.

The `dsconfig` command categorizes directory server configuration into *components* , also called *managed objects* . Actual components often inherit from a parent component type. For example, one component is a Connection Handler. An LDAP Connection Handler is a type of Connection Handler. You configure the LDAP Connection Handler component to specify how the server handles LDAP connections coming from client applications.

Configuration components have *properties* . For example, the LDAP Connection Handler component has properties such as `listen-port` and `allow-start-tls` . You can set the component's `listen-port` property to `389` to use the default LDAP port number. You can set the component's `allow-start-tls` property to `true` to permit LDAP client applications to use StartTLS. Much of the configuration you do with `dsconfig` involves setting component properties.

Options

The `dsconfig` command takes the following options:

Command options:

`--batch`

Reads from standard input a set of commands to be executed. Default: false

`--commandFilePath {path}`

The full path to the file where the equivalent non-interactive commands will be written when this command is run in interactive mode.

`--configFile {configFile}`

Path to the Directory Server configuration file. Default: /path/to/opendj/config/config.ldif

`--help-all`

Display all subcommands. Default: false

`--help-core-server`

Display subcommands relating to core server. Default: false

`--help-database`

Display subcommands relating to caching and backends. Default: false

`--help-logging`

Display subcommands relating to logging. Default: false

`--help-proxy`

Display subcommands relating to directory proxy. Default: false

`--help-replication`

Display subcommands relating to replication. Default: false

--help-security

Display subcommands relating to authentication and authorization. Default: false

--help-service-discovery

Display subcommands relating to service discovery mechanism. Default: false

--help-user-management

Display subcommands relating to user management. Default: false

--offline

Indicates that the command must be run in offline mode. Default: false

Configuration Options

--advanced

Allows the configuration of advanced components and properties. Default: false

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default: uid=admin

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

Utility input/output options:

-F | --batchFilePath {batchFilePath}

Path to a batch file containing a set of commands to be executed.

-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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-Q | --quiet

Use quiet mode. Default: false

-s | --script-friendly

Use script-friendly 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

Subcommands

The `dsconfig` command provides many subcommands.

Subcommands let you create, list, and delete entire configuration components, and get and set component properties. Subcommands have names that reflect these five actions:

- create- *component*
- list- *component* s
- delete- *component*
- get- *component* -prop
- set- *component* -prop

Here, *component* names are names of managed object types. Subcommand *component* names are lower-case, hyphenated versions of the friendly names. When you act on an actual configuration component, you provide the name of the component as an option argument.

For example, the Log Publisher component has these corresponding subcommands.

- `create-log-publisher`
- `list-log-publishers`
- `delete-log-publisher`
- `get-log-publisher-prop`
- `set-log-publisher-prop`

When you create or delete Log Publisher components and when you get and set their configuration properties, you provide the name of the actual log publisher, which you can find by using the `list-log-publishers` subcommand:

```
# Get the log publishers' names:
$ dsconfig \
 list-log-publishers \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
Log Publisher                       : Type                   : enabled
------------------------------------:------------------------:--------
...
Json File-Based Access Logger       : json-file-access       : true
...

# Use the name to read a property:
$ dsconfig \
 get-log-publisher-prop \
 --publisher-name "Json File-Based Access Logger" \
 --property rotation-policy \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
Property        : Value(s)
----------------:-----------------------------------------------------------
rotation-policy : 24 Hours Time Limit Rotation Policy, Size Limit Rotation
                : Policy
```

Many subcommands let you set property values. Notice in the reference for the subcommands below that specific options are available for handling multi-valued properties. Whereas you can assign a single property value by using the `--set` option, you assign multiple values to a multi-valued property by using the `--add` option. You can reset the values of the multi-valued property by using the `--reset` option.

Some property values take a time duration. Durations are expressed as numbers followed by units. For example `1 s` means one second, and `2 w` means two weeks. Some durations have minimum granularity or maximum units, so you cannot necessary specify every duration in milliseconds or weeks for example. Some durations allow you to use a special value to mean unlimited. Units are specified as follows.

- `ms` : milliseconds
- `s` : seconds
- `m` : minutes
- `h` : hours
- `d` : days
- `w` : weeks
- `y` : years

Use the `--help*` options described above to view help for subcommands.

For help with individual subcommands, either use `dsconfig subcommand --help` , or start `dsconfig` in interactive mode, without specifying a subcommand.

To view all component properties, use the `dsconfig list-properties` command.

Exit codes

0

The command completed successfully.

> 0

An error occurred.

Examples

The following example starts the `dsconfig` command in interactive, menu-driven mode:

```
$ dsconfig \
 --hostname localhost \
 --port 4444 \
 --bindDn "uid=admin" \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin

>>>> OpenDJ configuration console main menu

What do you want to configure?

    1)   Access Control Handler               21)  Log Publisher
    2)   Access Log Filtering Criteria        22)  Log Retention Policy
    3)   Account Status Notification Handler  23)  Log Rotation Policy
```

```
    4)   Administration Connector            24)  Mail Server
    5)   Alert Handler                       25)  Password Generator
    6)   Backend                             26)  Password Policy
    7)   Backend Index                       27)  Password Storage Scheme
    8)   Backend VLV Index                   28)  Password Validator
    9)   Certificate Mapper                  29)  Plugin
    10)  Connection Handler                  30)  Plugin Root
    11)  Crypto Manager                      31)  Replication Domain
    12)  Debug Target                        32)  Replication Server
    13)  Entry Cache                         33)  Root DSE Backend
    14)  Extended Operation Handler          34)  SASL Mechanism Handler
    15)  Global Access Control Policy        35)  Schema Provider
    16)  Global Configuration                36)  Service Discovery Mechanism
    17)  HTTP Authorization Mechanism        37)  Synchronization Provider
    18)  HTTP Endpoint                       38)  Trust Manager Provider
    19)  Identity Mapper                     39)  Virtual Attribute
    20)  Key Manager Provider                40)  Work Queue

    a)   show advanced components and properties
    q)   quit

Enter choice:
```

Use the interactive mode to discover the commands that you can reuse to script configuration changes. When you apply a change in interactive mode, the `dsconfig` command displays the corresponding command.

When the server is stopped, you can run the commands offline, and batch them together. The following example sets global properties, and creates a logger that writes messages to the console:

```
dsconfig --offline --no-prompt --batch << END_OF_COMMAND_INPUT
set-global-configuration-prop --set "server-id:&{ds.server.id|evaluation-only}"
set-global-configuration-prop --set "group-id:&{ds.group.id|default}"
set-global-configuration-prop --set "advertised-listen-address:&{ds.advertised.listen.address|localhost}"
create-log-publisher --type console-error --publisher-name "Console Error Logger" --set enabled:true
END_OF_COMMAND_INPUT
```

dskeymgr

`dskeymgr` — manage public key infrastructure in private deployments

Synopsis

```
dskeymgr {subcommand} {options}
```

Description

This utility can be used for provisioning and managing TLS certificates for use in private deployments.

Subcommands easily allow to:

- Create a deployment CA certificate
- Distribute the CA certificate to all deployed applications

- Provision each application with a TLS key pair signed by the deployment CA
- Rotate the TLS key pairs

Subcommands take several seconds to run because the tool uses a computationally expensive algorithm for hashing the deployment ID password.

Options

The `dskeymgr` command takes the following options:

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

General options:

`-V | --version`

Display Directory Server version information. Default: false

`-H | --help`

Display this usage information. Default: false

Subcommands

The `dskeymgr` command supports the following subcommands:

dskeymgr create-deployment-id

`dskeymgr create-deployment-id {options}`

Creates a new deployment ID.

Options

In addition to the global `dskeymgr` options, the `dskeymgr create-deployment-id` subcommand takes the following options:

`-f | --outputFile {outputFile}`

Optional path to a file where the deployment ID will be written, overwriting the file if it already exists.

`-v | --validity {validity}`

The duration for which the CA certificate associated with the deployment ID will be valid. Examples: '20years', '1days'. Default: 10 y

`-w | --deploymentIdPassword[:env|:file] {deploymentIdPassword}`

The deployment ID password.

dskeymgr create-tls-key-pair

`dskeymgr create-tls-key-pair {options}`

Creates a TLS key-pair signed by the CA associated with a deployment ID and exports it to a keystore or as a PEM file.

Options

In addition to the global `dskeymgr` options, the `dskeymgr create-tls-key-pair` subcommand takes the following options:

`-a | --alias {alias}`

The TLS key-pair alias, any entry with the same alias will be overwritten. Default: ssl-key-pair

`-f | --outputFile {pemFile}`

Optional path to a file with a .pem extension. The command writes the key(s) to the file in PEM format, overwriting the file if it exists.

`-h | --hostname {hostname}`

The hostname(s) that will be added to the TLS certificate alternative name extension. Multiple hostnames may be given by providing this argument multiple times. Hostnames can start with a wildcard. Default: localhost

`-k | --deploymentId {deploymentId}`

The deployment ID.

`-K | --keyStoreFile {keyStoreFile}`

Optional path to an existing PKCS12 keystore file or a path indicating where a new keystore file should be created.

`-r | --writableReplica`

Indicates that the server using the certificate is specifically allowed to send updates to other servers. Default: false

`-s | --subjectDn {subjectDn}`

The TLS certificate subject DN.

`-v | --validity {validity}`

The duration for which the TLS certificate will be valid. Examples: '1days', '12hours', '1d 12h'. Default: 1 y

`-w | --deploymentIdPassword[:env|:file] {deploymentIdPassword}`

The deployment ID password.

`-W | --keyStorePassword[:env|:file] {keyStorePassword}`

Keystore password which will be used as the cleartext configuration value.

dskeymgr export-ca-cert

`dskeymgr export-ca-cert {options}`

Exports the CA certificate associated with a deployment ID to a keystore or as a PEM file.

Options

In addition to the global `dskeymgr` options, the `dskeymgr export-ca-cert` subcommand takes the following options:

`-a | --alias {alias}`

The CA certificate alias, must not already exist in the keystore. Default: ca-cert

`-f | --outputFile {pemFile}`

Optional path to a file with a .pem extension. The command writes the key(s) to the file in PEM format, overwriting the file if it exists.

`-k | --deploymentId {deploymentId}`

The deployment ID.

`-K | --keyStoreFile {keyStoreFile}`

Optional path to an existing PKCS12 keystore file or a path indicating where a new keystore file should be created.

`-w | --deploymentIdPassword[:env|:file] {deploymentIdPassword}`

The deployment ID password.

`-W | --keyStorePassword[:env|:file] {keyStorePassword}`

Keystore password which will be used as the cleartext configuration value.

dskeymgr export-master-key-pair

`dskeymgr export-master-key-pair {options}`

Exports the master key pair associated with a deployment ID to a keystore or as a PEM file.

Options

In addition to the global `dskeymgr` options, the `dskeymgr export-master-key-pair` subcommand takes the following options:

`-a | --alias {alias}`

The master key pair alias, must not already exist in the keystore. Default: master-key

`-f | --outputFile {pemFile}`

Optional path to a file with a .pem extension. The command writes the key(s) to the file in PEM format, overwriting the file if it exists.

`-k | --deploymentId {deploymentId}`

The deployment ID.

-K | --keyStoreFile {keyStoreFile}

Optional path to an existing PKCS12 keystore file or a path indicating where a new keystore file should be created.

-w | --deploymentIdPassword[:env|:file] {deploymentIdPassword}

The deployment ID password.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

dskeymgr show-deployment-id

`dskeymgr show-deployment-id deployment-id`

Displays the deployment ID details.

Exit codes

0

The command completed successfully.

> 0

An error occurred.

Examples

The following example shows how to create a deployment ID for managing the public key infrastructure of a private deployment:

```
$ dskeymgr \
 create-deployment-id \
 --deploymentIdPassword password \
 --validity "10 years"
AFPxL0RlmdMZHeVkkcC3GYFsAHNlNQ5CBVN1bkVDM7FyW2gWxnvQdQ
```

The following examples show how to use a deployment ID to obtain the deployment CA certificate:

- Export the CA certificate to a file in PEM format:

```
$ dskeymgr \
 export-ca-cert \
 --deploymentId AFPxL0RlmdMZHeVkkcC3GYFsAHNlNQ5CBVN1bkVDM7FyW2gWxnvQdQ \
 --deploymentIdPassword password \
 > ca.pem
```

- Export the CA certificate to a PKCS#12 truststore, creating the truststore if it does not exist:

```
$ dskeymgr \
 export-ca-cert \
 --deploymentId AFPxL0RlmdMZHeVkkcC3GYFsAHNlNQ5CBVN1bkVDM7FyW2gWxnvQdQ \
 --deploymentIdPassword password \
 --keyStoreFile keystore \
 --keyStorePassword secret12 \
 --alias ca-cert
```

The following example shows how to use a deployment ID to generate a TLS key pair signed by the deployment CA certificate and add it to a PKCS#12 keystore, creating the keystore if the keystore file does not exist. In this example, the key pair must be used by an application hosted on `*.example.com` and the application's entry has the DN `cn=test account,cn=service` .

```
$ dskeymgr \
 create-tls-key-pair \
 --deploymentId AFPxL0RlmdMZHeVkkcC3GYFsAHNlNQ5CBVN1bkVDM7FyW2gWxnvQdQ \
 --deploymentIdPassword password \
 --subjectDn "cn=test account,cn=service" \
 --hostname "*.example.com" \
 --validity "1 days" \
 --keyStoreFile keystore \
 --keyStorePassword secret12 \
 --alias tls-key-pair
```

In the example above, the key pair is only valid for one day. When it is about to expire, run the same command again to replace the old key pair having the alias `tls-key-pair` with a new one.

The following example shows how to display important information about a deployment ID:

```
$ dskeymgr show-deployment-id AFPxL0RlmdMZHeVkkcC3GYFsAHNlNQ5CBVN1bkVDM7FyW2gWxnvQdQ

Not before: 2019-06-27T12:42:29Z
Not after: 2029-06-24T12:42:29Z
Version: 0
Serial number: 33B1725B6816C67BD075
Provider name: SunEC
```

dsrepl

`dsrepl` — Manages data synchronization between servers

Synopsis

`dsrepl {subcommand} {options}`

Description

This tool manages data synchronization between servers. For replication to work you must initialize the contents of one of the servers with the contents of the others using the 'initialize' subcommand.

Options

The `dsrepl` command takes the following options:

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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

General options:

-V | --version

Display Directory Server version information. Default: false

-H | --help

Display this usage information. Default: false

Subcommands

The `dsrepl` command supports the following subcommands:

dsrepl add-local-server-to-pre-7-0-topology

```
dsrepl add-local-server-to-pre-7-0-topology {options}
```

Adds the local server (with version 7.0 or more) to a topology with older server versions (prior to 7.0).

Options

In addition to the global `dsrepl` options, the `dsrepl add-local-server-to-pre-7-0-topology` subcommand takes the following options:

SubCommand Options:

-b | --baseDn {baseDN}

Base DN(s) to replicate.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default: cn=admin,cn=Administrators,cn=admin data

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

dsrepl cleanup-migrated-pre-7-0-topology

```
dsrepl cleanup-migrated-pre-7-0-topology {options}
```

Clean all the servers (with version 7.0 or more) that have been migrated from a topology of older servers (version prior to 7.0).

Options

In addition to the global `dsrepl` options, the `dsrepl cleanup-migrated-pre-7-0-topology` subcommand takes the following options:

SubCommand Options:

--bootstrapServer {serverSource}

Server ID of the server containing the source data.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default: uid=admin

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

dsrepl clear-changelog

`dsrepl clear-changelog`

Clears all replication server changelog data for the offline local server; the other replication servers in the topology will transfer any needed data when the server restarts.

dsrepl disaster-recovery

`dsrepl disaster-recovery {options}`

Performs disaster recovery on the local server. The subcommand has two forms.

The first form verifies each replica has the same data after recovery: on a replica, run

dsrepl disaster-recovery --baseDn dc=example,dc=com --generate-recovery-id

The command prints the identifier to use on all other servers with the --generated-id option:

dsrepl disaster-recovery --baseDn dc=example,dc=com --generated-id {identifier}

The second form uses an identifier you provide. It lets you automate the recovery process when you cannot use the first form. Do not use this form if the topology has standalone replication servers. With this form of the subcommand, you must ensure you recover each replica with the same data. Run the same subcommand on all servers.

Example:

dsrepl disaster-recovery --baseDn dc=example,dc=com --user-generated-id Recovery_Date_20240101

Read the documentation on disaster recovery carefully before using this command.

Options

In addition to the global `dsrepl` options, the `dsrepl disaster-recovery` subcommand takes the following options:

-b | --baseDn {baseDN}

Base DN of the domain to be recovered.

--generate-recovery-id

Generate a disaster recovery identifier during recovery. Use this for the first directory server in a replication topology with standalone RS servers. For all subsequent servers to recover, omit this option and use --generated-id {generatedRecoveryId} with the generated identifier. Default: false

`--generated-id {generatedRecoveryId}`

Use the disaster recovery identifier generated on the first server. You must use the same identifier for all servers involved in the same disaster recovery procedure.

`--user-generated-id {userGeneratedRecoveryId}`

Set the identifier for this recovery to {userGeneratedRecoveryId}, a string of your choice. Do not use this option if the replication topology has standalone RS servers. You must use the same identifier for all servers involved in the same disaster recovery procedure.

dsrepl initialize

```
dsrepl initialize {options}
```

Initialize replication data for the server.

Options

In addition to the global `dsrepl` options, the `dsrepl initialize` subcommand takes the following options:

SubCommand Options:

`-b | --baseDn {baseDN}`

Base DN(s) to use. Multiple base DNs can be provided by using this option multiple times.

`--fromServer {serverSource}`

Server ID of the server containing the source data.

`--toAllServers`

Initialize all the other servers in the topology. Default: false

`--toServer {serverToInitialize}`

Server ID of the server to be initialized.

LDAP connection options:

`--connectTimeout {timeout}`

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

`-D | --bindDn {bindDN}`

DN to use to bind to the server. Default: uid=admin

`-E | --reportAuthzId`

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

dsrepl purge-meta-data

`dsrepl purge-meta-data {options}`

Purges old replication meta-data from application data.

Options

In addition to the global `dsrepl` options, the `dsrepl purge-meta-data` subcommand takes the following options:

SubCommand Options:

-b | --baseDn {baseDN}

Base DN(s) to use. Multiple base DNs can be provided by using this option multiple times.

--completionNotify {emailAddress}

Email address of a recipient to be notified when the task completes. This option may be specified more than once.

--dependency {taskID}

ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution.

--description {description}

Gives a description to the task.

--errorNotify {emailAddress}

Email address of a recipient to be notified if an error occurs when this task executes. This option may be specified more than once.

--failedDependencyAction {action}

Action this task will take should one if its dependent tasks fail. The value must be one of PROCESS, CANCEL, DISABLE. If not specified defaults to CANCEL.

--maximumDuration {maximum duration in seconds}

Maximum duration of the command in seconds. Default: 3600

`--recurringTask {schedulePattern}`

Indicates the task is recurring and will be scheduled according to the value argument expressed in crontab(5) compatible time/date pattern. The schedule pattern for a recurring task supports only the following `crontab` features:

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names)
day of week	0-7 (0 or 7 is Sunday, or use names)

A field can contain an asterisk, `*` . An asterisk stands for `first-last` .

Fields can include ranges of numbers. A range is two numbers separated by a hyphen, and is inclusive. For example, `8-10` for an "hour" field means execution at hours 8, 9, and 10.

Fields can include lists. A list is a set of numbers or ranges separated by commas. For example, `4,8-10` for an "hour" field means execution at hours 4, 8, 9, and 10.

When using names for in "month" or "day of week" fields, use the first three letters of the particular month or day of the week. Case does not matter. Ranges and lists of names are not supported.

`-t | --start {startTime}`

Indicates the date/time at which this operation will start when scheduled as a server task expressed in YYYYMMDDhhmmssZ format for UTC time or YYYYMMDDhhmmss for local time. A value of '0' will cause the task to be scheduled for immediate execution. When this option is specified the operation will be scheduled to start at the specified time after which this utility will exit immediately.

`--taskId {taskID}`

Gives an ID to the task.

LDAP connection options:

`--connectTimeout {timeout}`

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

`-D | --bindDn {bindDN}`

DN to use to bind to the server. Default: uid=admin

`-E | --reportAuthzId`

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

dsrepl reset-change-number

`dsrepl reset-change-number {options}`

Re-synchronizes the change-log change number of the target server with the change-log change number of the source server.

Options

In addition to the global `dsrepl` options, the `dsrepl reset-change-number` subcommand takes the following options:

SubCommand Options:

--change-number {change number}

The change number to use as the basis for re-synchronization.

--sourceBindDn {bindDN}

DN to use to bind to the server. Default: uid=admin

--sourceBindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

--sourceHostname {host}

Directory server hostname or IP address. Default: localhost.localdomain

--sourcePort {port}

Directory server administration port number.

--targetBindDn {bindDN}

DN to use to bind to the server. Default: uid=admin

--targetBindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

--targetHostname {host}

Directory server hostname or IP address. Default: localhost.localdomain

--targetPort {port}

Directory server administration port number.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-E | --reportAuthzId

Use the authorization identity control. Default: false

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

dsrepl status

`dsrepl status {options}`

Displays the status of the replication service and various diagnostics about it. The information is derived from reading cn=monitor on all the servers in the replication topology. A server receives a LATE status when its replay delay exceeds five seconds.

Options

In addition to the global `dsrepl` options, the `dsrepl status` subcommand takes the following options:

SubCommand Options:

-b | --baseDn {baseDN}

Base DN(s) to display. Multiple base DNs can be provided by using this option multiple times. If no base DNs are provided, then all the base DNs will be displayed.

--showChangelogs

Displays individual changelog servers in the output. Default: false

--showGroups

Display replication group information in the output. Default: false

--showReplicas

Displays individual replicas in the output. Default: false

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default: uid=monitor

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

Exit codes

0

The command completed successfully.

> 0

An error occurred.

encode-password

`encode-password` — encode a password with a storage scheme

Synopsis

```
encode-password {options}
```

Description

This utility can be used to encode user passwords with a specified storage scheme, or to determine whether a given clear-text value matches a provided encoded password.

Options

The `encode-password` command takes the following options:

Command options:

`-a | --authPasswordSyntax`

Use the authentication password syntax rather than the user password syntax. Default: false

`-c | --clearPassword[:env|:file] {clearPW}`

Clear-text password to encode or to compare against an encoded password.

`-e | --encodedPassword[:env|:file] {encodedPW}`

Encoded password to compare against the clear-text password.

`-i | --interactivePassword`

The password to encode or to compare against an encoded password is interactively asked to the user. Default: false

`-l | --listSchemes`

List available password storage schemes. Default: false

`-r | --useCompareResultCode`

Use the LDAP compare result as an exit code for the password comparison. Default: false

`-s | --storageScheme {scheme}`

Scheme to use for the encoded password.

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.

5

The `-r` option was used, and the compare did not match.

6

The `-r` option was used, and the compare did match.

other

An error occurred.

export-ldif

`export-ldif` — export directory data in LDIF

Synopsis

```
export-ldif {options}
```

Description

This utility can be used to export data from a Directory Server backend in LDIF form.

Options

The `export-ldif` command takes the following options:

Command options:

`-a | --appendToLdif`

Append an existing LDIF file rather than overwriting it. Default: false

`-b | --includeBranch {branchDN}`

Base DN of a branch to include in the LDIF export.

`-B | --excludeBranch {branchDN}`

Base DN of a branch to exclude from the LDIF export.

`-c | --compress`

Compress the LDIF data as it is exported. Default: false

`-e | --excludeAttribute {attribute}`

Attribute to exclude from the LDIF export.

`--excludeFilter {filter}`

Filter to identify entries to exclude from the LDIF export.

-i | --includeAttribute {attribute}

Attribute to include in the LDIF export.

--includeFilter {filter}

Filter to identify entries to include in the LDIF export.

-l | --ldifFile {ldifFile}

Path to the LDIF file to be written.

-n | --backendId {backendName}

Backend ID for the backend to export.

-O | --excludeOperational

Exclude operational attributes from the LDIF export. Default: false

--offline

Indicates that the command must be run in offline mode. Default: false

Task Scheduling Options

--completionNotify {emailAddress}

Email address of a recipient to be notified when the task completes. This option may be specified more than once.

--dependency {taskID}

ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution.

--description {description}

Gives a description to the task.

--errorNotify {emailAddress}

Email address of a recipient to be notified if an error occurs when this task executes. This option may be specified more than once.

--failedDependencyAction {action}

Action this task will take should one if its dependent tasks fail. The value must be one of PROCESS, CANCEL, DISABLE. If not specified defaults to CANCEL.

--recurringTask {schedulePattern}

Indicates the task is recurring and will be scheduled according to the value argument expressed in crontab(5) compatible time/date pattern. The schedule pattern for a recurring task supports only the following `crontab` features:

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names)
day of week	0-7 (0 or 7 is Sunday, or use names)

A field can contain an asterisk, `*` . An asterisk stands for `first-last` .

Fields can include ranges of numbers. A range is two numbers separated by a hyphen, and is inclusive. For example, `8-10` for an "hour" field means execution at hours 8, 9, and 10.

Fields can include lists. A list is a set of numbers or ranges separated by commas. For example, `4,8-10` for an "hour" field means execution at hours 4, 8, 9, and 10.

When using names for in "month" or "day of week" fields, use the first three letters of the particular month or day of the week. Case does not matter. Ranges and lists of names are not supported.

`-t | --start {startTime}`

Indicates the date/time at which this operation will start when scheduled as a server task expressed in YYYYMMDDhhmmssZ format for UTC time or YYYYMMDDhhmmss for local time. A value of '0' will cause the task to be scheduled for immediate execution. When this option is specified the operation will be scheduled to start at the specified time after which this utility will exit immediately.

`--taskId {taskID}`

Gives an ID to the task.

Task Backend Connection Options

`--connectTimeout {timeout}`

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

`-D | --bindDn {bindDN}`

DN to use to bind to the server. Default: uid=admin

`-E | --reportAuthzId`

Use the authorization identity control. Default: false

`-h | --hostname {host}`

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

Utility input/output options:

--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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

--wrapColumn {wrapColumn}

Column at which to wrap long lines (0 for no wrapping). Default: 0

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.

> 0

An error occurred.

import-ldif

`import-ldif` — import directory data from LDIF

Synopsis

```
import-ldif {options}
```

Description

This utility can be used to import LDIF data into a Directory Server backend, overwriting existing data. It cannot be used to append data to the backend database.

Options

The `import-ldif` command takes the following options:

Command options:

`-A | --templateFile {templateFile}`

Path to a MakeLDIF template to use to generate the import data.

`-b | --includeBranch {branchDN}`

Base DN of a branch to include in the LDIF import.

`-B | --excludeBranch {branchDN}`

Base DN of a branch to exclude from the LDIF import.

`-c | --isCompressed`

LDIF file is compressed. Default: false

`--countRejects`

Count the number of entries rejected by the server and return that value as the exit code (values > 255 will be reduced to 255 due to exit code restrictions). Default: false

`-e | --excludeAttribute {attribute}`

Attribute to exclude from the LDIF import.

`--excludeFilter {filter}`

Filter to identify entries to exclude from the LDIF import.

`-F | --clearBackend`

Remove all entries for all base DNs in the backend before importing. Default: false

`-i | --includeAttribute {attribute}`

Attribute to include in the LDIF import.

--includeFilter {filter}

Filter to identify entries to include in the LDIF import.

-l | --ldifFile {ldifFile}

Path to the LDIF file to be imported.

-n | --backendId {backendName}

Backend ID for the backend to import.

-O | --overwrite

Overwrite an existing rejects and/or skip file rather than appending to it. Default: false

--offline

Indicates that the command must be run in offline mode. When using this option, the command writes to server files. Run the command as a user having the same filesystem permissions as the user running the server. Default: false

-R | --rejectFile {rejectFile}

Write rejected entries to the specified file.

-s | --randomSeed {seed}

Seed for the MakeLDIF random number generator. To always generate the same data with the same command, use the same non-zero seed value. A value of zero (the default) results in different data each time the tool is run. Default: 0

-S | --skipSchemaValidation

Skip schema validation during the LDIF import. Default: false

--skipFile {skipFile}

Write skipped entries to the specified file.

--threadCount {count}

Number of threads used to read LDIF files during import. If 0, the number of threads will be set to twice the number of CPUs. Default: 0

--tmpDirectory {directory}

Path to temporary directory for index scratch files during LDIF import. Default: import-tmp

Task Scheduling Options

--completionNotify {emailAddress}

Email address of a recipient to be notified when the task completes. This option may be specified more than once.

`--dependency {taskID}`

ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution.

`--description {description}`

Gives a description to the task.

`--errorNotify {emailAddress}`

Email address of a recipient to be notified if an error occurs when this task executes. This option may be specified more than once.

`--failedDependencyAction {action}`

Action this task will take should one if its dependent tasks fail. The value must be one of PROCESS, CANCEL, DISABLE. If not specified defaults to CANCEL.

`--recurringTask {schedulePattern}`

Indicates the task is recurring and will be scheduled according to the value argument expressed in crontab(5) compatible time/date pattern. The schedule pattern for a recurring task supports only the following `crontab` features:

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names)
day of week	0-7 (0 or 7 is Sunday, or use names)

A field can contain an asterisk, `*` . An asterisk stands for `first-last` .

Fields can include ranges of numbers. A range is two numbers separated by a hyphen, and is inclusive. For example, `8-10` for an "hour" field means execution at hours 8, 9, and 10.

Fields can include lists. A list is a set of numbers or ranges separated by commas. For example, `4,8-10` for an "hour" field means execution at hours 4, 8, 9, and 10.

When using names for in "month" or "day of week" fields, use the first three letters of the particular month or day of the week. Case does not matter. Ranges and lists of names are not supported.

`-t | --start {startTime}`

Indicates the date/time at which this operation will start when scheduled as a server task expressed in YYYYMMDDhhmmssZ format for UTC time or YYYYMMDDhhmmss for local time. A value of '0' will cause the task to be scheduled for immediate execution. When this option is specified the operation will be scheduled to start at the specified time after which this utility will exit immediately.

--taskId {taskID}

Gives an ID to the task.

Task Backend Connection Options

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default: uid=admin

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

Utility input/output options:

--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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-Q | --quiet

Use quiet mode (no output). 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.

> 0

An error occurred.

ldapcompare

`ldapcompare` — perform LDAP compare operations

Synopsis

```
ldapcompare {options} attribute:value DN
```

Description

This utility can be used to perform LDAP compare operations in the Directory Server.

Options

The `ldapcompare` command takes the following options:

Command options:

--assertionFilter {filter}

Use the LDAP assertion control with the provided filter.

-J | --control {controloid[:criticality[:value|::b64value|:<filePath]]}

Use a request control with the provided information. For some *controloid* values, you can replace object identifiers with user-friendly strings. The values are not case-sensitive:

Assertion , LdapAssertion

Assertion Request Control, Object Identifier: 1.3.6.1.1.12

AccountUsable , AccountUsability

Account Usability Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.8

AuthzId, AuthorizationIdentity

Authorization Identity Request Control, Object Identifier: 2.16.840.1.113730.3.4.16

Csn, ChangeNumber, ChangeSequenceNumber

Change Sequence Number Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.9 This is an internal DS server control.

EffectiveRights, GetEffectiveRights

Get Effective Rights Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.2

ManageDsaIt

Manage DSAIT Request Control, Object Identifier: 2.16.840.1.113730.3.4.2

Noop, No-Op

No-Op Request Control, Object Identifier: 1.3.6.1.4.1.4203.1.10.2

PwdPolicy, PasswordPolicy

Password Policy Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.8.5.1

PasswordQualityAdvice

Password Quality Advice Request Control, Object Identifier: 1.3.6.1.4.1.36733.2.1.5.5

PermissiveModify

Permissive Modify Request Control, Object Identifier: 1.2.840.113556.1.4.1413

PSearch, PersistentSearch

Persistent Search Request Control, Object Identifier: 2.16.840.1.113730.3.4.3

PostRead

Post Read Request Control, Object Identifier: 1.3.6.1.1.13.2

PreRead

Pre Read Request Control, Object Identifier: 1.3.6.1.1.13.1

ProxiedAuthV1

Proxied Authorization Request Control V1, Object Identifier: 2.16.840.1.113730.3.4.12

ProxiedAuth, ProxiedAuthV2

Proxied Authorization Request Control V2, Object Identifier: 2.16.840.1.113730.3.4.18

RealAttrsOnly, RealAttributesOnly

Real Attributes Only Request Control, Object Identifier: 2.16.840.1.113730.3.4.17

RelaxRules

Relax Rules Request Control, Object Identifier: 1.3.6.1.4.1.4203.666.5.12

TreeDelete, SubTreeDelete

Subtree Delete Request Control, Object Identifier: 1.2.840.113556.1.4.805

Sort, ServerSideSort

Server Side Sort Request Control, Object Identifier: 1.2.840.113556.1.4.473

PagedResults, SimplePagedResults

Simple Paged Results Control, Object Identifier: 1.2.840.113556.1.4.319

SubEntries

Sub-Entries Request Control, Object Identifier: 1.3.6.1.4.1.4203.1.10.1

TxnId, TransactionId

Transaction ID Control, Object Identifier: 1.3.6.1.4.1.36733.2.1.5.1 This is an internal ForgeRock control.

VirtualAttrsOnly, VirtualAttributesOnly

Virtual Attributes Only Request Control, Object Identifier: 2.16.840.1.113730.3.4.19

Vlv, VirtualListView

Virtual List View Request Control, Object Identifier: 2.16.840.1.113730.3.4.9

-m | --useCompareResultCode

Use the LDAP compare result as an exit code for the LDAP compare operations. Default: false

-n | --dry-run

Show what would be done but do not perform any operation and do not contact the server. Default: false

-S | --scriptFriendly

Use script-friendly mode. Default: false

-Y | --proxyAs {authzID}

Use the proxied authorization control with the given authorization ID.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default:

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server port number.

-q | --useStartTls

Use StartTLS to secure communication with the server. Default: false

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

-Z | --useSsl

Use SSL for secure communication with the server. Default: false

Utility input/output options:

--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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-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.

5

The LDAP compare operation did not match.

6

The `-m` option was used, and the LDAP compare operation did match.

ldap-error

An LDAP error occurred while processing the operation. LDAP result codes are described in RFC 4511. Also see the additional information for details.

89

An error occurred while parsing the command-line arguments.

Files

You can use `~/.opendj/tools.properties` to set the defaults for bind DN, host name, and port number as in the following example:

```
hostname=directory.example.com
port=1389
bindDN=uid=kvaughan,ou=People,dc=example,dc=com

ldapcompare.port=1389
ldapdelete.port=1389
ldapmodify.port=1389
ldappasswordmodify.port=1389
ldapsearch.port=1389
```

ldapdelete

`ldapdelete` — perform LDAP delete operations

Synopsis

```
ldapdelete {options} [DN]
```

Description

This utility can be used to perform LDAP delete operations in the Directory Server.

If standard input is used to specify entries to remove, end your input with EOF (Ctrl+D on UNIX, Ctrl+Z on Windows).

Options

The `ldapdelete` command takes the following options:

Command options:

`-c | --continueOnError`

Continue processing even if there are errors. Default: false

`-J | --control {controloid[:criticality[:value|::b64value|:<filePath]]}`

Use a request control with the provided information. For some *controloid* values, you can replace object identifiers with user-friendly strings. The values are not case-sensitive:

`Assertion, LdapAssertion`

Assertion Request Control, Object Identifier: 1.3.6.1.1.12

`AccountUsable, AccountUsability`

Account Usability Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.8

`AuthzId, AuthorizationIdentity`

Authorization Identity Request Control, Object Identifier: 2.16.840.1.113730.3.4.16

`Csn, ChangeNumber, ChangeSequenceNumber`

Change Sequence Number Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.9 This is an internal DS server control.

`EffectiveRights, GetEffectiveRights`

Get Effective Rights Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.2

`ManageDsaIt`

Manage DSAIT Request Control, Object Identifier: 2.16.840.1.113730.3.4.2

`Noop, No-Op`

No-Op Request Control, Object Identifier: 1.3.6.1.4.1.4203.1.10.2

`PwdPolicy, PasswordPolicy`

Password Policy Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.8.5.1

`PasswordQualityAdvice`

Password Quality Advice Request Control, Object Identifier: 1.3.6.1.4.1.36733.2.1.5.5

`PermissiveModify`

Permissive Modify Request Control, Object Identifier: 1.2.840.113556.1.4.1413

PSearch, PersistentSearch

Persistent Search Request Control, Object Identifier: 2.16.840.1.113730.3.4.3

PostRead

Post Read Request Control, Object Identifier: 1.3.6.1.1.13.2

PreRead

Pre Read Request Control, Object Identifier: 1.3.6.1.1.13.1

ProxiedAuthV1

Proxied Authorization Request Control V1, Object Identifier: 2.16.840.1.113730.3.4.12

ProxiedAuth, ProxiedAuthV2

Proxied Authorization Request Control V2, Object Identifier: 2.16.840.1.113730.3.4.18

RealAttrsOnly, RealAttributesOnly

Real Attributes Only Request Control, Object Identifier: 2.16.840.1.113730.3.4.17

RelaxRules

Relax Rules Request Control, Object Identifier: 1.3.6.1.4.1.4203.666.5.12

TreeDelete, SubTreeDelete

Subtree Delete Request Control, Object Identifier: 1.2.840.113556.1.4.805

Sort, ServerSideSort

Server Side Sort Request Control, Object Identifier: 1.2.840.113556.1.4.473

PagedResults, SimplePagedResults

Simple Paged Results Control, Object Identifier: 1.2.840.113556.1.4.319

SubEntries

Sub-Entries Request Control, Object Identifier: 1.3.6.1.4.1.4203.1.10.1

TxnId, TransactionId

Transaction ID Control, Object Identifier: 1.3.6.1.4.1.36733.2.1.5.1 This is an internal ForgeRock control.

VirtualAttrsOnly, VirtualAttributesOnly

Virtual Attributes Only Request Control, Object Identifier: 2.16.840.1.113730.3.4.19

Vlv, VirtualListView

Virtual List View Request Control, Object Identifier: 2.16.840.1.113730.3.4.9

-n | --dry-run

Show what would be done but do not perform any operation and do not contact the server. Default: false

--numConnections {numConnections}

Number of connections. Default: 1

-x | --deleteSubtree

Delete the specified entry and all entries below it. Default: false

-Y | --proxyAs {authzID}

Use the proxied authorization control with the given authorization ID.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default:

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server port number.

-q | --useStartTls

Use StartTLS to secure communication with the server. Default: false

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

-Z | --useSsl

Use SSL for secure communication with the server. Default: false

Utility input/output options:

--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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-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.

ldap-error

An LDAP error occurred while processing the operation. LDAP result codes are described in RFC 4511. Also see the additional information for details.

89

An error occurred while parsing the command-line arguments.

Files

You can use `~/.opendj/tools.properties` to set the defaults for bind DN, host name, and port number as in the following example:

```
hostname=directory.example.com
port=1389
bindDN=uid=kvaughan,ou=People,dc=example,dc=com
```

```
ldapcompare.port=1389
ldapdelete.port=1389
ldapmodify.port=1389
ldappasswordmodify.port=1389
ldapsearch.port=1389
```

ldapmodify

`ldapmodify` — perform LDAP modify, add, delete, mod DN operations

Synopsis

```
ldapmodify {options} [changes_files …]
```

Description

This utility can be used to perform LDAP modify, add, delete, and modify DN operations in the Directory Server. When not using file(s) to specify modifications, end your input with EOF (Ctrl+D on UNIX, Ctrl+Z on Windows).

Options

The `ldapmodify` command takes the following options:

Command options:

`--assertionFilter {filter}`

Use the LDAP assertion control with the provided filter.

`-c | --continueOnError`

Continue processing even if there are errors. Default: false

`-J | --control {controloid[:criticality[:value|::b64value|:<filePath]]}`

Use a request control with the provided information. For some *controloid* values, you can replace object identifiers with user-friendly strings. The values are not case-sensitive:

`Assertion , LdapAssertion`

Assertion Request Control, Object Identifier: 1.3.6.1.1.12

`AccountUsable , AccountUsability`

Account Usability Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.8

`AuthzId , AuthorizationIdentity`

Authorization Identity Request Control, Object Identifier: 2.16.840.1.113730.3.4.16

Csn, ChangeNumber, ChangeSequenceNumber

Change Sequence Number Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.9 This is an internal DS server control.

EffectiveRights, GetEffectiveRights

Get Effective Rights Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.2

ManageDsaIt

Manage DSAIT Request Control, Object Identifier: 2.16.840.1.113730.3.4.2

Noop, No-Op

No-Op Request Control, Object Identifier: 1.3.6.1.4.1.4203.1.10.2

PwdPolicy, PasswordPolicy

Password Policy Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.8.5.1

PasswordQualityAdvice

Password Quality Advice Request Control, Object Identifier: 1.3.6.1.4.1.36733.2.1.5.5

PermissiveModify

Permissive Modify Request Control, Object Identifier: 1.2.840.113556.1.4.1413

PSearch, PersistentSearch

Persistent Search Request Control, Object Identifier: 2.16.840.1.113730.3.4.3

PostRead

Post Read Request Control, Object Identifier: 1.3.6.1.1.13.2

PreRead

Pre Read Request Control, Object Identifier: 1.3.6.1.1.13.1

ProxiedAuthV1

Proxied Authorization Request Control V1, Object Identifier: 2.16.840.1.113730.3.4.12

ProxiedAuth, ProxiedAuthV2

Proxied Authorization Request Control V2, Object Identifier: 2.16.840.1.113730.3.4.18

RealAttrsOnly, RealAttributesOnly

Real Attributes Only Request Control, Object Identifier: 2.16.840.1.113730.3.4.17

RelaxRules

Relax Rules Request Control, Object Identifier: 1.3.6.1.4.1.4203.666.5.12

TreeDelete, SubTreeDelete

Subtree Delete Request Control, Object Identifier: 1.2.840.113556.1.4.805

Sort, ServerSideSort

Server Side Sort Request Control, Object Identifier: 1.2.840.113556.1.4.473

PagedResults, SimplePagedResults

Simple Paged Results Control, Object Identifier: 1.2.840.113556.1.4.319

SubEntries

Sub-Entries Request Control, Object Identifier: 1.3.6.1.4.1.4203.1.10.1

TxnId, TransactionId

Transaction ID Control, Object Identifier: 1.3.6.1.4.1.36733.2.1.5.1 This is an internal ForgeRock control.

VirtualAttrsOnly, VirtualAttributesOnly

Virtual Attributes Only Request Control, Object Identifier: 2.16.840.1.113730.3.4.19

Vlv, VirtualListView

Virtual List View Request Control, Object Identifier: 2.16.840.1.113730.3.4.9

-n | --dry-run

Show what would be done but do not perform any operation and do not contact the server. Default: false

--numConnections {numConnections}

Number of connections. Default: 1

--postReadAttributes {attrList}

Use the LDAP ReadEntry post-read control.

--preReadAttributes {attrList}

Use the LDAP ReadEntry pre-read control.

-Y | --proxyAs {authzID}

Use the proxied authorization control with the given authorization ID.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default:

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server port number.

-q | --useStartTls

Use StartTLS to secure communication with the server. Default: false

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

-Z | --useSsl

Use SSL for secure communication with the server. Default: false

Utility input/output options:

--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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-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.

ldap-error

An LDAP error occurred while processing the operation. LDAP result codes are described in RFC 4511. Also see the additional information for details.

89

An error occurred while parsing the command-line arguments.

Files

You can use `~/.opendj/tools.properties` to set the defaults for bind DN, host name, and port number as in the following example:

```
hostname=directory.example.com
port=1389
bindDN=uid=kvaughan,ou=People,dc=example,dc=com

ldapcompare.port=1389
ldapdelete.port=1389
ldapmodify.port=1389
ldappasswordmodify.port=1389
ldapsearch.port=1389
```

ldappasswordmodify

`ldappasswordmodify` — perform LDAP password modifications

Synopsis

```
ldappasswordmodify {options}
```

Description

This utility can be used to perform LDAP password modify operations in the Directory Server.

Options

The `ldappasswordmodify` command takes the following options:

Command options:

-a | --authzId {authzID}

Authorization ID for the user entry whose password should be changed. The authorization ID is a string having either the prefix "dn:" followed by the user's distinguished name, or the prefix "u:" followed by a user identifier that depends on the identity mapping used to match the user identifier to an entry in the directory. Examples include "dn:uid=bjensen,ou=People,dc=example,dc=com", and, if we assume that "bjensen" is mapped to Barbara Jensen's entry, "u:bjensen".

-c | --currentPassword[:env|:file] {currentPassword}

Current password for the target user.

-J | --control {controloid[:criticality[:value|::b64value|:<filePath]]}

Use a request control with the provided information. For some *controloid* values, you can replace object identifiers with user-friendly strings. The values are not case-sensitive:

Assertion, LdapAssertion

Assertion Request Control, Object Identifier: 1.3.6.1.1.12

AccountUsable, AccountUsability

Account Usability Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.8

AuthzId, AuthorizationIdentity

Authorization Identity Request Control, Object Identifier: 2.16.840.1.113730.3.4.16

Csn, ChangeNumber, ChangeSequenceNumber

Change Sequence Number Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.9 This is an internal DS server control.

EffectiveRights, GetEffectiveRights

Get Effective Rights Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.2

ManageDsaIt

Manage DSAIT Request Control, Object Identifier: 2.16.840.1.113730.3.4.2

Noop, No-Op

No-Op Request Control, Object Identifier: 1.3.6.1.4.1.4203.1.10.2

PwdPolicy, PasswordPolicy

Password Policy Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.8.5.1

PasswordQualityAdvice

Password Quality Advice Request Control, Object Identifier: 1.3.6.1.4.1.36733.2.1.5.5

PermissiveModify

Permissive Modify Request Control, Object Identifier: 1.2.840.113556.1.4.1413

PSearch, PersistentSearch

Persistent Search Request Control, Object Identifier: 2.16.840.1.113730.3.4.3

PostRead

Post Read Request Control, Object Identifier: 1.3.6.1.1.13.2

PreRead

Pre Read Request Control, Object Identifier: 1.3.6.1.1.13.1

ProxiedAuthV1

Proxied Authorization Request Control V1, Object Identifier: 2.16.840.1.113730.3.4.12

ProxiedAuth, ProxiedAuthV2

Proxied Authorization Request Control V2, Object Identifier: 2.16.840.1.113730.3.4.18

RealAttrsOnly, RealAttributesOnly

Real Attributes Only Request Control, Object Identifier: 2.16.840.1.113730.3.4.17

RelaxRules

Relax Rules Request Control, Object Identifier: 1.3.6.1.4.1.4203.666.5.12

TreeDelete, SubTreeDelete

Subtree Delete Request Control, Object Identifier: 1.2.840.113556.1.4.805

Sort, ServerSideSort

Server Side Sort Request Control, Object Identifier: 1.2.840.113556.1.4.473

PagedResults, SimplePagedResults

Simple Paged Results Control, Object Identifier: 1.2.840.113556.1.4.319

SubEntries

Sub-Entries Request Control, Object Identifier: 1.3.6.1.4.1.4203.1.10.1

TxnId, TransactionId

Transaction ID Control, Object Identifier: 1.3.6.1.4.1.36733.2.1.5.1 This is an internal ForgeRock control.

VirtualAttrsOnly, VirtualAttributesOnly

Virtual Attributes Only Request Control, Object Identifier: 2.16.840.1.113730.3.4.19

Vlv, VirtualListView

Virtual List View Request Control, Object Identifier: 2.16.840.1.113730.3.4.9

-n | --newPassword[:env|:file] {newPassword}

New password to provide for the target user.

-Y | --proxyAs {authzID}

Use the proxied authorization control with the given authorization ID.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default:

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server port number.

-q | --useStartTls

Use StartTLS to secure communication with the server. Default: false

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

-Z | --useSsl

Use SSL for secure communication with the server. Default: false

Utility input/output options:

--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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-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.

ldap-error

An LDAP error occurred while processing the operation. LDAP result codes are described in RFC 4511. Also see the additional information for details.

89

An error occurred while parsing the command-line arguments.

Files

You can use `~/.opendj/tools.properties` to set the defaults for bind DN, host name, and port number as in the following example:

```
hostname=directory.example.com
port=1389
bindDN=uid=kvaughan,ou=People,dc=example,dc=com

ldapcompare.port=1389
ldapdelete.port=1389
ldapmodify.port=1389
ldappasswordmodify.port=1389
ldapsearch.port=1389
```

ldapsearch

`ldapsearch` — perform LDAP search operations

Synopsis

```
ldapsearch {options} filter [attributes …]
```

Description

This utility can be used to perform LDAP search operations in the Directory Server.

Options

The `ldapsearch` command takes the following options:

Command options:

`-a | --dereferencePolicy {dereferencePolicy}`

Alias dereference policy ('never', 'always', 'search', or 'find'). Default: never

`-A | --typesOnly`

Only retrieve attribute names but not their values. Default: false

`--assertionFilter {filter}`

Use the LDAP assertion control with the provided filter.

`-b | --baseDn {baseDN}`

Search base DN.

`-c | --continueOnError`

Continue processing even if there are errors. Default: false

`-C | --persistentSearch ps[:changetype[:changesonly[:entrychgcontrols]]]`

Use the persistent search control. A persistent search allows the client to continue receiving new results whenever changes are made to data that is in the scope of the search, thus using the search as a form of change notification.

The optional `changetype` setting defines the kinds of updates that result in notification. If you do not set the `changetype` , the default behavior is to send notifications for all updates.

`add`

Send notifications for LDAP add operations.

`del,delete`

Send notifications for LDAP delete operations.

`mod,modify`

Send notifications for LDAP modify operations.

`moddn,modrdn,modifydn`

Send notifications for LDAP modify DN (rename and move) operations.

`all,any`

Send notifications for all LDAP update operations.

The optional `changesonly` setting defines whether the server returns existing entries as well as changes.

`true`

Do not return existing entries, but instead only notifications about changes. This is the default setting.

`false`

Also return existing entries.

The optional `entrychgcontrols` setting defines whether the server returns an Entry Change Notification control with each entry notification. The Entry Change Notification control provides additional information about the change that caused the entry to be returned by the search. In particular, it indicates the change type, the change number if available, and the previous DN if the change type was a modify DN operation.

`true`

Do request the Entry Change Notification control. This is the default setting.

`false`

Do not request the Entry Change Notification control.

`--countEntries`

Count the number of entries returned by the server. Default: false

`-e | --getEffectiveRightsAttribute {attribute}`

Specifies geteffectiverights control specific attribute list.

`-g | --getEffectiveRightsAuthzId {authzID}`

Use geteffectiverights control with the provided authzid.

`-G | --virtualListView {before:after:index:count | before:after:value}`

Use the virtual list view control to retrieve the specified results page.

-J | --control {controloid[:criticality[:value|::b64value|:<filePath]]}

Use a request control with the provided information. For some *controloid* values, you can replace object identifiers with user-friendly strings. The values are not case-sensitive:

Assertion, LdapAssertion

Assertion Request Control, Object Identifier: 1.3.6.1.1.12

AccountUsable, AccountUsability

Account Usability Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.8

AuthzId, AuthorizationIdentity

Authorization Identity Request Control, Object Identifier: 2.16.840.1.113730.3.4.16

Csn, ChangeNumber, ChangeSequenceNumber

Change Sequence Number Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.9 This is an internal DS server control.

EffectiveRights, GetEffectiveRights

Get Effective Rights Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.9.5.2

ManageDsaIt

Manage DSAIT Request Control, Object Identifier: 2.16.840.1.113730.3.4.2

Noop, No-Op

No-Op Request Control, Object Identifier: 1.3.6.1.4.1.4203.1.10.2

PwdPolicy, PasswordPolicy

Password Policy Request Control, Object Identifier: 1.3.6.1.4.1.42.2.27.8.5.1

PasswordQualityAdvice

Password Quality Advice Request Control, Object Identifier: 1.3.6.1.4.1.36733.2.1.5.5

PermissiveModify

Permissive Modify Request Control, Object Identifier: 1.2.840.113556.1.4.1413

PSearch, PersistentSearch

Persistent Search Request Control, Object Identifier: 2.16.840.1.113730.3.4.3

PostRead

Post Read Request Control, Object Identifier: 1.3.6.1.1.13.2

PreRead

Pre Read Request Control, Object Identifier: 1.3.6.1.1.13.1

ProxiedAuthV1

Proxied Authorization Request Control V1, Object Identifier: 2.16.840.1.113730.3.4.12

ProxiedAuth , ProxiedAuthV2

Proxied Authorization Request Control V2, Object Identifier: 2.16.840.1.113730.3.4.18

RealAttrsOnly , RealAttributesOnly

Real Attributes Only Request Control, Object Identifier: 2.16.840.1.113730.3.4.17

RelaxRules

Relax Rules Request Control, Object Identifier: 1.3.6.1.4.1.4203.666.5.12

TreeDelete , SubTreeDelete

Subtree Delete Request Control, Object Identifier: 1.2.840.113556.1.4.805

Sort , ServerSideSort

Server Side Sort Request Control, Object Identifier: 1.2.840.113556.1.4.473

PagedResults , SimplePagedResults

Simple Paged Results Control, Object Identifier: 1.2.840.113556.1.4.319

SubEntries

Sub-Entries Request Control, Object Identifier: 1.3.6.1.4.1.4203.1.10.1

TxnId , TransactionId

Transaction ID Control, Object Identifier: 1.3.6.1.4.1.36733.2.1.5.1 This is an internal ForgeRock control.

VirtualAttrsOnly , VirtualAttributesOnly

Virtual Attributes Only Request Control, Object Identifier: 2.16.840.1.113730.3.4.19

Vlv , VirtualListView

Virtual List View Request Control, Object Identifier: 2.16.840.1.113730.3.4.9

-l | --timeLimit {timeLimit}

Maximum length of time in seconds to allow for the search. Default: 0

--matchedValuesFilter {filter}

Use the LDAP matched values control with the provided filter.

-n | --dry-run

Show what would be done but do not perform any operation and do not contact the server. Default: false

-s | --searchScope {searchScope}

Search scope ('base', 'one', 'sub', or 'subordinates'). Note: 'subordinates' is an LDAP extension that might not work with all LDAP servers. Default: sub

-S | --sortOrder {sortOrder}

Use the server side sort control to have the server sort the results using the provided sort order. You can provide multiple comma separated sort keys. Sort key must respect the following pattern: "[-] attributeType [:OrderingRuleNameOrOID]". Minus character represent a descending sort order.

--simplePageSize {numEntries}

Use the simple paged results control with the given page size. Default: 1000

--subEntries

Use subentries control to specify that subentries are visible and normal entries are not. Default: false

-Y | --proxyAs {authzID}

Use the proxied authorization control with the given authorization ID.

-z | --sizeLimit {sizeLimit}

Maximum number of entries to return from the search. Default: 0

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default:

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server port number.

-q | --useStartTls

Use StartTLS to secure communication with the server. Default: false

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

-Z | --useSsl

Use SSL for secure communication with the server. Default: false

Utility input/output options:

--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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-t | --wrapColumn {wrapColumn}

Maximum length of an output line (0 for no wrapping). Default: 0

-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

Filters

The filter argument is a string representation of an LDAP search filter as in `(cn=Babs Jensen)`, `(&(objectClass=Person)(|(sn=Jensen)(cn=Babs J*)))`, or `(cn:caseExactMatch:=Fred Flintstone)`.

Attributes

The optional attribute list specifies the attributes to return in the entries found by the search. In addition to identifying attributes by name such as `cn sn mail` and so forth, you can use the following notations, too.

*****

Return all user attributes such as `cn`, `sn`, and `mail`.

+

Return all operational attributes such as `etag` and `pwdPolicySubentry` .

`@objectclass`

Return all attributes of the specified object class, where *objectclass* is one of the object classes on the entries returned by the search.

`1.1`

Return no attributes, only the DNs of matching entries.

Exit codes

0

The command completed successfully.

ldap-error

An LDAP error occurred while processing the operation. LDAP result codes are described in RFC 4511 . Also see the additional information for details.

89

An error occurred while parsing the command-line arguments.

Files

You can use `~/.opendj/tools.properties` to set the defaults for bind DN, host name, and port number as in the following example:

```
hostname=directory.example.com
port=1389
bindDN=uid=kvaughan,ou=People,dc=example,dc=com

ldapcompare.port=1389
ldapdelete.port=1389
ldapmodify.port=1389
ldappasswordmodify.port=1389
ldapsearch.port=1389
```

ldifdiff

`ldifdiff` — compare small LDIF files

Synopsis

```
ldifdiff {options} source target
```

Description

This utility can be used to compare two LDIF files and report the differences in LDIF format.

If standard input is used to specify source or target, end your input with EOF (Ctrl+D on UNIX, Ctrl+Z on Windows).

Options

The `ldifdiff` command takes the following options:

Command options:

`-B | --excludeBranch {branchDN}`

Base DN of a branch to exclude when comparing entries.

`-e | --excludeAttribute {attribute}`

Attribute to ignore when comparing entries.

`-o | --outputLdif {file}`

Write differences to {file} instead of stdout. Default: stdout

`-x | --exactMatch`

Match values byte-for-byte instead of using equality matching rules, which can be useful when comparing schema files. Default: false

Utility input/output options:

`-t | --wrapColumn {wrapColumn}`

Maximum length of an output line (0 for no wrapping). Default: 0

General options:

`-V | --version`

Display Directory Server version information. Default: false

`-H | --help`

Display this usage information. Default: false

Exit codes

0

No differences were found.

1

Differences were found.

other

An error occurred.

ldifmodify

`ldifmodify` — apply LDIF changes to LDIF

Synopsis

```
ldifmodify {options} source_file [changes_files…]
```

Description

This utility can be used to apply a set of modify, add, and delete operations to entries contained in an LDIF file.

If standard input is used to specify source or changes, end your input with EOF (Ctrl+D on UNIX, Ctrl+Z on Windows).

Options

The `ldifmodify` command takes the following options:

Command options:

`-c | --continueOnError`

Continue processing even if there are errors. Default: false

`-o | --outputLdif {file}`

Write updated entries to {file} instead of stdout. Default: stdout

Utility input/output options:

`-t | --wrapColumn {wrapColumn}`

Maximum length of an output line (0 for no wrapping). Default: 0

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.

> 0

An error occurred.

ldifsearch

`ldifsearch` — search LDIF with LDAP filters

Synopsis

```
ldifsearch {options} source filter [attributes …]
```

Description

This utility can be used to perform search operations against entries contained in an LDIF file.

If standard input is used to specify source, end your input with EOF (Ctrl+D on UNIX, Ctrl+Z on Windows).

Options

The `ldifsearch` command takes the following options:

Command options:

`-A | --typesOnly`

Only retrieve attribute names but not their values. Default: false

`-b | --baseDn {baseDN}`

The base DN for the search. If no base DN is provided, then the root DSE will be used. Default:

`-l | --timeLimit {timeLimit}`

Maximum length of time in seconds to allow for the search. Default: 0

`-o | --outputLdif {file}`

Write search results to {file} instead of stdout. Default: stdout

`-s | --searchScope {searchScope}`

Search scope ('base', 'one', 'sub', or 'subordinates'). Note: 'subordinates' is an LDAP extension that might not work with all LDAP servers. Default: sub

-z | --sizeLimit {sizeLimit}

Maximum number of entries to return from the search. Default: 0

Utility input/output options:

-t | --wrapColumn {wrapColumn}

Maximum length of an output line (0 for no wrapping). Default: 0

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.

> 0

An error occurred.

makeldif-template

`makeldif.template` — template file for the makeldif command

Synopsis

```
# Comment lines start with #.
#
# Notice that this synopsis includes blank lines after entries.
# In the same way you would use blank lines after entries in normal LDIF,
# leave empty lines after "entries" in template files.

# Optionally define constants used in the template.
# To reference constants later, put brackets around the name: [constant-name]
#
define  _constant-name_ = _value_
...

# Define branches by suffix DN, such as the following:
#
#  dc=example,dc=com
#  ou=People,dc=example,dc=com
```

```
#  ou=Groups,dc=example,dc=com
#
# makeldif generates the necessary object class definitions and RDNs.
#
# A branch can have subordinateTemplates that define templates to use for
# the branch entry. The optional  _number_  at the end
# of the subordinateTemplate specification defines how many entries to generate.
# If you do not specify a number, makeldif continues to generate entries
# indefinitely until you interrupt the command.
#
# A branch can have additional attributes generated on the branch entry. See
# the Description below for more information on specifying attribute values.
#
branch:  _suffix-dn_
objectClass: top
objectClass:  _suffix-object-class_
[subordinateTemplate:  _template-name_ [: _number_ ]
...]
[ _attribute_ :  _attr-value_
...]

...

# Define entries using templates.
#
# A template can extend another template.
# A template defines the RDN attribute(s) used for generated entries.
# A template can have a subordinateTemplate that defines a template to use for
# the generated entries.
#
# A template then defines attributes. See the Description below for more
# information on specifying attribute values.
#
template:  _template-name_
[extends:  _template-name_ ]
rdnAttr:  _attribute_ [+ _attribute_  ...]
[subordinateTemplate:  _template-name_ : _number_ ]
[ _attribute_ :  _attr-value_
...]

...
```

Description

Template files specify how to build LDIF. They allow you to define variables, insert random values from other files, and generally build arbitrarily large LDIF files for testing purposes. You pass template files to the `makeldif` command when generating LDIF.

The Synopsis above shows the layout for a `makeldif` template file. This section focuses on what you can do to specify entry attribute values, called *attr-value* in the Synopsis section.

When specifying attribute values in `makeldif` templates, you can use static text and constants that you have defined, enclosing names for constants in brackets, `[myConstant]` . You can use more than one constant per line, as in the following example:

You can also use two kinds of tags when specifying attribute values. One kind of tag is replaced with the value of another attribute in the generated entry. Such tags are delimited with braces, `{ }` . For example, if your template includes definitions for first name and last name attributes, use:

Then you can define a mail attribute that uses the values of both attributes, and an initials attribute that takes the first character of each:

The other kind of tag is delimited with `<` and `>` , as shown above in the example with `<first>` and `<last>` . Tag names are not case sensitive. Many tags can take arguments separated by colons, `:` , from the tag names within the tag.

Use backslashes to escape literal start tag characters (`< [ {`) as shown in the following example, and to escape literal end tag characters within tags (`> ] }`):

The `makeldif` command supports the following tags:

<DateTime>

The DateTime tag is replaced by a timestamp. The DateTime tag takes the form `<DateTime[:offsetInSeconds[:formatString]]>` , where:

- *offsetInSeconds* is the offset in seconds from the current time. The offset may be a positive or negative integer. Default: `0` (seconds).
- *formatString* is a date time pattern string. For details, see the Javadoc for the DateTimeFormat class. Default: `yyyyMMddHHmmss.SSS'Z'` .

<DN>

The DN tag is replaced by the distinguished name of the current entry. An optional integer argument specifies the subcomponents of the DN to generate. For example, if the DN of the entry is `uid=bjensen,ou=People,dc=example,dc=com` , then `<DN:1>` is replaced by `uid=bjensen` , and `<DN:-2>` is replaced by `dc=example,dc=com` .

<File>

The File tag is replaced by a line from a text file you specify. The File tag takes a required argument, the path to the text file, and an optional second argument, either `random` or `sequential` . For the file argument, either specify an absolute path to the file such as `<file:/path/to/myDescriptions>` , or specify a path relative to the template file such as `<file:streets>` . For the second argument, if you specify `sequential` then lines from the file are read in sequential order. Otherwise, lines from the file are read in random order.

<First>

The first name tag is replaced by a random line from `first.names` . Combinations of generated first and last names are unique, with integers appended to the name strings if not enough combinations are available.

<GUID>

The GUID tag is replaced by a 128-bit, type 4 (random) universally unique identifier, such as `f47ac10b-58cc-4372-a567-0e02b2c3d479` .

<IfAbsent>

The IfAbsent tag takes as its first argument the name of another attribute, and optionally, as its second argument, a value to use. This tag causes the attribute to be generated only if the named attribute is not present on the generated entry. Use this tag when you have used `<Presence>` to define another attribute that is not always present on generated entries.

<IfPresent>

The IfPresent takes as its first argument the name of another attribute, and optionally, as its second argument, a value to use. This tag causes the attribute to be generated only if the named attribute is also present on the generated entry. Use this tag when you have used `<Presence>` to define another attribute that is sometimes present on generated entries.

<Last>

The last name tag is replaced by a random line from the last names template file, `last.names` . Combinations of generated first and last names are unique, with integers appended to the name strings if not enough combinations are available.

<List>

The List tag is replaced by one of the values from the list of arguments you provide. For example, `<List:bronze:silver:gold>` is replaced with `bronze` , `silver` , or `gold` . You can weight arguments to ensure that some arguments are selected more often than others. For example, if you want two bronze for one silver and one gold, use `<List:bronze;2:silver;1:gold;1>` .

<ParentDN>

The ParentDN tag is replaced by the distinguished name of the parent entry. For example, if the DN of the entry is `uid=bjensen,ou=People,dc=example,dc=com` , `<ParentDN>` is replaced by `ou=People,dc=example,dc=com` .

<Presence>

The Presence tag takes a percent argument. It results in the attribute value being generated or not based on the percentage of entries you specify in the argument. For example, `description: <Presence:50>A description` generates `description: A description` on half the entries.

<Random>

The Random tag lets you generate a variety of random numbers and strings. The Random tag has the following subtypes, which you include as arguments, that is `<Random:subtype>` :

- `alpha:length`
- `alpha:min-length:max-length`
- `numeric:length`
- `numeric:minvalue:maxvalue`
- `numeric:minvalue:maxvalue:format` , where *format* is a `java.text.DecimalFormat` pattern
- `alphanumeric:length`
- `alphanumeric:min-length:max-length`

- `chars:characters:length`
- `chars:characters:min-length:max-length`
- `hex:length`
- `hex:min-length:max-length`
- `base64:length`
- `base64:min-length:max-length`
- `month`
- `month:max-length`
- `telephone` , a telephone number starting with the country code `+1`

<RDN>

The RDN tag is replaced with the RDN of the entry. Use this in the template after you have specified `rdnAttr` so that the RDN has already been generated when this tag is replaced. An optional integer argument specifies the subcomponents of the RDN to generate.

<Sequential>

The Sequential tag is replaced by a sequentially increasing generated integer. The first optional integer argument specifies the starting number. The second optional boolean argument specifies whether to start over when generating entries for a new parent entry. For example, `<Sequential:42:true>` starts counting from 42, and starts over when the parent entry changes from `o=Engineering` to `o=Marketing` .

<_DN>

The _DN tag is replaced by the DN of the current entry with underscores in the place of commas.

<_ParentDN>

The _ParentDN tag is replaced by the DN the parent entry with underscores in the place of commas.

Examples

The following example generates 10 organization units, each containing 50 entries. Add it next to the supporting files, such as `first.names` and `last.names` needed to generate the output:

```
define suffix=dc=example,dc=com
define maildomain=example.com
define numusers=50
define numorgs=10

branch: [suffix]
objectClass: top
objectClass: domain

branch: ou=People,[suffix]
objectClass: top
```

```
objectClass: organizationalUnit
subordinateTemplate: orgunit:[numorgs]
description: This is the People container
telephoneNumber: +33 00010002

template: orgunit
subordinateTemplate: person:[numusers]
rdnAttr: ou
ou: Org-<sequential:0>
objectClass: top
objectClass: organizationalUnit
description: This is the {ou} organizational unit

template: person
rdnAttr: uid
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
givenName: <first>
sn: <last>
cn: {givenName} {sn}
initials: {givenName:1}<random:chars:ABCDEFGHIJKLMNOPQRSTUVWXYZ:1>{sn:1}
employeeNumber: <sequential:0>
uid: user.{employeeNumber}
mail: {uid}@[maildomain]
userPassword: password
telephoneNumber: <random:telephone>
homePhone: <random:telephone>
pager: <random:telephone>
mobile: <random:telephone>
street: <random:numeric:5> <file:streets> Street
l: <file:cities>
st: <file:states>
postalCode: <random:numeric:5>
postalAddress: {cn}${street}${l}, {st}  {postalCode}
description: This is the description for {cn}.
```

See also

makeldif , the template files under the `config/MakeLDIF` directory

makeldif

`makeldif` — generate test LDIF

Synopsis

```
makeldif {options} template-file-path
```

Description

This utility can be used to generate LDIF data based on a definition in a template file.

The *template-file-path* can be one of the following:

- A full path to the template file such as `/path/to/opendj/config/MakeLDIF/example.template` .
- A relative path to the template file such as `../../my-test-data.template` .
- A file name that specifies one of the template files, such as `example.template` , or `people_and_groups.template` .

The following default template and data files are provided:

`cities`

List of more than 200 cities.

`example.template`

Template to generate a base entry and users in a branch `ou=people,[suffix]` , where the default setting for suffix is `suffix=dc=example,dc=com` .

`first.names`

List of more than 8000 first names.

`last.names`

List of more than 13000 last names.

`people_and_groups.template`

Template to generate a base entry, users, and groups.

`states`

List of US states by their two-character codes.

`streets`

List of more than 70 street names.

Options

The `makeldif` command takes the following options:

Command options:

`-c | --constant {name=value}`

A constant that overrides the value set in the template file.

-o | --outputLdif {file}

The path to the LDIF file to be written. If the filename ends in .gz, the output will be gzipped.

-r | --resourcePath {path}

Path to look for MakeLDIF resources (e.g., data files). The utility looks for resources in the following locations in this order:

1. The current directory where the command is run.
2. The resource path directory.
3. The built-in files.

-s | --randomSeed {seed}

The seed to use to initialize the random number generator. To always generate the same data with the same command, use the same non-zero seed value. A value of zero (the default) results in different data each time the tool is run. Default: 0

Utility input/output options:

-t | --wrapColumn {wrapColumn}

Maximum length of an output line (0 for no wrapping). Default: 0

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.

1

An error occurred.

See also

makeldif-template

manage-account

`manage-account` — manage state of OpenDJ server accounts

Synopsis

```
manage-account {subcommand} {options}
```

Description

This utility can be used to retrieve and manipulate the values of password policy state variables.

Options

The `manage-account` command takes the following options:

Command options:

`-b | --targetDn {targetDN}`

The DN of the user entry for which to get and set password policy state information.

LDAP connection options:

`--connectTimeout {timeout}`

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

`-D | --bindDn {bindDN}`

DN to use to bind to the server. Default: uid=admin

`-E | --reportAuthzId`

Use the authorization identity control. Default: false

`-h | --hostname {host}`

Fully-qualified server host name or IP address. Default: localhost.localdomain

`-N | --certNickname {nickname}`

Nickname of the certificate that should be sent to the server for SSL client authentication.

`-o | --saslOption {name=value}`

SASL bind options.

`-p | --port {port}`

Directory server administration port number.

`-T | --trustStorePassword[:env|:file] {trustStorePassword}`

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. 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

-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

Subcommands

The `manage-account` command supports the following subcommands:

manage-account add-authentication-failure-time

```
manage-account add-authentication-failure-time {options}
```

Add an authentication failure time to the user account. This should be used only for testing purposes.

Options

In addition to the global `manage-account` options, the `manage-account add-authentication-failure-time` subcommand takes the following options:

-O | --operationValue {time}

A timestamp value using the generalized time syntax. Multiple timestamp values may be given by providing this argument multiple times.

manage-account add-grace-login-use-time

```
manage-account add-grace-login-use-time {options}
```

Add a grace login use time to the user account. This should be used only for testing purposes.

Options

In addition to the global `manage-account` options, the `manage-account add-grace-login-use-time` subcommand takes the following options:

-O | --operationValue {time}

A timestamp value using the generalized time syntax. Multiple timestamp values may be given by providing this argument multiple times.

manage-account clear-account-expiration-time

```
manage-account clear-account-expiration-time
```

Clear account expiration time information from the user account.

manage-account clear-account-is-disabled

```
manage-account clear-account-is-disabled
```

Clear account disabled state information from the user account.

manage-account clear-authentication-failure-times

```
manage-account clear-authentication-failure-times
```

Clear authentication failure time information from the user's account. This should be used only for testing purposes.

manage-account clear-grace-login-use-times

```
manage-account clear-grace-login-use-times
```

Clear the set of grace login use times for the user. This should be used only for testing purposes.

manage-account clear-last-login-time

```
manage-account clear-last-login-time
```

Clear the time that the user last authenticated to the server. This should be used only for testing purposes.

manage-account clear-password-changed-by-required-time

```
manage-account clear-password-changed-by-required-time
```

Clear information about the required password change time with which the user last complied. This should be used only for testing purposes.

manage-account clear-password-changed-time

```
manage-account clear-password-changed-time
```

Clear information about the time that the user's password was last changed. This should be used only for testing purposes.

manage-account clear-password-expiration-warned-time

```
manage-account clear-password-expiration-warned-time
```

Clear information about the time that the user first received an expiration warning notice. This should be used only for testing purposes.

manage-account clear-password-history

```
manage-account clear-password-history
```

Clear password history state values for the user. This should be used only for testing purposes.

manage-account clear-password-is-reset

```
manage-account clear-password-is-reset
```

Clear information about whether the user will be required to change his or her password on the next successful authentication. This should be used only for testing purposes.

manage-account get-account-expiration-time

`manage-account get-account-expiration-time`

Display when the user account will expire.

manage-account get-account-is-disabled

`manage-account get-account-is-disabled`

Display information about whether the user account has been administratively disabled.

manage-account get-all

`manage-account get-all`

Display all password policy state information for the user.

manage-account get-authentication-failure-times

`manage-account get-authentication-failure-times`

Display the authentication failure times for the user.

manage-account get-grace-login-use-times

`manage-account get-grace-login-use-times`

Display the grace login use times for the user.

manage-account get-last-login-time

`manage-account get-last-login-time`

Display the time that the user last authenticated to the server.

manage-account get-password-changed-by-required-time

`manage-account get-password-changed-by-required-time`

Display the required password change time with which the user last complied.

manage-account get-password-changed-time

`manage-account get-password-changed-time`

Display the time that the user's password was last changed.

manage-account get-password-expiration-warned-time

`manage-account get-password-expiration-warned-time`

Display the time that the user first received an expiration warning notice.

manage-account get-password-is-reset

`manage-account get-password-is-reset`

Display information about whether the user will be required to change his or her password on the next successful authentication.

manage-account get-password-policy-dn

`manage-account get-password-policy-dn`

Display the DN of the password policy for the user.

manage-account get-remaining-authentication-failure-count

`manage-account get-remaining-authentication-failure-count`

Display the number of remaining authentication failures until the user's account is locked.

manage-account get-remaining-grace-login-count

`manage-account get-remaining-grace-login-count`

Display the number of grace logins remaining for the user.

manage-account get-seconds-until-account-expiration

`manage-account get-seconds-until-account-expiration`

Display the length of time in seconds until the user account expires.

manage-account get-seconds-until-authentication-failure-unlock

`manage-account get-seconds-until-authentication-failure-unlock`

Display the length of time in seconds until the authentication failure lockout expires.

manage-account get-seconds-until-idle-lockout

`manage-account get-seconds-until-idle-lockout`

Display the length of time in seconds until user's account is locked because it has remained idle for too long.

manage-account get-seconds-until-password-expiration

`manage-account get-seconds-until-password-expiration`

Display length of time in seconds until the user's password expires.

manage-account get-seconds-until-password-expiration-warning

`manage-account get-seconds-until-password-expiration-warning`

Display the length of time in seconds until the user should start receiving password expiration warning notices.

manage-account get-seconds-until-password-reset-lockout

`manage-account get-seconds-until-password-reset-lockout`

Display the length of time in seconds until user's account is locked because the user failed to change the password in a timely manner after an administrative reset.

manage-account get-seconds-until-required-change-time

`manage-account get-seconds-until-required-change-time`

Display the length of time in seconds that the user has remaining to change his or her password before the account becomes locked due to the required change time.

manage-account set-account-expiration-time

`manage-account set-account-expiration-time {options}`

Specify when the user account will expire.

Options

In addition to the global `manage-account` options, the `manage-account set-account-expiration-time` subcommand takes the following options:

`-O | --operationValue {time}`

A timestamp value using the generalized time syntax.

manage-account set-account-is-disabled

`manage-account set-account-is-disabled {options}`

Specify whether the user account has been administratively disabled.

Options

In addition to the global `manage-account` options, the `manage-account set-account-is-disabled` subcommand takes the following options:

`-O | --operationValue {true|false}`

'true' to indicate that the account is disabled, or 'false' to indicate that it is not disabled.

manage-account set-authentication-failure-times

`manage-account set-authentication-failure-times {options}`

Specify the authentication failure times for the user. This should be used only for testing purposes.

Options

In addition to the global `manage-account` options, the `manage-account set-authentication-failure-times` subcommand takes the following options:

-O | --operationValue {time}

A timestamp value using the generalized time syntax. Multiple timestamp values may be given by providing this argument multiple times.

manage-account set-grace-login-use-times

```
manage-account set-grace-login-use-times {options}
```

Specify the grace login use times for the user. This should be used only for testing purposes.

Options

In addition to the global `manage-account` options, the `manage-account set-grace-login-use-times` subcommand takes the following options:

-O | --operationValue {time}

A timestamp value using the generalized time syntax. Multiple timestamp values may be given by providing this argument multiple times.

manage-account set-last-login-time

```
manage-account set-last-login-time {options}
```

Specify the time that the user last authenticated to the server. This should be used only for testing purposes.

Options

In addition to the global `manage-account` options, the `manage-account set-last-login-time` subcommand takes the following options:

-O | --operationValue {time}

A timestamp value using the generalized time syntax.

manage-account set-password-changed-by-required-time

```
manage-account set-password-changed-by-required-time {options}
```

Specify the required password change time with which the user last complied. This should be used only for testing purposes.

Options

In addition to the global `manage-account` options, the `manage-account set-password-changed-by-required-time` subcommand takes the following options:

-O | --operationValue {time}

A timestamp value using the generalized time syntax.

manage-account set-password-changed-time

```
manage-account set-password-changed-time {options}
```

Specify the time that the user's password was last changed. This should be used only for testing purposes.

Options

In addition to the global `manage-account` options, the `manage-account set-password-changed-time` subcommand takes the following options:

`-O | --operationValue {time}`

A timestamp value using the generalized time syntax.

manage-account set-password-expiration-warned-time

```
manage-account set-password-expiration-warned-time {options}
```

Specify the time that the user first received an expiration warning notice. This should be used only for testing purposes.

Options

In addition to the global `manage-account` options, the `manage-account set-password-expiration-warned-time` subcommand takes the following options:

`-O | --operationValue {time}`

A timestamp value using the generalized time syntax.

manage-account set-password-is-reset

```
manage-account set-password-is-reset {options}
```

Specify whether the user will be required to change his or her password on the next successful authentication. This should be used only for testing purposes.

Options

In addition to the global `manage-account` options, the `manage-account set-password-is-reset` subcommand takes the following options:

`-O | --operationValue {true|false}`

'true' to indicate that the account is disabled, or 'false' to indicate that it is not disabled.

Exit codes

0

The command completed successfully.

89

An error occurred while parsing the command-line arguments.

manage-tasks

`manage-tasks` — manage server administration tasks

Synopsis

```
manage-tasks {options}
```

Description

This utility can be used to obtain a list of tasks scheduled to run within the Directory Server as well as information about individual tasks.

Options

The `manage-tasks` command takes the following options:

Command options:

`-c | --cancel {taskID}`

ID of a particular task to cancel.

`-i | --info {taskID}`

ID of a particular task about which this tool will display information.

`-s | --summary`

Print a summary of tasks. Default: false

`--status {taskStatus}`

Show only tasks with this status.

`-t | --type {taskType}`

Show only tasks of this type.

LDAP connection options:

`--connectTimeout {timeout}`

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

`-D | --bindDn {bindDN}`

DN to use to bind to the server. Default: uid=admin

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. 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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

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.

> 0

An error occurred.

modrate

`modrate` — measure modification throughput and response time

Synopsis

```
modrate {options} [(attribute:value template string) …]
```

Description

This utility can be used to measure modify throughput and response time of a directory service using user-defined modifications.

Example:

modrate -p 1636 -Z -X -D uid=admin -w password \

-F -c 4 -t 4 -b 'uid=user.{1},ou=people,dc=example,dc=com' \

-g 'rand(0,2000)' -g 'randstr(16)' 'description:{2}'

Before trying the example, import 2000 randomly generated users.

When you do not use the `-f` option to keep connections open and rebind on the connections, the tool can exhaust its available ports, causing the tool to crash. You can work around this problem on test systems by changing TCP settings on the system.

For example, on Linux systems, set the following parameters in the `/etc/sysctl.conf` file:

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
```

The parameter `net.ipv4.tcp_fin_timeout` sets the length of time in seconds to wait for a final FIN packet before forcing a close of the socket. The default is 60 (seconds).

The parameter `net.ipv4.tcp_tw_recycle` enables fast recycling of TIME_WAIT sockets. The default is 0 (false). Enabling this can cause Network Address Translation (NAT) issues.

The parameter `net.ipv4.tcp_tw_reuse` enables reuse of TIME_WAIT sockets for new connections. The default is 0 (false).

These settings are recommended only for testing, and *not for production systems* .

After making the changes to `/etc/sysctl.conf` , reload the configuration with the `sysctl` command:

```
# sysctl -p
```

Options

The `modrate` command takes the following options:

Command options:

`-b | --targetDn {targetDN}`

Target entry DN template string.

-B | --warmUpDuration {warmUpDuration}

Warm up duration in seconds. Default: 0

-c | --numConnections {numConnections}

Number of connections. Default: 1

-d | --maxDuration {maxDuration}

Maximum duration in seconds, 0 for unlimited. Default: 0

-e | --percentile {percentile}

Calculate max response time for a percentile of operations.

-f | --keepConnectionsOpen

Keep connections open. Default: false

-F | --noRebind

Keep connections open and do not rebind. Default: false

-g | --argument {generator function or static string}

Argument used to evaluate the template strings in program parameters (ie. Base DN, Search Filter). The set of all arguments provided form the argument list in order. Besides static string arguments, they can be generated per iteration with the following functions:

"inc({filename})" Consecutive, incremental line from file

"inc({min},{max})" Consecutive, incremental number

"rand({filename})" Random line from file

"rand({min},{max})" Random number

"randstr({length},*charSet*)" Random string of specified length and optionally from characters in the charSet string. A range of character can be specified with [start-end] charSet notation. If no charSet is specified, the default charSet of [A-Z][a-z][0-9] will be used.

These functions do not support formatted integers with comma due to the ambiguity between a comma used to separate function arguments and a comma used to separate digits in a formatted integer.

-i | --statInterval {statInterval}

Display results each specified number of seconds. Default: 5

-m | --maxIterations {maxIterations}

Max iterations, 0 for unlimited. Default: 0

-M | --targetThroughput {targetThroughput}

Target average throughput to achieve. Default: 0

-S | --scriptFriendly

Use script-friendly mode. Default: false

-t | --numConcurrentRequests {numConcurrentRequests}

Number of concurrent requests per connection. Default: 1

-Y | --proxyAs {authzID}

Use the proxied authorization control with the given authorization ID.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default:

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server port number.

-q | --useStartTls

Use StartTLS to secure communication with the server. Default: false

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

-Z | --useSsl

Use SSL for secure communication with the server. 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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-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.

89

An error occurred while parsing the command-line arguments.

Examples

The following example uses the `modrate` command to write random 16-character description values to user entries:

```
$ modrate \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDn uid=admin \
 --bindPassword password \
 --noRebind \
 --numConnections 4 \
 --numConcurrentRequests 4 \
 --maxDuration 30 \
 --argument "rand(0,2000)" --targetDn "uid=user.{1},ou=people,dc=example,dc=com" \
 --argument "randstr(16)" 'description:{2}'
---------------------------------------------------------------------------
|     Throughput    |                 Response Time                |          |
|    (ops/second)   |                (milliseconds)                |          |
|   recent  average |   recent  average    99.9%   99.99%  99.999% |  err/sec |
```

```
-------------------------------------------------------------------------------
|  11086.2  11086.2 |    1.404    1.404    14.88    19.14    23.86 |      0.0 |
|  15351.6  13218.9 |    1.031    1.187    13.89    17.04    23.86 |      0.0 |
|  15252.0  13896.6 |    1.038    1.133    13.24    17.30    23.33 |      0.0 |
|  15383.2  14268.3 |    1.029    1.105    13.37    18.22    51.64 |      0.0 |
|  15204.6  14455.5 |    1.041    1.091    13.83    17.83    51.64 |      0.0 |
|  15356.3  14605.3 |    1.030    1.080    13.63    17.83    51.64 |      0.0 |
```

This example uses the following options:

`--hostname localhost`, `--port 1636`, `--useSsl`, `--usePkcs12TrustStore /path/to/opendj/config/keystore`, `--trustStorePassword:file /path/to/opendj/config/keystore.pin`

Access the server running on the local system over a secure LDAPS connection to port 1636.

`--bindDn uid=admin`, `--bindPassword password`

Authenticate as the directory root user `uid=admin` with the bind password that is literally `password`. This user is not subject to access control, so rates may be higher than what you observe with a regular user.

`--noRebind`

Keep connections open and do not rebind.

`--numConnections 4`

Open 4 connections to the server.

`--numConcurrentRequests 4`

Perform up to 4 concurrent requests on each connection.

`--maxDuration 30`

Run for a maximum of 30 seconds.

`--argument "rand(0,2000)" --targetDn "uid=user.{1},ou=people,dc=example,dc=com"`

Target the entry with DN `uid=user.number,ou=people,dc=example,dc=com`, where *number* is a random number between 0 and 2000, inclusive.

`--argument "randstr(16)" 'description:{2}'`

Write a random, 16-character string to the `description` attribute of the target entry. The `randstr(16)` argument specifies only the length, which is 16. It does not have an optional second argument to specify a character set. Therefore, use the default character set, which is `[A-Z][a-z][0-9]`.

Notice the following characteristics of the output:

- The first two columns show the throughput in operations completed per second. The recent column shows the average rate for operations reflected in this row of output. The average column shows the average rate since the beginning of the run.

- The response time columns indicate characteristics of response latency in milliseconds. The recent column shows the average latency for operations reflected in this row of output. The average column shows the average latency since the beginning of the run. The "99.9%" column shows the latency after which 99.9% of operations have completed. Only 1 operation in 1000 took longer than this. The "99.99%" column shows the latency after which 99.99% of operations have completed. Only 1 operation in 10,000 took longer than this. The "99.999%" column shows the latency after which 99.999% of operations have completed. Only 1 operation in 100,000 took longer than this.
- The "err/sec" column show the rate of error results per second for this row of output. Unless you have intentionally set up the command to generate errors, this column should indicate `0.0` . Check that this column matches your expectations before looking at any other columns.

rebuild-index

`rebuild-index` — rebuild index after configuration change

Synopsis

```
rebuild-index {options}
```

Description

This utility can be used to rebuild index data within an indexed backend database.

Options

The `rebuild-index` command takes the following options:

Command options:

`-b | --baseDn {baseDN}`

Base DN of a backend supporting indexing. Rebuild is performed on indexes within the scope of the given base DN.

`--clearDegradedState`

Indicates that indexes do not need rebuilding because they are known to be empty and forcefully marks them as valid. This is an advanced option which must only be used in cases where a degraded index is known to be empty and does not therefore need rebuilding. This situation typically arises when an index is created for an attribute which has just been added to the schema. Default: false

`-i | --index {index}`

Names of index(es) to rebuild. For an attribute index this is simply an attribute name. At least one index must be specified for rebuild. Cannot be used with the "--rebuildAll" option.

`--offline`

Indicates that the command must be run in offline mode. When using this option, the command writes to server files. Run the command as a user having the same filesystem permissions as the user running the server. Default: false

--rebuildAll

Rebuild all indexes, including any DN2ID, DN2URI, VLV and extensible indexes. Cannot be used with the "-i" option or the "--rebuildDegraded" option. Default: false

--rebuildDegraded

Rebuild all degraded indexes, including any DN2ID, DN2URI, VLV and extensible indexes. Cannot be used with the "-i" option or the "--rebuildAll" option. Default: false

--tmpDirectory {directory}

Path to temporary directory for index scratch files during index rebuilding. Default: import-tmp

Task Scheduling Options

--completionNotify {emailAddress}

Email address of a recipient to be notified when the task completes. This option may be specified more than once.

--dependency {taskID}

ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution.

--description {description}

Gives a description to the task.

--errorNotify {emailAddress}

Email address of a recipient to be notified if an error occurs when this task executes. This option may be specified more than once.

--failedDependencyAction {action}

Action this task will take should one if its dependent tasks fail. The value must be one of PROCESS, CANCEL, DISABLE. If not specified defaults to CANCEL.

--recurringTask {schedulePattern}

Indicates the task is recurring and will be scheduled according to the value argument expressed in crontab(5) compatible time/date pattern. The schedule pattern for a recurring task supports only the following `crontab` features:

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names)

Field	Allowed Values
day of week	0-7 (0 or 7 is Sunday, or use names)

A field can contain an asterisk, `*` . An asterisk stands for `first-last` .

Fields can include ranges of numbers. A range is two numbers separated by a hyphen, and is inclusive. For example, `8-10` for an "hour" field means execution at hours 8, 9, and 10.

Fields can include lists. A list is a set of numbers or ranges separated by commas. For example, `4,8-10` for an "hour" field means execution at hours 4, 8, 9, and 10.

When using names for in "month" or "day of week" fields, use the first three letters of the particular month or day of the week. Case does not matter. Ranges and lists of names are not supported.

`-t | --start {startTime}`

Indicates the date/time at which this operation will start when scheduled as a server task expressed in YYYYMMDDhhmmssZ format for UTC time or YYYYMMDDhhmmss for local time. A value of '0' will cause the task to be scheduled for immediate execution. When this option is specified the operation will be scheduled to start at the specified time after which this utility will exit immediately.

`--taskId {taskID}`

Gives an ID to the task.

Task Backend Connection Options

`--connectTimeout {timeout}`

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

`-D | --bindDn {bindDN}`

DN to use to bind to the server. Default: uid=admin

`-E | --reportAuthzId`

Use the authorization identity control. Default: false

`-h | --hostname {host}`

Fully-qualified server host name or IP address. Default: localhost.localdomain

`-N | --certNickname {nickname}`

Nickname of the certificate that should be sent to the server for SSL client authentication.

`-o | --saslOption {name=value}`

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. 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

`--noPropertiesFile`

No properties file will be used to get default command line argument values. Default: false

`--propertiesFilePath {propertiesFilePath}`

Path to the file containing default property values used for command line arguments.

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.

> 0

An error occurred.

searchrate

`searchrate` — measure search throughput and response time

Synopsis

```
searchrate {options} [filter template string] [attributes …]
```

Description

This utility can be used to measure search throughput and response time of a directory service using user-defined searches.

Example:

searchrate -p 1636 -Z -X -D uid=admin -w password \

-F -c 4 -t 4 -b 'dc=example,dc=com' -g 'rand(0,2000)' '(uid=user.{})'

Before trying the example, import 2000 randomly generated users.

When you do not use the `-f` option to keep connections open and rebind on the connections, the tool can exhaust its available ports, causing the tool to crash. You can work around this problem on test systems by changing TCP settings on the system.

For example, on Linux systems, set the following parameters in the `/etc/sysctl.conf` file:

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
```

The parameter `net.ipv4.tcp_fin_timeout` sets the length of time in seconds to wait for a final FIN packet before forcing a close of the socket. The default is 60 (seconds).

The parameter `net.ipv4.tcp_tw_recycle` enables fast recycling of TIME_WAIT sockets. The default is 0 (false). Enabling this can cause Network Address Translation (NAT) issues.

The parameter `net.ipv4.tcp_tw_reuse` enables reuse of TIME_WAIT sockets for new connections. The default is 0 (false).

These settings are recommended only for testing, and *not for production systems* .

After making the changes to `/etc/sysctl.conf` , reload the configuration with the `sysctl` command:

```
# sysctl -p
```

Options

The `searchrate` command takes the following options:

Command options:

`-a | --dereferencePolicy {dereferencePolicy}`

Alias dereference policy ('never', 'always', 'search', or 'find'). Default: never

`-b | --baseDn {baseDN}`

Base DN template string.

`-B | --warmUpDuration {warmUpDuration}`

Warm up duration in seconds. Default: 0

`-c | --numConnections {numConnections}`

Number of connections. Default: 1

`-d | --maxDuration {maxDuration}`

Maximum duration in seconds, 0 for unlimited. Default: 0

`-e | --percentile {percentile}`

Calculate max response time for a percentile of operations.

-f | --keepConnectionsOpen

Keep connections open. Default: false

-F | --noRebind

Keep connections open and do not rebind. Default: false

-g | --argument {generator function or static string}

Argument used to evaluate the template strings in program parameters (ie. Base DN, Search Filter). The set of all arguments provided form the argument list in order. Besides static string arguments, they can be generated per iteration with the following functions:

"inc({filename})" Consecutive, incremental line from file

"inc({min},{max})" Consecutive, incremental number

"rand({filename})" Random line from file

"rand({min},{max})" Random number

"randstr({length},*charSet*)" Random string of specified length and optionally from characters in the charSet string. A range of character can be specified with [start-end] charSet notation. If no charSet is specified, the default charSet of [A-Z][a-z][0-9] will be used.

These functions do not support formatted integers with comma due to the ambiguity between a comma used to separate function arguments and a comma used to separate digits in a formatted integer.

-i | --statInterval {statInterval}

Display results each specified number of seconds. Default: 5

-m | --maxIterations {maxIterations}

Max iterations, 0 for unlimited. Default: 0

-M | --targetThroughput {targetThroughput}

Target average throughput to achieve. Default: 0

-s | --searchScope {searchScope}

Search scope ('base', 'one', 'sub', or 'subordinates'). Note: 'subordinates' is an LDAP extension that might not work with all LDAP servers. Default: sub

-S | --scriptFriendly

Use script-friendly mode. Default: false

-t | --numConcurrentRequests {numConcurrentRequests}

Number of concurrent requests per connection. Default: 1

-Y | --proxyAs {authzID}

Use the proxied authorization control with the given authorization ID.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default:

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server port number.

-q | --useStartTls

Use StartTLS to secure communication with the server. Default: false

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. Default: false

-Z | --useSsl

Use SSL for secure communication with the server. 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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-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.

89

An error occurred while parsing the command-line arguments.

Examples

The following example measures search performance:

```
$ searchrate \
 --hostname localhost \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --baseDn dc=example,dc=com \
 --numConnections 4 \
 --noRebind \
 --numConcurrentRequests 4 \
 --maxDuration 30 \
 --argument "rand(0,2000)" "(uid=user.{})"
-------------------------------------------------------------------------------------------
|     Throughput    |                 Response Time                 |       Additional      |
|    (ops/second)   |                (milliseconds)                 |       Statistics      |
|   recent  average |   recent  average    99.9%   99.99%  99.999% |  err/sec Entries/Srch |
-------------------------------------------------------------------------------------------
|  15669.9  15669.9 |    0.992    0.992    17.04    29.62    33.42 |      0.0          1.0 |
|  28143.6  21896.8 |    0.562    0.717    12.39    25.30    33.42 |      0.0          1.0 |
|  26761.8  23518.5 |    0.592    0.669    10.49    22.02    32.11 |      0.0          1.0 |
|  27053.2  24402.2 |    0.585    0.646     9.18    19.79    30.67 |      0.0          1.0 |
|  27555.4  25032.8 |    0.575    0.630     8.22    18.35    30.02 |      0.0          1.0 |
|  27745.3  25484.1 |    0.570    0.619     7.67    17.83    29.62 |      0.0          1.0 |
```

This example uses the following options:

`--hostname localhost , --port 1636 , --useSsl , --usePkcs12TrustStore /path/to/opendj/config/keystore , --trustStorePassword:file /path/to/opendj/config/keystore.pin`

Access the server running on the local system over a secure LDAPS connection to port 1636.

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

Search under the base DN `dc=example,dc=com` . This user is not subject to access control, so rates may be higher than what you observe with a regular user.

No* `--bindDn` *or* `--bindPassword` *options

Perform the search as an anonymous user.

`--numConnections 4`

Open 4 connections to the server.

`--noRebind`

Keep connections open and do not rebind.

`--numConcurrentRequests 4`

Perform up to 4 concurrent requests on each connection.

`--maxDuration 30`

Run for a maximum of 30 seconds.

`--argument "rand(0,2000)" "(uid=user.{})"`

Search for an entry with UID equal to `uid=user.number` , where *number* is a random number between 0 and 2000, inclusive.

Notice the following characteristics of the output:

- The first two columns show the throughput in operations completed per second. The recent column shows the average rate for operations reflected in this row of output. The average column shows the average rate since the beginning of the run.
- The response time columns indicate characteristics of response latency in milliseconds. The recent column shows the average latency for operations reflected in this row of output. The average column shows the average latency since the beginning of the run. The "99.9%" column shows the latency after which 99.9% of operations have completed. Only 1 operation in 1000 took longer than this. The "99.99%" column shows the latency after which 99.99% of operations have completed. Only 1 operation in 10,000 took longer than this. The "99.999%" column shows the latency after which 99.999% of operations have completed. Only 1 operation in 100,000 took longer than this.
- The additional statistics columns show information about what is happening during the run. The "err/sec" column shows the rate of error results per second for this row of output. Unless you have intentionally set up the command to generate errors, this column should indicate `0.0` . The "Entries/Srch" column shows the average number of entries returned for each search. If you expect one result entry per search, this column should indicate `1.0` . Check that these columns match your expectations before looking at any other columns.

setup-profile

`setup-profile` — configure profiles in an offline OpenDJ server instance

Synopsis

```
setup-profile {options}
```

Description

This utility configures profiles in an offline OpenDJ server instance. There are no setup profiles available for this OpenDJ version

.

Options

The `setup-profile` command takes the following options:

Command options:

`--help-profile {name[:version]}`

Display profile parameters.

`--instancePath {path}`

Path of the server instance where profiles should be setup. Default: /path/to/opendj

`--profile {name[:version]}`

Name of the profile to be configured. If the version is not specified, the most recent version older or equal to this OpenDJ version is used. Use this option multiple times to apply multiple profiles.

`--set[:env|:file] {[profileName/]parameterName:value}`

Assign a value to a setup profile parameter. Profile name must be provided if multiple profiles are provided, indicate the profile that a parameter applies to by using the profileName/parameterName format.

Utility input/output options:

`--noPropertiesFile`

No properties file will be used to get default command line argument values. Default: false

`--propertiesFilePath {propertiesFilePath}`

Path to the file containing default property values used for command line arguments.

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.

> 0

An error occurred.

setup

`setup` — install OpenDJ server

Synopsis

```
setup {options}
```

Description

This utility sets up an OpenDJ server. Use the --help-profiles option to list available profiles.

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.

`--bootstrapReplicationServer {bootstrapReplicationServer}`

The addresses of one or more replication servers within the topology which the server should connect to for discovering the rest of the topology. Use syntax "hostname:port" or "[IPv6Address]:port" for IPv6 addresses.

-D | --rootUserDn {rootUserDN}

DN for the initial root user for the Directory Server. Default: uid=admin

--deploymentId {deploymentId}

The deployment ID which should be used for securing the deployment. If no existing certificates are specified using the key-store and trust-store options then the deployment ID will also be used for securing all TLS network communication.

--deploymentIdPassword[:env|:file] {deploymentIdPassword}

Deployment ID password.

-h | --hostname {host}

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

--help-profile {name[:version]}

Display profile parameters.

--help-profiles

Display all available profiles. Default: false

--httpPort {port}

Port on which the server should listen for HTTP communication.

--httpsPort {port}

Port on which the server should listen for HTTPS communication.

--instancePath {path}

Path were the instance should be set up. Default: /path/to/opendj

--monitorUserDn {monitorUserDn}

DN of the default user allowed to query monitoring information. Default: uid=Monitor

--monitorUserPassword[:env|:file] {monitorUserPassword}

Password of the default user allowed to query monitoring information.

-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.

-p | --ldapPort {port}

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

--profile {name[:version]}

Setup profile to apply when initially configuring the server. If the version is not specified, the most recent version older or equal to this OpenDJ version is used. Use this option multiple times to apply multiple profiles. This option cannot be combined with data import options. There are no setup profiles available for this OpenDJ version.

-q | --enableStartTls

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

-Q | --quiet

Use quiet mode. Default: false

-r | --replicationPort {port}

Port used for replication protocol communications with other servers. Use this option to configure a local replication server. When this option is not used, this server is configured as a standalone DS (no local replication server).

-s | --start

Start the server when the configuration is completed. Default: false

-S | --skipPortCheck

Skip the check to determine whether the specified ports are usable. Default: false

--serverId {serverId}

Specify the server ID for this server. An acceptable ID is an ASCII alpha-numeric string; it may also contain underscore and hyphen characters provided they are not the first character.

--set[:env|:file] {[profileName/]parameterName:value}

Assign a value to a setup profile parameter. Profile name must be provided if multiple profiles are provided, indicate the profile that a parameter applies to by using the profileName/parameterName format.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

-u | --keyStorePasswordFilePath {path}

Path of the file containing the keystore password. The specified path will be used as the configuration value in the new server.

-U | --trustStorePasswordFilePath {path}

Path of the file containing the truststore password. The specified path will be used as the configuration value in the new server.

--useJavaKeyStore {keyStorePath}

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

--useJavaTrustStore {trustStorePath}

Use existing JKS truststore file for validating peer SSL certificates.

--useJceKeyStore {keyStorePath}

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

--useJceTrustStore {trustStorePath}

Use existing JCEKS truststore file for validating peer SSL certificates.

--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.

--usePkcs12TrustStore {trustStorePath}

Use existing PKCS12 truststore file for validating peer SSL certificates.

-w | --rootUserPassword[:env|:file] {rootUserPassword}

Password for the initial root user for the Directory Server.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Blindly trust peer SSL certificates. Default: false

-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 option is explicitly specified.

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.

> 0

An error occurred.

start-ds

`start-ds` — start OpenDJ server

Synopsis

```
start-ds {options}
```

Description

This utility can be used to start the Directory Server, as well as to obtain the server version and other forms of general server information.

Options

The `start-ds` command takes the following options:

Command options:

`-L | --useLastKnownGoodConfig`

Attempt to start using the configuration that was in place at the last successful startup (if it is available) rather than using the current active configuration. Default: false

`-N | --noDetach`

Do not detach from the terminal and continue running in the foreground. This option cannot be used with the -t, --timeout option. Default: false

`-s | --systemInfo`

Display general system information. Default: false

`-t | --timeout {seconds}`

Maximum time (in seconds) to wait before the command returns (the server continues the startup process, regardless). A value of '0' indicates an infinite timeout, which means that the command returns only when the server startup is completed. The default value is 60 seconds. This option cannot be used with the -N, --nodetach option. Default: 200

Utility input/output options:

-Q | --quiet

Use quiet 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.

> 0

An error occurred.

status

`status` — display basic OpenDJ server information

Synopsis

```
status {options}
```

Description

This utility can be used to display basic server information.

Options

The `status` command takes the following options:

Command options:

--offline

Indicates that the command must be run in offline mode. Default: false

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default: uid=admin

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. 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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-r | --refresh {period}

When this argument is specified, the status command will display its contents periodically. Used to specify the period (in seconds) between two displays of the status.

-s | --script-friendly

Use script-friendly 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.

> 0

An error occurred.

stop-ds

`stop-ds` — stop OpenDJ server

Synopsis

```
stop-ds {options}
```

Description

This utility can be used to request that the Directory Server stop running or perform a restart. When run without explicit connection options, this utility sends a signal to the OpenDJ process to stop the server. When run with explicit connection options, this utility connects to the OpenDJ administration port and creates a shutdown task to stop the server.

Options

The `stop-ds` command takes the following options:

Command options:

-r | --stopReason {stopReason}

Reason the server is being stopped or restarted.

-R | --restart

Attempt to automatically restart the server once it has stopped. Default: false

-t | --stopTime {stopTime}

Indicates the date/time at which the shutdown operation will begin as a server task expressed in format YYYYMMDDhhmmssZ for UTC time or YYYYMMDDhhmmss for local time. A value of '0' will cause the shutdown to be scheduled for immediate execution. When this option is specified the operation will be scheduled to start at the specified time after which this utility will exit immediately.

-Y | --proxyAs {authzID}

Use the proxied authorization control with the given authorization ID.

LDAP connection options:

--connectTimeout {timeout}

Maximum length of time (in milliseconds) that can be taken to establish a connection. Use '0' to specify no time out. Default: 30000

-D | --bindDn {bindDN}

DN to use to bind to the server. Default: uid=admin

-E | --reportAuthzId

Use the authorization identity control. Default: false

-h | --hostname {host}

Fully-qualified server host name or IP address. Default: localhost.localdomain

-N | --certNickname {nickname}

Nickname of the certificate that should be sent to the server for SSL client authentication.

-o | --saslOption {name=value}

SASL bind options.

-p | --port {port}

Directory server administration port number.

-T | --trustStorePassword[:env|:file] {trustStorePassword}

Truststore password which will be used as the cleartext configuration value.

--useJavaKeyStore {keyStorePath}

JKS keystore containing the certificate which should be used for SSL client authentication.

--useJavaTrustStore {trustStorePath}

Use a JKS truststore file for validating server certificate.

--useJceKeyStore {keyStorePath}

JCEKS keystore containing the certificate which should be used for SSL client authentication.

--useJceTrustStore {trustStorePath}

Use a JCEKS truststore file for validating server certificate.

--useJvmTrustStore

Use the JVM truststore for validating server certificate. Default: false

--usePasswordPolicyControl

Use the password policy request control. Default: false

--usePkcs11KeyStore

PKCS#11 keystore containing the certificate which should be used for SSL client authentication. Default: false

--usePkcs12KeyStore {keyStorePath}

PKCS#12 keystore containing the certificate which should be used for SSL client authentication.

--usePkcs12TrustStore {trustStorePath}

Use a PKCS#12 truststore file for validating server certificate.

-w | --bindPassword[:env|:file] {bindPassword}

Password to use to bind to the server. Omit this option while providing the bind DN to ensure that the command prompts for the password, rather than entering the password as a command argument.

-W | --keyStorePassword[:env|:file] {keyStorePassword}

Keystore password which will be used as the cleartext configuration value.

-X | --trustAll

Trust all server SSL certificates. 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

--noPropertiesFile

No properties file will be used to get default command line argument values. Default: false

--propertiesFilePath {propertiesFilePath}

Path to the file containing default property values used for command line arguments.

-Q | --quiet

Use quiet 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.

> 0

An error occurred.

supportextract

`supportextract` — extract support data

Synopsis

```
supportextract {options}
```

Description

This tool collects support data from the OpenDJ instance it is bound to.

Options

The `supportextract` command takes the following options:

Command options:

`-d | --outputDirectory {directory}`

The folder into which the files will be placed into.

`--logsAfterDate {date}`

Collect log files after this date. Format "YYYYMMDDhhmmss" like "20161123143612" = 23 November 2016, 14:36 12s. Overrides --maxLogFiles.

`--maxLogFiles {number}`

Maximum number of log files to collect. Ignored if --logsAfterDate is provided. Default: 100

`--needJavaHeapDump`

Specifies whether a Java Heap Dump (using jmap) should be produced. The binary file is generated at the same location as the ZIP archive before being added to it; please make sure that the target directory's volume has sufficient capacity. Default: false

--noAuditFiles

Specifies whether audit files are excluded. Default: false

--noKeystoreFiles

Specifies whether keystore files are excluded. Default: false

--noServerInteraction

Specifies that the tool should not interact with the server, that is no LDAP operation, and no jstack sampling. Default: false

--serverPID {pid}

When the server is embedded in OpenAM, there is no PID file. Therefore this option indicates the server PID of the OpenAM application server.

-t | --jdkToolsDirectory {directory}

Path to the JDK utility binaries directory such as jstack. Default: /opt/graalvm-ce-java11-22.3.3/bin

LDAP connection options:

-D | --bindDn {bindDN}

DN to use to bind to the server. Default:

-w | --bindPassword[:env|:file] {password}

Password to use to bind to the server.

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.

> 0

An error occurred.

Examples

The following example creates a support archive in a custom directory:

```
$ supportextract \
 --bindDn uid=admin \
 --bindPassword password \
 --outputDirectory /path/to/output/directory

The instance is running
No value was provided for --jdkToolsDirectory, JDK tool directory is set to
</path/to/jdk/bin>
VERSION: <version>
/path/to/output/directory/data/dev/opendj-support-data-<timestamp>.zip.lock
Collecting the monitoring info from cn=monitor
Collecting process statistics
Cannot extract process statistics (by running "top" command) on OS '<OS>'.
Only jcmd dump samples will be collected
- Generating stack dump, sample number : 1 using jcmd for pid <pid>
- Generating stack dump, sample number : 2 using jcmd for pid <pid>
- Generating stack dump, sample number : 3 using jcmd for pid <pid>
- Generating stack dump, sample number : 4 using jcmd for pid <pid>
- Generating stack dump, sample number : 5 using jcmd for pid <pid>
- Generating stack dump, sample number : 6 using jcmd for pid <pid>
- Generating stack dump, sample number : 7 using jcmd for pid <pid>
- Generating stack dump, sample number : 8 using jcmd for pid <pid>
- Generating stack dump, sample number : 9 using jcmd for pid <pid>
- Generating stack dump, sample number : 10 using jcmd for pid <pid>
Collecting the configuration files
- Adding rootUser.ldif
- Adding monitorUser.ldif
- Adding schema files
- Adding HTTP configuration file(s)
- Listing the security stores
* config/keystore
Collecting system node information
- OS information
- Network information
- Disk information
- Processor information
Collecting ChangelogDb information
- No changelogDb data found (is a DS or is not replicated)
Collecting backend statistics
- amCts: total jdb files 1
- Adding je.info.0
- Adding je.config.csv
- Adding je.stat.csv
Collecting the log files
- /path/to/output/directory/logs/access
- /path/to/output/directory/logs/filtered-ldap-access.audit.json
- /path/to/output/directory/logs/ldap-access.audit.json
- /path/to/output/directory/logs/errors
- /path/to/output/directory/logs/replication
- /path/to/output/directory/logs/server.out
Collecting the GC log files
```

```
- /path/to/output/directory/logs/cust11.log.0
- /path/to/output/directory/logs/cust11.log

The following archive has been created :
/path/to/output/directory/data/dev/opendj-support-data-<timestamp>.zip
```

upgrade

`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 performs only part of the upgrade process, which includes the following phases for a single server:

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

> **Important**
>
> This utility *does not back up your data before you upgrade, nor does it restore your data if the utility fails* . In order to revert a failed upgrade, make sure you back up directory data 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

`--dataOnly`

Upgrades only application data. OpenDJ configuration must have been upgraded before. 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 performed automatically. 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.

verify-index

`verify-index` — check index for consistency or errors

Synopsis

```
verify-index {options}
```

Description

This utility ensures that index data is consistent within an indexed backend database. Stop the server before running this tool.

Options

The `verify-index` command takes the following options:

Command options:

`-b | --baseDn {baseDN}`

Base DN of a backend supporting indexing. Verification is performed on indexes within the scope of the given base DN.

`-c | --clean`

Specifies that a single index should be verified to ensure it is clean. An index is clean if each index value references only entries containing that value. Only one index at a time may be verified in this way. Default: false

`--countErrors`

Count the number of errors found during the verification and return that value as the exit code (values > 255 will be reduced to 255 due to exit code restrictions). Default: false

-i | --index {index}

Name of an index to be verified. For an attribute index this is simply an attribute name. Multiple indexes may be verified for completeness, or all indexes if no indexes are specified. An index is complete if each index value references all entries containing that value.

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.

1

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.

10

The command found errors in the index, but the `--countErrors` option was not specified.

0-255

When the `--countErrors` option is specified, the exit code indicates the number of errors found.

windows-service

`windows-service` — register DS as a Windows Service

Synopsis

```
windows-service options
```

Description

This utility can be used to run the server as a Windows Service.

Service options

`-c, --cleanupServiceserviceName`

Disable the service and clean up the windows registry information associated with the provided service name

`-d, --disableService`

Disable the server as a Windows service and stop the server

`-e, --enableService`

Enable the server as a Windows service

`-s, --serviceState`

Provide information about the state of the server as a Windows service

General options

`-V, --version`

Display version information

`-?, -H, --help`

Display usage information

Exit codes

0

The command completed successfully.

> 0

An error occurred.