PingDS

June 5, 2025

PINGDS

Version: 7.5

Table of Contents

Start here

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

Product names changed when ForgeRock became part of Ping Identity. PingDS was formerly known as ForgeRock Directory Services, for example. Learn more about the name changes in New names for ForgeRock products in the Knowledge Base.

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, refer to 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 Learn HDAP, make sure the `curl` command is available.

   For details, refer to 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 Ping Identity Backstage.
3. Find and download the latest PingDS ZIP distribution.

Install a directory server

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

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

   For example:

Bash

```
$ unzip ~/Downloads/DS-7.5.2.zip -d /path/to
```

PowerShell

```
PS C:\path\to> Expand-Archive DS-7.5.2.zip C:\path\to
```

This example installs DS files with the cross-platform zip. When using the native installer, refer to Use the Windows MSI.

Zsh

```
% unzip ~/Downloads/DS-7.5.2.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.

> **Important**
>
> You must have write access to the folder where you install DS.

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's unique across servers in your deployment.

`--deploymentId <deployment-id>`

The *deployment ID* is a random string generated using the `dskeymgr` command. It's 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 all administrative operations and isn't subject to access control. It's called the *root user* due to the similarity to the Linux 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 default name is `uid=admin`.

`--monitorUserPassword password`

The monitor user has the privilege to read monitoring data. This example doesn't set the `--monitorUserDn` option, 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`. Use StartTLS to secure connections to this port. The connections aren't 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`. Secure connections to this port 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. Secure connections to this port 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.

This profile adds entries under the base DN `dc=example,dc=com`. A base DN is the suffix shared by all DNs in a set of directory data.

A directory arranges LDAP entries hierarchically. 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 that 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 doesn't start the server. This lets you complete any necessary configuration steps before starting the server for the first time, which may start 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.

You can also 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, refer to 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, refer to 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, refer to 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, refer to LDAP updates.

Learn HDAP

PingDS let you access LDAP data over HTTP using **H**TTP **D**irectory **A**ccess **P**rotocol (HDAP) APIs that transform HTTP operations into LDAP operations.

Before you try the examples, follow the instructions in Install DS.

Prepare

Get the deployment CA certificate to trust the server:

Bash

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

PowerShell

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 to 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`.

Zsh

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

Create

Use HDAP to create a user resource:

Bash

```
$ curl \
--request POST \
--cacert ca-cert.pem \
--user uid=admin:password \
--header 'Content-Type: application/json' \
--data '{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
}' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People?_prettyPrint=true'

{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
}
```

JavaScript

```
const { doRequest, getOptions } = require('./utils')

const options = getOptions({
    path: '/hdap/dc=com/dc=example/ou=People?_action=create',
    credentials: 'uid=admin:password',
    method: 'POST',
    body: {
        "_id": "dc=com/dc=example/ou=People/uid=newuser",
        "objectClass": ["person", "inetOrgPerson", "organizationalPerson", "top"],
        "cn": ["New User"],
        "givenName": ["New"],
        "mail": ["newuser@example.com"],
        "manager": ["dc=com/dc=example/ou=People/uid=bjensen"],
        "sn": ["User"],
        "telephoneNumber": ["+1 408 555 1212"],
        "uid": ["newuser"]
    }
})

doRequest('HDAP: create with POST', options)
    .then(response => { console.log(response) })
    .catch(error => { console.error(error) })
```

Source files for this sample: create-newuser.js, utils.js

PowerShell

```
PS C:\path\to> $Credentials =
[System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes("uid=admin:password"))
$Headers = @{
    Authorization = "Basic $Credentials"
}
Invoke-RestMethod `
 -Uri https://localhost:8443/hdap/dc=com/dc=example/ou=People `
 -Method Post `
 -Headers $Headers `
 -ContentType application/json `
 -Body @"
 {
    "_id" : "dc=com/dc=example/ou=People/uid=newuser",
    "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
    "cn" : [ "New User" ],
    "givenName" : [ "New" ],
    "mail" : [ "newuser@example.com" ],
    "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
    "sn" : [ "User" ],
    "telephoneNumber" : [ "+1 408 555 1212" ],
    "uid" : [ "newuser" ]
 }
"@ | ConvertTo-JSON

{
    "_id" : "dc=com/dc=example/ou=People/uid=newuser",
    "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
    "cn" : [ "New User" ],
    "givenName" : [ "New" ],
    "mail" : [ "newuser@example.com" ],
    "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
    "sn" : [ "User" ],
    "telephoneNumber" : [ "+1 408 555 1212" ],
    "uid" : [ "newuser" ]
}
```

Python

```
#!/usr/bin/env python3

import requests
from requests.auth import HTTPBasicAuth
import utils

body = {
    '_id': 'dc=com/dc=example/ou=People/uid=newuser',
    'objectClass': ['person', 'inetOrgPerson', 'organizationalPerson', 'top'],
    'cn': ['New User'],
    'givenName': ['New'],
    'mail': ['newuser@example.com'],
    'manager': ['dc=com/dc=example/ou=People/uid=bjensen'],
    'sn': ['User'],
    'telephoneNumber': ['+1 408 555 1212'],
    'uid': ['newuser']
}
headers = { 'Content-Type': 'application/json' }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    auth=HTTPBasicAuth('uid=admin', 'password'),
    headers=headers,
    json=body,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, create-newuser.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('', '')
options = { ca_file: utils.ca_pem }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, :basic, 'uid=admin', 'password'
end
body = {
    '_id' => "dc=com/dc=example/ou=People/uid=newuser",
    'objectClass' => ["person", "inetOrgPerson", "organizationalPerson", "top"],
    'cn' => ["New User"],
    'givenName' => ["New"],
    'mail' => ["newuser@example.com"],
    'manager' => ["dc=com/dc=example/ou=People/uid=bjensen"],
    'sn' => ["User"],
    'telephoneNumber' => ["+1 408 555 1212"],
    'uid' => ["newuser"]
}
response = hdap.post do |h|
    h.path = 'dc=com/dc=example/ou=People'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, create-newuser.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Zsh

```
% curl \
--request POST \
--cacert ca-cert.pem \
--user uid=admin:password \
--header 'Content-Type: application/json' \
--data '{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
}' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People?_prettyPrint=true'

{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
}
```

- 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`. HDAP uses the DN and password to perform a simple LDAP bind for authentication. The directory uses its LDAP-based access control mechanisms to authorize the operation.
- The successful response is the JSON resource that the command created.

  Fields names starting with an underscore like `_id` are reserved. For details, refer to HDAP API reference.

For additional details, refer to HDAP API reference and Create.

Read

Use HDAP to read a user resource:

Bash

```
$ curl \
--request GET \
--cacert ca-cert.pem \
--user dc=com/dc=example/ou=People/uid=bjensen:hifalutin \
--header 'Content-Type: application/json' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser?_prettyPrint=true'

{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "_rev" : "<revision>",
  "objectClass" : [ "top", "person", "organizationalPerson", "inetOrgPerson" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
}
```

JavaScript

```
const { doRequest, getOptions } = require('./utils')

const options = getOptions({
    path: '/hdap/dc=com/dc=example/ou=People/uid=newuser'
})

doRequest('HDAP: read with GET', options)
    .then(response => { console.log(response) })
    .catch(error => { console.error(error) })
```

Source files for this sample: read-newuser.js, utils.js

PowerShell

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

{
    "_id" : "dc=com/dc=example/ou=People/uid=newuser",
    "_rev" : "<revision>",
    "objectClass" : [ "top", "person", "organizationalPerson", "inetOrgPerson" ],
    "cn" : [ "New User" ],
    "givenName" : [ "New" ],
    "mail" : [ "newuser@example.com" ],
    "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
    "sn" : [ "User" ],
    "telephoneNumber" : [ "+1 408 555 1212" ],
    "uid" : [ "newuser" ]
}
```

Python

```
#!/usr/bin/env python3

import requests
from requests.auth import HTTPBasicAuth
import utils

response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=newuser',
    auth=HTTPBasicAuth('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery'),
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, read-newuser.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('', '')
options = { ca_file: utils.ca_pem }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, :basic, 'dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin'
end
response = hdap.get('dc=com/dc=example/ou=People/uid=newuser')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, read-newuser.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Zsh

```
% curl \
--request GET \
--cacert ca-cert.pem \
--user dc=com/dc=example/ou=People/uid=bjensen:hifalutin \
--header 'Content-Type: application/json' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser?_prettyPrint=true'

{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "_rev" : "<revision>",
  "objectClass" : [ "top", "person", "organizationalPerson", "inetOrgPerson" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
}
```

Authenticate when making 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, refer to HDAP API reference and Read. You can also query collections of resources, as described in Query.

Update

Use HDAP to update a user resource:

Bash

```
$ curl \
--request PUT \
--cacert ca-cert.pem \
--user uid=admin:password \
--header 'Content-Type: application/json' \
--header "If-Match: *" \
--data '{
  "cn" : [ "Updated User" ],
  "givenName" : [ "Updated" ],
  "mail" : [ "updated.user@example.com" ],
  "telephoneNumber" : [ "+1 234 567 8910" ]
}' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser?_prettyPrint=true'

{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "top", "person", "organizationalPerson", "inetOrgPerson" ],
  "cn" : [ "Updated User" ],
  "givenName" : [ "Updated" ],
  "mail" : [ "updated.user@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 234 567 8910" ],
  "uid" : [ "newuser" ]
}
```

JavaScript

```
const { doRequest, getOptions } = require('./utils')

const options = getOptions({
    path: '/hdap/dc=com/dc=example/ou=People/uid=newuser',
    credentials: 'uid=admin:password',
    method: 'PUT',
    body: {
        "cn": ["Updated User"],
        "givenName": ["Updated"],
        "mail": ["updated.user@example.com"],
        "telephoneNumber": ["+1 234 567 8910"]
    }
})

doRequest('HDAP: update newuser', options)
    .then(response => { console.log(response) })
    .catch(error => { console.error(error) })
```

Source files for this sample: update-newuser.js, utils.js

PowerShell

```
PS C:\path\to> $Credentials =
[System.Convert]::ToBase64String([System.Text.Encoding]::ASCII.GetBytes("uid=admin:password"))
$Headers = @{
    "Authorization" = "Basic $Credentials"
    "If-Match"      = "*"
}
Invoke-RestMethod `
 -Uri https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser `
 -Method Put `
 -Headers $Headers `
 -ContentType application/json `
 -Body @"
 {
    "cn" : [ "Updated User" ],
    "givenName" : [ "Updated" ],
    "mail" : [ "updated.user@example.com" ],
    "telephoneNumber" : [ "+1 234 567 8910" ]
 }
"@ | ConvertTo-JSON

{
    "_id" : "dc=com/dc=example/ou=People/uid=newuser",
    "objectClass" : [ "top", "person", "organizationalPerson", "inetOrgPerson" ],
    "cn" : [ "Updated User" ],
    "givenName" : [ "Updated" ],
    "mail" : [ "updated.user@example.com" ],
    "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
    "sn" : [ "User" ],
    "telephoneNumber" : [ "+1 234 567 8910" ],
    "uid" : [ "newuser" ]
}
```

Python

```
#!/usr/bin/env python3

import requests
from requests.auth import HTTPBasicAuth
import utils

body = {
    'cn': ['Updated User'],
    'givenName': ['Updated'],
    'mail': ['updated.user@example.com'],
    'telephoneNumber': ['+1 234 567 8910']
}
headers = { 'Content-Type': 'application/json' }
response = requests.put(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=newuser',
    auth=HTTPBasicAuth('uid=admin', 'password'),
    headers=headers,
    json=body,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, update-newuser.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('', '')
options = { ca_file: utils.ca_pem }
fields = { '_fields': 'telephoneNumber' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: fields, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, :basic, 'uid=admin', 'password'
end
body = {
    "cn" => ["Updated User"],
    "givenName" => ["Updated"],
    "mail" => ["updated.user@example.com"],
    "telephoneNumber" => ["+1 234 567 8910"]
}
response = hdap.put do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=newuser'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, update-newuser.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Zsh

```
% curl \
--request PUT \
--cacert ca-cert.pem \
--user uid=admin:password \
--header 'Content-Type: application/json' \
--header "If-Match: *" \
--data '{
  "cn" : [ "Updated User" ],
  "givenName" : [ "Updated" ],
  "mail" : [ "updated.user@example.com" ],
  "telephoneNumber" : [ "+1 234 567 8910" ]
}' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser?_prettyPrint=true'

{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "top", "person", "organizationalPerson", "inetOrgPerson" ],
  "cn" : [ "Updated User" ],
  "givenName" : [ "Updated" ],
  "mail" : [ "updated.user@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 234 567 8910" ],
  "uid" : [ "newuser" ]
}
```

HDAP versions resources with revision numbers. A revision is specified in the resource's `_rev` field.

The `--header "If-Match: *"` tells HDAP to replace the resource regardless of its revision. Alternatively, set `--header "If-Match: revision"` to replace the resource only if its revision matches.

For additional details, refer to HDAP API reference and Update. You can also patch resources instead of replacing them entirely. Refer to Patch.

Delete

Use HDAP to delete a user resource:

Bash

```
$ curl \
--request DELETE \
--cacert ca-cert.pem \
--user uid=admin:password \
--header 'Content-Type: application/json' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser?_prettyPrint=true'

{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "_rev" : "<revision>",
  "objectClass" : [ "top", "person", "organizationalPerson", "inetOrgPerson" ],
  "cn" : [ "Updated User" ],
  "givenName" : [ "Updated" ],
  "mail" : [ "updated.user@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 234 567 8910" ],
  "uid" : [ "newuser" ]
}
```

JavaScript

```
const { doRequest, getOptions } = require('./utils')

const options = getOptions({
    path: '/hdap/dc=com/dc=example/ou=People/uid=newuser',
    credentials: 'uid=admin:password',
    method: 'DELETE'
})

doRequest('HDAP: delete newuser', options)
    .then(response => { console.log(response) })
    .catch(error => { console.error(error) })
```

Source files for this sample: delete-newuser.js, utils.js

PowerShell

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

{
    "_id" : "dc=com/dc=example/ou=People/uid=newuser",
    "_rev" : "<revision>",
    "objectClass" : [ "top", "person", "organizationalPerson", "inetOrgPerson" ],
    "cn" : [ "Updated User" ],
    "givenName" : [ "Updated" ],
    "mail" : [ "updated.user@example.com" ],
    "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
    "sn" : [ "User" ],
    "telephoneNumber" : [ "+1 234 567 8910" ],
    "uid" : [ "newuser" ]
}
```

Python

```
#!/usr/bin/env python3

import requests
from requests.auth import HTTPBasicAuth
import utils

response = requests.delete(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=newuser',
    auth=HTTPBasicAuth('uid=admin', 'password'),
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, delete-newuser.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('', '')
options = { ca_file: utils.ca_pem }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, :basic, 'uid=admin', 'password'
end
response = hdap.delete('dc=com/dc=example/ou=People/uid=newuser')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, delete-newuser.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Zsh

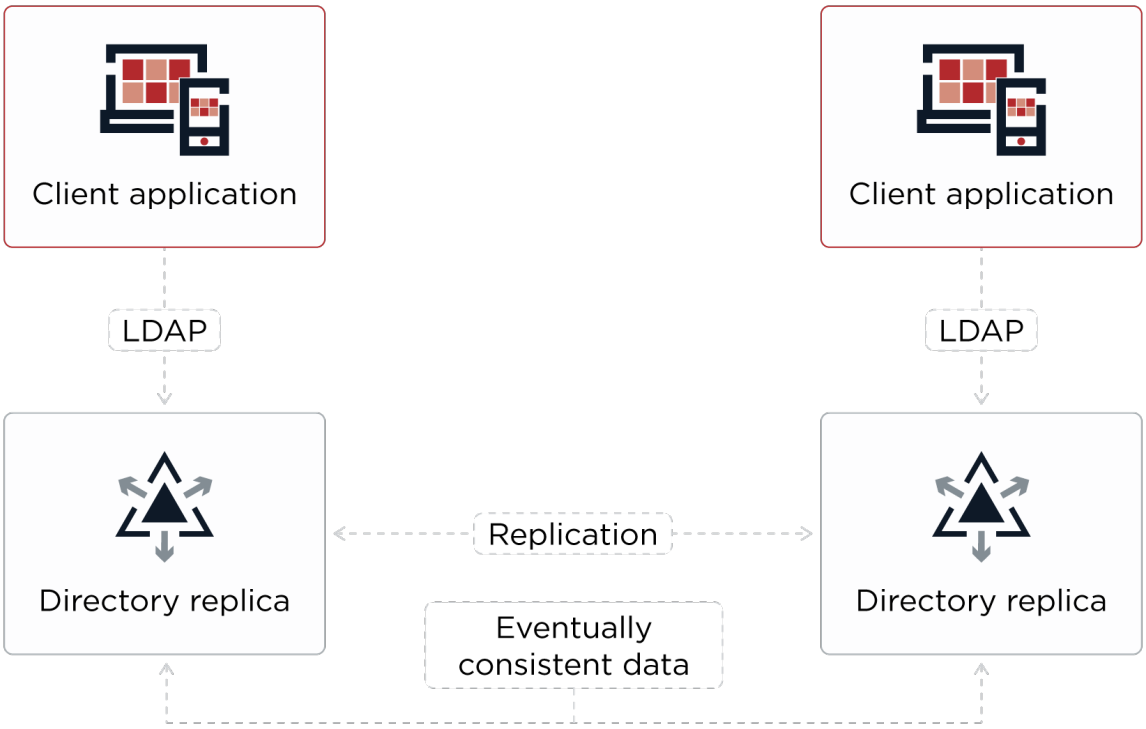
```
% curl \
--request DELETE \
--cacert ca-cert.pem \
--user uid=admin:password \
--header 'Content-Type: application/json' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser?_prettyPrint=true'

{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "_rev" : "<revision>",
  "objectClass" : [ "top", "person", "organizationalPerson", "inetOrgPerson" ],
  "cn" : [ "Updated User" ],
  "givenName" : [ "Updated" ],
  "mail" : [ "updated.user@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 234 567 8910" ],
  "uid" : [ "newuser" ]
}
```

For additional details, refer to HDAP API reference 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, refer to Replication and the related pages.

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-7.5.2.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-7.5.2.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-7.5.2.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, show 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
```

Show 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. Show this to check your understanding:

1. Stop the first server.

   Use the `stop-ds` command.

2. Modify an entry on the second server.

   Refer to 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.

   Refer to Search.

Notifications

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

Replicated DS directory servers publish an external change log over LDAP. This changelog lets authorized client applications 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, refer to 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, refer to 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, refer to 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, refer to 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 2020-2023 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, refer to the examples in Learn LDAP.

ACI syntax is powerful, and sometimes difficult to get right. For details, refer to 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.

PingDS implements the Lightweight Directory Access Protocol (LDAP). Nearly all of what follows is an introduction to LDAP.

PingDS 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:

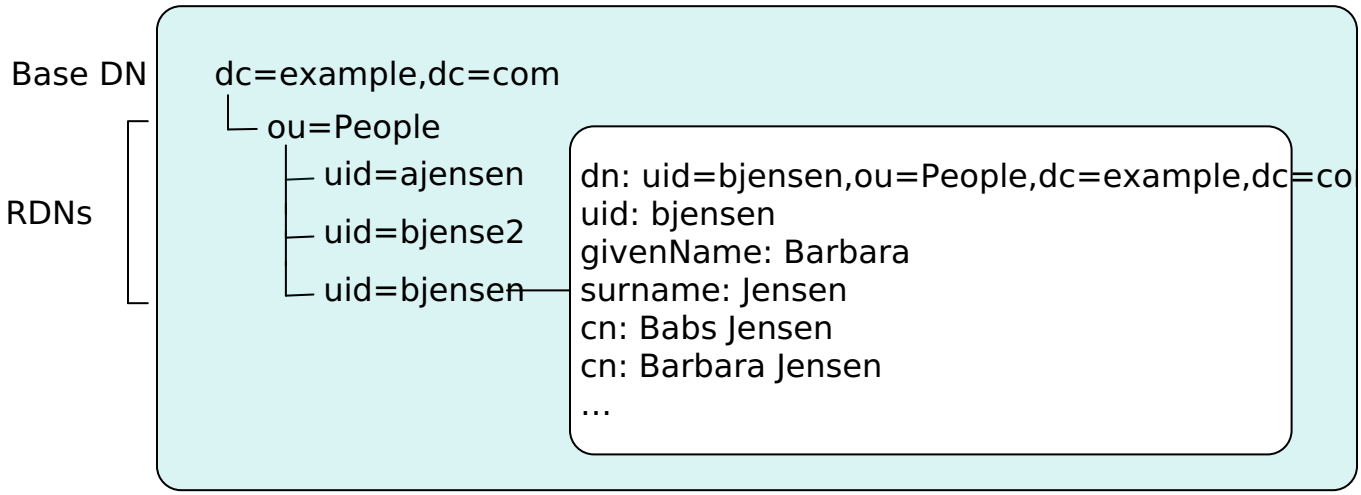
Bash

```
$ 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 C:\path\to\opendj\config\keystore `
 --trustStorePassword:file C:\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 takes 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, refer to 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, refer to 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 with understanding and managing indexes, read Indexes and the related pages.

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, refer to 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.

> **Important**
>
> The interface stability of this feature is *Deprecated*.

To set up DSMLv2 access, refer to Install a DSML gateway.

HTTP access

DS software maps LDAP data as JSON resources over HTTP for REST clients (HDAP).

LDAP schemas define the HDAP data model:

- LDAP entries hold sets of attributes, not arbitrarily nested objects.

  Each HDAP resource is an JSON object with fields at the top level:

  ```
  {
    "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
    "_rev" : "<revision>",
    "mail" : [ "bjensen@example.com" ],
    "cn" : [ "Barbara Jensen", "Babs Jensen" ],
    "sn" : [ "Jensen" ]
  }
  ```

- JSON has arrays, ordered collections that can contain duplicates.

  LDAP attributes are sets, unordered collections without duplicates.

  HDAP arrays have set semantics in which no duplicates are allowed, and the element order is arbitrary.

If you want a field with nested JSON or an array instead of a set, define a `json` syntax attribute. For details, refer to Schema and JSON.

You can deploy HDAP as a separate HDAP gateway servlet or through an HTTP connection handler on a DS server.

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 PingDS `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, PingDS 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":"<clientIp>","port":12345},"server":{"ip":"<serverIp>","port":1636},"request":
{"protocol":"LDAPS","operation":"CONNECT","connId":4},"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":"<serverIp>","port":1636},"request":
{"protocol":"LDAPS","operation":"TLS","connId":4},"transactionId":"0","response":
{"status":"SUCCESSFUL","statusCode":"0","elapsedTime":0,"elapsedTimeUnits":"MILLISECONDS"},"security":
{"protocol":"TLSv1.3","cipher":"TLS_AES_128_GCM_SHA256","ssf":128},"timestamp":"<timestamp>","_id":"<uuid>"}
{"eventName":"DJ-LDAP","client":{"ip":"<clientIp>","port":12345},"server":{"ip":"<serverIp>","port":1636},"request":
{"protocol":"LDAPS","operation":"BIND","connId":4,"msgId":1,"version":"3","dn":"cn=My
App,ou=Apps,dc=example,dc=com","authType":"SIMPLE"},"transactionId":"<uuid>","response":
{"status":"SUCCESSFUL","statusCode":"0","elapsedTime":1,"elapsedQueueingTime":0,"elapsedProcessingTime":
1,"elapsedTimeUnits":"MILLISECONDS","additionalItems":{"ssf":128}},"userId":"cn=My
App,ou=Apps,dc=example,dc=com","timestamp":"<timestamp>","_id":"<uuid>"}
{"eventName":"DJ-LDAP","client":{"ip":"<clientIp>","port":12345},"server":{"ip":"<serverIp>","port":1636},"request":
{"protocol":"LDAPS","operation":"SEARCH","connId":4,"msgId":
2,"dn":"dc=example,dc=com","scope":"sub","filter":"(uid=kvaughan)","attrs":
["isMemberOf"]},"transactionId":"<uuid>","response":{"status":"SUCCESSFUL","statusCode":"0","elapsedTime":
3,"elapsedQueueingTime":0,"elapsedProcessingTime":3,"elapsedTimeUnits":"MILLISECONDS","nentries":1,"entrySize":
430},"userId":"cn=My App,ou=Apps,dc=example,dc=com","timestamp":"<timestamp>","_id":"<uuid>"}
{"eventName":"DJ-LDAP","client":{"ip":"<clientIp>","port":12345},"server":{"ip":"<serverIp>","port":1636},"request":
{"protocol":"LDAPS","operation":"UNBIND","connId":4,"msgId":
3},"transactionId":"<uuid>","timestamp":"<timestamp>","_id":"<uuid>"}
{"eventName":"DJ-LDAP","client":{"ip":"<clientIp>","port":12345},"server":{"ip":"<serverIp>","port":1636},"request":
{"protocol":"LDAPS","operation":"DISCONNECT","connId":4},"transactionId":"0","response":
{"status":"SUCCESSFUL","statusCode":"0","elapsedTime":0,"elapsedTimeUnits":"MILLISECONDS","reason":"Client
Unbind"},"timestamp":"<timestamp>","_id":"<uuid>"}
```


Notice these features of the messages:

- The request operation types appear in upper case.
- The messages track the client information and identify the specific sequence of operations with connection ID (`connId`) and message ID (`msgID`) numbers.
- The `elapsedTime` for the response indicates the total time to complete the request. The `elapsedQueueingTime` is the time the request waited in the queue. The `elapsedProcessingTime` is the time actively processing the request.
- A status code 0 corresponds to a successful result, as described in RFC 4511.

For details about the message format, refer to Access log format.

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 the related pages.

Browse DS documentation

Category	Topics Covered
Release notes	DS features, fixes, and known issues
Use cases	Implementing common use cases for directory services
Deployment	Deploying PingDS in on-premises and cloud environments
Installation	Installing DS software
Upgrade	Upgrading DS software
Configuration	Configuring DS servers after installation
Security	Ensuring a PingDS 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 HDAP	How to configure and use DS REST APIs for HTTP access (HDAP)
Configuration reference	The `dsconfig` subcommands and server configuration properties
DS Javadoc	Evolving LDAP SDK and server APIs, including common APIs
LDAP reference	LDAP-specific features of DS software
LDAP schema reference	All default LDAP schema, including monitoring attributes and object classes

Category	Topics Covered
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.

Ping Identity does not endorse or support third-party tools.

Use DS with AM

- Backend directory servers in the AM *Deployment planning* documentation
- Prepare external stores in the AM *Installation* documentation
- Configure CTS token stores in the AM *Core token service* documentation.
- You can install DS directory servers for use as external AM stores.

  For details, refer to Setup profiles.

Use DS with IDM

- External DS repository and Select a repository in the IDM *Installation* documentation

  Also refer to 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* documentation
- DS repository configuration and Mappings with a DS repository in the IDM *Object modeling* documentation
- Synchronize passwords with DS in the IDM *Password synchronization* documentation

Remove DS software

For details, refer to 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.

Related: 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.

Related: 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.

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.

Related: 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.

Related: 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.

Related: 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.

HDAP

Short for **H**TTP **D**irectory **A**ccess **P**rotocol.

HDAP is not a standard. HDAP is the name of the feature providing REST APIs and HTTP access to directory data. HDAP translates HTTP requests to LDAP requests and LDAP responses to HTTP responses.

HDAP gateway

Standalone HDAP web application.

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.

Related: 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.

Refer to RFC 2849.

LDAP URL

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

Refer to 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.

Refer to 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.

Related: 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 server

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

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

Refer to: 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 Linux `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.

Related: 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 get dynamic groups represented as 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.

Use cases

These pages show you how to implement common use cases for directory services. If directory services are new to you, first work through the exercises in Start here.

The use cases show how IAM administrator Pat works in the directory services lab environment to develop and test processes and procedures before scaling them up for staging and production deployments.

Note

These pages walk you through key aspects of DS. They're a great starting point but do not make you an expert. Follow along to improve your practical know how. Use what you learned to help answer the additional questions. Be sure read the related documentation and think through the answers to your questions before deploying directory services into production.

Replication

Replicated DS across regions.

Backup

Back up and restore DS data.

Disaster recovery

Recover quickly after a disaster.

Password storage

Use stronger password storage.

LDAP schema

Change LDAP schema definitions.

CTS store

Replicate AM CTS data.

Enforceable limits

Enforce limits to protect directory services.

Cross-region replication

Simulate deploying replicated DS servers across multiple regions.

Description

Estimated time to complete: 25 minutes

DS replication works well across LANs and WANs. While some large and very high-performance deployments could call for optimizations to reduce latency or network bandwidth to a minimum, most deployments don't need them.

> **Tip**
>
> If you are running in Kubernetes, there's an easier way. Try the ForgeOps Cloud Deployment Model (CDM) reference implementation instead.

In this use case, you:

- Set up DS servers as if you were replicating across the WAN to different regions.
- Validate DS replicates data changes as expected.
- Review additional options to optimize performance if necessary.

Goals

In completing this use case, you learn to:

- Set up DS servers.
- Share secrets to protect network connections and encrypted data.
- Use appropriate bootstrap replication servers.
- Show replication in action.

Example scenario

As a directory service administrator, Pat plans to deploy directory services in multiple locations for redundancy.

Pat plans to show other identity administrators how the deployment would look and discuss whether the deployment would call for any optimizations.

Prerequisites

Knowledge

Before you start:

- Make sure you are familiar with the command line on your operating system.
- If you're new to directory services, work through the examples to learn LDAP and to learn replication.

Deployment

When deploying replicated DS servers, be aware of these constraints:

Network

- All DS servers must be able to connect to each other; their network must be routed.
- Each server FQDN must be unique and resolvable by all other DS servers; don't reuse FQDNs across regions.
- To recover from network partitions without intervention, DS servers must connect often enough to replay each other's changes before the end of the replication purge delay (default: 3 days).

DS configuration

Each DS server must:

- Share the same deployment ID.
- Have a unique server ID.
- Be able to contact its bootstrap replication servers.

  A *bootstrap replication server* is one of the replication servers in a deployment other DS servers contact to discover all the other DS servers in the deployment.

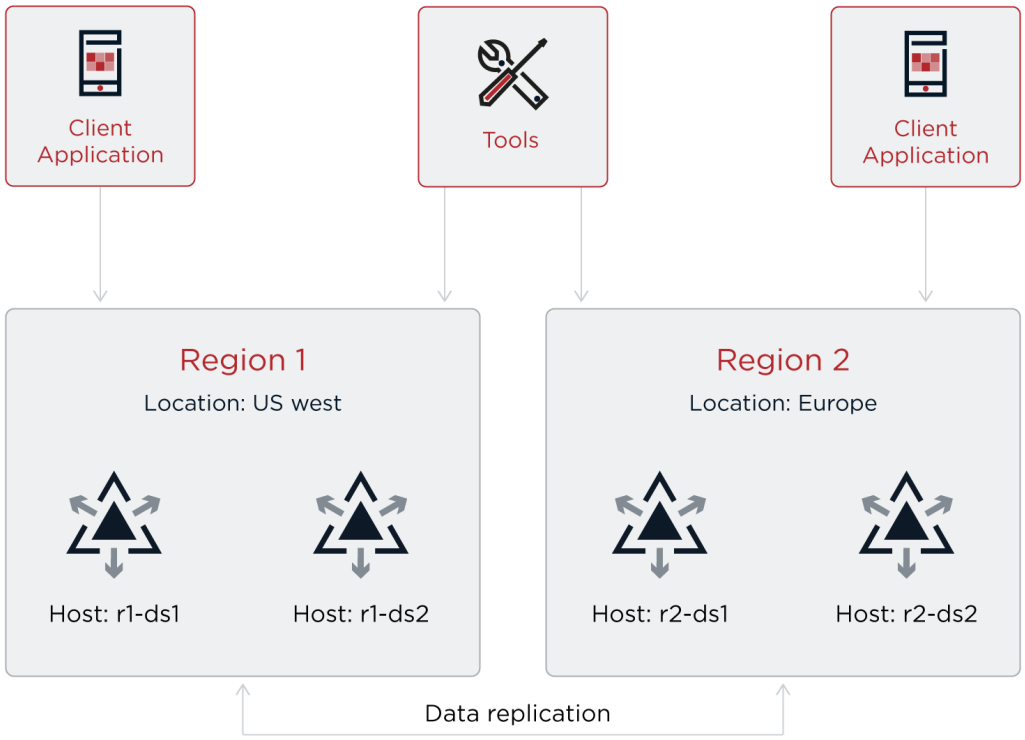
- Be able to verify and trust the digital certificates other DS servers use to establish their identities.

  DS tools must trust the server certificates to connect to DS servers securely. DS servers must trust each other's certificates to use secure connections for replication.

  This sample uses DS tools to simplify setting up a private PKI for this purpose. Your organization can use its own PKI in deployment.

Tasks

This sample deployment shows the steps to simulate a cross-region, replicated deployment on your computer. Use the same steps with geographically distributed computers or virtual machines for a real deployment.



- Two regions, each with two DS servers.
- The DS servers are fully meshed for replication; each server connects to the other server.
- You don't necessarily need this many DS servers. Two DS servers are the minimum for replication and availability. If the WAN has high bandwidth and low latency, one DS server per region is enough.
- DS servers function the same in a *simulated* cross-region deployment and an *actual* cross-region deployment.

  Replication requires distinct, stable server IDs and FQDNs. For replication, it doesn't matter whether the DS servers are on the same network interface or separated by a WAN.

Perform these tasks to simulate replicated DS servers across multiple regions.

Task 1: Prepare for installation

1. Make sure the DS server systems can connect to each other.

This sample simulates DNS on your computer by updating the hosts file with an alias for each DS server:

```
# Simulate DNS in a cross-region deployment
# with FQDN aliases for the loopback address:
127.0.0.1       r1-ds1.example.com
127.0.0.1       r1-ds2.example.com
127.0.0.1       r2-ds1.example.com
127.0.0.1       r2-ds2.example.com
```

When deploying in a production environment, make sure you have properly configured the DNS.

2. Unpack the DS server files once for each server to install.

   This sample uses folder locations aligned with the hostnames:

Base path	Description
`/path/to/r1-ds1`	Region 1, first server
`/path/to/r1-ds2`	Region 1, second server
`/path/to/r2-ds1`	Region 2, first server
`/path/to/r2-ds2`	Region 2, second server

3. Define the key configuration details for the deployment.

   This sample uses the following settings:

Server ID	Bootstrap replication servers
`r1-ds1`	`r1-ds1.example.com` `r2-ds1.example.com`
`r1-ds2`	
`r2-ds1`	
`r2-ds2`	

4. Define how the DS servers trust each other's certificates.

   This sample uses a private PKI based on the deployment ID. You generate a deployment ID for all DS servers using the `dskeymgr` command:

```
$ /path/to/r1-ds1/bin/dskeymgr \
 create-deployment-id \
 --deploymentIdPassword password
<deployment-id>
```

The deployment ID is a string. To use it, you must have the deployment ID password.

5. Determine the port numbers for the service.

   This sample uses different port numbers for each DS server because all the servers are on the same computer:

Sample server	Port numbers
`r1-ds1`	LDAP: 1389 LDAPS: 1636 HTTPS: 8443 Admin: 4444 Replication: 8989
`r1-ds2`	LDAP: 11389 LDAPS: 11636 HTTPS: 18443 Admin: 14444 Replication: 18989
`r2-ds1`	LDAP: 21389 LDAPS: 21636 HTTPS: 28443 Admin: 24444 Replication: 28989
`r2-ds2`	LDAP: 31389 LDAPS: 31636 HTTPS: 38443 Admin: 34444 Replication: 38989

   When installing each DS server on a different host, use the same port numbers everywhere.

Task 2: Install servers in "region 1"

Install servers in the first simulated region on your computer. In deployment, you would install each DS server on a separate host system:

1. Make sure you have the deployment ID required to install each DS server.

```
$ export DEPLOYMENT_ID=<deployment-id>
```

2. Install the first server in "region 1".

```
$ /path/to/r1-ds1/setup \
 --serverId r1-ds1 \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDn uid=admin \
 --rootUserPassword password \
 --monitorUserPassword password \
 --hostname r1-ds1.example.com \
 --ldapPort 1389 \
 --ldapsPort 1636 \
 --httpsPort 8443 \
 --adminConnectorPort 4444 \
 --replicationPort 8989 \
 --profile ds-evaluation \
 --bootstrapReplicationServer r1-ds1.example.com:8989 \
 --bootstrapReplicationServer r2-ds1.example.com:28989 \
 --start \
 --acceptLicense
```

3. Install the second server in "region 1".

```
$ /path/to/r1-ds2/setup \
 --serverId r1-ds2 \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDn uid=admin \
 --rootUserPassword password \
 --monitorUserPassword password \
 --hostname r1-ds2.example.com \
 --ldapPort 11389 \
 --ldapsPort 11636 \
 --httpsPort 18443 \
 --adminConnectorPort 14444 \
 --replicationPort 18989 \
 --profile ds-evaluation \
 --bootstrapReplicationServer r1-ds1.example.com:8989 \
 --bootstrapReplicationServer r2-ds1.example.com:28989 \
 --start \
 --acceptLicense
```

Task 3: Install servers in "region 2"

Install servers in the second simulated region on your computer. In deployment, you would install each DS server on a separate host system:

1. Make sure you have the deployment ID required to install each DS server.

```
$ export DEPLOYMENT_ID=<deployment-id>
```

2. Install the first server in "region 2".

```
$ /path/to/r2-ds1/setup \
 --serverId r2-ds1 \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDn uid=admin \
 --rootUserPassword password \
 --monitorUserPassword password \
 --hostname r2-ds1.example.com \
 --ldapPort 21389 \
 --ldapsPort 21636 \
 --httpsPort 28443 \
 --adminConnectorPort 24444 \
 --replicationPort 28989 \
 --profile ds-evaluation \
 --bootstrapReplicationServer r1-ds1.example.com:8989 \
 --bootstrapReplicationServer r2-ds1.example.com:28989 \
 --start \
 --acceptLicense
```

3. Install the second server in "region 2".

```
$ /path/to/r2-ds2/setup \
 --serverId r2-ds2 \
 --deploymentId $DEPLOYMENT_ID \
 --deploymentIdPassword password \
 --rootUserDn uid=admin \
 --rootUserPassword password \
 --monitorUserPassword password \
 --hostname r2-ds2.example.com \
 --ldapPort 31389 \
 --ldapsPort 31636 \
 --httpsPort 38443 \
 --adminConnectorPort 34444 \
 --replicationPort 38989 \
 --profile ds-evaluation \
 --bootstrapReplicationServer r1-ds1.example.com:8989 \
 --bootstrapReplicationServer r2-ds1.example.com:28989 \
 --start \
 --acceptLicense
```

Validation

Show updates to one simulated region getting replicated to the other region.

1. Modify an entry in the first region.

   The following command changes a description to `Description to replicate`:

```
$ /path/to/r1-ds1/bin/ldapmodify \
 --hostname r1-ds1.example.com \
 --port 1636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/r1-ds1/config/keystore \
 --trustStorePassword:file /path/to/r1-ds1/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: Description to replicate
EOF
```

2. Read the entry in the other region:

```
$ /path/to/r2-ds2/bin/ldapsearch \
 --hostname r2-ds2.example.com \
 --port 31636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/r2-ds2/config/keystore \
 --trustStorePassword:file /path/to/r2-ds2/config/keystore.pin \
 --bindDn uid=bjensen,ou=People,dc=example,dc=com \
 --bindPassword hifalutin \
 --baseDn dc=example,dc=com \
 "(uid=bjensen)" description
dn: uid=bjensen,ou=People,dc=example,dc=com
description: Description to replicate
```

Notice `description: Description to replicate` in the output.

You have shown replication works across regions.

What's next

After successfully showing the demonstration to other administrators, Pat doesn't stop there.

Pat leads the administrators to review the tradeoffs they can choose to make for the production deployment. Some of the questions to discuss include the following:

- Are there any applications we must direct to the nearest DS server on the network? For example, there could be applications with very low latency requirements, or the cost of network connections to remote servers could be much higher.

  If so, can those applications configure their own failover rules? Do we need a load balancer to do this for them?

- Do our DS replicas generate so much replication traffic that we should take steps to limit traffic between regions?

  If so, would standalone replication and directory servers be a good tradeoff? Should we configure replication group IDs to have directory servers in a region connect preferentially to replication servers in the same region?

- Should we use our own PKI to protect client-facing network connections over LDAP and HTTP?

This sample uses the server and CA certificates generated with the deployment ID and deployment ID password. You can set up DS with your own keys, using your own PKI to protect secure connections.

- How many DS servers do we really need?

  At a bare minimum, we need at least two DS servers to keep the service running while we upgrade, for example. The fewer servers we have, the easier it is to manage the service.

The answers to these questions depend on costs and service-level performance requirements. Don't optimize or pay extra for high performance unless you need it.

Explore further

Related use cases

- Backup and restore
- Disaster recovery

Reference material

Reference	Description
ForgeOps CDM	On Kubernetes? Use the ForgeOps CDM reference implementation instead.
Bootstrap replication servers	Configure bootstrap replication servers.
Cryptographic keys	Understand how DS uses secrets and keys.
Installation	Install directory services.
Install standalone servers (advanced)	Optimize network bandwidth for deployments with many servers.
On load balancers	Read this before configuring a load balancer.
Performance tuning	When performance is a concern, measure, tune, and test.
Replication	Background and procedures for working with DS replication.
Use your own cryptographic keys	Opt for your own PKI to protect network connections.

Backup and restore

Plan DS backup and restore procedures for your deployment.

Description

Estimated time to complete: 20 minutes

Safely and regularly back up your directory data to recover quickly when accidents happen.

In this use case, you:

- Back up directory data using DS tools.
- Cause an incident requiring recovery.
- Restore directory data after an incident.
- Validate the data restore procedure.

Goals

In completing this use case, you learn to:

- Use DS backup and restore tools.
- Schedule a recurring backup task.
- Restore directory data from backup files.
- Purge outdated backup files.

Example scenario

As a directory service administrator, Pat plans to deploy directory services for critical identity data such as login credentials.

Pat knows good backup and restore plans are a must for identity and access services. If the data is lost, end users cannot authenticate, and account profiles are lost.

Pat plans to show other identity administrators how the backup and restore procedures work and get them to review the process before deployment.

Prerequisites

Knowledge

Before you start, bring yourself up to speed with Pat:

- Pat is familiar with the command line on the target operating system, a Linux distribution in this example.
- Pat knows how to use basic LDAP commands, having worked examples to learn LDAP.
- Pat has already successfully completed directory service installation and setup procedures.

Actions

Before you try this example, set up two replicated DS directory servers on your computer as described in Install DS and Learn replication.

Tasks

Pat demonstrates how to back up and restore DS directory data from the evaluation profile. The order of the tasks is the same in deployment, but the directory data is different.

Task 1: Schedule a recurring backup operation

When you use the DS tools, backup operations are incremental. You can take regular backups with a reasonable amount of disk space relative to your data.

Configure backup tasks

1. Schedule a regular backup task.

   The following example schedules an hourly backup task:

   ```
   $ /path/to/opendj/bin/dsbackup \
    create \
    --backupLocation bak \
    --recurringTask "00 * * * *" \
    --description "Back up every hour" \
    --taskId HourlyBackup \
    --completionNotify diradmin@example.com \
    --errorNotify diradmin@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
   ```

2. Schedule a task to remove backup data older than the replication purge delay.

   When you restore data from backup, the backup you restore must be more recent than the replication purge delay. If you restore from older data, the replica you restore can't replicate with other servers. The default replication purge delay is three days.

   The following example schedules an hourly task to remove outdated backup data:

```
$ /path/to/opendj/bin/dsbackup \
 purge \
 --backupLocation bak \
 --recurringTask "00 * * * *" \
 --description "Purge old backups every hour" \
 --olderThan "3 days" \
 --taskId HourlyPurge \
 --completionNotify diradmin@example.com \
 --errorNotify diradmin@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
```

This task ensures you won't fill up the disk with old backup data.

(Optional) Back up data now

At this point, the recurring backup task is scheduled; however, the next backup operation won't start until the top of the hour. If you want to continue this example without waiting for the task to run, you can back up the data now.

These steps demonstrate offline backup:

1. Stop the server:

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

2. Back up the data with the server offline:

```
$ /path/to/opendj/bin/dsbackup \
 create \
 --backupLocation bak \
 --offline
```

The command writes the backup data to the `bak/` directory under the server installation directory.

3. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

Task 2: Simulate the loss of a server

1. Make sure you have at least one set of backup files:

```
$ /path/to/opendj/bin/dsbackup \
 list \
 --backupLocation bak \
 --offline
```

This command runs on the files and can run in offline mode even if the server is up.

If you are waiting for the hourly backup task to run, there may not be any backup files yet.

2. Simulate the loss of a server by stopping it abruptly and deleting the files.

   This example removes the `second-ds` server:

```
$ kill -9 <second-ds-pid>
$ rm -rf /path/to/replica
```

3. Change an entry.

   You use this change later to validate the restore procedure and show replication replays changes occurring after the last backup operation:

```
$ /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: Updated after the replica crashed
EOF
```

Task 3: Recover and restore the lost server

1. Replace the lost server with the same configuration but don't start it.

   This example uses the evaluation profile:

```
$ cd ~/Downloads && unzip ~/Downloads/opendj-7.5.2-20250513124640-de1088550ebabfaf1b3577b53f3eb9fc3b03739c.zip
&& mv opendj /path/to/replica
$ export DEPLOYMENT_ID=<deployment-id>
$ /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 \
 --set ds-evaluation/generatedUsers:0 \
 --acceptLicense
```

Rebuilding the basic server configuration depends on your deployment. For testing and deployment, adapt the commands to fit your process.

2. Restore the server data from backup:

```
$ /path/to/replica/bin/dsbackup \
 restore \
 --offline \
 --backendName dsEvaluation \
 --backupLocation /path/to/opendj/bak
```

3. Start the server:

```
$ /path/to/replica/bin/start-ds
```

After the server starts and connects to other servers, replication replays changes from after the backup operation.

Validation

Demonstrate the server you restored has the same data as the other replica.

1. Read the description you changed after the backup operation and server crash on `first-ds`:

```
$ /path/to/opendj/bin/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
dn: uid=bjensen,ou=People,dc=example,dc=com
description: Updated after the replica crashed
```

2. Read the same data on the `second-ds` server you restored from backup:

```
/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: Updated after the replica crashed
```

The data is the same on both servers. You have shown your backup and restore procedure is sound.

What's next

After demonstrating the process, Pat implements backup and restore procedures for testing and deployment. These procedures become part of the organization's runbook, so operators can implement them quickly and easily.

Pat realizes disaster recovery is more than restoring backup files. Pat also implements disaster recovery procedures for testing and deployment as part of the organization's runbook.

Explore further

This use case can serve as a template for DS test and production deployments. Adapt this example for deployment:

- Make sure the backup tasks run on more than one DS replica to avoid a single point of backup failure.
- To keep things simple, this example shows a backup on the local filesystem.

  In testing and deployment, make sure you store backup files remotely in a shared location. For example, consider backing up to cloud storage.

  A shared remote location for backup files makes it easier to restore from the same backup on multiple replicas.

- If the filesystem on your servers supports atomic snapshots, consider backing up DS with filesystem snapshots.

Related use cases

- Disaster recovery

Reference material

Reference	Description
Backup and restore	Includes detailed examples and alternatives for backing up and restoring directory data
Cryptographic keys	About keys, including those for encrypting and decrypting backup files
dsbackup	Reference for the command-line tool
Server tasks	On server tasks, like recurring backup operations

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 a service 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.

Description

Estimated time to complete: 30 minutes

In this use case, you:

- Back up a DS directory service.
- Simulate a disaster.
- Restore the service to a known state.
- Validate the procedure.

Goals

In completing this use case, you learn to:

- Back up and restore directory data.
- Restart cleanly from backup files to recover from a disaster.

Example scenario

Pat has learned how to install and configure replicated directory services and recognizes broken directory services could bring identity and access management services to a halt, too.

Pat understands replication protects directory services from single points of failure. However, what happens if a misbehaving application or a mistaken operator deletes all the user accounts, for example? Pat realizes replication replays the operations everywhere. In the case of an error like this, replication could amplify a big mistake into a system-wide disaster. (For smaller mistakes, refer to Recover from user error.)

Pat knows the pressure on the people maintaining directory services to recover quickly would be high. It would be better to plan for the problem in advance and to provide a scripted and tested response. No one under pressure should have to guess how to recover a critical service.

Pat decides to demonstrate a safe, scripted procedure for recovering from disaster:

- Start with a smoothly running, replicated directory service.
- Cause a "disaster" by deleting all the user accounts.
- Recover from the disaster by restoring the data from a recent backup.
- Verify the results.

Pat knows this procedure loses changes between the most recent backup operation and the disaster. Losing some changes is still better than a broken directory service. If Pat can discover the problem and repair it quickly, the procedure minimizes lost changes.

Prerequisites

Knowledge

Before you start, bring yourself up to speed with Pat:

- Pat is familiar with the command line and command-line scripting on the target operating system, a Linux distribution in this example. Pat uses shell scripts to automate administrative tasks.
- Pat knows how to use basic LDAP commands, having worked examples to learn LDAP.
- Pat has already scripted and automated the directory service installation and setup procedures. Pat already saves copies of the following items:
  - 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.

This example scenario focuses on the application and user data, not the directory setup and configuration. For simplicity, Pat chooses to demonstrate disaster recovery with two replicated DS servers set up for evaluation.

- Pat has a basic understanding of DS replication, including how replication makes directory data eventually consistent.

Actions

Before you try this example, set up two replicated DS directory servers on your computer as described in Install DS and Learn replication.

Tasks

Pat demonstrates this recovery procedure on a single computer. In deployment, the procedure involves multiple computers, but the order and content of the tasks remain the same.

> **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?

- 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.

1. Back up the 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, you will 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: Simulate a disaster

1. Delete all user entries in the evaluation backend:

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

This command takes a few seconds to remove over 100,000 user entries. It takes a few seconds more for replication to replay all the deletions on the other DS replica.

Why is this a disaster? Suppose you restore a DS replica from the backup to recreate the missing user entries. After the restore operation finishes, replication replays each deletion again, ensuring the user entries are gone from all replicas.

Although this example looks contrived, it is inspired by real-world outages. You cannot restore the entries permanently without a recovery procedure.

Task 3: Recover from the 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`.

3. Record the ID of the last good backup before the disaster.

   The following command lists backups for the `dsEvaluation` backend, the one affected by the disaster:

```
$ /path/to/opendj/bin/dsbackup \
 list \
 --backupLocation /path/to/opendj/bak \
 --backendName dsEvaluation \
 --last \
 --offline
```

The output for a backup operation includes the date and the ID. Use the date shown in the output to find the ID of the last good backup:

```
…
Backend name:       dsEvaluation
Server ID:          first-ds
Backup Date:        17/Feb/2025 16:23:26 [Europe/Paris]
Backup ID:          dsEvaluation_20250217152326308
…
```

Make sure you find *the ID of the last backup before the disaster*. This isn't necessarily the same ID as the last successful backup. If the disaster only broke your data, not the service, the last successful backup could've run after the disaster.

4. Make sure the files for the last backup before the disaster are available to all DS servers you'll recover.

   Disaster recovery includes restoring all directory server replicas from the same good backup files.

   If you store backup files locally on each server, copy the backup files to each server. You can optionally purge backup files you won't use to avoid copying more files than necessary. Only use the backup from the backup files you copied and not the local backups on the server you're recovering. You'll recover the server from the good backup files and won't use the local files.

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.

   The following command restores data from the last good backup based on the ID you found in Prepare for recovery:

```
$ /path/to/opendj/bin/dsbackup \
 restore \
 --offline \
 --backupId ${BACKUP_ID} \
 --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.

   The new backup is for potential future recoveries, not the current disaster recovery. To be safe, take new backups as soon as you allow external applications to make changes again.

   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:

- Access to the good backup files and the ID you found for the last good backup in Prepare for recovery.
  Use the last good backup from before the disaster for all servers to recover.
- The 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 last good backup files you used for the first server:

```
$ /path/to/replica/bin/dsbackup \
 restore \
 --offline \
 --backupId ${BACKUP_ID} \
 --backupLocation /path/to/opendj/bak
```

3. Run the recovery command.

   The following command uses a generated ID. It verifies this server's data match 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 a user 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.

What's next

Example scenario

With the plan for disaster recovery off to a good start, Pat's next steps are to:

- Develop tests and detailed procedures for recovering from a disaster in deployment.
- Put in place backup plans for directory services.

  The backup plans address these more routine maintenance cases and keep the directory service running smoothly.
- Document the procedures in the deployment runbook.

Explore further

This use case can serve as a template for DS test and production deployments. Adapt this example for deployment:

- Back up files as a regularly scheduled task to ensure you always have a recent backup of each backend.
- Regularly export the data to LDIF from at least one DS replica in case all backups are lost or corrupted. This LDIF serves as a last resort when you can't recover the data from backup files.
- Store the backup files remotely with multiple copies in different locations.
- Purge old backup files to avoid filling up the disk space.
- Be ready to restore each directory database backend.

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.

Related use cases

- Backup and restore

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
Installation	Examples you can use when scripting installation procedures
Configuration	Examples you can use when scripting server configuration

Change password storage

What seemed a secure password storage scheme a few years ago no longer looks safe. You can configure DS to migrate to stronger password storage.

Description

Estimated time to complete: 30 minutes

With a reversible encryption scheme, an attacker who gains access to the server files can recover all the plaintext passwords. With a strong one-way hash scheme, the attacker must use brute force methods for each password.

However, not all one-way hash schemes are safe, either. Older password storage schemes, such as the salted Secure Hash Algorithm (SHA-1) schemes, use one-way hash functions designed for message authentication and digital signatures. SHA-1 schemes are fast; a server processes authentications with low latency and high throughput. On the downside, high-performance algorithms also make brute force attack techniques more effective. Modern off-the-shelf GPUs can calculate billions of SHA-1 hashes per second. Dedicated hardware can calculate even more hashes per second.

Goals

In completing this use case, you learn to:

- Discover password policies using outdated password storage schemes.
- List accounts using outdated password storage schemes.
- Create a replicated password policy and configure its password storage scheme settings.
- Assign accounts a password policy.

Example scenario

The security team where Pat works has mandated passwords must be stored with a computationally intensive one-way hash, such as Argon2, Bcrypt, PBKDF2, or PKCS5S2.

Pat knows the default password storage scheme for the DS directory service has not changed in years. Many user accounts still have salted SHA-1-based password storage.

Pat considers the options and decides to move to a PBKDF2-based scheme. Pat plans to show how to switch to PBKDF2 and to get the other identity administrators to review the process.

At this point, the security team has not communicated a due date to implement the mandate. Pat expects the change to be transparent for users and application developers.

As a directory service administrator, Pat must work with the deployment team to make sure DS systems have enough CPU. PBKDF2 uses far more CPU resources than outdated storage schemes.

Prerequisites

Knowledge

Before you start, make sure you have the same background knowledge as Pat:

- Pat is familiar with the command line on the target operating system, a Linux distribution in this example.
- Pat knows how to use basic LDAP commands, having worked through the examples to learn LDAP.
- Pat has already successfully completed directory service installation and setup procedures.

Background

The problem

Sometimes, people ask why DS doesn't provide a tool to move passwords from one storage scheme to another.

Pat explains DS uses one-way hash functions to store passwords. These are *one-way* functions because going from a password to a hash is deterministic and straightforward. Going from a hash to a password is hard. For computationally intensive schemes like PBKDF2, going from a hash to a password is effectively impossible.

Even given the PBKDF2-based password hashes for all the accounts in the directory service, you'd spend plenty of money and computer resources cracking any of them to recover an original password.

Any tool to move passwords from one storage scheme to another must first crack every password hash. For this reason, DS does not provide such a tool, and there are no plans to develop one.

The solution

One possible solution is to change the storage scheme in password policies, disable the target storage schemes, and require users to reset the passwords for their accounts; however, this can be disruptive.

Pat knows a less disruptive solution is to wait until the next successful authentication, then let DS store the password with the new storage scheme.

When you authenticate with a DN and password—an LDAP simple bind—you supply the password. If the authentication succeeds, the password is valid. DS still has the password at this time, so it can hash the password according to the new scheme and remove the hash computed by the old scheme.

In DS, the password policy defines the storage scheme to use. As an administrator, Pat configures a password policy to deprecate the old scheme in favor of the new scheme. Pat then waits for accounts to bind and lets DS update the storage scheme.

Constraints

Waiting for accounts to bind is not a problem unless there are time constraints.

For example, if there's a mandate to move away from the deprecated scheme by a target date, then Pat will have to effectively lock "inactive" accounts. Those accounts must reset their passwords after the date.

As an administrator, Pat can implement this by disabling the deprecated password storage scheme on the target date. Accounts cannot bind with a password stored using a disabled scheme.

Pat knows to warn application owners and developers of end-user UIs and self-service account management tools "inactive" accounts cannot authenticate when their passwords still use the old scheme after the target date.

Applications can rely on account usability features to discover why LDAP binds fail. Developers of end-user tools can use the hints in their applications to reset user passwords and prompt users to set new passwords.

Tasks

Pat demonstrates how to change password storage with a single DS server using the evaluation profile. The order of the tasks is the same in deployment, but the target storage schemes can differ.

Pat shows the process with a subentry password policy. You create an LDAP subentry and DS replicates it to the other replicas. If you use per-server password policies instead, you must edit the configuration for each DS replica.

Task 1: Set up DS

1. Create a deployment ID to use when setting up DS:

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

2. Set up a single DS server using the evaluation profile with an outdated password storage scheme:

```
$ /path/to/opendj/setup \
 --serverId evaluation-only \
 --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 \
 --set ds-evaluation/useOutdatedPasswordStorage:true \
 --start \
 --acceptLicense
```

The `useOutdatedPasswordStorage` sets the password storage scheme for users to `Salted SHA-512`.

Task 2: List password policies using outdated schemes

To show the process, Pat deprecates the outdated storage scheme in favor of a new stronger storage scheme.

1. List all available password storage schemes:

```
$ /path/to/opendj/bin/dsconfig \
 list-password-storage-schemes \
 --hostname localhost \
 --port 4444 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --no-prompt
Password Storage Scheme : Type               : enabled
------------------------:--------------------:--------
3DES (LEGACY)           : triple-des         : false
AES (LEGACY)            : aes                : false
Argon2                  : argon2             : true
Base64 (LEGACY)         : base64             : false
Bcrypt                  : bcrypt             : true
Blowfish (LEGACY)       : blowfish           : false
Clear (LEGACY)          : 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 (LEGACY)   : salted-sha1        : false
Salted SHA-256          : salted-sha256      : false
Salted SHA-384          : salted-sha384      : false
Salted SHA-512          : salted-sha512      : true
SCRAM-SHA-256           : scram-sha256       : true
SCRAM-SHA-512           : scram-sha512       : true
SHA-1 (LEGACY)          : sha1               : false
```

Accounts cannot authenticate with a password if their password policy depends on a disabled password storage scheme. Only the enabled password storage schemes (`enabled: true`) matter for this procedure:

- `Argon2`
- `Bcrypt`
- `PBKDF2-HMAC-SHA256`
- `PBKDF2-HMAC-SHA512`
- `Salted SHA-512`
- `SCRAM-SHA-256`
- `SCRAM-SHA-512`

For this example, Pat migrates passwords away from `Salted SHA-512` . The others are stronger password storage schemes.

2. List the per-server password policies to identify any that use the outdated scheme.

```
$ /path/to/opendj/bin/dsconfig \
 list-password-policies \
 --hostname localhost \
 --port 4444 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --no-prompt
Password Policy         : Type            : password-attribute : default-password-storage-scheme
------------------------:-----------------:--------------------:-------------------------------
Default Password Policy : password-policy : userPassword       : Salted SHA-512
Root Password Policy    : password-policy : userPassword       : PBKDF2-HMAC-SHA256
```

The `Default Password Policy` uses the outdated storage scheme:

- The `Default Password Policy` applies to accounts in user and application data.
- The `Root Password Policy` applies to DS service accounts, such as the directory superuser (`uid=admin`).

3. List subentry password policies to check for any using the outdated scheme.

```
$ /path/to/opendj/bin/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 "" \
 "(&(objectClass=subEntry)(objectClass=ds-pwp-password-policy))"
```

The command returns nothing; DS has no subentry password policies configured for the evaluation profile.

Task 3: List accounts using outdated schemes

DS has a `userPassword` index to the directory entries using each password scheme.

1. List the accounts using the outdated scheme.

   This command uses a filter with an extensible match comparison, `1.3.6.1.4.1.36733.2.1.4.14:=Salted SHA-512`. The object identifier corresponds to password storage scheme quality match syntax. The filter matches entries whose password is stored with `Salted SHA-512` (`SSHA512`):

```
$ /path/to/opendj/bin/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 \
  "(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SSHA512)" 1.1
```

An attribute list of `1.1` means the search should not return attribute values, just DNs.

If you have multiple password policies with outdated storage schemes, search like this for each one.

The response can be empty, meaning no accounts use the storage scheme. If a password policy uses an outdated password storage scheme, but no accounts use it, update the password policy to deprecate the outdated scheme. Double-check the response is still empty, and disable the outdated scheme in each DS configuration to prevent its use.

2. If you want to check which password policy an account has, request `pwdPolicySubentry`:

```
$ /path/to/opendj/bin/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)" \
 pwdPolicySubentry
dn: uid=bjensen,ou=People,dc=example,dc=com
pwdPolicySubentry: cn=Default Password Policy,cn=Password Policies,cn=config
```

The `pwdPolicySubentry` has the DN of the applicable password policy for the entry. You could use `pwdPolicySubentry` instead of `1.1` in the previous step to show the attribute for each user.

Task 4: Move accounts to the new scheme

These steps deprecate `Salted SHA-512` in favor of `PBKDF2-HMAC-256`:

1. Configure a password policy to deprecate the outdated scheme.

```
$ /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=admin \
 --bindPassword password  << EOF
dn: cn=New password 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: New password policy
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-length-based-min-password-length: 8
subtreeSpecification: {base "", specificationFilter "(userPassword=*)" }
EOF
# ADD operation successful for DN cn=New password policy,dc=example,dc=com
```

The `subtreeSpecification` applies the password policy to all accounts under `dc=example,dc=com` with a `userPassword` attribute.

2. Check the new policies apply as expected.

   The following command shows the new policy applies to a user account:

```
$ /path/to/opendj/bin/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)" \
 pwdPolicySubentry userPassword
dn: uid=bjensen,ou=People,dc=example,dc=com
userPassword: {SSHA512}<hash>
pwdPolicySubentry: cn=New password policy,dc=example,dc=com
```

   The password is still hashed with the old scheme. The user hasn't authenticated since the password policy change.

3. Wait for accounts to bind with password-based authentication.

   You can check progress using the searches described in Task 3: List accounts using outdated schemes.

4. (Optional) When enough accounts have changed storage schemes, disable stale password policies and the outdated scheme.

Task 5: Plan any necessary communications

When you have no time constraints, there's nothing to communicate to application developers or end users. Make sure DS systems have the resources to process the stronger password policy; communicate about this with those providing systems for testing and deployment. Eventually, DS updates the password storage scheme for all active accounts.

If you have a due date to finish the move, you must disable the outdated scheme at that time:

```
$ /path/to/opendj/bin/dsconfig \
 set-password-storage-scheme-prop \
 --scheme-name "Salted SHA-512" \
 --set enabled:false \
 --hostname localhost \
 --port 4444 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password \
 --no-prompt
```

This has the effect of locking inactive accounts—those who didn't authenticate before the date—because they are stuck with the disabled storage scheme. An administrator must reset the passwords to activate the accounts.

- Plan with other identity administrators and identity application developers how to automate the password reset and change process to active locked accounts.

  In a Ping Identity Platform deployment, you can configure self-service features to help end users help themselves.

- If possible, let end users know they need to sign on before the due date to keep their accounts active.

  Let them know inactive accounts are locked out after the due date, and describe how they can activate their accounts after the lockout.

Validation

Display a user's password before and after authentication to confirm the policy causes DS to update how it stores the password.

1. Read a `userPassword` as directory superuser to display the password storage scheme:

   ```
   $ /path/to/opendj/bin/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)" \
    userPassword
   dn: uid=bjensen,ou=People,dc=example,dc=com
   userPassword: {SSHA512}<hash>
   ```

The attribute shows the password storage scheme in braces before the hash. The user has not authenticated since the policy change. The scheme is still `Salted SHA-512` (`SSHA512`).

2. Read the `userPassword` again *as the user* to display the password storage scheme:

```
$ /path/to/opendj/bin/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)" \
 userPassword
dn: uid=bjensen,ou=People,dc=example,dc=com
userPassword: {PBKDF2-HMAC-SHA256}10:<hash>
```

The `--bindDn` and `--bindPassword` indicate the user authenticates with an LDAP simple bind. DS updates the hash when the user authenticates. The scheme is now `PBKDF2-HMAC-SHA256`.

What's next

After demonstrating the process, Pat implements plans to deprecate outdated password storage schemes in deployment.

Pat is careful to make sure DS systems have the resources to process PBKDF2 hashes, in particular for binds. For example, Pat can use the `authrate` command to generate LDAP binds before and after the change. Pat can also review logs and monitoring data from the deployment to estimate peak bind rates.

> **Note**
>
> When you install DS, the `setup` command configures a `PBKDF2-HMAC-SHA256` password storage scheme with 10 iterations instead of the default 10,000 iterations.
> The server's default password policy uses this storage scheme.

When DS systems have sufficient resources, Pat can increase the number of iterations for the `PBKDF2-HMAC-SHA256` scheme; for example, setting `pbkdf2-iterations: 10000` and `rehash-policy: only-increase` in the `PBKDF2-HMAC-SHA256` scheme configuration. DS updates the password storage hash for an account on the next successful authentication.

Explore further

This use case can serve as a template for DS test and production deployments. Adapt this example for deployment:

- Review the password storage schemes used in deployment to determine what to change.
- Make sure the directory service has appropriate resources to sustain authentication rates after moving to a resource-intensive password storage scheme.
- Plan communications as necessary.

Reference material

Reference	Description
Passwords	DS password policy and storage schemes
Password Storage Scheme	Supported password storage schemes
Authentication (binds)	About LDAP bind operations
authrate	Performance tool for generating LDAP bind operations
Passwords and accounts, Actions	Client-side password and account management

Change LDAP schema

Learn how to change LDAP schema definitions online and offline.

Description

Estimated time to complete: 30 minutes

LDAP schema definitions determine the kinds of information in the directory and how the information is related. You can update the schema definitions online and offline to change what the directory allows.

Develop and test schema changes online to catch any errors in the updated definitions. After you validate the schema changes, you can deploy them online with the `ldapmodify` command or offline by copying updated schema files. Replication replays the LDAP schema changes to other DS servers.

In this use case, you:

- Understand a scenario where schema changes make sense.
- Understand how schema changes can require rebuilding indexes.
- Develop and test schema changes.
- Practice rolling out schema changes by copying updated schema files.

Goals

In completing this use case, you learn to:

- Use the `ldapmodify` command to change LDAP schema.
- Rebuild indexes affected by schema changes.
- Review and remove replication metadata from changed schema files.

Example scenario

One of the directory application owners asks Pat to let their application page through accounts by class of service.

Pat's directory deployment uses the definition for the `classOfService` attribute based on the evaluation profile.

Pat can add an index for the `classOfService` attribute, but wonders if the application owner has additional requirements. In discussion with the application owner, Pat learns the application owner:

- Found the class of service attribute can accept any random string value.

  They ask Pat if class of service could be restricted to an enumeration of `bronze`, `silver`, `gold`, and `platinum`.

- Wants a `sharedQuota` attribute like the `diskQuota` and `mailQuota` attributes.

  The application owner doesn't use `sharedQuota` yet, but plans to use it in a few weeks.

Prerequisites

Knowledge

Before you start:

- Make sure you are familiar with the command line on your operating system.
- If you're new to directory services, work through the examples to learn LDAP.

Actions

Before you try this example, install a DS server in evaluation mode.

Tasks

Pat shows the tasks with DS servers in evaluation mode. The order and content of the tasks for production deployments are the same.

Task 1: Add a `classOfService` index

The application owner wants to page through accounts by class of service. Class of service has only a few values, and every user account could have the attribute. This is a good match for a big index.

1. Create the index:

```
$ /path/to/opendj/bin/dsconfig \
 create-backend-index \
 --backend-name dsEvaluation \
 --index-name classOfService \
 --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
```

2. Build the new index:

```
$ /path/to/opendj/bin/rebuild-index \
 --baseDn dc=example,dc=com \
 --index classOfService \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

Applications can now use the simple paged results control to page through entries with a specified class of service.

Task 2: Develop schema changes

Pat notices the `classOfService` attribute has `SYNTAX 1.3.6.1.4.1.1466.115.121.1.15` (directory string syntax). Pat can change the schema definition to use a custom enumeration syntax, so DS only allows applications to set one of the desired values. Pat can update the schema again to extend the enumeration as necessary.

Pat also adds a new `sharedQuota` attribute modeled on the `diskQuota` and `mailQuota` attributes.

Pat knows DS rejects malformed online modifications to schema definitions. Pat develops and tests the schema changes with the `ldapmodify` command.

> **Important**
>
> When changing a schema definition, delete the existing value and add the new value as part of the same modification. Otherwise, there's a window after you delete a definition and before you add the new one where an update could fail or an index could become degraded due to the missing schema definition.
> The definition you delete must match the definition in the schema LDIF exactly, not including space characters.
> When you update schema definitions online, DS sets the `X-SCHEMA-FILE` value even if you don't.

1. Update the schema definitions.

   The following example command:

   - Adds an enumeration syntax for class of service
   - Updates the `classOfService` attribute to use the enumeration syntax

- Adds a `sharedQuota` attribute to the `cos` object class for class of service attributes

```
$ /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=admin \
 --bindPassword password << EOF
dn: cn=schema
changetype: modify
add: ldapSyntaxes
ldapSyntaxes: ( example-custom-syntax-oid
  DESC 'Enumeration syntax for class of service'
  X-ENUM ( 'bronze' 'silver' 'gold' 'platinum' )
  X-ORIGIN 'DS Documentation Examples' )
-
delete: 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-attribute-type
  NAME 'classOfService'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreOrderingMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX example-custom-syntax-oid
  SINGLE-VALUE
  USAGE userApplications
  X-ORIGIN 'DS Documentation Examples' )
-
add: attributeTypes
attributeTypes: ( example-class-of-service-shared-quota
  NAME 'sharedQuota'
  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' )
-
delete: objectClasses
objectClasses: ( example-class-of-service-object-class
  NAME 'cos'
  SUP top
  AUXILIARY
  MAY ( classOfService $ diskQuota $ mailQuota )
  X-ORIGIN 'DS Documentation Examples' )
-
add: objectClasses
objectClasses: ( example-class-of-service-object-class
  NAME 'cos'
```

```
  SUP top
  AUXILIARY
  MAY ( classOfService $ diskQuota $ mailQuota $ sharedQuota )
  X-ORIGIN 'DS Documentation Examples' )
EOF
```

2. Rebuild affected indexes.

   This update changes the `classOfService` syntax, so rebuild the index to use the new syntax:

```
$ /path/to/opendj/bin/rebuild-index \
 --baseDn dc=example,dc=com \
 --index classOfService \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

   If the enumeration syntax changes again, rebuild the `classOfService` index.

Task 3: Save changed schema files

For the production servers, Pat doesn't change the schema online. Pat keeps schema definition files under source control to track all schema changes.

After modifying the schema online, Pat locates the schema definitions added and changed in the `db/schema` LDIF files. Pat notices DS rewrites the updated LDIF files with one schema definition per line.

Before putting the changed LDIF files under source control, Pat takes care to remove the operational attributes including the `ds-sync-generation-id` and `ds-sync-state` attributes. Using the wrong values for those attributes could break schema replication. Pat lets DS replication manage the operational attributes.

In Pat's copies of the LDIF files, the schema definitions are folded for readability. Each line continuation starts with *two spaces* before a schema element keyword. LDIF continuation consumes the first space. The second space separates the keyword from the preceding text.

```
dn: cn=schema
objectclass: top
objectclass: ldapSubentry
objectclass: subschema
cn: schema
ldapSyntaxes: ( example-custom-syntax-oid
  DESC 'Enumeration syntax for class of service'
  X-ENUM ( 'bronze' 'silver' 'gold' 'platinum' )
  X-ORIGIN 'DS Documentation Examples'
  X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
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'
  X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
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'
  X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
attributeTypes: ( example-class-of-service-shared-quota
  NAME 'sharedQuota'
  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'
  X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
attributeTypes: ( json-attribute-oid
  NAME 'json'
  EQUALITY caseIgnoreJsonQueryMatch
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1
  X-ORIGIN 'DS Documentation Examples'
  X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
attributeTypes: ( oauth2token-attribute-oid
  NAME 'oauth2Token'
  EQUALITY caseIgnoreOAuth2TokenQueryMatch
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1
  X-ORIGIN 'DS Documentation Examples'
  X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
attributeTypes: ( jsonToken-attribute-oid
  NAME 'jsonToken'
  EQUALITY caseIgnoreJsonTokenIDMatch
  SYNTAX 1.3.6.1.4.1.36733.2.1.3.1
  SINGLE-VALUE
  X-ORIGIN 'DS Documentation Examples'
  X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
attributeTypes: ( example-class-of-service-attribute-type
  NAME 'classOfService'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreOrderingMatch
```

```
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX example-custom-syntax-oid
    SINGLE-VALUE
    USAGE userApplications
    X-ORIGIN 'DS Documentation Examples'
    X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
objectClasses: ( json-object-class-oid
    NAME 'jsonObject'
    SUP top
    AUXILIARY
    MAY json
    X-ORIGIN 'DS Documentation Examples'
    X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
objectClasses: ( oauth2token-object-class-oid
    NAME 'oauth2TokenObject'
    SUP top
    AUXILIARY
    MAY oauth2Token
    X-ORIGIN 'DS Documentation Examples'
    X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
objectClasses: ( json-token-object-class-oid
    NAME 'JsonTokenObject'
    SUP top
    AUXILIARY
    MAY jsonToken
    X-ORIGIN 'DS Documentation Examples'
    X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
objectClasses: ( example-class-of-service-object-class
    NAME 'cos'
    SUP top
    AUXILIARY
    MAY ( classOfService $ diskQuota $ mailQuota $ sharedQuota )
    X-ORIGIN 'DS Documentation Examples'
    X-SCHEMA-FILE '60-ds-evaluation-schema.ldif' )
```

```
dn: cn=schema
objectclass: top
objectclass: ldapSubentry
objectclass: subschema
cn: schema
```

Pat also keeps copies of the original DS schema files under source control. When upgrading, Pat compares the original files with the upgraded files and applies any changes to the modified production files as necessary.

Task 4: Deploy changed schema files

To make a schema change in deployment, stop the server, add the custom schema, and restart the server.

1. Prepare to show schema change deployment by setting up two replicated DS directory servers as described in Install DS and Learn replication.
2. Make sure you have local copies of the changed schema definition files:

   60-ds-evaluation-schema.ldif

   This file contains the changed schema definitions.

99-user.ldif

This file removes the replication metadata.

3. Stop a server:

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

4. Add the custom schema files and start the replica:

```
$ cp 60-ds-evaluation-schema.ldif /path/to/opendj/db/schema/
```

5. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

Replication applies the changes to other servers.

Task 5: Deploy the `classOfService` index

Create and build the index on each replica an application uses for searches:

1. Create the index on the first server:

```
$ /path/to/opendj/bin/dsconfig \
 create-backend-index \
 --backend-name dsEvaluation \
 --index-name classOfService \
 --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
```

2. Build the new index on the first server:

```
$ /path/to/opendj/bin/rebuild-index \
 --baseDn dc=example,dc=com \
 --index classOfService \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin
```

3. Create the index on the second server:

```
$ /path/to/replica/bin/dsconfig \
 create-backend-index \
 --backend-name dsEvaluation \
 --index-name classOfService \
 --set index-type:big-equality \
 --hostname localhost \
 --port 14444 \
 --bindDN uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/replica/config/keystore \
 --trustStorePassword:file /path/to/replica/config/keystore.pin \
 --no-prompt
```

4. Build the new index on the second server:

```
$ /path/to/replica/bin/rebuild-index \
 --baseDn dc=example,dc=com \
 --index classOfService \
 --hostname localhost \
 --port 14444 \
 --bindDn uid=admin \
 --bindPassword password \
 --usePkcs12TrustStore /path/to/replica/config/keystore \
 --trustStorePassword:file /path/to/replica/config/keystore.pin
```

The new schema definitions and indexes are ready to use.

Validation

After you deploy the changed schema definitions and `classOfService` indexes, follow these steps to check you can use the updated schema definitions and index.

1. Page through entries with `gold` class of service on the second replica as a user who doesn't have the `unindexed-search` privilege:

```
$ ldapsearch \
 --hostname localhost \
 --port 11636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/replica/config/keystore \
 --trustStorePassword:file /path/to/replica/config/keystore.pin \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDn dc=example,dc=com \
 --simplePageSize 5 \
 "(classOfService=gold)" \
 mail
dn: uid=abarnes,ou=People,dc=example,dc=com
mail: abarnes@example.com

dn: uid=ahall,ou=People,dc=example,dc=com
mail: ahall@example.com

dn: uid=aknutson,ou=People,dc=example,dc=com
mail: aknutson@example.com

dn: uid=alutz,ou=People,dc=example,dc=com
mail: alutz@example.com

dn: uid=ashelton,ou=People,dc=example,dc=com
mail: ashelton@example.com

Press RETURN to continue
```

2. Show users can now have `platinum` class of service:

```
$ /path/to/replica/bin/ldapmodify \
 --hostname localhost \
 --port 11636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/replica/config/keystore \
 --trustStorePassword:file /path/to/replica/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: classOfService
classOfService: platinum
EOF
```

3. Show users can’t have a random string for class of service:

```
$ /path/to/replica/bin/ldapmodify \
 --hostname localhost \
 --port 11636 \
 --useSsl \
 --usePkcs12TrustStore /path/to/replica/config/keystore \
 --trustStorePassword:file /path/to/replica/config/keystore.pin \
 --bindDN uid=admin \
 --bindPassword password << EOF
dn: uid=bjensen,ou=People,dc=example,dc=com
changetype: modify
replace: classOfService
classOfService: custom extended service
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 classOfService, value "custom extended service" was found to be invalid
according to the associated syntax: The provided value "custom extended service" cannot be parsed because it
is not allowed by enumeration syntax with OID "example-custom-syntax-oid"
```

What's next

Pat knows schema definition changes are safe in files under source control. The *reasons* for the schema changes are not so well known. Pat plans to start and maintain a schema dictionary. The schema dictionary will describe each attribute known to be in use. It will track:

- Who uses the attribute, including their contact information, and how they use it
- What data it stores, and who owns the data, including contact information
- Where the data comes from, especially if it comes from another system
- When there are maintenance windows for the attribute (for reindexing and so on)

In addition, Pat has more to discuss with the application owner, who asked for the new `sharedQuota` attribute. The `diskQuota` and `mailQuota` attributes depend on the `classOfService` attribute for their values.

- How should DS define `sharedQuota` values?
- What should the quotas be for `classOfService: platinum`?

Explore further

Reference material

Reference	Description
Indexes	Background and how-to instructions for working with indexes
LDAP schema	An in-depth look at LDAP schema definitions
LDAP schema	LDAP schema in client applications

Reference	Description
JSON schema	Schema for HTTP client applications
About This Reference	A reference for all default schema definitions

DS for AM CTS

Show how to replicate AM core token service (CTS) data and fail over when a DS server is unavailable.

Description

Estimated time to complete: 45 minutes

AM uses DS to store CTS data, such as session tokens, data for SAML v2.0 and OAuth 2.0 applications, and push notifications.

Replicate the CTS data as you would any other directory data for availability, but realize AM applications are not necessarily built with DS eventual consistency in mind. For this reason, configure AM to use affinity load balancing when connecting to the DS CTS store. Affinity load balancing ensures each request for the same entry goes to the same DS server. If the DS server becomes unavailable, AM fails over to another DS server.

Suppose an AM application makes several AM calls in quick succession, and each call requires AM to retrieve a CTS entry from DS. Without affinity, if AM updates the CTS entry on one DS then reads it from another DS, it's possible replication won't have had time to replay the changes between the update and the subsequent read. The application could get a confusing response when it appears AM "forgets" the update.

With affinity, both the update and the read target the same DS server. The AM client application gets the expected response each time.

In this use case, you:

- Set up DS for AM CTS, configuration, and identity data.
- Set up and configure AM to use the DS service with affinity and failover.
- Show AM continues to work as expected when a DS server is unavailable.

Goals

In completing this use case, you learn to:

- Set up DS and AM together.
- Configure affinity and failover for AM connections to DS.
- Replicate CTS data effectively while minimizing the impact on AM clients.

Example scenario

As a directory service administrator, Pat plans to deploy directory services for AM CTS data.

Pat knows AM has a number of configuration options for CTS, but wants to clarify the basic deployment principles before tuning the service for their specific deployment.

Pat plans to show the AM administrators the basic approach, and then discuss additional options.

Prerequisites

Knowledge

Before you start:

- Make sure you are familiar with the command line on your operating system.
- If you're new to directory services, consider working through the examples to learn LDAP and to learn replication.
- If you're new to AM, consider working through the AM evaluation tasks.

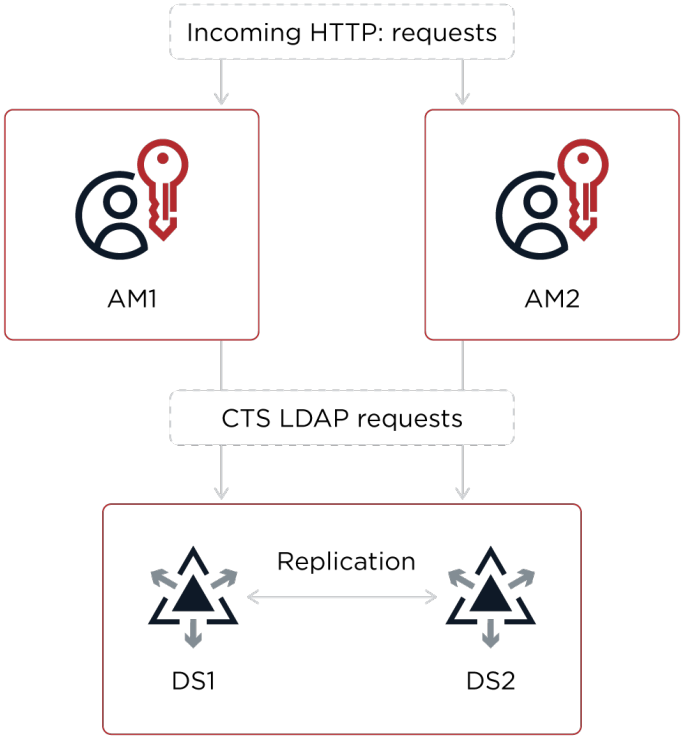
Actions

Before you start, download:

- The AM .war file
- An appropriate version of Apache Tomcat
- The DS .zip file

Tasks

This sample deployment shows the steps to replicate CTS data on your computer. Use the same steps with geographically distributed computers or virtual machines for a real deployment.



- Two AM servlets run in Apache Tomcat and serve HTTP requests from AM client applications.
- Two replicated DS servers provide storage for AM.
- Each AM servlet makes LDAP requests to DS for CTS data.

Task 1: Prepare for installation

1. Make sure there's an FQDN for AM.

   The cookie domain for AM session cookies depends on the FQDN, because the browser uses it to connect to AM.

   This sample simulates DNS on your computer by updating the hosts file with an alias for each DS server:

   ```
   # Simulate DNS with an FQDN alias for the loopback address:
   127.0.0.1       am.example.com
   ```

   When deploying in a production environment, make sure you have properly configured the DNS.

2. Unpack the server files once for each server to install.

   This sample uses folder locations aligned with the hostnames:

Base path	Description
`/path/to/ds1`	First DS server
`/path/to/ds2`	Second DS server

Base path	Description
`/path/to/tomcat`	Apache Tomcat server

3. Determine the port numbers for the service.

   This sample uses different port numbers for each server because all the servers are on the same computer:

Sample server	Port numbers
`ds1`	LDAP: 1389 LDAPS: 1636 Admin: 4444 Replication: 8989
`ds2`	LDAP: 11389 LDAPS: 11636 Admin: 14444 Replication: 18989
Tomcat	HTTPS: 8080

   When installing each DS server on a different host, use the same port numbers everywhere.

4. Set the `JAVA_HOME` environment variable to a supported JDK home if it isn't already set:

```
$ export JAVA_HOME=<supported-jdk-home>
```

5. Define how the DS servers trust DS server certificates.

   This sample uses a private PKI based on the deployment ID. You generate a deployment ID for all DS servers using the `dskeymgr` command:

```
$ /path/to/ds1/bin/dskeymgr \
create-deployment-id \
--deploymentIdPassword password
<deployment-id>
```

   The deployment ID is a string. To use it, you must have the deployment ID password.

   Once you generate the ID, set a `DEPLOYMENT_ID` environment variable for use in other steps of this sample:

```
$ export DEPLOYMENT_ID=<deployment-id>
```

6. Make sure Tomcat and AM trust DS server certificates for secure LDAPS connections.

This sample uses the private PKI based on the deployment ID you generated. Prepare a truststore with the DS CA certificate for Tomcat:

```
$ /path/to/ds1/bin/dskeymgr \
export-ca-cert \
--deploymentId $DEPLOYMENT_ID \
--deploymentIdPassword password \
--outputFile /path/to/ca-cert.pem

$ keytool \
-importcert \
-trustcacerts \
-alias ca-cert \
-file /path/to/ca-cert.pem \
-keystore /path/to/truststore \
-storepass changeit \
-storetype JKS \
-noprompt

$ export TRUSTSTORE=/path/to/truststore
```

Task 2: Set up DS

These sample commands prepare DS servers for AM CTS, configuration, and identities. They depend on the `DEPLOYMENT_ID` environment variable you set.

1. Set up the first DS server:

```
$ /path/to/ds1/setup \
--deploymentId $DEPLOYMENT_ID \
--deploymentIdPassword password \
--rootUserDN uid=admin \
--rootUserPassword password \
--monitorUserPassword password \
--hostname localhost \
--adminConnectorPort 4444 \
--ldapPort 1389 \
--enableStartTls \
--ldapsPort 1636 \
--replicationPort 8989 \
--bootstrapReplicationServer localhost:8989 \
--bootstrapReplicationServer localhost:18989 \
--profile am-config \
--set am-config/amConfigAdminPassword:5up35tr0ng \
--profile am-cts \
--set am-cts/amCtsAdminPassword:5up35tr0ng \
--profile am-identity-store \
--set am-identity-store/amIdentityStoreAdminPassword:5up35tr0ng \
--acceptLicense \
--start
```

2. Set up the second DS server:

```
$ /path/to/ds2/setup \
--deploymentId $DEPLOYMENT_ID \
--deploymentIdPassword password \
--rootUserDN uid=admin \
--rootUserPassword password \
--monitorUserPassword password \
--hostname localhost \
--adminConnectorPort 14444 \
--ldapPort 11389 \
--enableStartTls \
--ldapsPort 11636 \
--replicationPort 18989 \
--bootstrapReplicationServer localhost:8989 \
--bootstrapReplicationServer localhost:18989 \
--profile am-config \
--set am-config/amConfigAdminPassword:5up35tr0ng \
--profile am-cts \
--set am-cts/amCtsAdminPassword:5up35tr0ng \
--profile am-identity-store \
--set am-identity-store/amIdentityStoreAdminPassword:5up35tr0ng \
--acceptLicense \
--start
```

At this point, both DS servers are running and replicating changes to each other.

Task 3: Set up Tomcat

1. Update Tomcat settings for AM:

   This command uses the `TRUSTSTORE` environment variable you set:

   ```
   echo "export CATALINA_OPTS=\"\$CATALINA_OPTS \
   -Djavax.net.ssl.trustStore=${TRUSTSTORE} \
   -Djavax.net.ssl.trustStorePassword=changeit \
   -Djavax.net.ssl.trustStoreType=jks \
   -server \
   -Xmx2g \
   -XX:MetaspaceSize=256m \
   -XX:MaxMetaspaceSize=256m\"" > /path/to/tomcat/bin/setenv.sh
   ```

2. Make the Tomcat scripts executable:

   ```
   $ chmod +x /path/to/tomcat/bin/*.sh
   ```

3. Start Tomcat:

   ```
   $ /path/to/tomcat/bin/startup.sh
   ```

At this point, Tomcat is ready for you to set up AM.

Task 4: Set up AM

These steps prepare AM to use DS with affinity load balancing and failover.

1. Copy the AM .war file to `/path/to/tomcat/webapps/am1.war` and `/path/to/tomcat/webapps/am2.war`.
2. Configure AM at http://am.example.com:8080/am1 and http://am.example.com:8080/am2.

   Use the following configuration settings:

Setting	Choice
Configuration Options	**Create New Configuration**.
Server Settings > Default User Password	`Passw0rd`
Server Settings > Server URL	`http://am.example.com:8080`
Server Settings > Cookie Domain	`example.com`
Server Settings > Platform Locale	`en_US`
Server Settings > Configuration Directory	`/path/to/am1` or `/path/to/am2`
Configuration Data Store Settings > Configuration Data Store	**External DS**
Configuration Data Store Settings > SSL/TLS Enabled	Enable
Configuration Data Store Settings > Host Name	`localhost`
Configuration Data Store Settings > Port	`1636`
Configuration Data Store Settings > Encryption Key	Save the generated key (example: `w72dwbuhsLQzFNcUftA8eMCaw3a5ayhL`) from `am1` to use when configuring `am2`.
Configuration Data Store Settings > Root Suffix	`ou=am-config`
Configuration Data Store Settings > Login ID	`uid=am-config,ou=admins,ou=am-config`
Configuration Data Store Settings > Password	`5up35tr0ng`

Setting	Choice
Configuration Data Store Settings > Server configuration	**New deployment** (`am1`) or **Additional server for existing deployment** (`am2`) **Note** The `am2` server uses the same stores as those of the existing deployment. This choice causes the configurator to skip to the **Site Configuration** settings for `am2` .
User Data Store Settings > User Data Store Type	**ForgeRock Directory Services (DS)**
User Data Store Settings > SSL/TLS Enabled	Enable
User Data Store Settings > Directory Name	`localhost`
User Data Store Settings > Port	`1636`
User Data Store Settings > Root Suffix	`ou=identities`
User Data Store Settings > Login ID	`uid=am-identity-bind-account,ou=admins,ou=identities`
User Data Store Settings > Password	`5up35tr0ng`
Site Configuration	**No**

3. Configure the CTS store with affinity load balancing to both DS servers.

   On the `am1` servlet, make these configuration changes, which are shared with the `am2` servlet:

   1. Log in to the AM admin UI at http://am.example.com/am1 as `amadmin` with `Passw0rd` .
   2. Browse to **Configure > Server Defaults > CTS**.
   3. Use the following CTS settings, saving changes before switching tabs:

Setting	Choice
CTS Token Store > Store Mode	**External Token Store**
CTS Token Store > Root Suffix	`ou=famrecords,ou=openam-session,ou=tokens`
External Store Configuration > SSL/TLS Enabled	Enable
External Store Configuration > Connection String(s)	`localhost:1636,localhost:11636`
External Store Configuration > Login Id	`uid=openam_cts,ou=admins,ou=famrecords,ou=openam-session,ou=tokens`

Setting	Choice
External Store Configuration > Password	`5up35tr0ng`
External Store Configuration > Affinity Enabled	Enable

4. Configure the identity store with affinity load balancing to both DS servers.

   In the `am1` admin UI, while connected as `amadmin`:

   1. Browse to **Top Level Realm > Identity Stores > ds1 > Server Settings**.
   2. Update the following identity settings:

Setting	Choice
LDAP Server	Add `localhost:11636`.
Affinity Enabled	Enable
Affinty Level	**Bind**

   3. Save your changes.

5. Configure the configuration store to use both DS servers.

   In the `am1` admin UI, while connected as `amadmin`:

   1. Browse to **Deployment > Servers**.
   2. *For each AM servlet*:
      - Browse to ***Server URL*** **> Directory Configuration > Server**.
      - Add an entry for the second DS server and save the changes:

Setting	Choice
NAME	`ds2`
HOST NAME	`localhost`
PORT NUMBER	`11636`
CONNECTION TYPE	**SSL**

      - Save your changes.

6. Log out of the AM admin UI.

7. Restart Tomcat to take the configuration changes into account:

```
$ /path/to/tomcat/bin/shutdown.sh
# Wait a moment for Tomcat to shut down cleanly before starting it again.
$ /path/to/tomcat/bin/startup.sh
```

At this point, AM is ready to use.

Task 5: Create a test user

You will use this account for validation.

1. Log in to the AM admin UI at http://am.example.com/am1 as `amadmin` with `Passw0rd`.
2. Browse to **Top Level Realm > Identities** and click **+ Add Identity**.
3. Use the following settings for the test user:

Setting	Choice
User ID	`bjensen`
Password	`hifalutin`
Email Address	`bjensen@example.com`
First Name	`Babs`
Last Name	`Jensen`
Full Name	`Barbara Jensen`

4. Log out of the AM admin UI.

Validation

To validate your work, check:

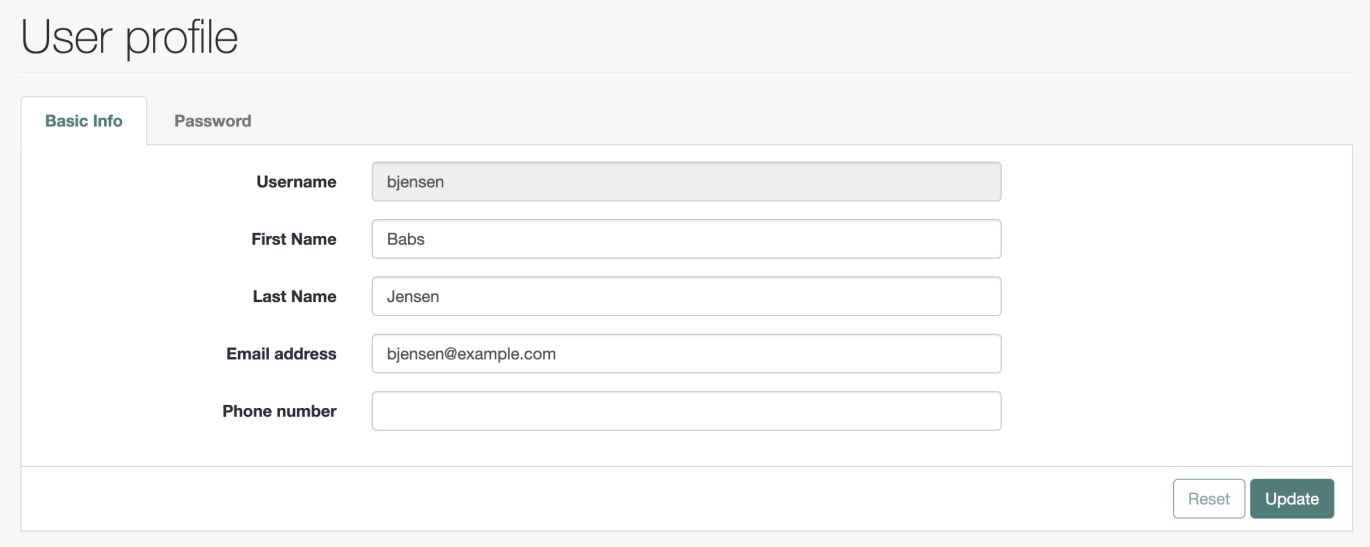
- A user can log in to one AM servlet and access the other with the same session while all servers are up.
- The session is still honored when a CTS store is unavailable.

The following sections show how to do this in detail.

Access AM as the test user

1. Log in to AM at http://am.example.com/am1 as `bjensen` with password `hifalutin`.

   The AM UI displays the user profile page:



2. Switch AM servlets by updating the URL in the browser address bar, replacing `am1` with `am2`.

   The AM UI displays the same user profile page again.

3. On the command line, find the associated CTS token in DS:

```
$ /path/to/ds1/bin/ldapsearch \
--hostname localhost \
--port 1636 \
--useSsl \
--useJavaTrustStore "${TRUSTSTORE}" \
--trustStorePassword changeit \
--bindDn uid=openam_cts,ou=admins,ou=famrecords,ou=openam-session,ou=tokens \
--bindPassword 5up35tr0ng \
--baseDn ou=famrecords,ou=openam-session,ou=tokens \
"(coreTokenUserId=id=bjensen,ou=user,ou=am-config)" \
coreTokenObject
dn: coreTokenId=<token-id>,ou=famrecords,ou=openam-session,ou=tokens
coreTokenObject: {"clientDomain":"ou=am-config","clientID":"id=bjensen,ou=user,ou=am-config","...":...}
```

   Notice the CTS does not reference the test user account by its DN, but instead by its AM universal ID.

```
{
"clientDomain": "ou=am-config",
"clientID": "id=bjensen,ou=user,ou=am-config",
"creationTimeInMillis": 1706087705386,
"listeners": {
"8f51ba31-a2e8-4f44-a998-91b411ffde3e": true,
"f0e6df25-2a9c-4be7-a5bb-1ae22c834190": true
},
"maxCachingTimeInMinutes": 3,
"maxIdleTimeInMinutes": 30,
"maxSessionTimeInMinutes": 120,
"restrictedTokensBySessionID": {},
"sessionEventURLs": {},
"sessionID": {
"encryptedString":
"ecJSF_y5EMdaJhQ4oJ01JGiXAyU.*AAJTSQACMDEAAlNLABxraFNLaytaenFKMlBtYjNmelpBdG9JTUU3ZEE9AAR0eXBlAANDVFMAAlMxAAA.
*"
},
"sessionProperties": {
"Locale": "en_GB",
"authInstant": "2024-01-24T09:15:05Z",
"Organization": "ou=am-config",
"UserProfile": "Required",
"Principals": "bjensen",
"successURL": "/am1/console",
"CharSet": "UTF-8",
"Service": "ldapService",
"Host": "127.0.0.1",
"FullLoginURL": "/am1/UI/Login?realm=%2F",
"AuthLevel": "0",
"clientType": "genericHTML",
"AMCtxId": "4cc3e651-f4eb-4bd6-9355-03b2b0abb45b-319",
"loginURL": "/am1/UI/Login",
"UserId": "bjensen",
"AuthType": "DataStore",
"sun.am.UniversalIdentifier": "id=bjensen,ou=user,ou=am-config",
"HostName": "127.0.0.1",
"amlbcookie": "01",
"Principal": "id=bjensen,ou=user,ou=am-config",
"UserToken": "bjensen"
},
"sessionState": "VALID",
"sessionType": "USER",
"timedOutTimeInSeconds": 0
}
```


You have shown the test user session works for either AM servlet.

Test CTS failover

1. Stop the first DS server to force AM to use the second DS server:

```
$ /path/to/ds1/bin/stop-ds
```

2. Verify you can still access both AM servlets as `bjensen`.

For `am1` and `am2`, the AM UI displays the user profile page.

3. Start the first DS server and stop the second:

```
$ /path/to/ds1/bin/start-ds
$ /path/to/ds2/bin/stop-ds
```

4. Verify again you can still access both AM servlets as `bjensen`.

   For `am1` and `am2`, the AM UI displays the user profile page.

You have demonstrated how AM can use DS as a CTS store with affinity load balancing and failover.

What's next

After successfully showing the sample to AM administrators, Pat leads a discussion to review the tradeoffs they can choose to make for the production deployment. Some of the questions to discuss include the following:

- Do we back up CTS data?

  If CTS data is lost, users must authenticate again.

  If that's acceptable, then we won't back up CTS data, which is volatile and potentially large.

- Should there be a separate DS service for CTS data?

  CTS access patterns are very different from identity store access patterns. They cause DS to fill and empty its database cache in very different ways.

  In a high-volume deployment, it may make sense to split the data up.

- What AM features are in use?

  Could we have DS reap expired tokens (optional) instead of AM (default)?

AM administrators can bring their own questions to the discussion.

Explore further

Related use cases

- Cross-region replication

Reference material

Reference	Description
Core Token Service (CTS)	In-depth information on setting up AM CTS with explanations of the tradeoffs
Install DS for AM CTS	Details about DS for CTS
Install DS for AM configuration	Details about DS for AM configuration

Reference	Description
Install DS for platform identities	Details about DS for AM identities
Entry expiration	Settings for letting DS reap expired tokens

Enforce limits

Enforce application and user limits to protect against a denial of service.

Description

Estimated time to complete: 20 minutes

DS has many settings to prevent client applications from using more than their share of directory resources.

> **Important**
>
> Don't disable global limit settings.
> Lift restrictions for specific trusted client applications, accounts, or groups.

Goals

In completing this use case, you learn:

- The DS alternatives for enforcing limits.
- How to change limits.
- The result codes when an application exceeds a limit.

Example scenario

As a directory service administrator, Pat knows directory services are critical for identity applications.

To prevent performance problems and denial of service, Pat wants to restrict what a misbehaving client can do. Pat also wants to make it easy for applications and users to take advantage of directory services.

Pat knows DS offers many options to set limits and aims to review them in light of the directory service requirements.

Prerequisites

Knowledge

Before you start:

- Make sure you are familiar with working with the command line on your operating system.
- If you're new to directory services, work through the examples to learn LDAP.

Actions

Before you try the sample commands, install a DS server in evaluation mode.

Tasks

Task 1: Review enforceable limits

The following tables list available options for enforcing limits. To change limits for:

- A single `ldapsearch` command, use the size or time limit options.
- An application, user or group of accounts, set operational attributes.
- A DS server, update configuration settings with the `dsconfig` command.

`ldapsearch` *options*

Limit	Option to use
Size limit	`ldapsearch --sizeLimit <number>`
Time limit	`ldapsearch --timeLimit <number-of-seconds>`

Operational attributes

Attribute	What it overrides
`ds-rlim-idle-time-limit: <number-of-seconds>`	How long an idle connection remains open.
`ds-rlim-size-limit: <number>`	The maximum number of entries returned for a search.
`ds-rlim-time-limit: <number-of-seconds>`	The maximum processing time for a search operation.

Request limit settings

Setting	Scope	Description
max-request-size	Connection handler[1]	The maximum size request this connection handler allows. When client applications add groups with large numbers of members, for example, requests can exceed the default limit. This setting affects only the size of requests, not responses. Default: 5 megabytes

[1] HTTP and LDAP connection handlers have this setting.

Connection limits[1]

Setting	Scope	Description
allowed-client	Global, Connection handler[2]	The client applications that DS accepts connections from identified by hostname or IP address. Default: not set
denied-client	Global, Connection handler[2]	The client applications that DS refuses connections from identified by hostname or IP address. Default: not set
idle-time-limit	Global	The maximum number of seconds a client connection may remain established since its last completed operation. If the network drops idle connections, set this to a lower value than the idle time limit for the network. This is particularly useful when networks drop idle connections without notification and without closing the connection. It ensures DS shuts down idle connections in an orderly fashion. DS servers do not enforce idle timeout settings for persistent searches. Default: `0` (seconds), meaning no limit
max-allowed-client-connections	Global	The total number of concurrent client connections DS accepts. Each connection uses memory. On Linux systems, each connection uses a file descriptor. Default:`0`, meaning no limit
restricted-client	Global, Connection handler[2]	The client applications DS limits to `restricted-client-connection-limit` connections. Default: not set
restricted-client-connection-limit	Global, Connection handler[2]	The maximum number of concurrent connections for specified clients. Default: `100` (connections)

[1] DS applies the settings in this order:

1. If the `denied-client` property is set, DS denies connections from any client matching the settings.
2. If the `restricted-client` property is set, DS counts the connections from any client matching the settings.

   If a matching client exceeds `restricted-client-connection-limit` connections, DS refuses additional connections.
3. If the `allowed-client` property is set, DS lets any client matching the settings connect.
4. If the limits are not set, DS lets any client connect.

[2] The settings on a connection handler override the global settings.

Search limit settings

Setting	Scope	Description
max-psearches	Global	The maximum number of concurrent persistent searches. Default: `-1`, meaning no limit
size-limit	Global	The maximum number of entries returned for a single search. Default: `1000` (entries)
time-limit	Global	The maximum number of seconds to process a single search. Default: `0` (seconds), meaning no limit

Task 2: Override account limits

1. Give an administrator access to update the 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: 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. Override the limits for a single 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
add: ds-rlim-size-limit
ds-rlim-size-limit: 10
EOF
```

When Babs Jensen performs an indexed search returning more than 10 entries, she reads 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
```

Task 2: Override group limits

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 override the limits with 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 with administrator entries. For details on how subentries apply, refer to About subentry scope.

3. Show an administrator account has limits set to `0` (no limit):

```
$ 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
```

Task 3: Limit persistent searches

An LDAP persistent search maintains an open connection until the client application ends the search. Whenever a modification changes data in the search scope, DS returns a search result. The more concurrent persistent searches, the more work the server has to do for each modification:

Set the global property `max-psearches` to limit total concurrent persistent searches.

The following command sets a maximum of 30 persistent searches:

```
$ 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
```

Task 4: Limit connections

- Limit the total concurrent connections DS accepts.

  The following command sets the limit to 64K (the minimum number of file descriptors to make available to DS on a Linux system):

  ```
  $ 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
  ```

- Set an idle timeout of 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
  ```

- Limit access to clients in 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
```

- Limit clients on the `10.0.0.*` network to 1000 concurrent connections each:

```
$ 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
```

Task 5: Permit large requests

The following command increases the limit to 20 MB for the LDAP connection handler. This lets client applications add large static group entries, for example:

```
$ 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
```

Result codes

When an LDAP application exceeds a limit, DS responds with the appropriate result code:

- `3: Time Limit Exceeded` when the request took too long to process.
- `4: Size Limit Exceeded` when the request returned too many entries.
- `11: Administrative Limit Exceeded` when the request exceeded a limit imposed by one of the other settings.

Refer to any additional information DS returns with the result to determine what action to take.

Deployment

This guide focuses on how to use PingDS 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

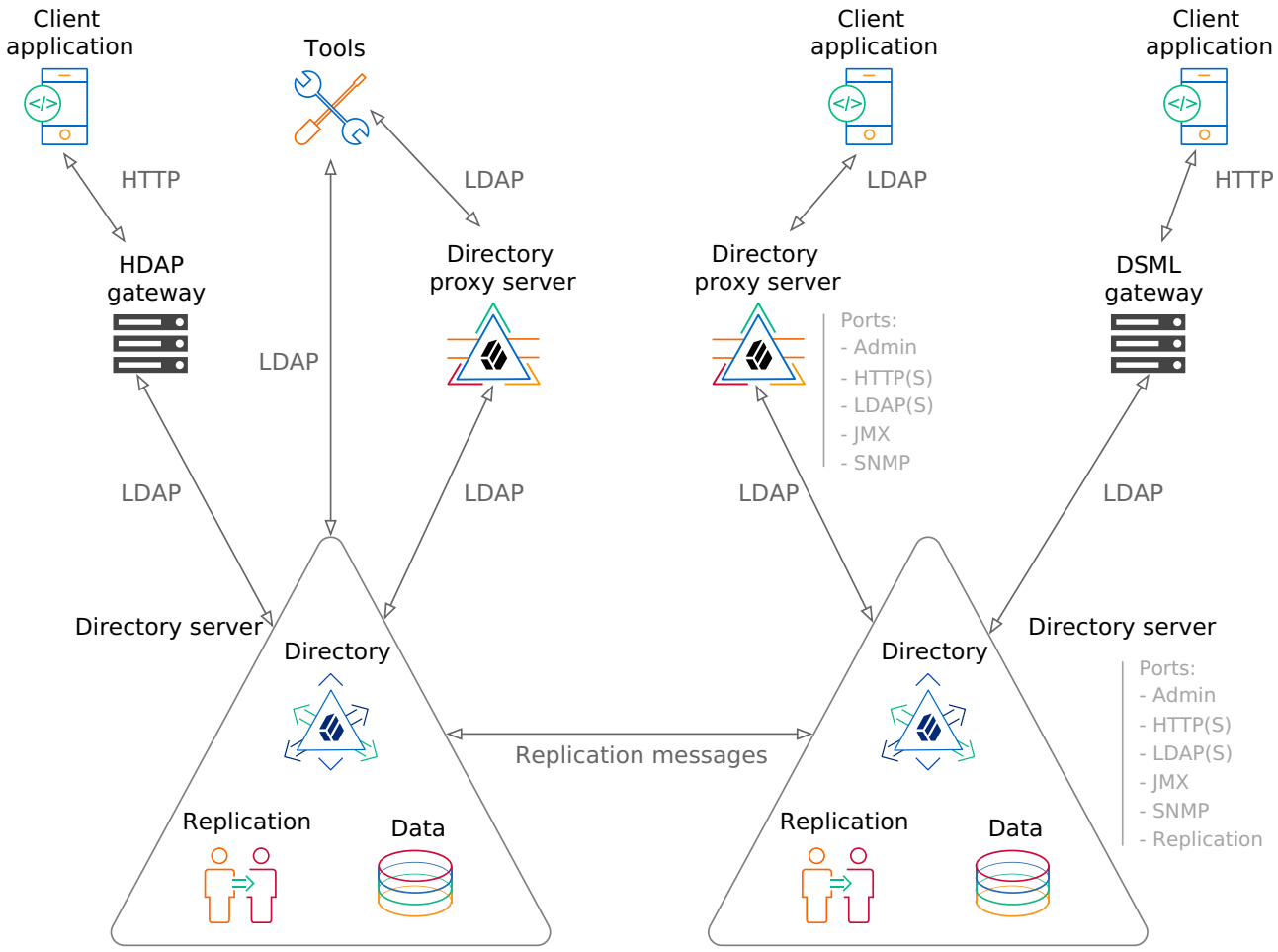
Prepare systems and hardware.

Checklists

Follow checklists.

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* maintain and serve requests for directory data.

  Directory servers use data *replication* to ensure their data sets eventually converge everywhere. This documentation refers to a replicated directory server as a *replica*.

- (Optional) *Directory proxy servers* forward LDAP requests to directory servers and return responses to client applications.

- (Optional) *Replication servers* transmit data replication messages among replicas.

  Some advanced deployments use *standalone* replication servers. These servers only broker replication change messages and do not store directory data. In most deployments, each directory server acts as a replication server as well.

- (Optional) *DSML gateways* intermediate between DSML client applications and an LDAP directory.

- (Optional) *HDAP gateways* intermediate between RESTful HTTP client applications and LDAP directories.

- LDAP client tools and server administration tools serve to test and configure 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 re-authenticating 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.

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 has 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 depending on the pattern of update operations.

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

Metadata for replication

To avoid single points of failure, almost all real-world deployments depend on replication. Each directory server is a replica of other directory servers, meaning it holds an eventually consistent copy of the data on the other replicas.

When serving a request to update directory data, the directory server modifies its data and makes a request to a replication server. The replication server is usually but not always part of the same Java process as the directory server. The replication server ensures 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, refer to 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 in LDIF and JSON files, LDAP schema definitions in LDIF files, keystores, and some additional data.

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

When upgrading a server, the `upgrade` command applies necessary changes to the configuration data.

Log files

The server writes 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, refer to 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, refer to 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

A replication server is usually but not always part of the same Java process as a directory server.

Replication servers have the following 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.

- 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 and 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 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.

To avoid a single point of failure, always install two or more replication servers. For example, install at least two directory servers operating as replication servers as well.

Data

In addition to the libraries and tools delivered with the server distribution, a replication server has 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 makes a request to a replication server. The replication server makes sure 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. Replication servers store changes in change log files. For details, refer to 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 in LDIF and JSON files, LDAP schema definitions in LDIF files, keystores, and some additional data.

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

When upgrading a server, the `upgrade` command applies necessary changes to the configuration data.

Log files

The server writes 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, refer to 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

Some deployments use 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 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 depends 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 offer HTTP access to remote LDAP directory servers, use a gateway instead. For details, refer to HDAP gateway.

Data

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

Server configuration

Each server has configuration data in its `config` directory. This includes the server configuration in LDIF and JSON files, LDAP schema definitions in LDIF files, keystores, and some additional data.

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

When upgrading a server, the `upgrade` command applies necessary changes to the configuration data.

Log files

The server writes 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, refer to Logging. This means, however, that messages are eventually lost unless you move old files to another system for permanent storage.

System resources

A directory proxy server decodes incoming and encodes outgoing requests and responses. When you deploy directory proxy servers, the volume of decoding and encoding means 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 a server, its files include tools for setup, upgrade, configuration, and maintenance, and LDAP command-line tools for sending LDAP requests and measuring directory service performance.

For details, refer to Server commands.

DSML gateway

> **Important**
>
> The interface stability of this feature is *Deprecated*.

The standalone DSML gateway web application has the following characteristics.

You can install this component independently of directory services. For details, refer to 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.

HDAP gateway

The standalone HDAP 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, refer to Install an HDAP gateway.

Roles

HDAP 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.

An HDAP gateway runs in a Java web application container. You can configure the gateway to contact multiple LDAP directory servers.

Data

An HDAP gateway maintains only its own service configuration files. It depends on the host web application container for other services, such as logging.

System resources

An HDAP 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 Ping Identity 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.

Ping Identity training regularly offers training courses for DS topics.

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.

Ping Identity also offers an accreditation program for partners, including an in-depth assessment of business and technical skills for each Ping Identity 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**
>
> Ping Identity 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, refer to 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, refer to 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, refer to 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?

  Refer to LDAP schema extensions.

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

  Refer to The DIT.

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

  Refer to 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?

  Refer to Data management.

- What groups will be defined in your directory service?

  Refer to Groups.

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

Refer to Shared data.

- How should you cache data for appropriate performance?

  Refer to Caching.

- How will identities be managed in your deployment?

  Refer to 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 LDAP 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

DS offers LDAP and HTTP connection handlers.

Connection handlers govern connection security and access from specified client hostnames or address masks.

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.

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.

  PingAM 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.

  PingIDM integrates directory account management features into full identity management services.

Further Reading on Managing Identities

Topics	References
Account Management	• Accounts • Active accounts

Topics	References
Authentication	• Authentication mechanisms • Authentication (binds) • Certificate-based authentication • Pass-through authentication • HDAP API reference
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, refer to 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, refer to 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, refer to Authentication mechanisms.

Proxy layer

DS directory proxy servers and the DS HDAP gateway application offer access to remote 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, refer to 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, refer to LDAP proxy.

HTTP access

Refer to HTTP access or Install an HDAP gateway.

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 Ping Identity Platform components offer such higher-level abstractions. For example, PingAM software lets you plug into directory services for authentication and account profiles, and then orchestrate powerful authentication and authorization scenarios. PingIDM 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 about the Ping Identity Platform.

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, refer to 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 servers in case one fails. Deploy more servers where necessary, knowing more servers means more complexity for those managing the service.

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, refer to 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, refer to Changelog for notifications.

Standalone replication servers

> **Tip**
>
> This information applies to *advanced* deployments.

In most modern deployments, each directory server acts as a replication server as well.

For deployments with many servers over slow or high-latency networks, DS software makes it possible to configure standalone replication servers and directory servers.

All replication servers 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 ensure directory servers only connect to local replication servers until they must fail over to remote replication servers. This limits the replication traffic over slow network links to messages between replication servers, except when all local replication servers are down. For details, refer to Install standalone servers (advanced) and Replication groups (advanced).

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.

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 previously, replication lets you use redundant servers and systems to 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, refer to High availability.

As shown in that section, your deployment may involve multiple locations. Some deployments even use separate replication topologies, for example, 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 slow links to communications between replication servers. Directory proxy servers can direct client traffic to local servers until it becomes necessary to fail over 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 and the related pages. 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, refer to 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, refer to 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, refer to 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, refer to 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/prometheus` endpoint for Prometheus monitoring software.

  For details, refer to Use administrative APIs.

- LDAP connection handlers expose a `cn=monitor` branch that offers LDAP access to monitoring data.

  For details, refer to LDAP-based monitoring.

- JMX connection handlers offer remote access.

  For details, refer to JMX-based monitoring.

- You can configure alerts to be sent over JMX or SMTP (mail).

  For details, refer to Alerts.

- Replication conflicts are found in the directory data.

  For details, refer to Replication conflicts.

- Server tools, such as the `status` command, can run remotely.

  For details, refer to 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 eventually removes 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, refer to 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 Ping Identity already tests DS servers and gateways functionally, focus your functional testing on customization 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 performance bottlenecks.

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 tasks to:

- Develop and maintain performance tests.
- Provision and maintain a pre-production test environment that mimics your production environment.

  Security measures in your test environment must mimic your production environment. Security measures can affect 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 raise performance requirements. Also, consider the thresholds that you can monitor in the production system to estimate when your system might no longer meet performance requirements.

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.

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 the DS gateway web applications.

A gateway does not have external configuration APIs. You must restart it 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 Ping Identity 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, first try more powerful servers and faster storage.

Single directory services can 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, refer to High scalability.

Directory service upgrades

Ping Identity 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

> **Tip**
>
> This information applies to *all* deployments.

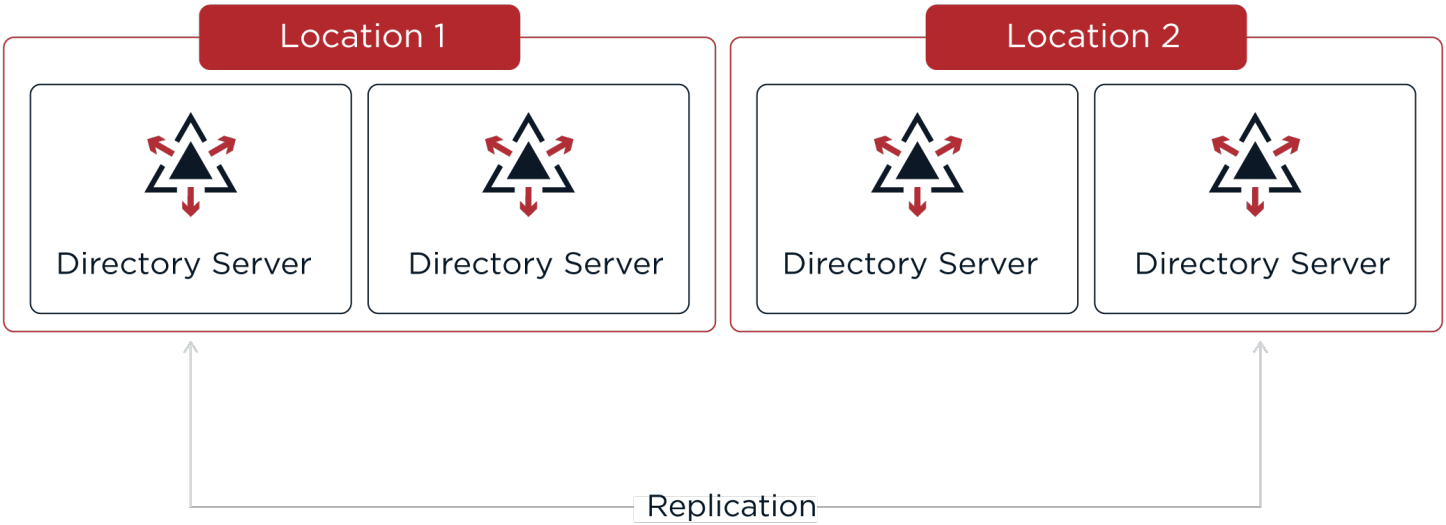
When you deploy DS servers into a highly available directory service, you are implementing the primary use case for which DS software is designed:

- Data replication lets you eliminate single points of failure.

  Replication favors availability over immediate consistency.

- 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 of redundant servers in multiple locations. If possible, use redundant networks within and between locations to limit network partitions.

Effective disaster recovery

> **Tip**
>
> This information applies to *all* deployments.

Avoiding downtime depends on redundant servers and operational readiness to recover quickly and effectively. Prepare and test your plans. Even if disaster strikes, you will 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, refer to 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. This operation happens automatically when you restart the server.

  In case of disaster, be prepared to rebuild the entire service. For details, refer to Disaster recovery.

- **Part of the service is down.**

  Fail client applications over to healthy servers, and restart or rebuild servers that are down.

  Directory proxy servers can fail over automatically and retry requests for certain types of failure. For details, refer to 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 cannot catch up. For longer network outages, you must 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 any standalone replication servers.

3. Rebuild directory servers.

4. Rebuild any directory proxy servers.

Start up order

> **Tip**
>
> This information applies to *advanced* deployments.

Some advanced deployments compose the directory service using separate component servers for different functions. Bring component servers online in the following order:

1. **Standalone Replication Servers**

   Replication servers are the foundation for high availability. They communicate change messages to directory server replicas. 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**

   DS directory proxy servers discover DS replicas by querying the replication service. They forward requests to the directory server replicas, and responses to the client applications.

High scalability

> **Tip**
>
> This information applies to *advanced* deployments.

A high-scale directory service requires very high throughput, very low response times, or both. It might have a large data set, such as 100 million entries or more. 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 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

> **Tip**
>
> This information applies to *advanced* deployments.

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, your SLOs, and on directory data like what you expect in production:

1. Import a known fraction of the expected initial data with the server configured for production.

   For help, refer to 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

> **Tip**
>
> This information applies to *advanced* deployments.

When scaling up, each server system must have the resources to run a high-scale DS server. As described in Scaling replication, each directory server replica only absorbs its share of the full read load, but each replica absorbs *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 each server on a separate system.
- Use fast disks with good IOPS, and 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

> **Tip**
>
> This information applies to *advanced* deployments.

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. You can distribute entries based on location, network, or other characteristics. Branches can join at a base DN to bring 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 (advanced). 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 access.

Data sovereignty

> **Tip**
>
> This information applies to *advanced* deployments.

In many countries, how you store and process user accounts and profile information is subject to regulations and restrictions that protect user privacy. Data sovereignty legislation is beyond the scope of this document, but DS servers do include features to build services in compliance with data sovereignty requirements:

- Data replication
- Subtree replication
- Fractional replication

The following deployment patterns address questions of data storage. When planning your deployment, 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 dynamic access management, use PingDS together with PingAM software.

Replication and data sovereignty

> **Tip**
>
> This information applies to *advanced* deployments.

Data replication is critical to a high-scale, highly available directory service. For deployments where data protection is also critical, you must 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 change data, including data governed by privacy regulations:

- For details on replicating data that must not leave a given location, refer to Subtree replication.
- For details on replicating only part of the data set outside a given location, refer to Fractional replication.

Subtree replication

> **Tip**
>
> This information applies to *advanced* deployments.

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, only replicate data 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.

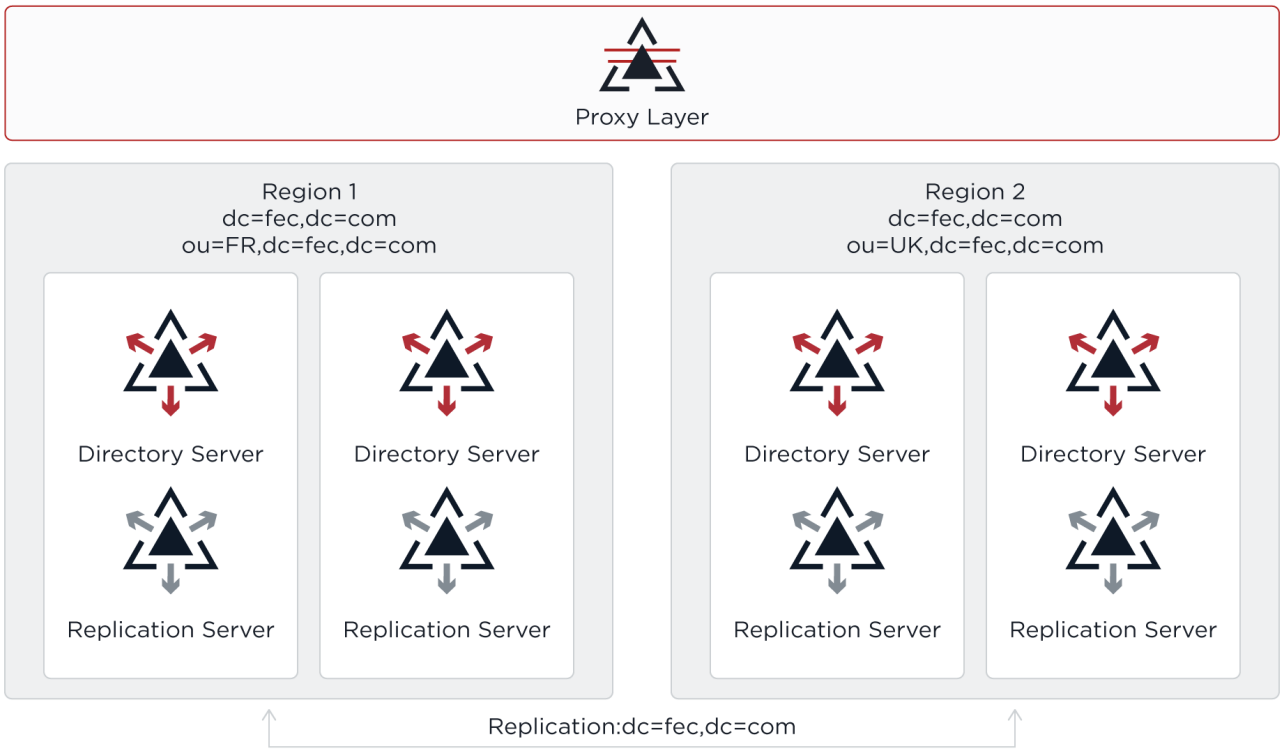


Figure 1. Separate replication domains for data sovereignty

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.

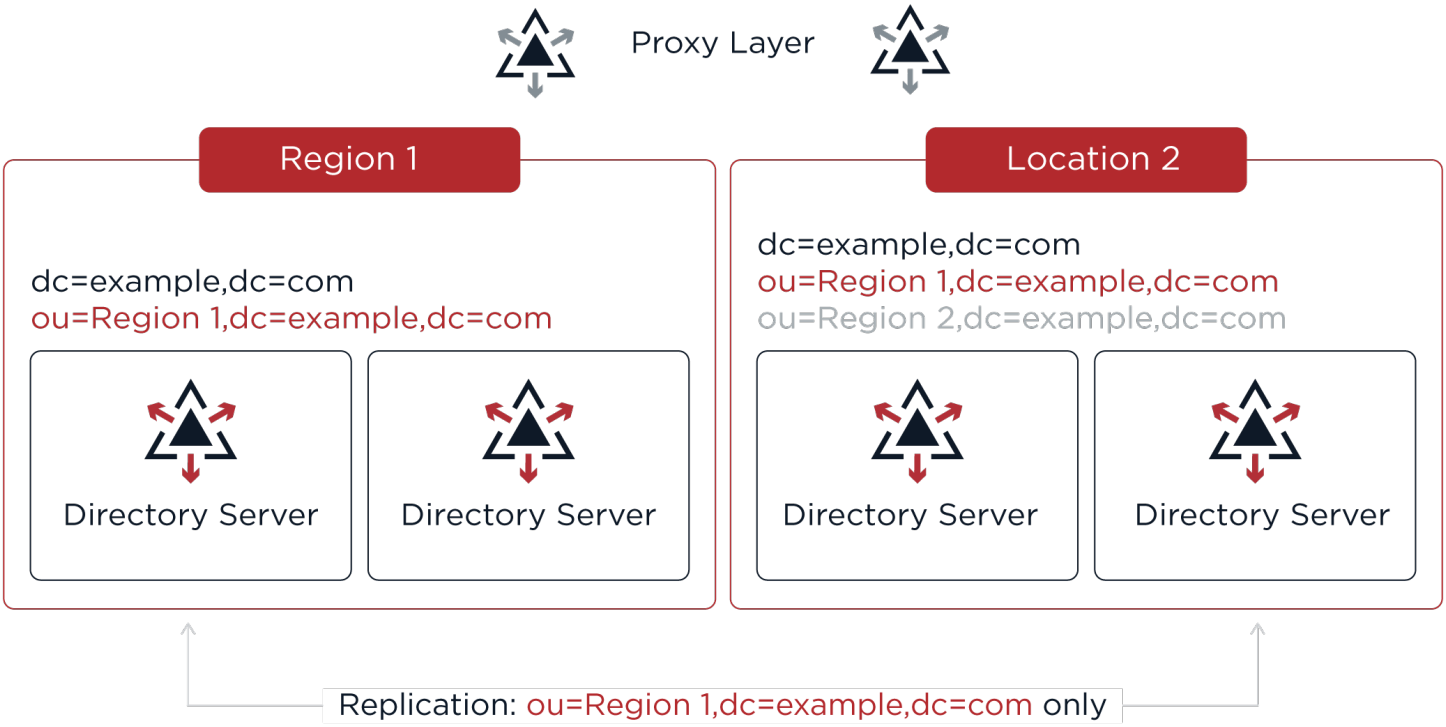


Figure 2. Mixed replication domains for data sovereignty

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 locally available data.

Subtree replication and subordinate backends include important requirements and limitations. For more information, refer to Subtree replication (advanced), and Subordinate backends.

Fractional replication

> **Tip**
>
> This information applies to *advanced* deployments.

In some deployments, regulations let you replicate some user attributes. For example, 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 travel outside the location where the full entries are processed.

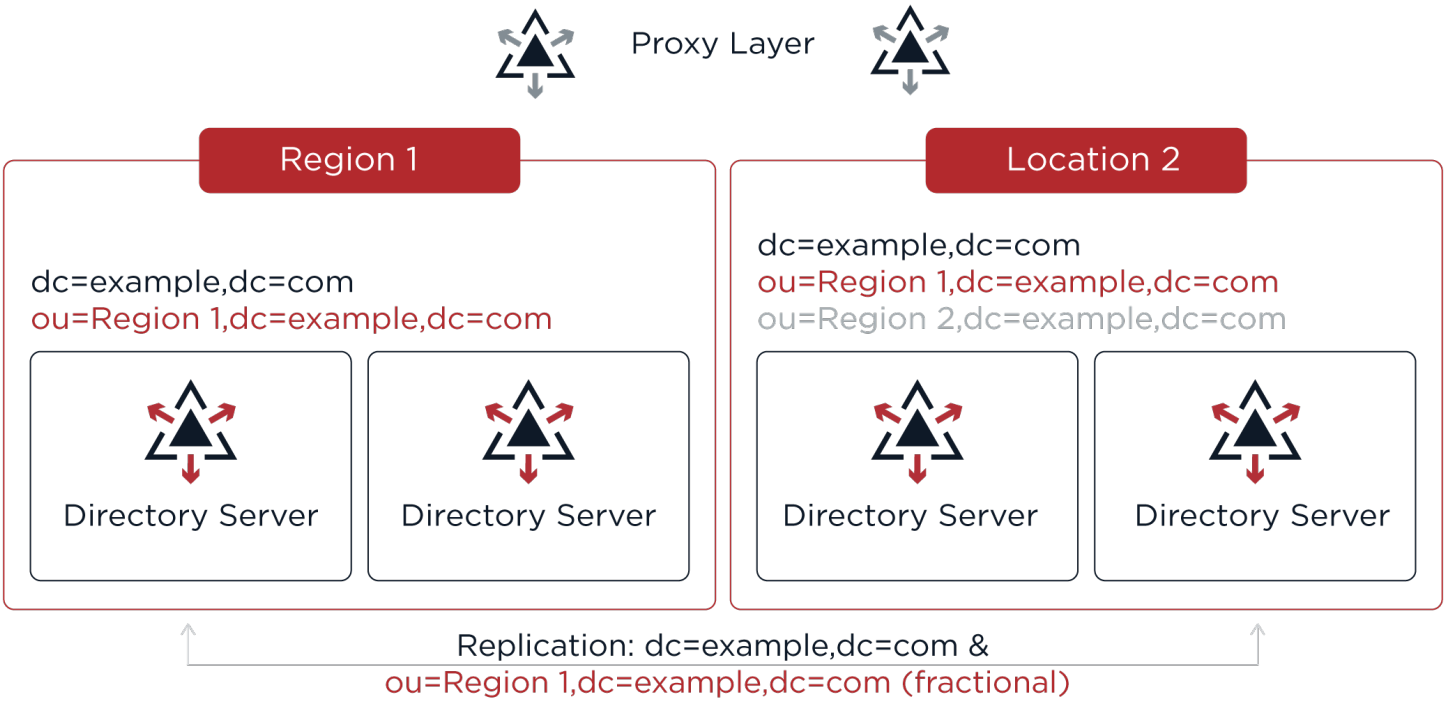


Figure 3. Fractional replication for protected data

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, refer to LDAP schema.

For additional information, refer to Fractional replication (advanced).

Interoperability

The following use cases involve interoperability with other directory software.

Use Case	Refer to...
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

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, refer to 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, refer to 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 PingIDM software or a similar solution. PingIDM 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. PingIDM 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.

Provisioning systems

Before running PingDS software in production, review the Requirements section 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, refer to 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.

In most modern deployments, WAN links are fast and responsive enough to prevent the extra traffic from causing problems. Adapt your deployment if you measure that some network links are too slow or the latency is too high.

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-level 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 paths to the scripts differ on Linux and Windows systems. Find 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 tools to help register the server as a service of the operating system. These scripts are `create-rc-script` (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	☐
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 Ping Identity and partners	☐
Find out how to get training from Ping Identity 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	☐

Task	Done?
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	☐
Plan how to adapt the deployment to new and changing requirements	☐

Installation

This guide shows you how to install and remove PingDS 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 (deprecated)	DSML support is available through the gateway, which is a Java web application that you install in a web container.
HDAP gateway	The HDAP gateway is a Java web application offering REST access to a remote LDAPv3 directory service.
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.

Product names changed when ForgeRock became part of Ping Identity. PingDS was formerly known as ForgeRock Directory Services, for example. Learn more about the name changes in New names for ForgeRock products in the Knowledge Base.

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.

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.
   - Adds documentation files under the `/usr/share/doc/opendj` directory.
   - Adds man pages under the `/opt/opendj/share/man` directory.
   - Generates systemd service files `/etc/default/opendj` and `/etc/systemd/system/opendj.service`.

   By default, the system superuser (`root`) owns the files. The DS server can listen on privileged ports like `389` and `636`.

3. (Optional) Change the systemd configuration:

   - Edit `/etc/default/opendj` directly to set any environment variables DS requires.

     For example, set environment variables for property value substitutions.

   - Use the `systemctl edit` command to change the service configuration; for example, to run DS as a specific user.

     The command makes the changes in a new `override.conf` file that systemd reads automatically.

   The changes you make in this way are independent of upgrades and changes to the package defaults. To avoid compatibility problems, don't edit `/etc/systemd/system/opendj.service` directly.

4. 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:
root# rpm -ivh jre-*.rpm
```

2. Install the server package:

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

The RPM package:

- Installs server files in the `/opt/opendj` directory.
- Adds man pages under the `/opt/opendj/share/man` directory.
- Generates systemd service files `/etc/default/opendj` and `/etc/systemd/system/opendj.service`.

By default, the system superuser (`root`) owns the files. The DS server can listen on privileged ports like `389` and `636`.

3. (Optional) Change the systemd configuration:

- Edit `/etc/default/opendj` directly to set any environment variables DS requires.

  For example, set environment variables for property value substitutions.

- Use the `systemctl edit` command to change the service configuration; for example, to run DS as a specific user.

  The command makes the changes in a new `override.conf` file that systemd reads automatically.

The changes you make in this way are independent of upgrades and changes to the package defaults. To avoid compatibility problems, don't edit `/etc/systemd/system/opendj.service` directly.

4. 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

Make sure you can log on as Windows Administrator to install the files and run the `setup.bat` command.

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:

`C:\path\to\opendj\bat\`
: Windows command-line tools

`/path/to/opendj/bin/`
: Linux command-line tools

`/path/to/opendj/extlib/`
: Optional `.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, refer to 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.

GUI

1. Review requirements for installation.

2. Start the wizard as Windows Administrator:

    1. If you are logged on as Administrator, double-click the Windows installer package, `DS-7.5.2.msi`.

    2. If you are logged on as a regular user, hold the shift key while right-clicking `DS-7.5.2.msi`, select **Run as different user**, and run the installer as Windows Administrator.

3. (Optional) Set the **Destination Folder** to the location for DS 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)\ForgeRock Directory Services\`.

- The Windows installer has 32-bit dependencies but DS runs as a 64-bit Java application.

    1. Complete the wizard.

The installation program writes DS server files to the destination folder.

You must run the `setup.bat` command in the destination folder *as Administrator* to set up DS.

PowerShell

1. Review requirements for installation.
2. Start PowerShell as Windows Administrator:
    1. If you are logged on as Windows Administrator, double-click **Start > Windows PowerShell**.
    2. If you are logged on as a regular user, hold the shift key while right-clicking **Start > Windows PowerShell** and select **Run as Administrator**.
3. Use the Microsoft `msiexec.exe` command to install the files.

    The following example installs DS server files under `C:\Users\opendj\ds`. It writes an `install.log` file in the current folder:

    ```
    C:\> msiexec /i C:\Users\opendj\Downloads\DS-7.5.2.msi /l* install.log /q OPENDJ="C:\Users\opendj\ds"
    ```

    The installation program writes DS server files to the destination folder.

    You must run the `setup.bat` command in the destination folder *as Administrator* to set up DS.

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, refer to 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`

Parameter	Description	Option(s)
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, refer to Deployment IDs.	`--deploymentId`
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 Linux root user. The default name is: `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]`

Parameter	Description	Option(s)
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`

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, refer to 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`
		`--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`

Parameter	Description	Option(s)

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`

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`.	`--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, refer to 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 Ping 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.

Different data under different base DNs

Nothing prevents you from configuring multiple setup profiles to use the same base DN for different directory data. Keep different directory data under different base DNs.

When the different data sets are incompatible, reusing a base DN can lead to errors, such as the following:

```
category=CONFIG severity=ERROR msgID=116 msg=An error occurred while trying
to initialize a backend loaded from class org.opends.server.backends.jeb.JEBackend
with the information in configuration entry ds-cfg-backend-id=cfgStore,cn=Backends,cn=config:
An error occurred while attempting to register the base DNs [dc=reused,dc=base,dc=dn] in the Directory Server:
Unwilling to Perform: Unable to register base DN dc=reused,dc=base,dc=dn with the Directory Server
for backend cfgStore because that base DN is already registered for backend amCts.
This backend will be disabled.
```

Check profiles

The `opendj/profiles.version` file lists the profiles selected at setup time:

```
$ cat /path/to/opendj/config/profiles.version
ds-evaluation:7.5.2
```

Default indexes

For new backends, setup profiles create the following default indexes:

- `ds-certificate-fingerprint` (equality index)
- `ds-certificate-subject-dn` (equality index)
- `member` (equality index)
- `uid` (equality index)
- `uniqueMember` (equality index)

When a profile adds a backend with default user indexes, it also creates the following default indexes:

- `cn` (equality and substring indexes)
- `givenName` (equality and substring indexes)
- `mail` (equality and substring indexes)
- `sn` (equality and substring indexes)
- `telephoneNumber` (equality and substring indexes)

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 7.5.0

The `am-identity-store:7.5.0` 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

AM Identity Data Store 7.3.0

The `am-identity-store:7.3.0` 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 7.2.0

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 7.5.0

The `ds-evaluation:7.5.0` 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

`useOutdatedPasswordStorage`

Use Salted SHA-512 as the password storage scheme for the import and default password policy for users.
Default: `--set ds-evaluation/useOutdatedPasswordStorage:false`

This parameter takes one of the following values:

- `true`
- `false`

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 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:

- `true`
- `false`

IDM External Repository 7.5.0

The `idm-repo:7.5.0` 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

IDM External Repository 7.4.0

The `idm-repo:7.4.0` 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 7.3.0

The `idm-repo:7.3.0` 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 7.2.0

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. Refer to Parameters.
`profile.groovy` (required)	Groovy script that makes changes to the server. Refer to Profile script.
`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`.

File or Directory	Description
`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, refer to 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. For new backends, setup profiles create the following default indexes: • `ds-certificate-fingerprint` (equality index) • `ds-certificate-subject-dn` (equality index) • `member` (equality index) • `uid` (equality index) • `uniqueMember` (equality index) When a profile adds a backend with default user indexes, it also creates the following default indexes: • `cn` (equality and substring indexes) • `givenName` (equality and substring indexes) • `mail` (equality and substring indexes) • `sn` (equality and substring indexes) • `telephoneNumber` (equality and substring indexes)
`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.

Method	Description
`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.
`void ds.importLdifTemplate(String… ldifFilePaths)` `void ds.importLdifTemplate(Collection<Path> 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.importLdif(String… ldifFilePaths)` `void ds.importLdif(Collection<Path> 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.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, refer to 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.
`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, refer to 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, refer to 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, refer to 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.
- HDAP 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";)`

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 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, refer to Unpack files.

2. Run the appropriate `setup` command with the `--profile am-cts` option.

> **Important**
>
> Installation settings depend on AM token expiration and session capability requirements. Letting DS expire tokens is efficient, but affects sending AM notifications about session expiration and timeouts to AM policy agents.
>
> - For details about AM token expiration options, refer to Manage expired CTS tokens.
> - For details about the mechanism DS uses to expire tokens, refer to Entry expiration.

1. AM reaper manages all token expiration (AM default):

```
$ /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
```

In 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`.

AM and IDM expect exclusive access to the data in each setup profile. *Keep the data separate by using distinct base DNs and domains for each setup profile.* Don't accidentally mix the data by choosing a base DN under another base DN.

- The `am-cts` profile excludes the base DN from change number indexing.

For the full list of profiles and parameters, refer to Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, refer to 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, refer to 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 and IDM expect exclusive access to the data in each setup profile. *Keep the data separate by using distinct base DNs and domains for each setup profile.* Don't accidentally mix the data by choosing a base DN under another base DN.

For the full list of profiles and parameters, refer to Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, refer to Install DS for custom cases.

4. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

Install DS for platform identities

Use this profile when setting up DS as an identity repository and user data store for AM alone or shared with IDM in a Ping Identity Platform deployment. It includes the additional LDAP schema and indexes required to store the identities:

Important

When AM and IDM share multiple DS replicas for identities:

- Configure IDM for failover.
- Configure AM to fail over when connecting to DS replicas in the same order as IDM.

1. Before proceeding, install the server files.
   For details, refer to 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 and IDM expect exclusive access to the data in each setup profile. *Keep the data separate by using distinct base DNs and domains for each setup profile.* Don't accidentally mix the data by choosing a base DN under another base DN.

For the full list of profiles and parameters, refer to Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, refer to Install DS for custom cases.

4. Start the server:

```
$ /path/to/opendj/bin/start-ds
```

Install DS as an IDM repository

> **Important**
>
> When IDM uses multiple DS replicas, configure IDM for failover.

1. Before proceeding, install the server files.
   For details, refer to 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`.

     AM and IDM expect exclusive access to the data in each setup profile. *Keep the data separate by using distinct base DNs and domains for each setup profile.* Don't accidentally mix the data by choosing a base DN under another base DN.
   - IDM requires change number indexing with the default settings.

   For the full list of profiles and parameters, refer to Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, refer to 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, refer to Install DS for platform identities instead.

To import generated sample user data, refer to Install DS for evaluation instead:

1. Before proceeding, install the server files.

   For details, refer to 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, refer to Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, refer to 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, refer to 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, refer to Custom indexes for JSON.

- Configure password storage.

  For details, refer to Configure password policies.

  Take care to configure the password policy import plugin as well. For details on the settings, refer to Password Policy Import Plugin.

- Add custom LDAP schema.

  For details, refer to LDAP schema.

- Configure one or more backends for your data.

  For details, refer to 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, refer to 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, refer to Replace deployment IDs.

- Initialize replication.

  For example, import the data from LDIF, or restore the data from backup.

  For details, refer to 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 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, refer to Replication and the related pages.

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. Ping Identity does not test all servers listed:

- PingDS
- PingDirectory
- ApacheDS
- NetIQ eDirectory
- OpenDJ
- OpenLDAP
- Oracle Directory Server Enterprise Edition
- 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.

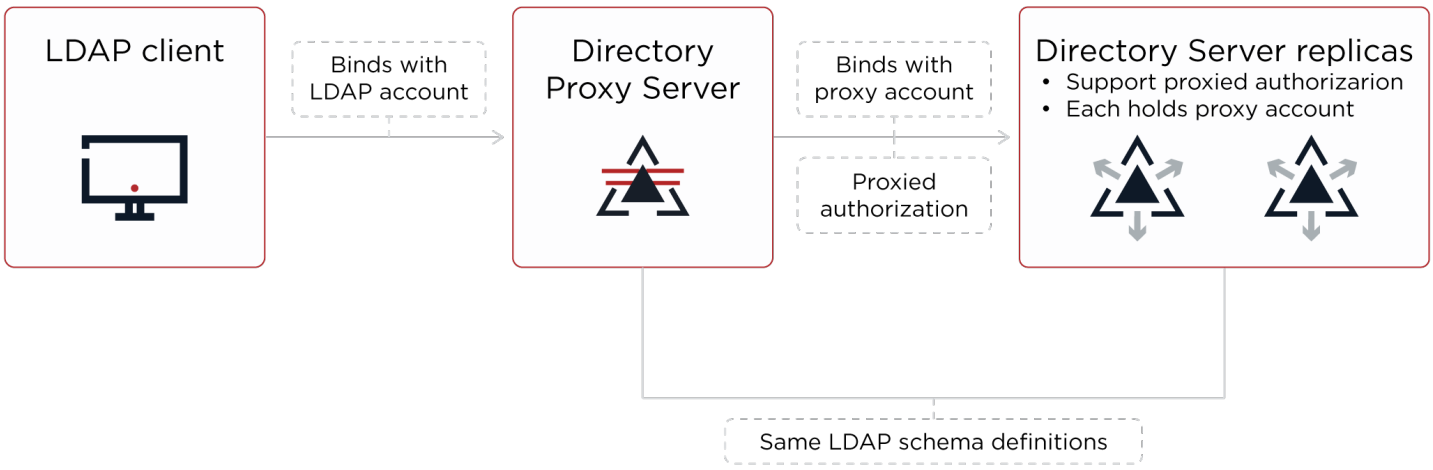


Figure 1. Proxy Configuration

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-7.5.2.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, refer to 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, refer to 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, refer to 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, refer to 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, refer to 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, refer to 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, refer to 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, refer to 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, refer to 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 have 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, refer to 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. Avoid doing 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, refer to 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, refer to Default setup profiles.

3. Finish configuring the server *before you start it*.

   For a list of optional steps at this stage, refer to Install DS for custom cases.

4.

Start the server:

```
$ /path/to/opendj/bin/start-ds
```

Install standalone servers (advanced)

> **Tip**
>
> This information applies to *advanced* deployments.

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 number of replication servers squared. Large deployments that span slow, high-latency links can 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 connections for replication over slow links:

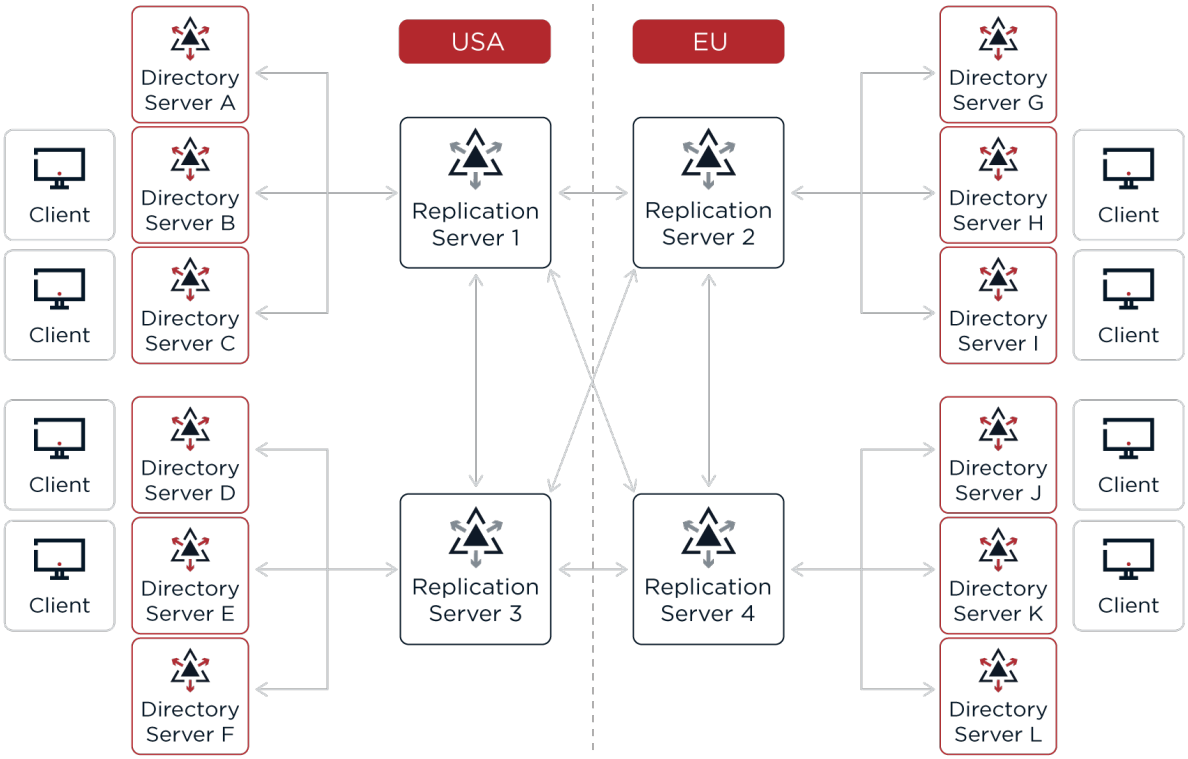


Figure 1. Deployment for multiple data centers

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, refer to 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, refer to 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, refer to 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.

Use your own cryptographic keys

Important

When you set up a DS server with your own keys for PKI, account for the following points:

- You must also use a deployment ID and password.
  Some DS features depend on the shared master key generated from the deployment ID and password. For example, the `dsbackup` command depends on the shared master key for encryption.
- 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, refer to PKCS#11 hardware security module.
- Make sure AM, IDM, and all other client applications can trust DS certificates.

The `setup` command has options to simplify setting up a server with existing keys:

For…	Use…

Keystores containing server key pairs	`--useJavaKeyStore`
	`--useJceKeyStore`
	`--usePkcs11KeyStore`
	`--providerArg` (for PKCS#11)
	`--providerClass` or
	`--providerName` (for PKCS#11)
	`--usePkcs12KeyStore`
	`-W, --keyStorePassword[:env|:file]`
	`-N, --certNickname`

Truststores containing trusted CA or self-signed certificates	`--useJavaTrustStore`
	`--useJceTrustStore`
	`--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 read PKCS#11 hardware security module.

- If you are using PEM format keys, read PEM format keys.

- CAs can optionally set X.509 key usage extensions in server certificates.

  If the CA does set key usage extensions, make sure it includes at least the required settings:

Protocol	X.509 extension	Required settings
HTTP	`KeyUsage`	`digitalSignature` `keyEncipherment`
	`ExtendedKeyUsage`	`serverAuth` (TLS server authentication)
LDAP	`KeyUsage`	`digitalSignature` `keyEncipherment`
	`ExtendedKeyUsage`	`serverAuth` (TLS server authentication)
Replication	`KeyUsage`	`digitalSignature` `keyEncipherment`
	`ExtendedKeyUsage`	`clientAuth` (TLS client authentication)[1] `serverAuth` (TLS server authentication) `1.3.6.1.4.1.36733.2.1.10.1` (for Trusted replicas (advanced))

[1] Replication requires both TLS server and TLS client roles.

Follow steps similar to these to install a DS replica with existing cryptographic keys:

1. Before proceeding, install the server files.
   For details, refer to 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, refer to Property value substitution.

Install an HDAP gateway

The DS HDAP gateway web application translates HTTP requests in LDAP requests:

hdap

The HDAP gateway functions as a web application in a web application container. It runs independently of the LDAPv3 directory service. The LDAPv3 directory service must support proxied authorization. In particular, this means you can use the HDAP gateway with current and previous versions of DS.

Installation

1. Review the requirements for installation to verify the HDAP gateway supports your web application container.
2. Deploy the .war file according to the instructions for your web application container; for example:

   ```
   $ cp DS-hdap-servlet-7.5.2.war /path/to/tomcat/webapps/
   ```

   If you use Wildfly, you must unzip the .war file into the deployment directory.
3. Edit the configuration in the deployed gateway web application:

   `WEB-INF/classes/config.json`

   This file defines how the HDAP gateway connects to and interacts with LDAP directory servers.

   At minimum, set the directory server hostnames, port numbers, and proxy user credentials. The proxy user LDAP account performs proxied authorization. In a DS directory server set up for evaluation, the account with simple bind credentials `cn=My App,ou=Apps,dc=example,dc=com` and `password` can act as a proxy user.

   When connecting to the remote directory service over LDAPS or LDAP and StartTLS (recommended), configure the gateway client-side trust manager to trust the server certificates. For help, refer to the examples showing how to trust DS server certificates.

   `WEB-INF/classes/logging.properties`

   This file defines logging properties when you run the gateway in Apache Tomcat.
4. (Optional) Adjust the log level.

   At the default log level of `INFO`, the HDAP gateway logs messages about HTTP requests. For log level definitions, refer to java.util.logging.Level.

   If the HDAP gateway runs in Apache Tomcat, edit the `logging.properties` file. Otherwise, set the log level as described in the container documentation.
5. (Recommended) Configure the web application container to use HTTPS for secure connections to the gateway.

   Refer to the container documentation for details.
6. Restart the HDAP gateway or the web application container.

   The gateway reloads its configuration.
7. Verify the directory service is up and the gateway connects correctly.

Verification

Install and configure the HDAP gateway before following these steps:

1. Set up a DS directory server for evaluation.
2. Read Babs Jensen's resource through the gateway.

If necessary, adjust the protocol (`https`), port (`8443`), and base path (`/hdap`) for your configuration:

```
$ curl \
 --user dc=com/dc=example/ou=People/uid=bjensen:hifalutin \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=cn&_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
  "_rev" : "<revision>",
  "cn" : [ "Barbara Jensen", "Babs Jensen" ]
}
```

You have demonstrated the HDAP gateway works as expected.

Install a DSML gateway

The DSML gateway web application translates each HTTP request into one or more LDAP requests.

> **Important**
>
> The interface stability of this feature is *Deprecated*.

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, refer to Configure DSML access.
4. Configure your web application container to use HTTPS for secure connections to the gateway.

   Refer to 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, refer to 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:

   ```
   root# 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.

GUI

1. Open Control Panel as Windows Administrator.
2. Browse to the page to uninstall a program.
3. Find the **ForgeRock Directory Service** in the list and uninstall it.
4. Manually remove the server configuration files and any directory data.

PowerShell

1. Open PowerShell as Windows Administrator.
2. Use the `msiexec` command.

   The following command quietly removes installed files:

   ```
   C:\> msiexec /x DS-7.5.2.msi /q
   ```

3. 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`	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, refer to 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, and audit logs.
`logs/server.pid`	Contains the process ID for a running server.
`opendj_logo.png`	DS splash logo.
`README`	About DS servers.

File or directory	Description
`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`	Linux setup tool.
`setup.bat`	Windows setup tool.
`template`	Templates for setting up a server instance.
`template/setup-profiles`	Profile scripts to configure directory servers for specific use cases.
`upgrade`	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. Ping Identity 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, refer to the accompanying `README.md` files.

Upgrade

This guide shows you how to upgrade PingDS software.

About upgrades

Read this first.

Upgrade strategies

Choose between upgrading in place and upgrading by adding servers.

Upgrade in place

Overwrite old software to upgrade on the same server.

Upgrade by adding servers

Add new servers then retire old ones.

Read the Release notes before you upgrade DS software.

Product names changed when ForgeRock became part of Ping Identity. PingDS was formerly known as ForgeRock Directory Services, for example. Learn more about the name changes in New names for ForgeRock products in the Knowledge Base.

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

While the replication protocol remains fully compatible with earlier versions, you *must* upgrade from DS 6 or later. Learn more in Supported upgrades.

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.

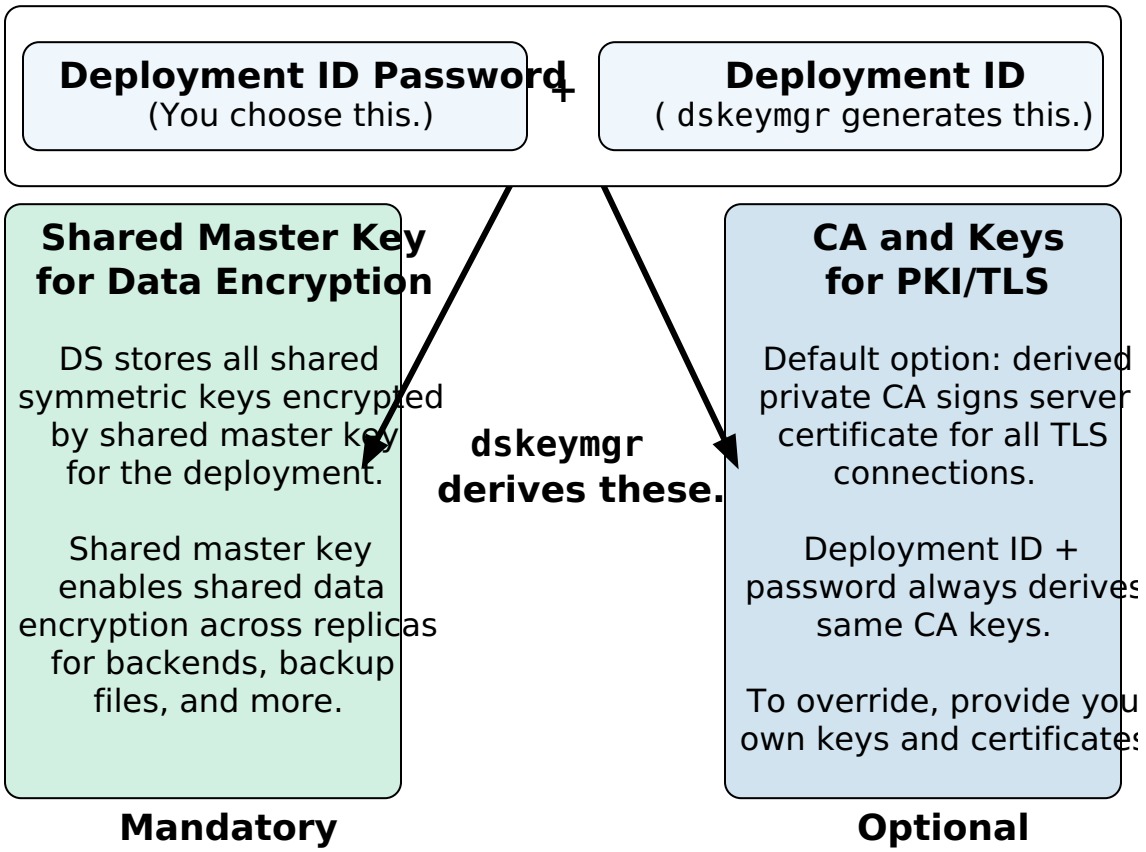
Deployment ID required

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.

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 that 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 add new servers.

Supported upgrades

From…	To…	Important Notes
Official DS release, version 6.0 or later	Official DS release, same edition of directory server or replication server	Supported.
Official ForgeRock release, 2.6.x to 5.5.x	Official DS 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 DS 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*.
Evaluation release	Official DS 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	Official DS release	Not supported.

Upgrade strategies

When you upgrade to a new DS version, you choose between:

- Upgrade in place: unpack the new software over the old software and run the `upgrade` command.
- Add new servers then retire old ones.

> **Important**
>
> For some scenarios, like upgrading Docker images in a Kubernetes deployment, you must upgrade in place.

Upgrade in place

The most straightforward option when upgrading DS servers is to upgrade in place. DS software provides an `upgrade` command to simplify the process.

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 the upgrade, the host system must meet the requirements for both the older version and the new release. For example, you may 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 the 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.
> If the deployment includes DS 7.4.0 servers with data encryption using default settings, you must add new servers instead. For details, read Upgrade from DS 7.4.0.

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 the 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 the 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 that use `dsreplication status` and the newer servers that use `dsrepl status`. Run both commands to get a more complete picture of replication status.
	You must manually enable new features after the upgrade.

Strategy: in-place upgrade

These pages cover *in-place upgrades*. For servers, you unpack the new software over old and run the `upgrade` command, reusing the same host systems. For details about alternative upgrade strategies, refer to Upgrade strategies.

> **Note**
>
> If you use encryption and the default `cipher-transformation` settings, you cannot upgrade in place from DS 7.4.0. Upgrade by adding new servers instead and refer to Upgrade from DS 7.4.0 for details.

Next steps

- ☐ Perform these steps before you upgrade
- ☐ Upgrade each:
  - ☐ Directory server
  - ☐ Directory proxy

- ☐ Replication server
- ☐ HDAP gateway
- ☐ DSML gateway

☐ Perform these steps after you upgrade

Before you upgrade in place

Fulfill these requirements before upgrading PingDS software, especially before upgrading the software in a production environment. Also refer to the requirements listed in release notes.

Global server IDs

Before upgrading, make sure you use unique global server IDs. Prior to DS 7.0, each server could have multiple server IDs. Global server IDs were supported but optional in DS 6.5.

To update each DS 6.5 to use a unique global server ID, set the `server-id` global configuration property. The following example sets the global server ID to `1`:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname opendj.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --set server-id:1 \
 --trustAll \
 --no-prompt
```

Server IDs were originally numeric for compatibility with DS 6.5 and earlier servers. In DS 7.0 and later, use strings as server IDs.

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 and refer to CAs from deployment IDs.

DS software supports the following Java environments:

Vendor	Versions
OpenJDK, including OpenJDK-based distributions: • AdoptOpenJDK/Eclipse Temurin Java Development Kit (Adoptium) • Amazon Corretto • Azul Zulu • Red Hat OpenJDK Ping Identity tests most extensively with AdoptOpenJDK/Eclipse Temurin. Use the HotSpot JVM if possible.	17[1], 21
Oracle Java	17[1], 21

[1] DS requires Java 17.0.8 or later.

TLS cipher support depends solely on the JVM. For details, refer to TLS settings.

CAs from deployment IDs

Due to a change to the Java platform between versions 11 and 17, the key pairs you generate with the `dskeymgr` and `setup` commands using Java 11 are incompatible with keys generated using Java 17 and later.

> **Note**
>
> Running DS servers with incompatible Java versions is a problem when you use deployment ID-based CA certificates. If you use your own CA, not one derived from a deployment ID, skip this section.

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.

Find troubleshooting suggestions in Incompatible Java versions.

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.

Next steps

Perform these steps before you upgrade

- ☐ Upgrade each:
  - ☐ Directory server
  - ☐ Directory proxy
  - ☐ Replication server
  - ☐ HDAP gateway
  - ☐ DSML gateway
- ☐ Perform these steps after you upgrade

Directory server

This page shows how to upgrade a directory server in place.

> **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. Make sure you've completed the tasks in Before you upgrade in place.
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.
        - If the existing server uses a Java version that is no longer supported, follow the steps in Java updates, but do not restart the server yet.
        - 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.

Although unlikely, when the server configuration has changed in an incompatible way with the previous release, the `upgrade` command can fail when performing property value substitution for a configuration expression. If this happens, change the configuration to a static value during upgrade. Use the configuration expression again after you successfully run the `upgrade` command.

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.

5. Start the upgraded server.

   At this point, the upgrade process is complete. Refer to 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.

6. If you disabled the Windows service to upgrade, enable it again:

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

Next steps

Perform these steps before you upgrade

- ☐ Upgrade each:
  - Directory server
  - ☐ Directory proxy
  - ☐ Replication server
  - ☐ HDAP gateway
  - ☐ DSML gateway
- ☐ Perform these steps after you upgrade

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. Make sure you've completed the tasks in Before you upgrade in place.
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.
        - If the existing server uses a Java version that is no longer supported, follow the steps in Java updates, but do not restart the server yet.
        - 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.

    Although unlikely, when the server configuration has changed in an incompatible way with the previous release, the `upgrade` command can fail when performing property value substitution for a configuration expression. If this happens, change the configuration to a static value during upgrade. Use the configuration expression again after you successfully run the `upgrade` command.
4. Start the upgraded server.

    At this point, the upgrade process is complete. Refer to 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
```

Next steps

Perform these steps before you upgrade

- ☐ Upgrade each:
  - Directory server
  - Directory proxy
  - ☐ Replication server
  - ☐ HDAP gateway
  - ☐ DSML gateway
- ☐ Perform these steps after you upgrade

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, refer to Directory server 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. Make sure you've completed the tasks in Before you upgrade in place.
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.
      - If the existing server uses a Java version that is no longer supported, follow the steps in Java updates, but do not restart the server yet.
      - 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.

Although unlikely, when the server configuration has changed in an incompatible way with the previous release, the `upgrade` command can fail when performing property value substitution for a configuration expression. If this happens, change the configuration to a static value during upgrade. Use the configuration expression again after you successfully run the `upgrade` command.

4. Start the upgraded server.

   At this point, the upgrade process is complete. Refer to 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
```

Next steps

Perform these steps before you upgrade

- ☐ Upgrade each:
  - Directory server
  - Directory proxy
  - Replication server
  - ☐ HDAP gateway
  - ☐ DSML gateway
- ☐ Perform these steps after you upgrade

HDAP gateway

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

Next steps

Perform these steps before you upgrade

- ☐ Upgrade each:
  - Directory server
  - Directory proxy
  - Replication server
  - HDAP gateway
  - ☐ DSML gateway

- ☐ Perform these steps after you upgrade

DSML gateway

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

Next steps

Perform these steps before you upgrade

Upgrade each:

Directory server

Directory proxy

Replication server

HDAP gateway

DSML gateway

- ☐ Perform these steps after you upgrade

After you upgrade in place

The DS server upgrade process preserves the existing configuration as much as possible. This maintains compatibility, but there are more steps you must take.

> **Note**
>
> Many example commands on 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.

Checklist

Use this checklist to make sure you don't miss these important post-upgrade tasks:

- ☐ Use the new security model (for upgrades from DS 6.5 and earlier).
- ☐ Back up your directory data.[1]
- ☐ Update your scripts to account for incompatible changes.
- ☐ Plan your move away from deprecated features.
- ☐ Move to dedicated service accounts for your directory applications.[2]
- ☐ Manually review and purge the DS server configurations for stale references to old servers.[3]
- ☐ Review what's new and changed and adopt useful improvements.

- ☐ Eliminate outdated password storage.
- ☐ Clean up admin data (for upgrades from DS 6.5 and earlier).[4]
- ☐ 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.
- ☐ Set the cloud storage endpoint for backup if you back up to cloud storage.

[1] Backup files are *not* compatible between versions. Before starting this task, complete the work to Use the new security model.

[2] You would not run all your applications as the Linux root user or the Windows Administrator. Stop using superuser 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, refer to the `base-entries.ldif` files in setup profiles under the `opendj/template/setup-profiles/AM` directory.

[3] 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.

[4] Before starting this task, complete the work to Use the new security model and to Eliminate outdated password storage.

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 can't add new servers using normal procedures until you have completed these steps*. Some new features depend on the new model.

DS release 7 changes the security model to configure replication at setup time, make disaster recovery more straightforward, and simplify symmetric key distribution:

- In prior releases, trust and symmetric key distribution in a replication topology depend 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 offer 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, refer to 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) have been 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

Follow these steps:

1. On at least one DS replica, add an index for passwords using deprecated or outdated storage schemes.

   The following example creates the index on an upgraded server with data for `dc=example,dc=com` in a backend called `userRoot`. The directory superuser account on the upgraded server has DN `cn=Directory Manager`:

```
dsconfig \
 create-backend-index \
 --backend-name userRoot \
 --type generic \
 --index-name userPassword \
 --set index-type:big-extensible \
 --set big-index-included-attribute-value:3DES \
 --set big-index-included-attribute-value:AES \
 --set big-index-included-attribute-value:Base64 \
 --set big-index-included-attribute-value:Blowfish \
 --set big-index-included-attribute-value:CRYPT \
 --set big-index-included-attribute-value:Clear \
 --set big-index-included-attribute-value:PBKDF2 \
 --set big-index-included-attribute-value:PKCS5S2 \
 --set big-index-included-attribute-value:RC4 \
 --set big-index-included-attribute-value:SHA-1 \
 --set big-index-included-attribute-value:Salted\ SHA-1 \
 --set big-index-included-attribute-value:Salted\ SHA-256 \
 --set big-index-included-attribute-value:Salted\ SHA-384 \
 --set big-index-included-attribute-value:Salted\ SHA-512 \
 --set big-index-extensible-matching-rule:1.3.6.1.4.1.36733.2.1.4.14 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --bindPassword password \
 --trustAll \
 --no-prompt

rebuild-index \
 --baseDN dc=example,dc=com \
 --index userPassword \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --trustAll
```

The `ds-evaluation` setup profile, described in Install DS for evaluation, includes a `userPassword` big index for reversible password storage schemes.

2. Search for accounts using these password storage schemes on the replica where you added the index:

```
ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSSL \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --trustAll \
 --simplePageSize 100 \
 --baseDn dc=example,dc=com \
  "(|(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=3DES)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=AES)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=BASE64)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=BLOWFISH)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=CRYPT)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=CLEAR)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=PBKDF2)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=PKCS5S2)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=RC4)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SHA)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SSHA)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SSHA256)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SSHA384)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SSHA512))" \
 1.1
```

If the search returns no matches, set `enabled: false` for the unused storage schemes in each server configuration. You can skip to the rest of the steps in this procedure.

3. If the search returns any DNs, migrate active accounts to another storage scheme, such as `PBKDF2-HMAC-SHA256`.

   For details, refer to Deprecate a password storage scheme. When a user binds successfully with their existing password or changes their password, DS stores the password with the new scheme.

4. Wait for all active accounts to bind or to update their passwords.

   The definition of *active* depends on the deployment. You decide how long a user can go without binding before you consider their account inactive. For details, refer to Active accounts.

5. Run the search again, adding a filter to match active accounts.

   After the migration, the search ideally returns no results.

6. Once you are confident active accounts no longer use deprecated or outdated storage schemes, set `enabled: false` for the unused storage schemes in each server configuration.

   Inactive accounts—those with no binds during the migration—must now reset their passwords before they can bind.

For additional examples, refer to How do I change a password storage scheme and apply a new password policy to users in DS (All versions)?

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.

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.

   As explained in Checklist 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. 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 enough to run the command once.

   After upgrading from DS 6.5 or earlier, *make sure you set the* `--bootstrapServer` *option*. Target at least two servers for redundancy. The following example uses the servers with IDs `1` and `2` as the bootstrap servers:

   ```
   $ dsrepl \
    cleanup-migrated-pre-7-0-topology \
    --bootstrapServer 1 \
    --bootstrapServer 2 \
    --bindDn "cn=Directory Manager" \
    --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:

```
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
```

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
```

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`	An update of the `mail` attribute to support UTF-8 characters and 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 for the following attributes, which DS considers degraded:

   ◦ `automountInformation`

   ◦ `automountKey`

   ◦ `automountMapName`

- `gecos`
- `ipHostNumber`
- `ipNetworkNumber`
- `mail`
- `memberNisNetGroup`
- `memberUid`
- `nisMapEntry`
- `nisNetgroupTriple`

For details, refer to 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. As part of the upgrade process, adjust the settings appropriately to align your deployment with the new software version.

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.

1. Unpack the libraries required to store backup files in the cloud:

```
$ cd /path/to/opendj
$ unzip -o -q extensions/backup-cloud-extension.zip
```

2. Restart DS.
3. 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
```

Set the cloud storage endpoint for backup

If you back up to cloud storage, set the storage endpoint to control where your backup files go.

Use one of these `dsbackup` options:

- `--storage-property endpoint:endpoint-url`
- `--storage-property endpoint.env.var:environment-variable-for-endpoint-url`

For details, refer to Cloud storage endpoint.

Upgrade complete

Perform these steps before you upgrade

Upgrade each:

Directory server

Directory proxy

Replication server

HDAP gateway

DSML gateway

Perform these steps after you upgrade

Strategy: new servers

These pages cover *upgrade by adding new servers* on new host systems, then retiring old servers. For details about alternative upgrade strategies, refer to Upgrade strategies.

Next steps

- ☐ Perform these steps before you add servers
- ☐ Add new servers:
  - ☐ Follow these instructions when upgrading from DS 6.5 or earlier
  - ☐ Follow these instructions when upgrading from DS 7.4.0
- ☐ Perform these steps after you finish adding servers

Before you add new servers

Fulfill these requirements before upgrading PingDS 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 and refer to CAs from deployment IDs.

DS software supports the following Java environments:

Vendor	Versions
OpenJDK, including OpenJDK-based distributions: • AdoptOpenJDK/Eclipse Temurin Java Development Kit (Adoptium) • Amazon Corretto • Azul Zulu • Red Hat OpenJDK Ping Identity tests most extensively with AdoptOpenJDK/Eclipse Temurin. Use the HotSpot JVM if possible.	17[(1)], 21
Oracle Java	17[(1)], 21

[(1)] DS requires Java 17.0.8 or later.

TLS cipher support depends solely on the JVM. For details, refer to TLS settings.

CAs from deployment IDs

Due to a change to the Java platform between versions 11 and 17, the key pairs you generate with the `dskeymgr` and `setup` commands using Java 11 are incompatible with keys generated using Java 17 and later.

Note

Running DS servers with incompatible Java versions is a problem when you use deployment ID-based CA certificates. If you use your own CA, not one derived from a deployment ID, skip this section.

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.

Find troubleshooting suggestions in Overcome incompatible Java versions when adding new servers.

Next steps

Perform these steps before you add servers

- ☐ Add new servers:
  - ☐ Follow these instructions when upgrading from DS 6.5 or earlier
  - ☐ Follow these instructions when upgrading from DS 7.4.0
- ☐ Perform these steps after you finish adding servers

Add 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:

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`

- For `cn=admin data` and `cn=schema`, use the `dsrepl initialize` command(s) from the previous step.
- For other base DNs, if initializing over the network is appropriate—for example, because there is little directory data under the base DN compared to available network bandwidth—use the `dsrepl initialize` command.

  Otherwise, initialize from LDIF or from backup taken on another new server of the same version. For details, refer to Manual initialization.

> **Tip**
>
> Test the initialization process to make sure you understand the duration and ramifications of the chosen initialization method.
> Use the results to make an evidence-based decision on whether to use backup/restore or export/import instead of online initialization.

6. Align the change number indexing settings on the new servers to match the same settings on the existing servers.
7. Rebuild "degraded" `mail` indexes for the change to the `mail` attribute schema definition to allow UTF-8 characters.

   The definitions for DS 7.3 and later allow UTF-8, whereas earlier versions allow only ASCII. The change does not affect the data, but does affect `mail` indexes.

   For details, refer to Automate index rebuilds.
8. 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.

9. If necessary, add the deprecated `/admin` and `/api` (REST to LDAP) endpoints to the server configuration:

```
# /admin
$ dsconfig \
 create-http-endpoint \
 --endpoint-name /admin \
 --type admin-endpoint \
 --set authorization-mechanism:"HTTP Basic" \
 --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

# /api (REST to LDAP)
$ dsconfig \
 create-http-endpoint \
 --endpoint-name /api \
 --type rest2ldap-endpoint \
 --set authorization-mechanism:"HTTP Basic" \
 --set config-directory:config/rest2ldap/endpoints/api \
 --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
```

Next steps

Perform these steps before you add servers

Add new servers:

Follow these instructions when upgrading from DS 6.5 or earlier

☐ Follow these instructions when upgrading from DS 7.4.0

☐ Perform these steps after you finish adding servers

Upgrade from DS 7.4.0

If the deployment includes a DS 7.4.0 server with data encryption using default settings, follow the procedures in this page.

If the deployment has no DS 7.4.0 servers or does not use data encryption, skip this page.

The problem

Due to an issue (OPENDJ-10211) in the way DS 7.4.0 encrypts data on disk when using the default `cipher-transformation: AES/GCM/NoPadding` setting, the backend or changelog data on disk and encrypted with 7.4.0 is incompatible with all other DS versions.

If the deployment is configured with non-default `cipher-transformation` settings that do not use the AES algorithm and GCM mode, the problem doesn't affect the deployment. In this case, skip this page.

Otherwise, the directory data on disk uses incompatible encryption. Any binary backups of the backend data are also affected. You can't use the `upgrade` command to upgrade a DS server to 7.4.0 from earlier versions or from 7.4.0 to later versions.

The solution

You *can* upgrade by adding new DS servers; follow these steps:

1. Upgrade by adding new servers, leaving existing 7.4.0 servers in operation during the upgrade.

   When initializing new servers, *do not use backup files*, as they use incompatible encryption. Instead, either initialize data over the network or initialize all replicas from plaintext LDIF.

2. Change the bootstrap replication servers for each server to stop using the DS 7.4.0 servers.
3. If you use backup files, create them from the new servers with compatible encryption.
4. Stop directing client application traffic to the DS 7.4.0 servers.
5. Wait until the replication purge delay has elapsed (default: 3 days) and retire the DS 7.4.0 servers.

Next steps

Perform these steps before you add servers

Add new servers:

Follow these instructions when upgrading from DS 6.5 or earlier

Follow these instructions when upgrading from DS 7.4.0

☐ Perform these steps after you finish adding servers

After you add new servers

The DS server upgrade process preserves the existing configuration as much as possible. This maintains compatibility, but there are more steps you must take.

Checklist

Use this checklist to make sure you don't miss these important post-upgrade tasks:

- ☐ Back up your directory data.[1]
- ☐ Update your scripts to account for incompatible changes.
- ☐ Plan your move away from deprecated features.
- ☐ Move to dedicated service accounts for your directory applications.[2]
- ☐ Manually review and purge the DS server configurations for stale references to old servers.[3]
- ☐ Update bootstrap replication servers.[4]
- ☐ Review what's new and changed and adopt useful improvements.
- ☐ Eliminate outdated password storage.
- ☐ Clean up admin data (for upgrades from DS 6.5 and earlier).[5]
- ☐ Add a monitor user account.
- ☐ Update LDAP schema.
- ☐ Tune settings.
- ☐ Set the cloud storage endpoint for backup if you back up to cloud storage.

[1] Backup files are *not* compatible between versions. Before starting this task, complete the work to [upgrade-deployment-ids].

[2] You would not run all your applications as the Linux root user or the Windows Administrator. Stop using superuser 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, refer to the `base-entries.ldif` files in setup profiles under the `opendj/template/setup-profiles/AM` directory.

[3] 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.

[4] 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.

[5] Before starting this task, complete the work to Eliminate outdated password storage.

Eliminate outdated password storage

Reversible password storage schemes (3DES, AES, Blowfish, RC4) have been 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

Follow these steps:

1. On at least one DS replica, add an index for passwords using deprecated or outdated storage schemes.

   The following example creates the index on an upgraded server with data for `dc=example,dc=com` in a backend called `userRoot`. The directory superuser account on the upgraded server has DN `cn=Directory Manager`:

```
dsconfig \
 create-backend-index \
 --backend-name userRoot \
 --type generic \
 --index-name userPassword \
 --set index-type:big-extensible \
 --set big-index-included-attribute-value:3DES \
 --set big-index-included-attribute-value:AES \
 --set big-index-included-attribute-value:Base64 \
 --set big-index-included-attribute-value:Blowfish \
 --set big-index-included-attribute-value:CRYPT \
 --set big-index-included-attribute-value:Clear \
 --set big-index-included-attribute-value:PBKDF2 \
 --set big-index-included-attribute-value:PKCS5S2 \
 --set big-index-included-attribute-value:RC4 \
 --set big-index-included-attribute-value:SHA-1 \
 --set big-index-included-attribute-value:Salted\ SHA-1 \
 --set big-index-included-attribute-value:Salted\ SHA-256 \
 --set big-index-included-attribute-value:Salted\ SHA-384 \
 --set big-index-included-attribute-value:Salted\ SHA-512 \
 --set big-index-extensible-matching-rule:1.3.6.1.4.1.36733.2.1.4.14 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --bindPassword password \
 --trustAll \
 --no-prompt

rebuild-index \
 --baseDN dc=example,dc=com \
 --index userPassword \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --trustAll
```

The `ds-evaluation` setup profile, described in Install DS for evaluation, includes a `userPassword` big index for reversible password storage schemes.

2. Search for accounts using these password storage schemes on the replica where you added the index:

```
ldapsearch \
 --hostname localhost \
 --port 1636 \
 --useSSL \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --trustAll \
 --simplePageSize 100 \
 --baseDn dc=example,dc=com \
  "(|(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=3DES)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=AES)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=BASE64)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=BLOWFISH)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=CRYPT)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=CLEAR)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=PBKDF2)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=PKCS5S2)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=RC4)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SHA)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SSHA)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SSHA256)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SSHA384)\
(userPassword:1.3.6.1.4.1.36733.2.1.4.14:=SSHA512))" \
 1.1
```

If the search returns no matches, set `enabled: false` for the unused storage schemes in each server configuration. You can skip to the rest of the steps in this procedure.

3. If the search returns any DNs, migrate active accounts to another storage scheme, such as `PBKDF2-HMAC-SHA256`.

   For details, refer to Deprecate a password storage scheme. When a user binds successfully with their existing password or changes their password, DS stores the password with the new scheme.

4. Wait for all active accounts to bind or to update their passwords.

   The definition of *active* depends on the deployment. You decide how long a user can go without binding before you consider their account inactive. For details, refer to Active accounts.

5. Run the search again, adding a filter to match active accounts.

   After the migration, the search ideally returns no results.

6. Once you are confident active accounts no longer use deprecated or outdated storage schemes, set `enabled: false` for the unused storage schemes in each server configuration.

   Inactive accounts—those with no binds during the migration—must now reset their passwords before they can bind.

For additional examples, refer to How do I change a password storage scheme and apply a new password policy to users in DS (All versions)?

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 still have DS 6.5 or earlier servers, retire them before continuing.

   As explained in Checklist 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. 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:

   ```
   $ 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.

3. Remove unused configuration settings:

```
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
```

4. 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.

5. 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
```

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`	An update of the `mail` attribute to support UTF-8 characters and 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 for the following attributes, which DS considers degraded:

   ◦ `automountInformation`

   ◦ `automountKey`

   ◦ `automountMapName`

- `gecos`
- `ipHostNumber`
- `ipNetworkNumber`
- `mail`
- `memberNisNetGroup`
- `memberUid`
- `nisMapEntry`
- `nisNetgroupTriple`

For details, refer to 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. As part of the upgrade process, adjust the settings appropriately to align your deployment with the new software version.

For information and suggestions on tuning, read the Release notes and Performance tuning.

Set the cloud storage endpoint for backup

If you back up to cloud storage, set the storage endpoint to control where your backup files go.

Use one of these `dsbackup` options:

- `--storage-property endpoint:endpoint-url`
- `--storage-property endpoint.env.var:environment-variable-for-endpoint-url`

For details, refer to Cloud storage endpoint.

Upgrade complete

Perform these steps before you add servers

Add new servers:

Follow these instructions when upgrading from DS 6.5 or earlier

Follow these instructions when upgrading from DS 7.4.0

Perform these steps after you finish adding servers

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.

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, refer to 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, refer to Use administrative APIs.

Configure HTTP authorization

HTTP authorization mechanisms map HTTP credentials to LDAP credentials.

Multiple HTTP authorization mechanisms can be enabled simultaneously.

These HTTP authorization mechanisms are supported:

HDAP (enabled by default)

Process anonymous, basic and bearer authorization requests.

This mechanism treats anonymous requests like the HTTP Anonymous mechanism.

For HTTP Basic requests, this mechanism matches an HDAP resource `_id` to the DN. The `_id` matches the suffix of the path to the resource. For example, the default directory superuser `_id` is `uid=admin`. Babs Jensen's `_id` is `dc=com/dc=example/ou=People/uid=bjensen`.

For HTTP Bearer requests, this mechanism uses a JSON Web Token (JWT). Get the JWT with the HDAP `authenticate` action. For details, refer to Bearer auth.

HTTP Anonymous (enabled by default)

Process anonymous HTTP requests, optionally binding with a specified DN.

If the client does not specify a bind DN (default), it binds as an anonymous LDAP user.

HTTP Basic (enabled by default)

Process HTTP Basic authorization requests by mapping the HTTP Basic identity to a user's directory account.

By default, DS uses the exact match identity mapper with its default configuration to map the HTTP Basic username to an LDAP `uid`. DS searches all local public naming contexts to find the user's entry based in the `uid` value. For details, refer to Identity mappers.

HTTP OAuth2 CTS

Process OAuth 2.0 requests as a resource server, acting as the AM Core Token Service (CTS) store.

When the client bearing an OAuth 2.0 access token presents the token to access the JSON resource, DS tries to resolve the access token against its CTS data. If DS finds the access token and it has not expired, DS extracts the user identity and OAuth 2.0 scopes. If the required scopes are present and the token is valid, DS maps the user identity to a directory account.

This mechanism makes an internal request, avoiding a request to AM. *This mechanism does not ensure the token has been replicated to the DS serving the request.*

The AM 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, DS requests token information from AM. If the access token is valid, DS extracts the user identity and OAuth 2.0 scopes. If the required scopes are present, DS maps the user identity to a directory account.

Send the requests to AM over HTTPS. Configure a truststore manager if necessary to trust the AM authorization server certificate. Configure a keystore manager if necessary to send the DS server certificate for 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, DS requests token introspection from the authorization server. If the access token is valid, DS extracts the user identity and OAuth 2.0 scopes. If the required scopes are present, DS maps the user identity to a directory account.

Send the requests to the authorization server over HTTPS. Configure a truststore manager if necessary to trust the authorization server certificate. Configure a keystore manager if necessary to send the DS server certificate for 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, DS applies the mechanisms in the following order:

- If the client request has an `Authorization` header and an OAuth 2.0 mechanism is specified, DS 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, DS attempts to apply the Basic Auth mechanism.
- Otherwise, if an HTTP anonymous mechanism is specified, and none of the previous mechanisms apply, DS 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 PingAM 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`

Read the PingAM 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 HDAP 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 "/hdap" \
 --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": "<access-token>"",
 "scope": "uid read write",
 "token_type": "Bearer",
 "expires_in": 3599
}
```

   Use HTTPS when obtaining access tokens.

4. Request a resource with HTTP Bearer authentication with the access token:

```
$ curl \
--header "Authorization: Bearer <access-token>" \
--cacert ca-cert.pem \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_prettyPrint=true'
```

   Use HTTPS when presenting access tokens.

Use administrative APIs

The APIs for configuring and monitoring DS servers are under the following endpoints:

`/alive`

Check whether the server is currently *alive*, meaning 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, refer to Server is alive (HTTP).

`/healthy`

Check whether the server is currently *healthy*, meaning it is alive, the replication server is accepting connections on the configured port, and any replication delays are below the configured threshold.

By default, this endpoint returns a status code to anonymous requests, and supports authenticated requests. For details, refer to Server health (HTTP).

`/metrics/prometheus`

Access the server monitoring information in Prometheus monitoring software format.

By default, DS protects this endpoint with the HTTP Basic authorization mechanism. Users reading monitoring information must have the `monitor-read` privilege.

To use these APIs, follow these steps:

1. Grant access to the `/metrics/prometheus` endpoint, if necessary, by assigning the `monitor-read` privilege.

   For details, refer to Administrative privileges.

   Alternatively, create a monitor user with the `setup` command when installing DS.

2. Adjust the `authorization-mechanism` settings for the Admin endpoint.

   By default, DS 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:

   - If the request has an `Authorization: Basic` header for HTTP Basic authentication, DS extracts the username and password.
   - If the request has `X-OpenIDM-Username` and `X-OpenIDM-Password` headers, DS extracts the username and password.
   - DS 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 LDAP terms, it searches all user 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, refer to Identity mappers and Configure HTTP authorization.

3. Test access to the endpoint as an authorized user.

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, refer to Supported locales. For the list of other matching rules, refer to 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, refer to 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, refer to 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: 7.5.2`

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 class of service'
  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.

There is no requirement to change LDAP schema definitions before using this feature. To get started, refer to HDAP API reference.

- **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`
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). Refer to 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, refer to JSON Ordering Matching Rule.

For examples showing how to add LDAP schema for new attributes, refer to Update LDAP Schema. For examples showing how to index JSON attributes, refer to 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, refer to 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.

File	Description
`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.
`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

These pages show you how to configure and debug indexes so DS can respond quickly to searches:

- About indexes
- What to index
- Index types
- Indexing tools
- Configure indexes
- Verify indexes
- Debug a missing index

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

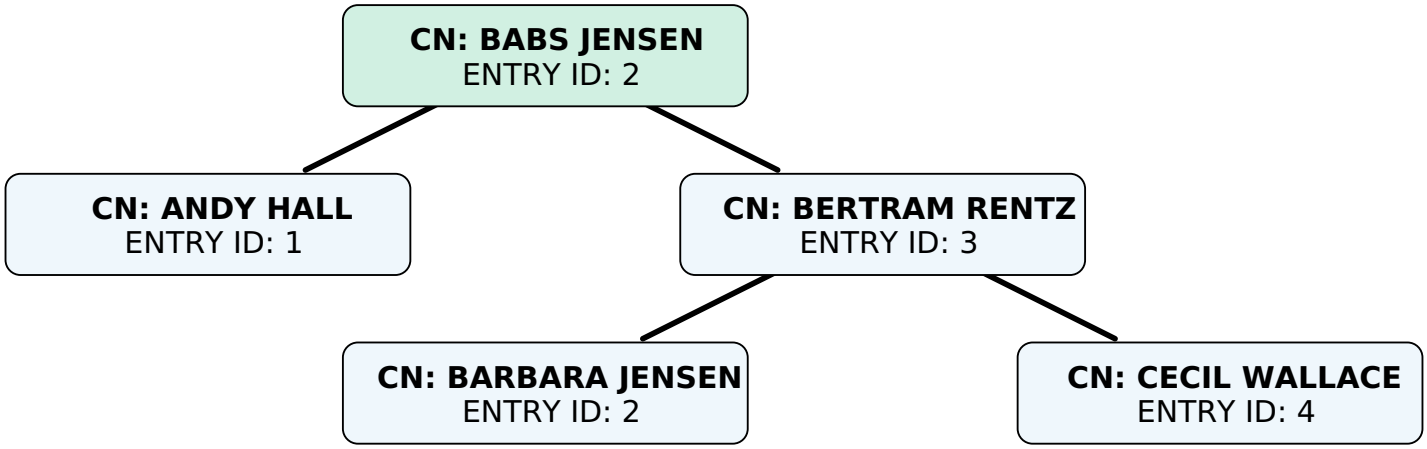
The role of an index is to answer the question, "Which entries have an attribute with this corresponding value?"

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 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 peak hours.

  Make sure the application has no resource limits. For details, refer to Enforce 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 a large company, or all the users in the state of California. For details, refer to 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, refer to 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 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.

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.

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 find in a search filter? If an attribute is going to show up frequently in reasonable search filters, then index it.

Standard indexes by search filter

LDAP filter	HDAP filter	Recommended indexes[1]	Explanation
`(&)` `(objectClass=*)`	`_queryFilter=true`	None	This filter matches every entry in scope. Indexes cannot help. Narrow the scope of the search as much as possible.
`(uid=*)`	`_queryFilter=uid+pr`	`uid` presence index	A presence index matches an attribute present on the entry, regardless of the value.
`(uid=bjensen)`	`_queryFilter=uid+eq+'bjensen'`	`uid` equality index	An equality index matches values that correspond exactly to those in search filters.
`(uid=*jensen*)`	`_queryFilter=uid+co+'jensen'`	`uid` substring index	A substring index matches values specified with wildcards in the filter.
`(uid=jensen*)`	`_queryFilter=uid+sw+'jensen'`	`uid` equality index or `uid` ordering index	The filter value `jensen*` is also called an *initial substring*. When filters use *only* initial substrings, configure an equality or ordering index for the attribute.[1]
`(&(uid=*jensen*)` `(cn=babs*))`	`_queryFilter=(uid+co+'jensen'and+cn+sw'babs')`	`uid` substring index, `cn` substring index	Both filters match substrings.
`(!(uid=*jensen*))`	`_queryFilter=!` `(uid+co+'jensen')`	None	This NOT filter matches all entries except a few. Extend this to an AND filter if possible where the other component filters make use of indexes.
`(uid<=jensen)`	`_queryFilter=uid+le+'jensen'`	`uid` ordering index	An ordering index is used to match values for a filter that specifies a range.
`(uid>=jensen)`	`_queryFilter=uid+ge+'jensen'`	`uid` ordering index	The greater than or equal comparison also uses ordering.

LDAP filter	HDAP filter	Recommended indexes[1]	Explanation
`(sn~=jansen)`	Not applicable	`sn` approximate index	An approximate index matches values that "sound like" those provided in the filter.
`(st=CA)`	`_queryFilter=st+eq+'CA'`	`st` big index	Big indexes fit the case where a large number of entries have the same attribute value. In this example, every user in California matches `(st=CA)`. Client applications must page through the results to avoid exceeding resource limits. For details, refer to Indexes for attributes with few unique values.

[1] DS servers optimize the search internally to:

- Rearrange the search filter terms as long as the logical result is the same.
- Ignore indexes for some filters when other filters and their indexes sufficiently narrow the list of candidates. For example, if the search filter is `(&(sn=Jensen)(st=CA))`, DS might not use the `st` equality index.
- Use alternative indexes depending on the matching rule from the filter operator, falling back to an alternative index when the best-choice index does not exist. This works for standard matching rules as shown in the following table, not for extensible matching rules.

Index use by matching rule

Matching rule	Example search filter	DS index use
-	`(&)`	For the "always true" filter, potentially fall back to a VLV index if the search has the same sort keys as the index.

Matching rule	Example search filter	DS index use
`equality`	`(uid=bjensen)`	Try an equality or ordering index; fall back to a presence index. Fall back to a VLV index when the search filter is either: • An equality filter that targets a single-valued attribute matching a VLV sort key. • A complex intersection (`&`) filter where one or more equality sub-filters target single-valued attributes, and the sub-filter attributes for equality assertions followed by the sort keys for the search match the sort keys for the VLV index. DS can use this optimization even when some of the sub-filters are not equality assertions.
`ordering`	`(uid>=bjensen)`	Try an equality or ordering index; fall back to a presence index.
`approximate`	`(sn~=jansen)`	Try an approximate index; fall back to a presence index.
`presence`	`(uid=*)`	Try a presence index; fall back to an equality or ordering index.
`substring`	`(uid=*babs*)`	Try an equality or ordering index for *initial substrings*; otherwise, use a substring index and fall back to a presence index.

Compare your guesses with what you observe actually happening in the directory. For more complex filters, refer to Debug search indexes.

Unindexed searches

Directory users might complain their searches fail because they're unindexed.

Ask for the result code, additional information, and search filter. 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": true
    }
  }
}
```

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 observe a message like this in the server log, take these actions:

- Figure out which indexes are missing, and add them.

  For details, refer to Index analysis metrics, Debug search indexes, and Configure indexes.

- Check the integrity of the indexes.

  For details, refer to 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, refer to 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 recommended 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-indexed ds-mon-backend-filter-unindexed ds-mon-backend-filter-use-start-time ds-mon-backend-
filter-use

dn: ds-cfg-backend-id=dsEvaluation,cn=backends,cn=monitor
ds-mon-backend-filter-indexed: 2
ds-mon-backend-filter-unindexed: 3
ds-mon-backend-filter-use-start-time: <timestamp>
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-indexed` and `ds-mon-backend-filter-unindexed` metrics are available even when index filter analysis is disabled.

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 observe 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 either:

- The access log message `response` > `additionalItems` > `debugSearchIndex` object for the unindexed search.
- A search for the `debugsearchindex` attribute.

The default global access control only allows the directory superuser to read the `debugsearchindex` attribute. To allow another user 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 objects are intended to be read by human beings, not scripts. If you do write scripts to interpret `debugsearchindex` values and `debugSearchIndex` objects, be aware the format is not stable. Be prepared to adapt your scripts for every upgrade or patch.

The `debugsearchindex` / `debugSearchIndex` object indicates how the server would process the search. The server uses 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": "FIRST_OF",
        "rank": "RANGE_MATCH",
        "filter": "(givenName=aa*)",
        "subQueries": [{
          "query": "FIRST_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": "FIRST_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.

If an index already exists, but you suspect it is not working properly, refer to 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 attribute 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 index is unused before taking action.

You can remove unused attribute indexes with one of the following commands, depending on the type of index:

- `dsconfig delete-backend-index`
- `dsconfig delete-backend-vlv-index`

> **Important**
>
> Never remove a system index.

Index cost

DS maintains metrics about index cost, which lets you determine which indexes are causing write contention. The metrics count the number of updates and how long they took 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-cost
```

Prometheus

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null |
grep index_cost
```

If the cost is persistently very high, and an attribute index is not used, or hardly used, then consider removing it. Never remove a system index.

Indexing tools

Command	Use this to...
`backendstat`	Drill down into the details and inspect index contents during debugging.

Command	Use this to...
`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, refer to 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, refer to Active accounts.

Name: **`relativeTimeOrderingMatch.gt`**

Greater-than relative time matching rule for time-based searches.

For an example using this matching rule, refer to Active accounts.

Name: **`relativeTimeOrderingMatch.lt`**

Less-than relative time matching rule for time-based searches.

For an example using this matching rule, refer to 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 the `index-extensible-matching-rule` setting takes an OID, not the name of the matching rule.

Indexes for attributes with few unique values

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. Use the following options to set the index type and the matching rule. You must set at least one `big-index-extensible-matching-rule`:

- `--set index-type:big-extensible`
- `--set big-index-extensible-matching-rule:OID`

  Notice the `big-index-extensible-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, to create a big index for the `sn` (surname) attribute with `caseIgnoreMatch`, use `--set big-index-extensible-matching-rule:2.5.13.2`. To create a big index for password storage schemes, use `--set big-index-extensible-matching-rule:1.3.6.1.4.1.36733.2.1.4.14`.

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, refer to JSON query matching rule index
Custom JSON equality or ordering matching rule	Specific field(s)	Yes, refer to 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 an `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, refer to 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, refer to 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 displaying 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
```

You can also list the results with HDAP:

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user uid=admin:password \
 --data '_queryFilter=true' \
 --data '_fields=cn,l,sn' \
 --data '_sortKeys=l,sn,cn' \
 --data '_pageSize=5' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
```

```
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=user.93953",
    "_rev" : "<revision>",
    "cn" : [ "Access Abedi" ],
    "l" : [ "Abilene" ],
    "sn" : [ "Abedi" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.40283",
    "_rev" : "<revision>",
    "cn" : [ "Achal Abernathy" ],
    "l" : [ "Abilene" ],
    "sn" : [ "Abernathy" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.67240",
    "_rev" : "<revision>",
    "cn" : [ "Alaine Alburger" ],
    "l" : [ "Abilene" ],
    "sn" : [ "Alburger" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.26994",
    "_rev" : "<revision>",
    "cn" : [ "Alastair Alexson" ],
    "l" : [ "Abilene" ],
    "sn" : [ "Alexson" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.53853",
    "_rev" : "<revision>",
    "cn" : [ "Alev Allen" ],
    "l" : [ "Abilene" ],
    "sn" : [ "Allen" ]
  } ],
  "resultCount" : 5,
  "pagedResultsCookie" : "<cookie>",
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```


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 an 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
```

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.

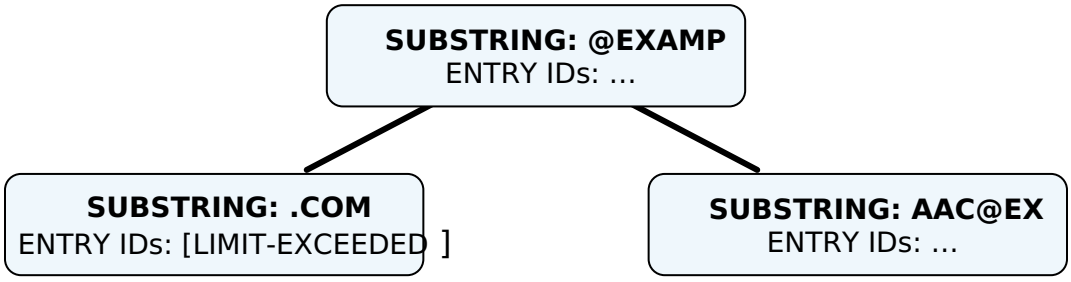
Index entry limits

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, refer to 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 observing many unindexed searches targeting 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 observe 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": "FIRST_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": "FIRST_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 are stored on the `mobile` attribute in the directory.

The LDAP client application has 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 observe 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": true
    },
    "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               ...
```

You can determine why a filter is unindexed by:

- Referring to the `response` > `additionalItems` > `debugSearchIndex` object on the access log message for the unindexed search.
- Running the search with the `debugsearchindex` attribute.

If the data and indexes changed significantly since the search ran, the `debugSearchIndex` object and `debugsearchindex` output can be different.

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": "FIRST_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 has a working search:

```
$ 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 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 platform 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

Important

Separate backends let you use different configuration settings for different data with different access patterns. While doing this allows you more flexibility, it also comes at the price of more administrative and maintenance tasks to manage. Because of this, don't create more backends than you need.
Each new backend implies new administrative tasks. When you create a backend:

- Add backup tasks for the new backend.
- Define restore procedures for the new backend.
- Include the backend in your disaster recovery plans.
- Consider whether shared cache for all JE database backends still makes sense for your deployment.

Before creating a backend *whose base DN is a child of an existing backend*, also read Subordinate backends carefully. Subordinate backends include important requirements and limitations.

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

Index	Approx.	Equality	Ordering	Presence	Substring	Entry Limit
`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 all replicas from LDIF for context.

Important

- Importing from LDIF overwrites the data in the target backend with entries from the LDIF data.
  For a backend with multiple base DNs, importing the data for one of the base DNs doesn't affect the data for the other base DNs.
- 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.

Use the manage-tasks command to manage scheduled tasks. For background, read Server tasks. For an example command, refer to Status and tasks.

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.

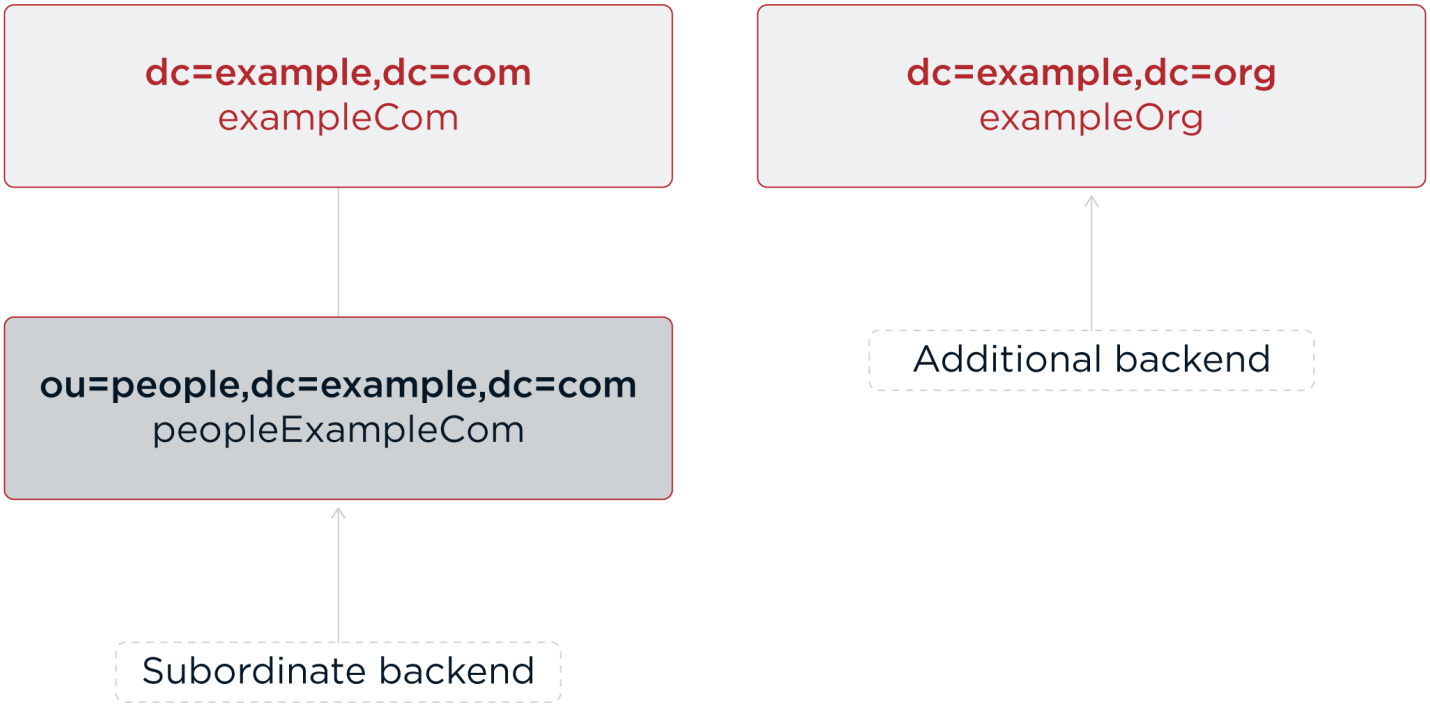
You can specify a time for the task to start using the format yyyymmddHHMMSS. For example, `20250101043000` specifies a start time of 4:30 AM on January 1, 2025.

If the server is not running at the time specified, it attempts to perform the task after it restarts.

Use the manage-tasks command to manage scheduled tasks. For background, read Server tasks. For an example command, refer to Status and tasks.

Subordinate backends

A *subordinate backend* has a base DN making it a child of an existing backend. For example, a DS server has an `exampleCom` backend for `dc=example,dc=com`, a subordinate `peopleExampleCom` backend for `ou=People,dc=example,dc=com`, and an `exampleOrg` backend:



(Technically, all DS backends are subordinate to the root DSE backend whose base DN is the empty string. You don't manage the root DSE, however.)

Subordinate backends include **important requirements and limitations**:

- DS supports paging, sorting, and VLV search results for searches in a single backend. DS doesn't support paging, sorting, and VLV searches traversing backends.

  Client applications know base DNs, however, not backends. Use of subordinate backends can lead to unexpected, logically wrong search results when the scope extends from the parent to the child backend.

- Like other backends, you must add backup tasks and restore procedures, and include subordinate backends in disaster recovery plans.

  Unlike other backends, you must perform all tasks and procedures for subordinate backends *at the same time you perform them for parent backends*. This ensures the replicated data remains aligned.

- Subordinate backends don't reduce network traffic between replication servers, and the replication changelog doesn't use less disk space. Instead, the servers use slightly more system resources in file descriptors and threads for each added backend.

Tip

If you migrate to DS from another directory service, don't use subordinate backends simply because you used them in the other directory service.
Try concatenating the data and using a single backend instead:

1. Exporting existing backends to LDIF.
2. Concatenate the child LDIF after the parent LDIF in a single LDIF file.
3. Import the concatenated LDIF into a single DS backend.

If you accept the requirements and limitations and the deployment appears to require a subordinate backend, contact Ping Identity to validate your plans before deploying in production.

To set up a subordinate backend, stop the servers and follow this example for each server:

1. Create an `exampleCom` backend with a `dc=example,dc=com` replication domain.

   This backend holds all Example.com data except `ou=People,dc=example,dc=com`. The backend `base-dn` must match the replication domain `base-dn`:

```
# Create a backend for the data not under the subordinate base DN:
$ dsconfig \
 create-backend \
 --backend-name exampleCom \
 --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 exampleCom \
 --excludeBranch ou=People,dc=example,dc=com \
 --ldifFile example.ldif \
 --offline
```

2. Create a `peopleExampleCom` backend with an `ou=People,dc=example,dc=com` replication domain.

   This backend holds only Example.com data for `ou=People`. The backend `base-dn` must match the replication domain `base-dn`:

```
# Create a backend for the data under the subordinate base DN:
$ dsconfig \
 create-backend \
 --backend-name peopleExampleCom \
 --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 peopleExampleCom \
 --includeBranch ou=People,dc=example,dc=com \
 --ldifFile example.ldif \
 --offline
```

3. Start the server.

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": true}` 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 display 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 Group membership and Referential integrity.

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 Group 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.

Group membership

DS servers let you verify which groups a user belongs to by reading their entry. The virtual `isMemberOf` attribute shows which groups a user is in.

Reading the user entry is more efficient than reading large static group entries and checking the lists of members:

```
$ 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.

For a dynamic group, you can check the membership directly on the candidate member's entry using the same search criteria as the dynamic group's member URL.

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, refer to 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 dc=com/dc=example/ou=People/uid=bjensen:hifalutin \
--cacert ca-cert.pem \
--get \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=mail&_prettyPrint=true'

{
  "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
  "mail" : [ "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 observe 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
```

Rename an attribute

You can rename a collective attribute with a special form of `inheritAttribute` value. In the collective attribute subentry, set:

`inheritAttribute: <collective-attribute>:<inherited-attribute>`

The following example inherits the `postalAddress` from the manager with the RFC 3671 name, `c-PostalAddress`:

```
dn: cn=Inherit postal address from manager,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: inheritedCollectiveAttributeSubentry
objectClass: inheritedFromDNCollectiveAttributeSubentry
cn: Inherit postal address from manager
subtreeSpecification: { base "ou=People", specificationFilter "(objectClass=posixAccount)" }
inheritFromDNAttribute: manager
inheritAttribute: postalAddress:c-PostalAddress
```

Note

The <collective-attribute> must be defined in the LDAP schema.
If the client uses the <inherited-attribute to filter or sort, it must also be defined in the schema.
In this example, the default LDAP schema defines both attributes.

Try the example with the following steps:

1. Add the subentry:

```
$ 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=Inherit postal address from manager,dc=example,dc=com
objectClass: top
objectClass: subentry
objectClass: inheritedCollectiveAttributeSubentry
objectClass: inheritedFromDNCollectiveAttributeSubentry
cn: Inherit postal address from manager
subtreeSpecification: { base "ou=People", specificationFilter "(objectClass=posixAccount)" }
inheritFromDNAttribute: manager
inheritAttribute: postalAddress:c-PostalAddress
EOF
```

2. Add a `postalAddress` to the manager'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=trigden,ou=people,dc=example,dc=com
changetype: modify
add: postalAddress
postalAddress: 1234 Main St.$Anytown, CA 12345$USA
EOF
```

3. Read `c-PostalAddress` on the employee's entry:

```
$ 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)" \
 c-PostalAddress
dn: uid=bjensen,ou=People,dc=example,dc=com
c-PostalAddress: 1234 Main St., CA 12345
```

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.

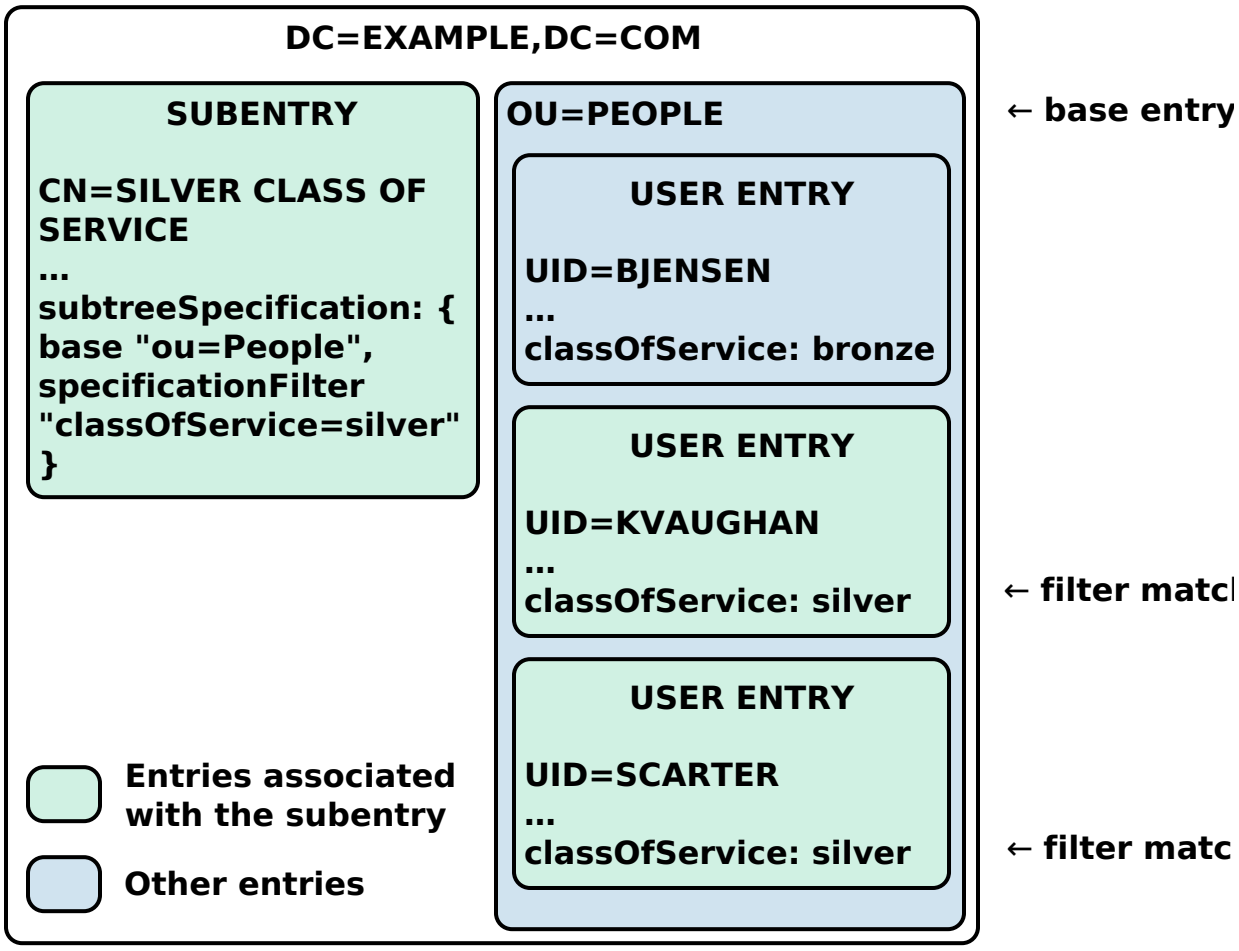
`specificationFilter`

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
- Manual initialization
- Add a new replica
- Replication status

- Manual purge
- Replication groups (advanced)
- Subtree replication (advanced)
- Fractional replication (advanced)
- Read-only replicas
- Trusted replicas (advanced)
- Listen addresses
- Disk space thresholds
- Recover from user error
- Replication conflicts
- Bootstrap replication servers
- Disable replication
- Monitor replication

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 fast and slow networks. For advanced deployments across multiple sites with many replicas and slow links, consider standalone servers. For details, refer to Install standalone servers (advanced).

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), can 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, can 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, refer to 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:

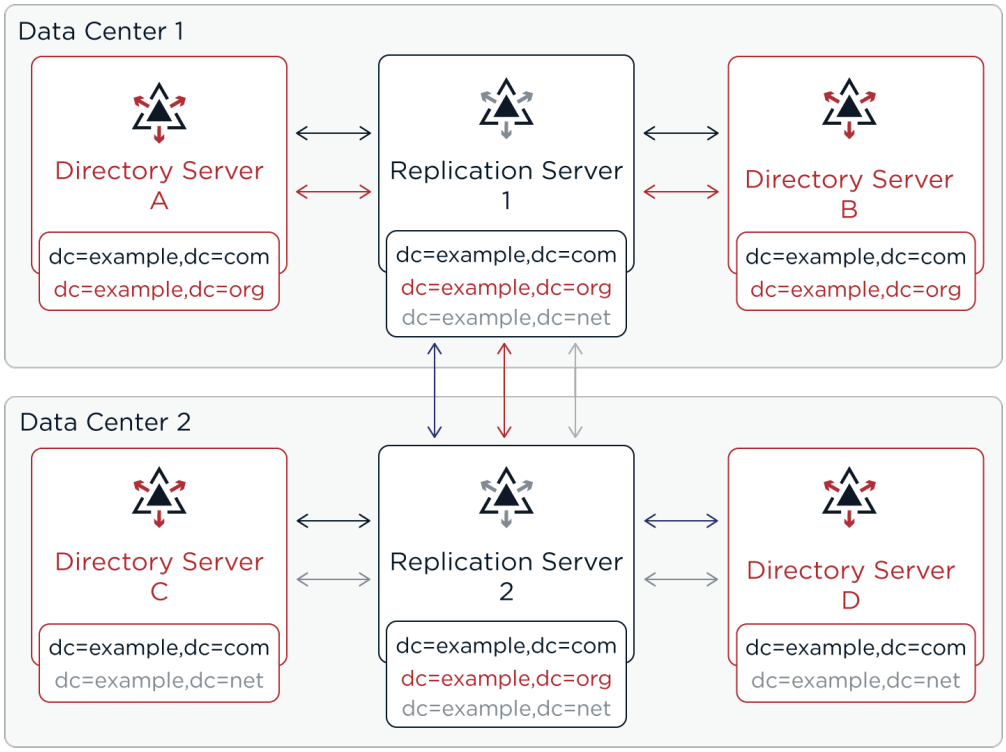


Figure 1. Correct replication configuration

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:

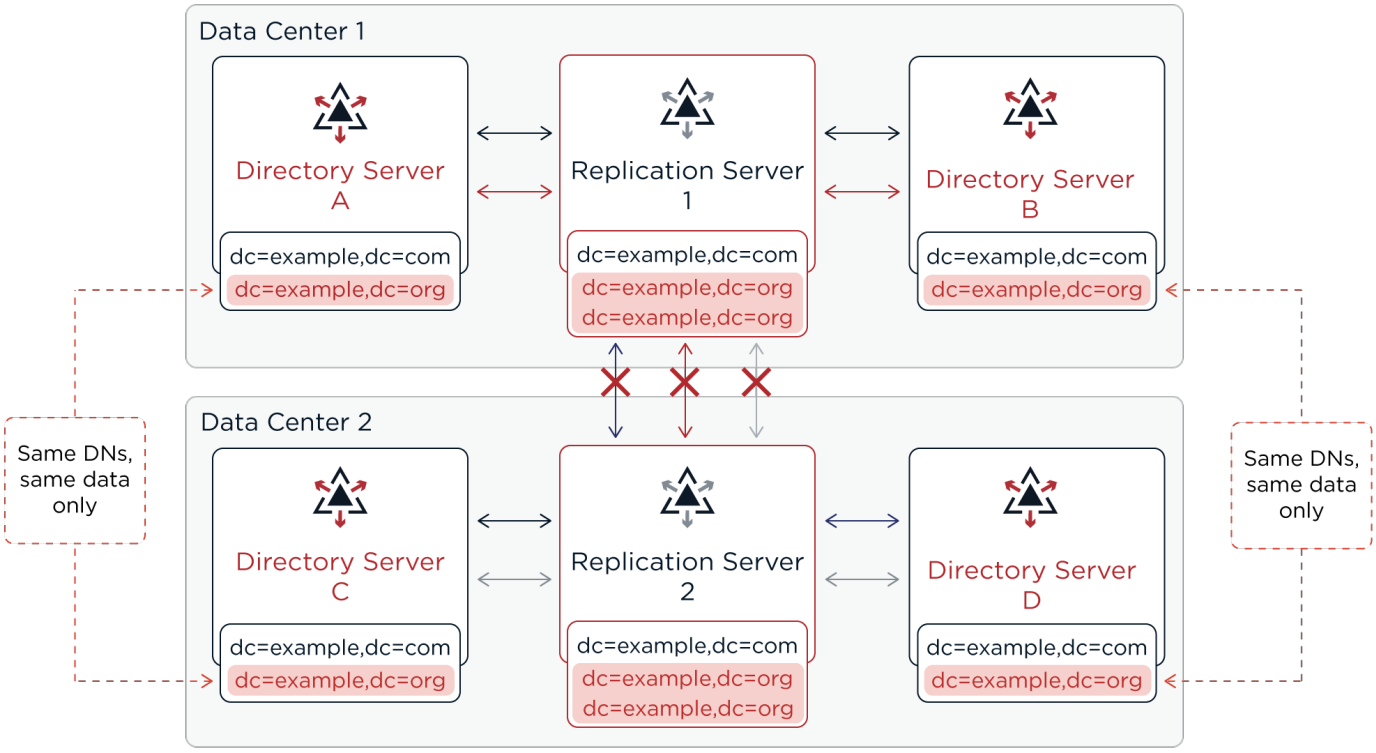


Figure 2. Incorrect replication configuration

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 (advanced)
Some replicas have only a subset of LDAP attributes.	Fractional replication (advanced)

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:

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.

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:

Replication connection selection

DS servers can provide both directory services and replication services. The two services are not the same, even if they usually run alongside each other in the same DS server.

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 over the network when replicas start from the same initial data.

If this is not the case, manually initialize replication.

> **Tip**
>
> Test the initialization process to make sure you understand the duration and ramifications of the chosen initialization method.
> Use the results to make an evidence-based decision on whether to use backup/restore or export/import instead of online initialization.

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 Plenty of bandwidth to replicate data	Initialize over the network *Limitations:* • Transmits all data over the network, so requires ample bandwidth; can be a problem for slow links. • Rebuilds indexes and consumes significant system resources; can impact service performance.
New directory service, medium to large data set (> 1 million entries; limited bandwidth)	Initialize all replicas from LDIF *Limitations:* • Rebuilds indexes and consumes significant system resources; can impact service performance. • Doesn't correct bad data in the changelog database.
Existing directory service, medium to large data set (> 1 million entries; limited bandwidth)	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. • Doesn't correct bad data in the changelog database.
New backend	Create a backend, then one of: • Initialize over the network • Initialize all replicas 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 all replicas from LDIF

> **Important**
>
> If you aim to initialize replicas with LDIF from a non-replicated environment, follow the steps in Disaster recovery instead.

Review Initialization options before following these steps:

Initialize each replica with the same LDIF:

1. Stop the server.
2. If desired, enable data confidentiality.

    Learn more in Data encryption and Encrypt External Changelog Data.

3. Import the LDIF.

    Learn more in 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.

   Learn more in Restore.
3. Start the replica.

   Replication replays changes from other replicas that have happened since the backup was created.

Add a new replica

Adding a new replica means installing a new DS server.

- If you have the user data as LDIF and can import it using one of the setup profiles, set the new replica up with the appropriate setup profiles, importing the data from LDIF at setup time.
- If you must import user data from LDIF you can't import through a setup profile, or you opt to get the user data by restoring backup files from another replica:

  1. Set the new replica up with the appropriate setup profiles, *omitting the* `setup --start` *option to prevent the new server from starting*.
  2. With the new replica stopped, import the LDIF *offline* or restore the data from backup *offline*.
  3. Start the new replica.

  > **Important**
  >
  > If you accidentally started the new replica before you restored the data from backup, remove the new replica and start over.

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, refer to 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 (advanced)

> **Tip**
>
> This information applies to *advanced* deployments.

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 (advanced)

> **Tip**
>
> This information applies to *advanced* deployments.

To configure subtree replication, split the data across more than one backend, each with its own replication domain.

For example, use different backends for `ou=People,dc=example,dc=com` and other `dc=example,dc=com` data:

- Configure a backend and replication domain for everything in `dc=example,dc=com` except `ou=People,dc=example,dc=com`.
- Configure a separate backend and replication domain for `ou=People,dc=example,dc=com`.

The backend `base-dn` and the replication domain `base-dn` settings must match in each case.

Subordinate backends include important requirements and limitations. Before you implement this feature, read Subordinate backends carefully.

Fractional replication (advanced)

> **Tip**
>
> This information applies to *advanced* deployments.

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 (advanced)

> **Tip**
>
> This information applies to *advanced* deployments.

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, refer to 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 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 `rm` (remove) 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, refer to Changelog for notifications. In particular, refer to 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:

```
# ; conn=; op=
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:

# ; conn=; op=
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:

# ; conn=; op=
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 Disaster recovery. This alternative is only recommended in the case of a disaster.

- 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 blank 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, refer to 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, refer to 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, refer to 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, refer to 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, refer to the Upgrade documentation instead.

To remove a server that is no longer needed:

1. Uninstall the server.

   For details, refer to Uninstallation.

2. If the server is referenced in other servers' `bootstrap-replication-server` settings, remove it.

   For details, refer to 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, refer to Read-only replicas.

2. Perform these steps in parallel on all affected servers:

   1. Disable the replication mechanism.

      For details, refer to 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, refer to 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.

Monitor replication

To keep replication running smoothly, monitor DS servers for problems such as excessive and persistent replication delay or blocked change number indexing.

DS servers give you a choice for monitoring replication. You can use HTTP and Prometheus software or LDAP access to `cn=monitor`. For details on what to watch and how to interpret the metrics, refer to the appropriate documentation:

- Monitoring replication over HTTP
- Monitoring replication over LDAP

For additional troubleshooting instructions, also refer to Replication problems.

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 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 \
 --control "ecl:false" \
 "(&)"

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, refer to 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 generate symmetric keys when needed and store them, 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, refer to 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 support applications using the Internet-Draft: Definition of an Object Class to Hold LDAP Change Records instead of changelog cookies.

Replication servers must perform Change number indexing to get the objects specified for this format, and you must perform some steps 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. Enable change number indexing on all replication servers.
2. Make sure 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.
3. If you must start the new server's change numbering from a specific change, decide which `changeNumber` to use.

   The `changeNumber` must be from a change that has not yet been purged following the replication purge delay (default: 3 days).
4. 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 server's changelog.

Change number indexing

By default, replication servers let application use cookies instead of change numbers when searching the changelog.

Adjust the replication server `changelog-enabled` settings appropriately on each replication server in your deployment:

- If applications need change notifications and use changelog cookies, leave the default setting `changelog-enabled: enabled-cookie-mode-only`.
- If applications need change notifications and use change numbers, use `changelog-enabled: enabled`.

> **Important**
>
> Change number indexing requires more CPU, disk access, and storage. Enable it only when applications require change number-based browsing.

- If no applications need notifications, use `changelog-enabled: disabled`.
- If applications need notifications for some domains but not for others, refer to Exclude a domain.

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. This example 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
```

Based on the referral entry, the server returns a referral for operations under `ou=Subscribers`:

LDAP

```
$ 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
```

HDAP

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data '_queryFilter=uid+pr' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=subscribers'
{
  "code" : 404,
  "reason" : "Not Found",
  "message" : "Referral: A referral entry ou=Subscribers,dc=example,dc=com indicates that the operation must
be processed at a different server"
}
```

To access the entry instead of the referral, use the Manage DSAIT control:

LDAP

```
$ 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
```

HDAP

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data '_fields=ref' \
 --data '_prettyPrint=true' \
 --data 'manageDsaIT=true' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=subscribers'
{
  "_id" : "dc=com/dc=example/ou=Subscribers",
  "_rev" : "<revision>",
  "ref" : [ "ldaps://referral.example.com:1636/ou=Subscribers,dc=example,dc=com" ]
}
```

You can use the Manage DSAIT control to change the referral:

LDAP

```
$ 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
```

HDAP

```
$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --data '[{
  "operation": "replace",
  "field": "ref",
  "value": "ldap://localhost:1636/ou=People,dc=example,dc=com"
 }]' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=subscribers?manageDsaIT=true'
```

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, refer to 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.

   Refer to 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, stores accounts because Linux 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. Refer to 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.

LDAP proxy

PingDS 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 PingDS, 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 observe 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.

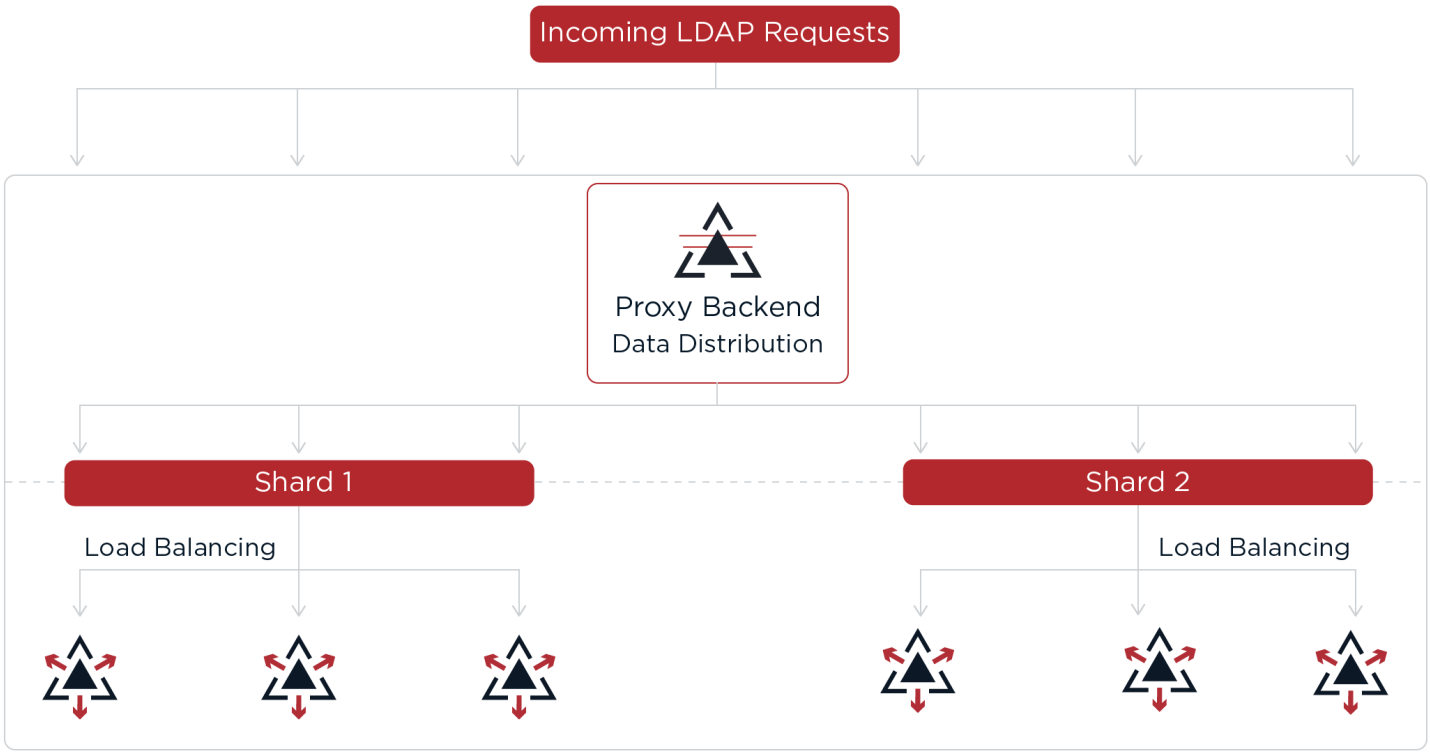
Make sure client applications for configuration and monitoring access servers directly for the server-specific data they need.

- 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.

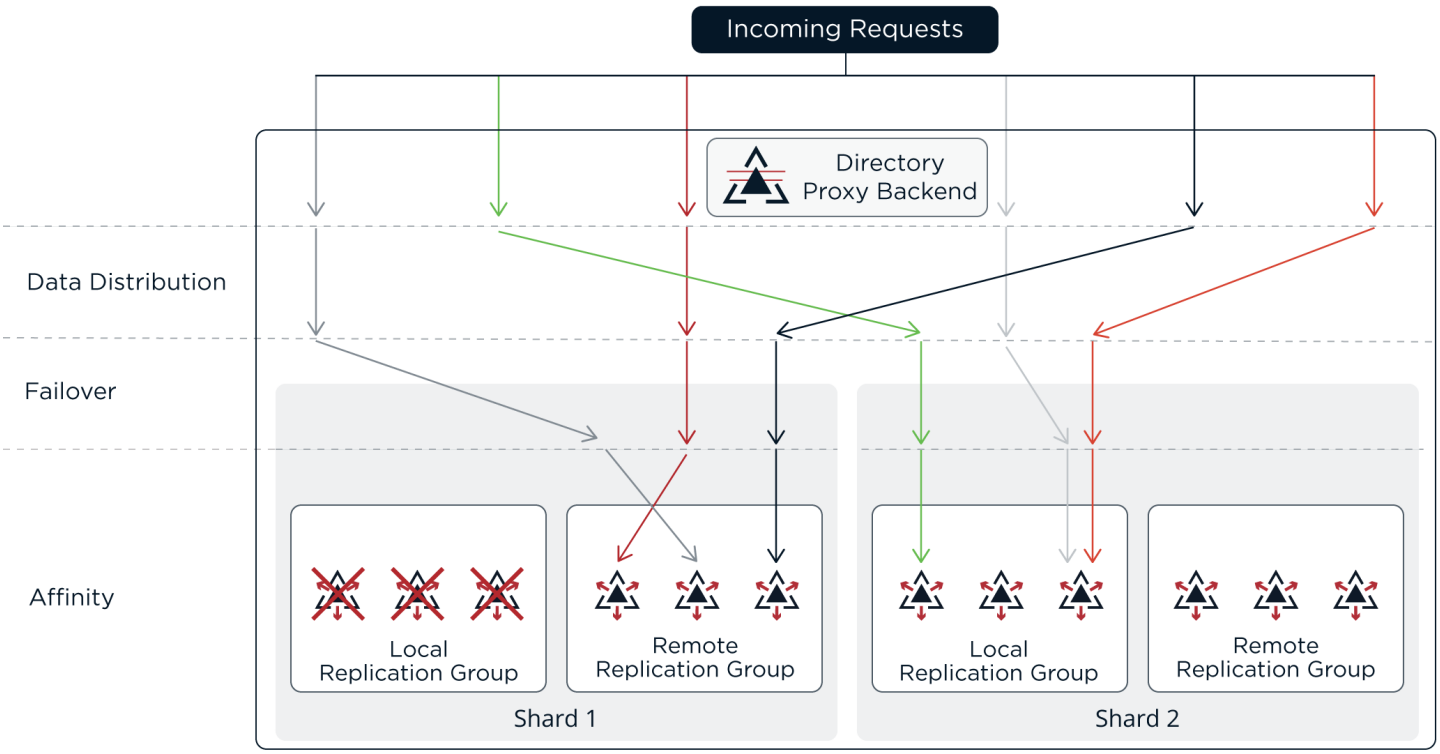
Load balancing

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* Refer to Data distribution.
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. For details, refer to 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

Feature	Characteristics
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 observed 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, refer to LDAP schema.

If the remote servers are not DS servers, refer to 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, refer to 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, refer to 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. Refer to 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 observe 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, refer to 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 on how to update a server's configuration, refer to Enforce limits.

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.

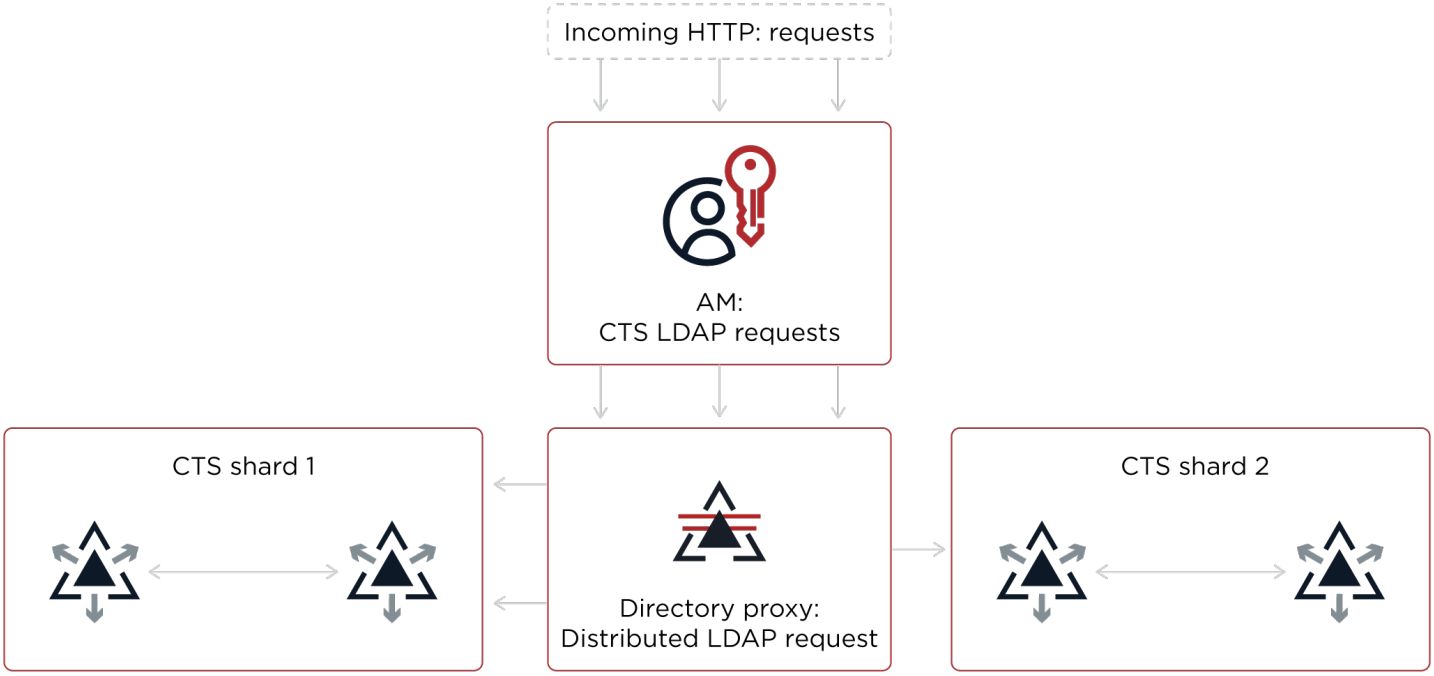
- Subtree and one-level searches can be relatively expensive, as the proxy must potentially forward the search request to every shard to retrieve all search results.

  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-7.5.2.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 2018-2023 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-7.5.2-20250513124640-de1088550ebabfaf1b3577b53f3eb9fc3b03739c.zip
BASE_DIR=/path/to
FQDN=localhost
DEPLOYMENT_ID=ASIQ8nH5BypHIB7AmmmJZ5VfCVKZXg5CBVN1bkVDAIT3myJF1rRsipI
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 2018-2023 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-7.5.2-20250513124640-de1088550ebabfaf1b3577b53f3eb9fc3b03739c.zip
BASE_DIR=/path/to
FQDN=localhost
DEPLOYMENT_ID=ASIQ8nH5BypHIB7AmmmJZ5VfCVKZXg5CBVN1bkVDAIT3myJF1rRsipI
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, refer to the AM installation documentation.

   You can use the default configuration options during installation.

7. Configure the AM server to use the directory proxy server.

   For details, refer to Configure 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 get 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 observe 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, refer to Test session 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 2018-2023 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 using the create-connection-handler command, and setting 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 gets different data 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, Ping 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 Ping Identity Platform 7.5 component

Do not use a load balancer between Ping Identity Platform components and the DS directory service.

Ping 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, refer to the reference documentation.

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, refer to 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, as described in the reference documentation.

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.

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, refer to 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, refer to Authentication mechanisms.

- Encourage best practices for client applications, such as scrubbing input to avoid injection vulnerabilities.

  For details, refer to 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.

- 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, refer to 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, refer to 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, refer to 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, refer to the suggestions in Insecure client applications, Poor password management, and Access control.

To protect against unauthorized access by administrators, refer to 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, refer to 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, refer to 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, refer to Access control.

For details about privileges, refer to 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.
- 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.

For details, refer to Monitoring.

Operating systems

When you deploy PingDS 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.
- 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.

Protocols	Ports[1]	Active by default?	Description
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.

[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 HDAP gateways run as web applications in containers like Apache Tomcat. Security settings depend on the container and on 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, refer to SSL/TLS Configuration HOW-TO.
- For other security-related settings, refer to 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.

HDAP 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.

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 users to run commands, such as the `ldapsearch` command.
- Prevent other users from reading configuration files.

  On Linux, 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, refer to 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 Linux `umask` or Windows file access control settings.

Changelog files

To encrypt change log files, refer to Encrypt 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 Linux 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 Linux `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 Linux `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 Linux systems. It does not apply to Common Audit file-based log publishers. Its value is a Linux 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.

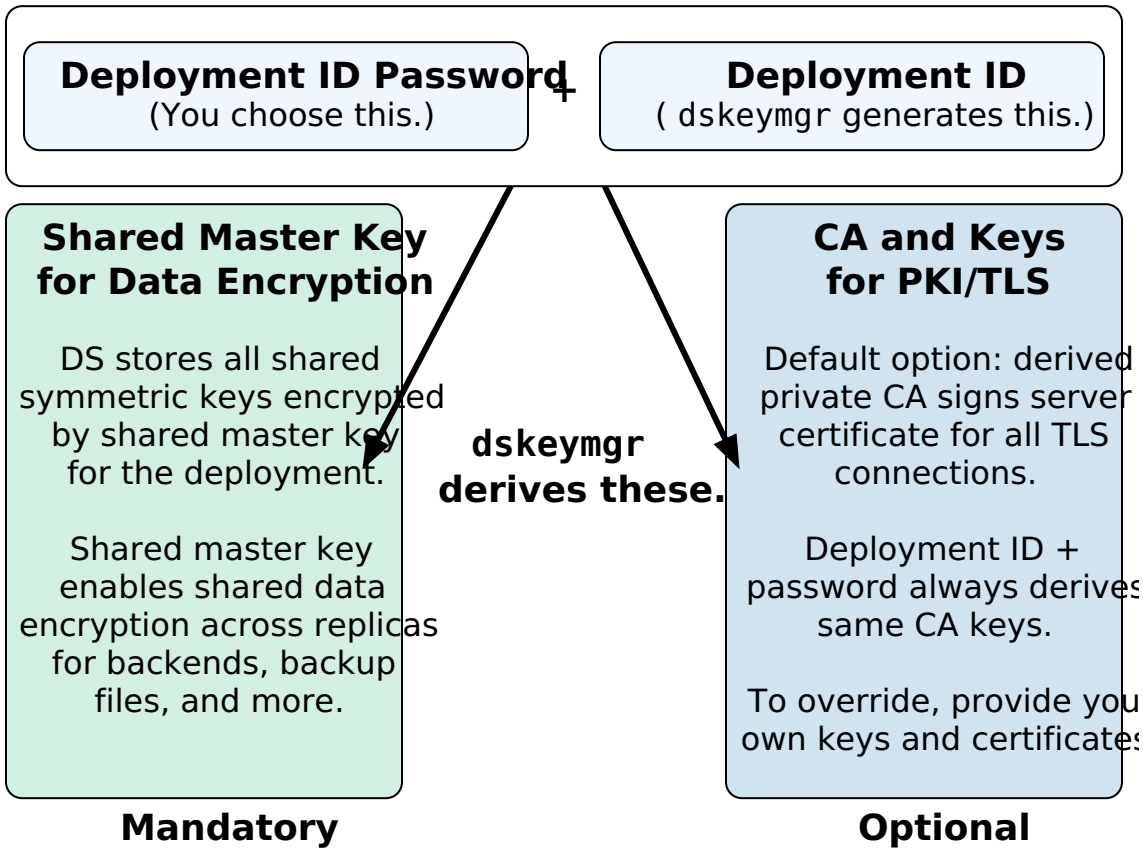
  To avoid exposing secret keys, DS servers encrypt secret keys with a shared master key. DS software uses a deployment ID and its password to derive the master key.

- (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:

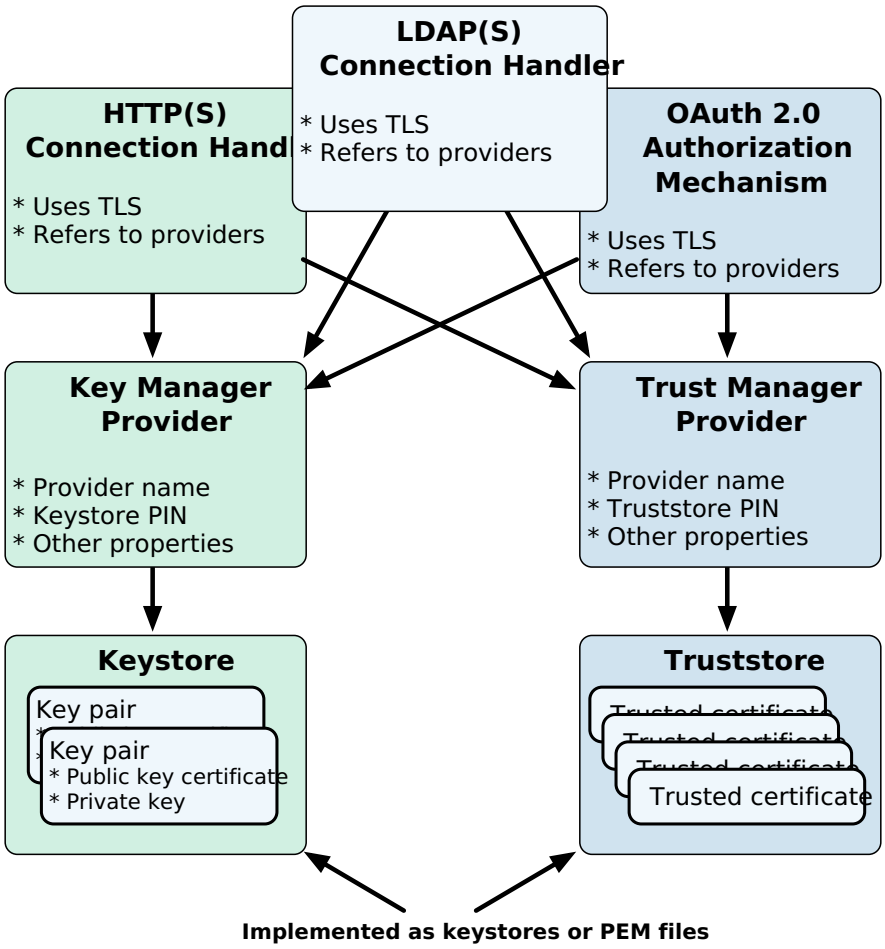
- *Symmetric keys* (also known as secret keys)
- *Asymmetric key pairs* (public/private key pairs)

	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 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 generate symmetric keys when needed and store them, 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.

Use	Private CA	Public CA	Rationale
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
```

The default validity for the certificate is one year.

If the command displays an error about interoperability problems with the deployment ID, read Overcome incompatible Java versions to generate new keys.

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, refer to 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.

      If the command displays an error about interoperability problems with the deployment ID, read Overcome incompatible Java versions to generate new keys.

   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.

   The server keeps the retired key in the backend indefinitely for decryption.

Replace deployment IDs

Follow these steps if you must:

- Switch to a new deployment ID after moving from Java 11 or earlier to Java 17 or later.
- Replace a lost or compromised deployment ID password.
- Renew an expiring deployment ID-based CA.

  The default validity period is 10 years.

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.

   Only remove the obsolete shared master key after all these conditions are met:

   - All servers have restarted at least once since you replaced the shared master key.
   - The time since you replaced the shared master key is longer ago than the DS replication purge delay (default: three days).
   - All backups taken before you replaced the shared master key are purged.

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.

   Refer to 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, refer to 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, refer to 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, refer to 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, refer to 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, refer to 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, refer to 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, refer to 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 ASIQ8nH5BypHIB7AmmmJZ5VfCVKZXg5CBVN1bkVDAIT3myJF1rRsipI
Not before: 2024-03-15T06:42:38Z
Not after: 2034-03-13T06:42:38Z
Version: 1
Serial number: 84F79B2245D6B46C8A92
Provider name: SunEC
```

If you depend on an expiring deployment ID CA, refer to 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, refer to 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, Store 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, refer to 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

Note

- The examples on this page use a SoftHSM PKCS#11 software device for evaluation, development, and testing.
  The command to configure the SoftHSM build in this example is the following:
  `./configure --with-openssl=/usr/local/opt/openssl --with-sqlite3=/usr/local/opt/sqlite --with-objectstore-backend-db --disable-p11-kit`
  Your build configuration options might be different.
- The examples use the `sun.security.pkcs11.SunPKCS11` provider implementation with SoftHSM.
  If you use a different Java implementation for the provider, refer to the documentation for details on how to use PKCS#11 devices with your JVM.
- Adapt the examples for use with your HSM.

1. Install SoftHSM.

   For details, refer to the SoftHSM documentation for details.

2. Initialize the SoftHSM 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.

3. Prepare a Java PKCS#11 configuration file to generate key pairs and configure DS.

   For example, to use the Java `keytool` command with the device, create a PKCS#11 configuration file for the security provider implementation.

   The file name is `/path/to/hsm.conf`. Replace <number> in the following example 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:

```
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 platform software.

4. Verify Java can access the device using your configuration.

   The following example shows how 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 following examples 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
```

Refer to 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. Create the PKCS#11 key manager provider configuration.

   The following example creates a provider for SoftHSM:

   - The `pkcs11-provider-name:SunPKCS11` setting uses the Java name for the PKCS#11 provider.
   - The `pkcs11-provider-arg:/path/to/hsm.conf` setting holds the absolute path to the Java PKCS#11 configuration file.
   - The configuration uses the `SOFTHSM_USER_PIN` environment variable to get the PIN.

     DS key manager providers also support storing the PIN in the configuration, in a file, or in a Java property.

```
$ dsconfig \
 create-key-manager-provider \
 --provider-name SoftHSM \
 --type pkcs11 \
 --set enabled:true \
 --set pkcs11-provider-name:SunPKCS11 \
 --set pkcs11-provider-arg:/path/to/hsm.conf \
 --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
```

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 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 you will discard.

However, if you start DS with any data at all, it protects secret keys for encryption with the generated shared master key, and DS must continue to use that shared master key to decrypt the data.

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 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.
5. Start each DS server.

Secure connections

Securing connections depends on PKI and asymmetric, public/private key pairs. For details, refer to Cryptographic keys.

About secure connections

Incoming connections are from clients to the directory. To match port numbers with protocols, refer to 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 (deprecated)	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.
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 username/password credentials. Client applications can use SSL client authentication. When using DS HDAP gateway, use HTTPS to protect client connections.
JMX	Secure JMX access with the SSL/TLS-related properties, such as `use-ssl` and others.

Protocol	Recommendations
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 username/password credentials. Client applications can use SSL client authentication. Refer to 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.

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 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 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 common audit event handlers to use HTTPS or TLS when connecting to external log services.
DSML gateway (deprecated)	The DS DSML gateway connects to remote LDAP directory servers. Use LDAP and StartTLS or LDAPS to protect the connections.
HDAP gateway	The DS HDAP gateway connects to remote LDAP directory servers. Use LDAP with 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.

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, refer to 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, refer to 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, refer to `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, refer to 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, refer to 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, refer to *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, refer to Certificate-based authentication.

OCSP and obtaining CRLs depend on network access to the CA. If DS servers or the DSML or HDAP 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, refer to 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 \
 --set ssl-protocol:TLSv1.3 \
 --set ssl-cipher-suite:TLS_AES_128_GCM_SHA256 \
 --set 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 \
 --set ssl-protocol:TLSv1.3 \
 --set ssl-cipher-suite:TLS_AES_128_GCM_SHA256 \
 --set 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, refer to 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, refer to 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, refer to Certificate-based authentication.

GSSAPI-based authentication is useful for interoperation with Kerberos. For additional information, refer to 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, refer to 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, refer to Pass-through authentication.

DS servers and HDAP gateways 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 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 logging can display exceptions about unsupported encryption types when a mismatch occurs. The exceptions are visible only when you activate debug-level 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 (FQDN) 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.

   The following example binds the server to `server-fqdn`, the DNS-resolvable FQDN for the system:

   ```
   $ 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
   ```

> **Tip**
>
> To accept Kerberos authentication requests for more than one service principal, add the `--set bind-to-server-fqdn:false` advanced option.
> In addition, the JVM may require setting `--set server-fqdn:*` instead of using the FQDN.
> Some JVMs may not support running without binding to the server FQDN.

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, refer to 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. Password policies govern passwords:

- Which password policy applies
- Configure password policies
- List subentry password policies

- Assign password policies
- Strong and safe passwords
- Sample password policies
- About password policies

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.

Which password policy applies

The operational attribute, `pwdPolicySubentry`, identifies an account's password policy. The default global access control instructions grant no access to this operational attribute. 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

> **Important**
>
> Make sure you keep per-server password policy settings aligned across replicated DS servers. When per-server password policy settings differ between replicas, the results can be surprising to end users.
> As an example, suppose the user's password policy depends on a password storage scheme enabled on the replica where the user changes their password and disabled on the replica where they later authenticate:
>
> - The user changes their password on the first replica.
>   The password update succeeds.
>   Replication replays the change.
> - The user authenticates on the second replica.
>   Authentication fails even though the second replica has the correct password hash. The password storage scheme is disabled in the local per-server configuration.

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
```

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, refer to Assign a password policy to a user.

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, refer to 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, refer to About subentry scope.

3. Check 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.

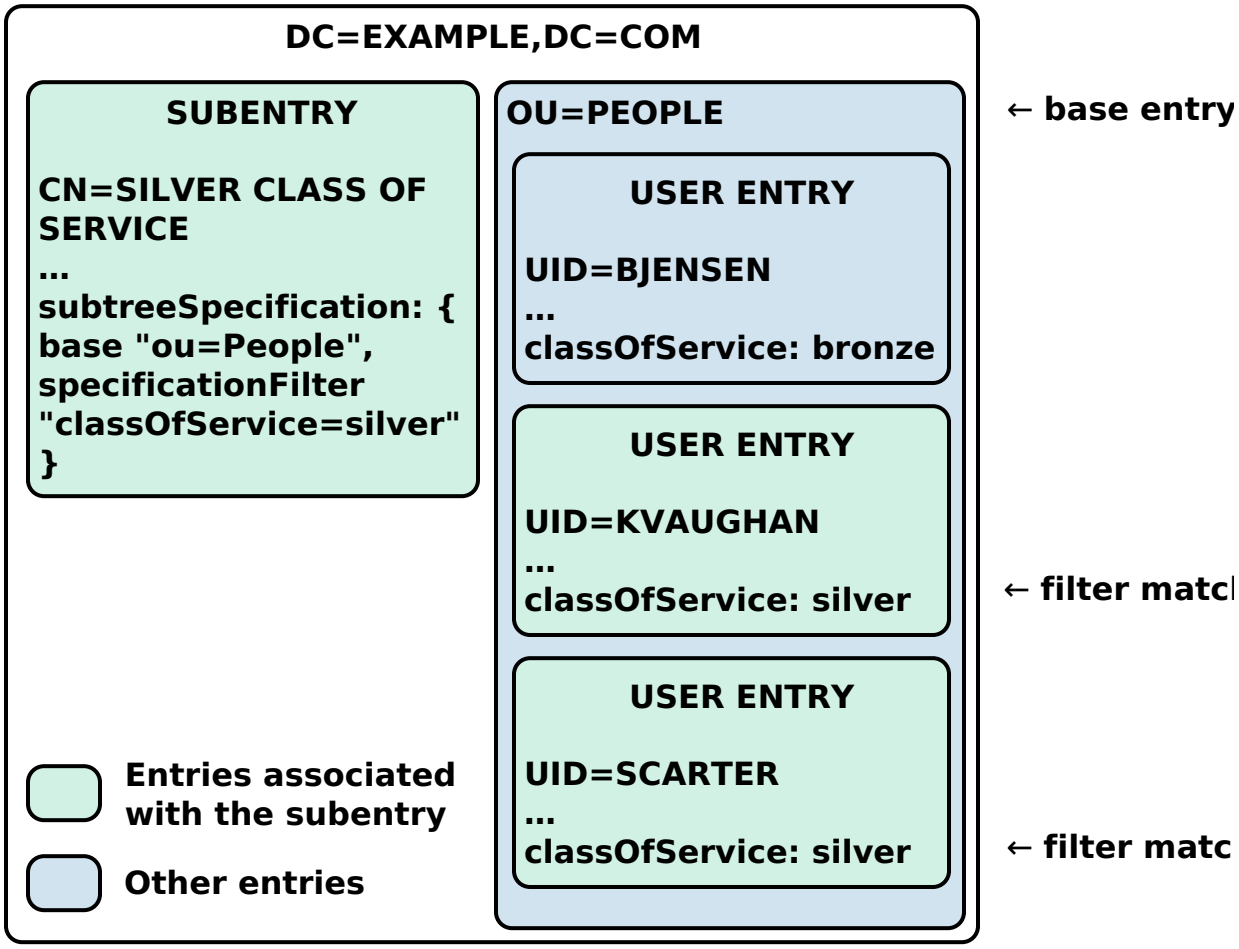
`specificationFilter`

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
```

Important

Make sure you keep per-server password policy settings aligned across replicated DS servers. When per-server password policy settings differ between replicas, the results can be surprising to end users.
As an example, suppose the user's password policy depends on a password storage scheme enabled on the replica where the user changes their password and disabled on the replica where they later authenticate:

- The user changes their password on the first replica.
  The password update succeeds.
  Replication replays the change.
- The user authenticates on the second replica.
  Authentication fails even though the second replica has the correct password hash. The password storage scheme is disabled in the local per-server configuration.

For details, refer to Password Validator.

For an example showing how to test password quality, refer to 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 (LEGACY)           : triple-des         : false
AES (LEGACY)            : aes                : false
Argon2                  : argon2             : true
Base64 (LEGACY)         : base64             : false
Bcrypt                  : bcrypt             : true
Blowfish (LEGACY)       : blowfish           : false
Clear (LEGACY)          : 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 (LEGACY)   : 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 (LEGACY)          : 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, refer to 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.*
PBKDF2	One-way hash	Computationally intensive hashing function, based on PBKDF2 algorithm defined in RFC 8018, 5.2. PBKDF2. Default iterations: 10,000. The pseudorandom function for the algorithm corresponds to the HMAC based on SHA-1.

Name	Type of Algorithm	Notes
PBKDF2-HMAC-SHA256	One-way hash	Computationally intensive hashing function using PBKDF2. Default iterations: 10,000. **Note** When you install DS, the `setup` command configures a `PBKDF2-HMAC-SHA256` password storage scheme with 10 iterations instead of the default 10,000 iterations. The server's default password policy uses this storage scheme. 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: 10,000. 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: 10,000.
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: 10,000. 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. To stop using them, refer to Eliminate outdated password storage (after upgrading in place) or Eliminate outdated password storage (after adding new servers).

(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: 15,360.
	`argon2-memory-pool-size`	The amount of memory dedicated to Argon2 password hashing, expressed in kibibytes. Default: 122,880.
	`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. Default: 10,000. **Note** When you install DS, the `setup` command configures a `PBKDF2-HMAC-SHA256` password storage scheme with 10 iterations instead of the default 10,000 iterations. The server's default password policy uses this storage scheme.
	`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. Default: 10,000.

Change a password storage scheme

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.

The setting in the default password policy is a per-server setting.

Important

Make sure you keep per-server password policy settings aligned across replicated DS servers. When per-server password policy settings differ between replicas, the results can be surprising to end users.
As an example, suppose the user's password policy depends on a password storage scheme enabled on the replica where the user changes their password and disabled on the replica where they later authenticate:

- The user changes their password on the first replica.
  The password update succeeds.
  Replication replays the change.
- The user authenticates on the second replica.
  Authentication fails even though the second replica has the correct password hash. The password storage scheme is disabled in the local per-server configuration.

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 determine the password storage scheme in use. 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, refer to 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
```

Note

When modifying a password, DS checks the new password against each hashed password value in the entry. If the password policy specifies a computationally intensive password storage scheme, such as Argon2, Bcrypt, PKCS5S2, or a PBKDF2-based scheme, enabling password history multiplies the cost of changing the password. DS must calculate the computationally intensive hash from the new password separately for each comparison with a hashed password in the password history. As a result, if it takes 100 ms to calculate the hash for a new password, and the applicable password policy has a password history count of 7, the calculations to modify the password can take up to 700 ms.

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
```

The `last-login-time-format` setting must:

- Use `GeneralizedTime` syntax.
- Be a valid format string for the `java.text.SimpleDateFormat` class.

With the setting shown in the example, `last-login-time-format:"yyyyMMddHH'Z'"`, DS records last login time to the nearest hour. For each bind where the timestamp changes, DS updates the timestamp on the entry. So this recommended setting avoids updating entries often for users who bind repeatedly over a short period. If the deployment requires a fine-grained last login timestamp, use a format that includes minutes or seconds. For example, to get last login times that are accurate to the second, use `last-login-time-format:"yyyyMMddHHmmss'Z'"`.

For examples using last login time in LDAP searches, refer to Active accounts.

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.

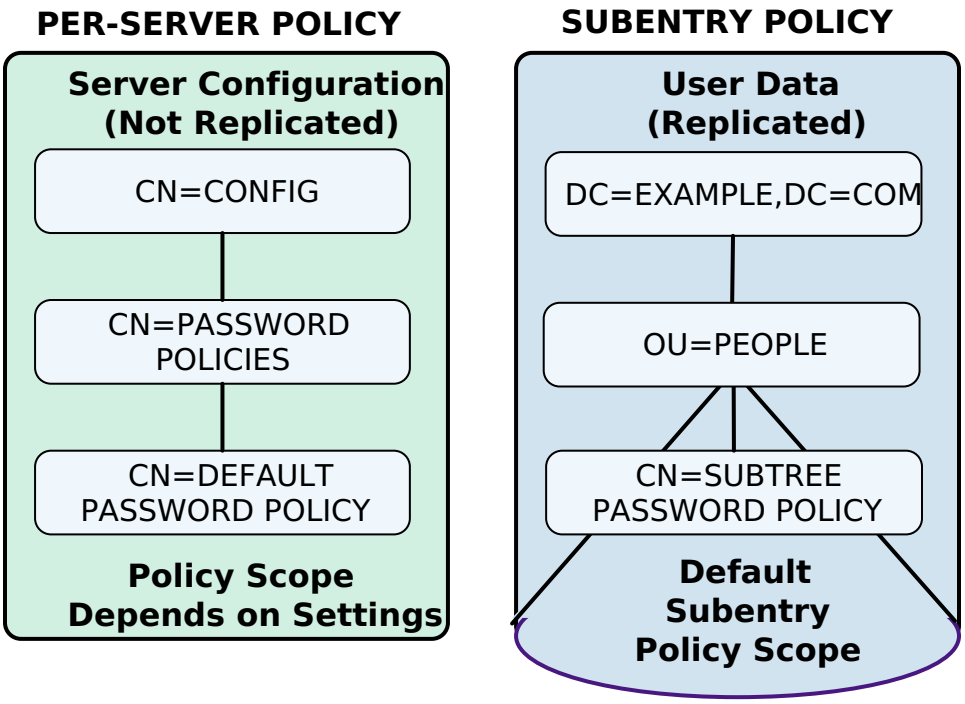


Figure 1. Per-Server and subentry password policies

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`.

Important

Make sure you keep per-server password policy settings aligned across replicated DS servers. When per-server password policy settings differ between replicas, the results can be surprising to end users.
As an example, suppose the user's password policy depends on a password storage scheme enabled on the replica where the user changes their password and disabled on the replica where they later authenticate:

- The user changes their password on the first replica.
  The password update succeeds.
  Replication replays the change.
- The user authenticates on the second replica.
  Authentication fails even though the second replica has the correct password hash. The password storage scheme is disabled in the local per-server configuration.

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, refer to 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:

Core policy attributes

Object class: `ds-pwp-password-policy`

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 received 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. Use the `ds-pwp-max-password-reset-age` property to control how long users have to change their passwords. 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. Use the `ds-pwp-max-password-reset-age` property to control how long users have to change their passwords. 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).

Attribute	Description
`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).
`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. Users are only required to change their password after it is reset if `ds-pwp-force-change-on-add` or `ds-pwp-force-change-on-reset` is true. 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.

Attribute	Description
`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. **Note** When modifying a password, DS checks the new password against each hashed password value in the entry. If the password policy specifies a computationally intensive password storage scheme, such as Argon2, Bcrypt, PKCS5S2, or a PBKDF2-based scheme, enabling password history multiplies the cost of changing the password. DS must calculate the computationally intensive hash from the new password separately for each comparison with a hashed password in the password history. As a result, if it takes 100 ms to calculate the hash for a new password, and the applicable password policy has a password history count of 7, the calculations to modify the password can take up to 700 ms.
`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.

Value validator attributes

Object class: `ds-pwp-attribute-value-validator`

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 and portions of attribute values against the password string.
When false, the server checks whether the entire password matches any user attribute values. When true, the server checks whether the password contains portions of attribute values and whether the attribute values contain portions of the password.
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`	Failure: 19 (Constraint Violation)
`ds-pwp-attribute-value-check-substrings: true`	`bjensen`	Failure: 19 (Constraint Violation)

<table>
<tr><th>Attribute</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Setting</th><th>New Password</th><th>Password Modification Result</th></tr>
<tr><td><code>ds-pwp-attribute-value-check-substrings: true</code></td><td><code>bjensens</code></td><td>Failure: 19 (Constraint Violation)</td></tr>
</table>
In summary:

<code>bjense</code> is allowed when the setting is false because the password does not exactly match and rejected when the setting is true because Babs's user ID contains the password.
<code>bjensen</code> is rejected in both cases because the password exactly matches and contains Babs's user ID.
<code>bjensens</code> 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.
</td></tr>
<tr><td><code>ds-pwp-attribute-value-min-substring-length</code></td><td>The minimal length of the substring within the password when substring checking is enabled.
Default: 0.</td></tr>
</table>

Character set attributes

Object class: `ds-pwp-character-set-validator`

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.

Attribute	Description
`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.
`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.

Dictionary attributes

Object class: `ds-pwp-dictionary-validator`

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.
`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.

Password length attributes

Object class: `ds-pwp-length-based-validator`

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.

Repeated characters attributes

Object class: `ds-pwp-repeated-characters-validator`

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).

Attribute	Description
`ds-pwp-repeated-characters-case-sensitive-validation`	Whether this password validator should treat password characters in a case-sensitive manner. Default: false.

Similarity attributes

Object class: `ds-pwp-similarity-based-validator`

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).

Unique characters attributes

Object class: `ds-pwp-unique-characters-validator`

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).

Generator attributes

Object class: `ds-pwp-random-generator`

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 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:

Internet-Draft attributes

Object class: `pwdPolicy`

Attribute	Description
`pwdAttribute` (required)	The attribute type used to hold user passwords.
`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).

Attribute	Description
`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.

Overrides

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`

Internet-Draft policy attribute	Overrides this server policy property
`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`

Ignored attributes

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.

Inheritance

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`.
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`.

Tasks	Required Access[1] and Privileges[2]
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/errors`. 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 change log levels, 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.
Delegate administration within their organizations	Access to write administration-related attributes on others' entries. DS server privileges: `modify-acl`, `privilege-change`.

Tasks	Required Access[1] and Privileges[2]
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 Linux `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, refer to Enforce limits.

2. Assign any necessary administrative privileges.

   For details, refer to Administrative privileges.

3. Add any necessary ACIs.

   For details, refer to 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/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, refer to 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 PingIDM. For details, refer to 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, refer to 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 display the rights the server grants.

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 gets 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, refer to 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 access, 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 observe 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 observe 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. Refer to 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 generate symmetric keys when needed and store them, 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, refer to 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, refer to 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 PingDS `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, by providing access to monitoring information over multiple protocols, and by recording events in log files that can be shared with 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.

Move a server

Move a server to a new host.

Tuning

Tune server performance.

Troubleshooting

Solve common problems.

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.

`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.

Commands	Constraints
`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.
`create-rc-script` (Linux)	Generate a `systemd` service to start, stop, and restart the server, either directly or at system boot and shutdown. This lets you register and manage DS servers as services on 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.

Command[1]	Description
`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.
`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.

Command[1]	Description
`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] Linux 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 decide whether it does trust the server certificate and continues to negotiate a secure connection, or doesn't 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 `%USERPROFILE%` on Windows. The keystore password is `OpenDJ`. Do not change the file name or the password. • 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.

Truststore Option	Truststore Used
`--use<Type>TrustStore {trustStorePath}`	DS only uses the specified truststore. The <*Type*> in the option name reflects the trust store type. The tool fails with an error if it can't 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=editable:dsAdminDN["uid=admin"]
bindPassword\:file=/path/to/.pwd
useSsl=true
trustAll=true
```

When you use an option with a colon, such as `bindPassword:file`, escape the colon with a backslash (`\:`) in the properties file.

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.
  ```

- (Linux) Use a `systemd` service to manage DS.

```
# Create the service.
# Unless you run DS as root,
# set --userName to the user who installed the server:
$ sudo create-rc-script \
--systemdService /etc/systemd/system/opendj.service \
--userName opendj

# Register the service you created:
$ sudo systemctl daemon-reload
```

Manage the service with `systemctl`.

```
$ sudo systemctl start opendj
```

- (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.
```

- (Linux) Use a `systemd` service to manage DS.

```
# Create the service.
# Unless you run DS as root,
# set --userName to the user who installed the server:
$ sudo create-rc-script \
--systemdService /etc/systemd/system/opendj.service \
--userName opendj

# Register the service you created:
$ sudo systemctl daemon-reload
```

Manage the service with `systemctl`.

```
$ sudo systemctl stop opendj
```

- (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.
```

- (Linux) Use a `systemd` service to manage DS.

```
# Create the service.
# Unless you run DS as root,
# set --userName to the user who installed the server:
$ sudo create-rc-script \
--systemdService /etc/systemd/system/opendj.service \
--userName opendj

# Register the service you created:
$ sudo systemctl daemon-reload
```

Manage the service with `systemctl`.

```
$ sudo systemctl restart opendj
```

- (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. Use the manage-tasks command to manage scheduled tasks. For an example command, refer to Status and tasks.

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*.

How backup works

DS directory servers store data in *backends*. The amount of data in a backend varies depending on your deployment. It can range from very small to very large. A JE backend can hold billions of LDAP entries, for example.

Backup process

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.

The DS backup process takes advantage of this log file structure. Together, a set of log files represents a backend at a point in time. The backup process essentially copies the log files to the backup directory. DS also protects the data and adds metadata to keep track of the log files it needs to restore a JE backend to the state it had when the backup task completed.

Cumulative backups

DS backups are cumulative in nature. Backups reuse the JE files that did not change since the last backup operation. They only copy the JE files the backend created or changed. Files that did not change are shared between backups.

A set of backup files is fully standalone.

Purge old backups

Backup tasks keep JE files until you purge them.

The backup purge operation prevents an endless increase in the size of the backup folder on disk. The purge operation does not happen automatically; you choose to run it. When you run a purge operation, it removes the files for old or selected backups. The purge does not impact the integrity of the backups DS keeps. It only removes log files that do not belong to any remaining backups.

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, refer to 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, refer to dsbackup.

Use the manage-tasks command to manage scheduled tasks. For background, read Server tasks. For an example command, refer to Status and tasks.

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, refer to 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

Use 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.*

  For example, Kubernetes deployments can use volume snapshots when the underlying storage supports atomic snapshots. Learn more in Backup and restore using volume snapshots.

  In contrast, *do not use VMWare snapshots* to back up a running DS server.

- 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, refer to 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.

Backup and restore options

	`dsbackup` commands	Snapshots
What is backed up	DS backend data only	Potentially everything; at minimum DS backends, changelogs
Incremental backups	Yes	Depends on the snapshot tools
Portability	Yes; restore backend data on any DS of the same major/minor version	Depends; potentially limited to the same environment as with Kubernetes volume snapshots
Disaster recovery	Optimal; restore data and delete old changelog	Potentially restores changelog only to clear it during recovery
Recover single server	Potentially slower while rebuilding the local changelog; impacts the change number index (if enabled)	Optimal; restores everything to the previous state
Choice of what to restore	Good; you choose which backends to restore	Bad; you restore the file system, potentially rolling back multiple backends at once
Ease of use	Medium; you must understand `dsbackup` commands and choose what to restore	Medium; you must understand platform tools and impact of restoring everything at once

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, refer to 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, refer to 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, refer to 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

Use 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, refer to 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, refer to 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, refer to the access control documentation for your provider.

   If you are not yet familiar with cloud storage, refer to 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, refer to 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, refer to 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, refer to the Google documentation on Getting Started with Authentication. For details about creating and managing storage buckets, refer to 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`

In production environments, also set the cloud storage 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 endpoint

Backup to cloud storage can use a default endpoint, which can simplify evaluation and testing.

Control where your backup files go. Add one of the following options:

- `--storage-property endpoint:endpoint-url`
- `--storage-property endpoint.env.var:environment-variable-for-endpoint-url`

The *endpoint-url* depends on your provider. Refer to their documentation for details. 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 samples demonstrate how to use the setting.

Cloud storage samples

Click the samples for your storage provider to expand the section and display 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
```

Efficiently store backup files

DS backups are collections of files in a backup directory. To restore from backup, DS requires a coherent collection of backup files.

You can use the dsbackup command to purge stale backup files from a backup directory. When you purge stale backup files, the command leaves a coherent collection of files you can use to restore data.

You should also store copies of backup files remotely to guard against the loss of data in a disaster.

Remote storage

Perform the following steps to store copies of backup files remotely in an efficient way. These steps address backup of directory data, which is potentially very large, not backup of configuration data, which is almost always small:

1. Choose a local directory or local network directory to hold backup files.

   Alternatively, you can back up to cloud storage.

2. Schedule a regular backup task to back up files to the directory you chose.

Make sure that the backup task runs more often than the replication purge delay. For example, schedule the backup task to run every three hours for a default purge delay of three days. Each time the task runs, it backs up only new directory backend files.

For details, refer to the steps for backing up directory data.

3. Store copies of the local backup files at a remote location for safekeeping:

    1. Purge old files in the local backup directory.

        As described in How backup works, DS backups are cumulative in nature; DS reuses common data that has not changed from previous backup operations when backing up data again. The set of backup files is fully standalone.

        The purge removes stale files without impacting the integrity of newer backups, reducing the volume of backup files to store when you copy files remotely.

    2. Regularly copy the backup directory and all the files it holds to a remote location.

        For example, copy all local backup files every day to a remote directory called `bak-date`:

        ```
        $ ssh user@remote-storage mkdir /path/to/bak-date
        $ scp -R /path/to/bak/* user@remote-storage:/path/to/bak-date/
        ```

4. Remove old `bak-date` directories from remote storage in accordance with the backup policy for the deployment.

Restore from remote backup

For each DS directory server to restore:

1. Install DS using the same cryptographic keys and deployment ID.

    Backup files are protected using keys derived from the DS deployment ID and password. You must use the same ones when recovering from a disaster.

2. Restore configuration files.

3. Restore directory data from the latest remote backup folder.

After restoring all directory servers, validate that the restore procedure was a success.

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. Refer to 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`

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.

Linux file systems

Write barriers and journaling mode for Linux file systems help avoid directory database file corruption. They make sure writes to the file system are ordered even after a crash or power failure. Make sure these features are enabled.

Some Linux distributions permanently enable write barriers. There is no administrative action to take.

Other Linux systems leave the decision to you. If your Linux system lets you configure write barriers and journaling mode for the file system, refer to the options for your file system in the `mount` command manual page for details on enabling them.

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 details, refer to 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, refer to 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 production systems, test and run with a tuned JVM.

Tip

To apply JVM settings, either:

- Set the `OPENDJ_JAVA_ARGS` environment variable and restart the server:
  `export OPENDJ_JAVA_ARGS="-Xmx<size> -XX:MaxTenuringThreshold=1 -Djava.security.egd=file:/dev/urandom"`
- Edit `config/java.properties` to update `start-ds.java-args` and restart the server:
  `start-ds.java-args=-server -Xmx<size> -XX:MaxTenuringThreshold=1 -Djava.security.egd=file:/dev/urandom`
  As the name indicates, the `start-ds.java-args` settings apply only to the `start-ds` command. To set JVM options for offline LDIF import, edit `import-ldif.offline.java-args`, for example.

With Java 21 and later you can try experimental virtual thread support, a DS Technology Preview (not for production use). DS uses virtual threads for core processing, not for network I/O, replication, or other features. The use of virtual threads is expected to expand to other features in future releases. To enable virtual thread support, either:

- Add `-Dorg.forgerock.opendj.useVirtualThreads=true` to `OPENDJ_JAVA_ARGS` and restart the server.
- Use the `dsconfig` command to set the global configuration variable `use-virtual-threads-if-available:true`.

1. Use the most recent supported Java environment.

   Refer to the release notes section on Java for details.

2. Set the maximum heap size with `-Xmx<size>`.

   Use at least a 2 GB heap unless your data set is small.

   For additional details, refer to Cache internal nodes.

   For Java 17 and later, there is no need to set the minimum heap size.

3. Choose the appropriate garbage collection (GC) settings:

Java	Recommendations
17	Use the default garbage collection (GC) settings, equivalent to `-XX:+UseG1GC`. Set the maximum tenuring threshold to reduce unnecessary copying with `-XX:MaxTenuringThreshold=1`. This option sets the maximum number of GC cycles an object stays in survivor spaces before it is promoted into the old generation space. The recommended setting 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.
21	For most deployments, follow the recommendations for Java 17. For deployments needing low latency and high scalability, try Generational ZGC with `-XX:+UseZGC -XX:+ZGenerational`. Generational ZGC aims to minimize GC pauses. ZGC can require more overhead, including more memory compared to G1GC for the same maximum heap size settings. Be sure to test and measure how DS behaves under load before rolling the settings out to production servers.

4. (Optional) Review the following additional details for specific use cases:

`-XX:+DisableExplicitGC`

When using JMX and G1GC, 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 GC to manage memory and references to memory-mapped files.

`-Xlog:gc=level:file`

When diagnosing JVM tuning problems, log GC messages. You can turn the option off when everything is running smoothly.

Always specify the output file for the GC 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 GC messages to the file, `/path/to/gc.log`.

For details, use the `java -Xlog:help` command.

`-XX:TieredStopAtLevel=1`

When trying to reduce startup time for short-lived client tools, such as the `ldapsearch` command, use this setting as shown.

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, refer to 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
139242470     47887210   187129680
/dc=com,dc=example/cn.caseIgnoreSubstringsMatch:6                                               858657
5106079    204936276   210042355
/dc=com,dc=example/counter.objectClass.big.objectIdentifierMatch                                2
34           2           36
/dc=com,dc=example/counter.userPassword.big.passwordStorageSchemeEqualityMatch                  0
0            0           0
/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
51690     20017387    20069077
/dc=com,dc=example/givenName.caseIgnoreSubstringsMatch:6                                        19651
97664     48312525    48410189
/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.caseIgnoreMatch                                                         10000152
238891751     47887168   286778919
/dc=com,dc=example/mail.caseIgnoreSubstringsMatch: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                                                                 26
1335           25        1360
/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
/dc=com,dc=example/userPassword.big.passwordStorageSchemeEqualityMatch                   0
0             0          0

Total: 34

# Calculate the sum of total keys, the average key size, and the average value size.
# Sum of total keys: 73255334
# Average key size: sum of key sizes/sum of total keys = 1015212568 / 73255334 ~= 13.86
# Average value size: sum of values sizes/sum of total keys = 6399663760 / 73255334 ~= 87.36

# Use the results rounded to the nearest integer as arguments to the DbCacheSize tool:
$ java -cp /path/to/opendj/lib/opendj.jar com.sleepycat.je.util.DbCacheSize \
 -records 73255334 -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.

Log settings

Debug-level log settings trace the internal workings of DS servers, and should be used sparingly. Be careful when activating debug-level 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

To let legacy applications get update notifications by change number, as described in Align draft change numbers, enable change number indexing.

Change number indexing requires more CPU, disk access, and storage. Enable it only when applications require change number-based browsing.

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. The command output shows a message like the following:

```
See opendj-setup-profile-*.log for a detailed log of the failed 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:

`C:\path\to\opendj\bat\`

Windows command-line tools

`/path/to/opendj/bin/`

Linux command-line tools

`/path/to/opendj/extlib/`

Optional `.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, refer to 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 read an error message in the server log like 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
```

NoSuchAlgorithmException

When running the `dskeymgr create-deployment-id` or `setup` command on an operating system with no support for the `PBKDF2WithHmacSHA256 SecretKeyFactory` algorithm, the command displays this error:

```
NoSuchAlgorithmException: PBKDF2WithHmacSHA256 SecretKeyFactory not available
```

This can occur on operating systems where the default settings limit the available algorithms.

To fix the issue, enable support for the algorithm and run the command again.

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:
   ```

   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-level logging

DS error log message severity levels are:

- `ERROR` (highest severity)
- `WARNING`
- `NOTICE`
- `INFO`
- `DEBUG` (lowest severity)

By default, DS error log severity levels are set as follows:

- Log `ERROR`, `WARNING`, `NOTICE`, and `INFO` replication (`SYNC`) messages.
- Log `ERROR`, `WARNING`, and `NOTICE` messages for other message categories.

You can change these settings when necessary to log debug-level messages.

> **Caution**
>
> DS debug-level logging can generate a high volume of messages. Use debug-level logging very sparingly on production systems.

1. Choose the category you want to debug:

Category	Description
`BACKEND`	Server backends
`BACKUP`	Backup procedures
`CONFIG`	Configuration management
`CORE`	Core server operations
`DEFAULT`	Messages with no specific category

Category	Description
`EXTENSIONS`	Reserved for custom extensions
`EXTERNAL`	External libraries
`JVM`	Java virtual machine information
`LOGGING`	Server log publishers
`PLUGIN`	Server plugins
`PROTOCOL`	Server protocols
`PROTOCOL.ASN1`	ASN.1 encoding
`PROTOCOL.HTTP`	HTTP
`PROTOCOL.JMX`	JMX
`PROTOCOL.LDAP`	LDAP
`PROTOCOL.LDAP_CLIENT`	LDAP SDK client features
`PROTOCOL.LDAP_SERVER`	LDAP SDK server features
`PROTOCOL.LDIF`	LDIF
`PROTOCOL.SASL`	SASL
`PROTOCOL.SMTP`	SMTP
`PROTOCOL.SSL`	SSL and TLS
`SCHEMA`	LDAP schema
`SECURITY`	Security features
`SECURITY.AUTHENTICATION`	Authentication
`SECURITY.AUTHORIZATION`	Access control and privileges
`SERVICE_DISCOVERY`	Service discovery
`SYNC`	Replication
`SYNC.CHANGELOG`	Replication changelog
`SYNC.CHANGENUMBER`	Replication change number and change number index

Category	Description
`SYNC.CONNECTIONS`	Replication connections
`SYNC.HEARTBEAT`	Replication heartbeat checks
`SYNC.LIFECYCLE`	Replication lifecycle
`SYNC.PROTOCOL_MSGS`	Replication protocol messages excluding updates and heartbeat checks
`SYNC.PURGE`	Replication changelog and historical data purge events
`SYNC.REPLAY`	Replication replays and conflicts
`SYNC.STATE`	Replication state changes including generation ID
`SYNC.TOPOLOGY`	Replication topology
`SYNC.UPDATE_MSGS`	Replication update messages
`TASK`	Server tasks
`TOOLS`	Command-line tools

A monitoring user can read the list of supported categories over LDAP.

2. Override the error log level specifically for the category or categories of interest.

   The following example enables debug-level logging for the replication lifecycle. As debug-level logging is of lower severity than the defaults, all the default log levels remain in effect:

```
$ dsconfig \
 set-log-publisher-prop \
 --add override-severity:SYNC.LIFECYCLE=DEBUG \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Error Logger" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

   The server immediately begins to write additional messages to the error log.

3. Read the messages:

```
$ tail -f /path/to/opendj/logs/errors
```

4. Restore the default settings as soon as debug-level logging is no longer required:

```
$ dsconfig \
 set-log-publisher-prop \
 --remove override-severity:SYNC.LIFECYCLE=DEBUG \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --publisher-name "File-Based Error Logger" \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file /path/to/opendj/config/keystore.pin \
 --no-prompt
```

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 in the state Completed successfully
```

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 \
 --bindDN uid=kvaughan,ou=People,dc=example,dc=com \
 --bindPassword bribery \
 --baseDN "" \
 --searchScope base \
 "(objectclass=*)" \
 +

The LDAP bind request failed: 49 (Invalid Credentials)
```

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 in the state Completed successfully
```

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 compared to Java 17 and later. When running the `dskeymgr` and `setup` commands, use the same Java environment everywhere in the deployment.

> **Note**
>
> Running DS servers with incompatible Java versions is a problem when you use deployment ID-based CA certificates. If you use your own CA, not one derived from a deployment ID, skip this section.

When you run the commands with a Java version that doesn't match the deployment ID, DS displays a message such as the following:

```
The specified deployment ID with version '0' will cause interoperability problems with servers
running Java versions less than 17 if the deployment uses deployment ID-based PKI.
Follow the steps in the troubleshooting section of the documentation to resolve compatibility issues
with deployment IDs generated using a Java version prior to 17.
```

Overcome incompatible Java versions to generate new keys

When you upgrade servers in place, moving all servers from Java 11 or earlier to Java 17 or later, you first encounter the error message when using the `dskeymgr` command to renew or replace server TLS certificates.

Overcome the problem by switching to a new deployment ID. The procedure includes replacement of server certificates.

Overcome incompatible Java versions when adding new servers

When you add new servers running on Java 17 or later to a deployment of replicated servers running Java 11 or earlier, the new servers can't replicate with the old servers. Replication breaks as soon as you use the `setup` command for a new server. 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. Install the most recent supported Java 17 or later environment on each DS server system.
2. Update all DS servers to use the most recent supported Java environment.

   If the default Java environment on the system isn't the most recent, 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.
3. 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
      ```
4. 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
   ```
5. 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.

Learn more about related key management procedures in these sections:

- Replace deployment IDs and deployment ID-based CAs
- Replace a TLS key pair

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 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 display 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, refer to 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.

In the access log, each message is a JSON object. This example formats each message to make it easier to read:

```
{
  "eventName": "DJ-LDAP",
  "client": {
    "ip": "<clientIp>",
    "port": 12345
  },
  "server": {
    "ip": "<serverIp>",
    "port": 1636
  },
  "request": {
    "protocol": "LDAPS",
    "operation": "BIND",
    "connId": 3,
    "msgId": 1,
    "version": "3",
    "dn": "uid=kvaughan,ou=people,dc=example,dc=com",
    "authType": "SIMPLE"
  },
  "transactionId": "<uuid>",
  "response": {
    "status": "SUCCESSFUL",
    "statusCode": "0",
    "elapsedTime": 1,
    "elapsedQueueingTime": 0,
    "elapsedProcessingTime": 1,
    "elapsedTimeUnits": "MILLISECONDS",
    "additionalItems": {
      "ssf": 128
    }
  },
  "userId": "uid=kvaughan,ou=People,dc=example,dc=com",
  "timestamp": "<timestamp>",
  "_id": "<uuid>"
}
{
  "eventName": "DJ-LDAP",
  "client": {
    "ip": "<clientIp>",
    "port": 12345
  },
  "server": {
    "ip": "<serverIp>",
    "port": 1636
  },
  "request": {
    "protocol": "LDAPS",
    "operation": "SEARCH",
    "connId": 3,
    "msgId": 2,
    "dn": "dc=example,dc=com",
    "scope": "sub",
    "filter": "(uid=bjensen)",
    "attrs": ["cn"]
  },
  "transactionId": "<uuid>",
  "response": {
    "status": "SUCCESSFUL",
    "statusCode": "0",
    "elapsedTime": 3,
```

```
    "elapsedQueueingTime": 0,
    "elapsedProcessingTime": 3,
    "elapsedTimeUnits": "MILLISECONDS",
    "nentries": 1,
    "entrySize": 591
  },
  "userId": "uid=kvaughan,ou=People,dc=example,dc=com",
  "timestamp": "<timestamp>",
  "_id": "<uuid>"
}
{
  "eventName": "DJ-LDAP",
  "client": {
    "ip": "<clientIp>",
    "port": 12345
  },
  "server": {
    "ip": "<serverIp>",
    "port": 1636
  },
  "request": {
    "protocol": "LDAPS",
    "operation": "UNBIND",
    "connId": 3,
    "msgId": 3
  },
  "transactionId": "<uuid>",
  "timestamp": "<timestamp>",
  "_id": "<uuid>"
}
{
  "eventName": "DJ-LDAP",
  "client": {
    "ip": "<clientIp>",
    "port": 12345
  },
  "server": {
    "ip": "<serverIp>",
    "port": 1636
  },
  "request": {
    "protocol": "LDAPS",
    "operation": "DISCONNECT",
    "connId": 3
  },
  "transactionId": "0",
  "response": {
    "status": "SUCCESSFUL",
    "statusCode": "0",
    "elapsedTime": 0,
    "elapsedTimeUnits": "MILLISECONDS",
    "reason": "Client Unbind"
  },
  "timestamp": "<timestamp>",
  "_id": "<uuid>"
}
```


For details about the messages format, refer to Access log format.

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, refer to Effective rights.

Simple paged results

For some versions of Linux, you read 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 read messages like the following in the `logs/errors` 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 refer to 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.

Persistently long replication delays can be a problem for client applications. A client application gets an unexpectedly old view of the data when reading from a very delayed replica. Monitor replication delay and take action when you observe persistently long delays. For example, make sure the network connections between DS servers are functioning normally. Make sure the DS server systems are sized appropriately.

For detailed suggestions about monitoring replication delays, refer to either of the following sections:

- Replication delay (LDAP)
- Replication delay (Prometheus)

Use the logs

By default, replication records messages in the log file, `logs/errors`. Replication messages 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 replicas maintain historical information to bring replicas up to date and to resolve conflicts. To prevent historical information from growing without limit, DS replicas purge historical information after the replication-purge-delay (default: 3 days).

A replica becomes irrevocably out of sync when, for example:

- You restore it from backup files older than the purge delay.
- You stop it for longer than the purge delay.
- The replica stays out of contact with other DS servers for longer than the purge delay.

When this happens to a single replica, reinitialize the replica.

For detailed suggestions about troubleshooting stale replicas, refer to either of the following sections:

- Replication status (LDAP)
- Replication status (Prometheus)

Change number indexing

DS replication servers maintain a changelog database to record updates to directory data. The changelog database serves to:

- Replicate changes, synchronizing data between replicas.
- Let client applications get change notifications.

DS replication servers purge historical changelog data after the `replication-purge-delay` in the same way replicas purge their historical data.

Client applications can get changelog notifications using cookies (recommended) or change numbers.

To support change numbers, the servers maintain a change number index to the replicated changes. A replication server maintains the index when its configuration properties include `changelog-enabled:enabled`. (Cookie-based notifications do not require a change number index.)

The change number indexer must not be interrupted for long. Interruptions can arise when, for example, a DS server:

- Stays out of contact, not sending any updates or heartbeats.
- Gets removed without being shut down cleanly.
- Gets lost in a system crash.

Interruptions prevent the change number indexer from advancing. When a change number indexer cannot advance for almost as long as the purge delay, it may be unable to recover as the servers purge historical data needed to determine globally consistent change numbers.

For detailed suggestions about monitoring changelog indexing, refer to either of the following sections:

- Change number indexing (LDAP)
- Change number indexing (Prometheus)

Take action based on the situation:

Situation	Actions to take
The time since last indexing is much smaller than the purge delay.	No action required.
The time since last indexing is approaching the purge delay.	Begin by determining why. The fix depends on the exact symptoms.
A DS server was removed without a clean shutdown.	Rebuild an identical server and shut it down cleanly before removing it.
A DS server disappeared in a crash.	Rebuild an identical server.

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, refer to Replication and the related pages.

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 observe the problem:

```
$ status --offline --version

ForgeRock Directory Services 7.5.2-20250513124640-de1088550ebabfaf1b3577b53f3eb9fc3b03739c
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, refer to 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.

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.
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, refer to Enable an audit log.

Type	Description
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 server messages by category, refer to Log message reference. DS error log message severity levels are: • `ERROR` (highest severity) • `WARNING` • `NOTICE` • `INFO` • `DEBUG` (lowest severity) To log debug-level messages for a category of interest, refer to Debug-level logging. Use the external changelog to get notifications about changes to directory data. For details, refer to 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 a common audit event framework. The framework deals with any event you can audit, not only the data updates recorded in a directory audit log. The framework provides log handlers for publishing to local files or to remote systems.

Access log format

DS servers support a 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 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 Ping Identity Platform. You can plug in custom handlers that comply with the framework without having to upgrade the server.
- The framework includes handlers for logging to local files and to external services.

  Although the 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.

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": [{                         // Request controls
      "control": string,                   // Control name or OID
      "critical": boolean,                 // true, false
      "value": string                      // Base64-encoded
    }],
    "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
    "modifications": [{                    // Attributes targeted for modification[1]
      "modification": string,              // Modification type
      "attribute": string,                 // Targeted attribute
      "values": [ string ]                 // Modification values
    }]
    "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", "TLS"
    "opType": "sync",                      // Replication operation
    "protocol": string,                    // "LDAP", "LDAPS"
    "runAs": string,                       // Authorization ID
    "scope": string,                       // Search scope such as "sub"
    "version": string                      // Version "2", "3"
  },
  "response": {
    "additionalItems": object              // Additional information
    "controls": [{                         // Response controls
      "control": string,                   // Control name or OID
      "critical": boolean                  // true, false
    }],
    "elapsedTime": number,                 // Total time queuing and processing the request
    "elapsedQueueingTime": number,         // Time the request spent waiting in the queue
    "elapsedProcessingTime": number,       // Time actively processing the request
    "elapsedTimeUnits": string,            // Time unit such as "MILLISECONDS"
    "entrySize": number,                   // Size in bytes of the largest entry
                                           // read from disk while processing
                                           // the operation
    "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
    },
    "security": {                           // Connection security, such as TLS handshake data
      "protocol": string,                   // Protocol, such as "TLSv1.3"
      "cipher": string,                     // Cipher suite
      "ssf": number                         // Security strength factor
    },
    "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
  }
```

[1] When the advanced configuration property for the log publisher `log-modified-attribute-values:true`, DS logs an array of attributes targeted by modification requests. Each item indicates:

- The modification type: `add`, `delete`, `increment`, or `replace`.
- The attribute name.
- The values for the modification, if allowed according to the log publisher configuration.

  Set one of the advanced attributes `exclude-values-of-attributes` or `include-values-of-attributes` to specify which attribute values DS logs.

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,                 // Total time queuing and processing the request
    "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 read. 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, queuing time, and processing time
- Result codes (only log error results, for example)
- Search response criteria (number of entries returned, unindexed search, and others)
- Target DN
- TLS handshakes
- User DN and group membership

A log publisher's filtering policy determines whether to include or exclude log messages that match the criteria.

Refer to the examples in 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 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 interface stability of this feature is *Deprecated*.
> 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 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, refer to 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`	The 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.
`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 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 hide controls 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-controls:false \
    --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, refer to LDAP result codes.

For details about how filtering works, refer to Access log filtering.

Log modifications

You can configure a DS log publisher for JSON-based LDAP access logs to record the attributes targeted by modification requests.

The following example enables the default `Json File-Based Access Logger` to log modifications in its log file, `logs/ldap-access.audit.json`:

```
$ dsconfig \
 set-log-publisher-prop \
 --publisher-name "Json File-Based Access Logger" \
 --set log-modified-attribute-values: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 example uses the default settings for the related advanced properties:

- `exclude-values-of-attributes` includes `authPassword` and `userPassword` to avoid logging hashed passwords.
- `include-values-of-attributes` isn't set.

When a client application requests a modification, DS includes the modifications in the audit event message `request` object. The following example shows an extract of the audit event message for a request changing Babs's `description` to `New description`:

```
{
  "request": {
    "protocol": "LDAPS",
    "operation": "MODIFY",
    "dn": "uid=bjensen,ou=people,dc=example,dc=com",
    "modifications": [{
      "modification": "replace",
      "attribute": "description",
      "values": ["New description"]
    }]
  }
}
```

CSV format

A CSV handler sends messages to a comma-separated variable (CSV) file.

> **Important**
>
> The interface stability of this feature is *Deprecated*.
> 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 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, refer to 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

> **Important**
>
> The interface stability of this feature is *Deprecated*.

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

> **Important**
>
> The interface stability of this feature is *Deprecated*.

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. Refer to the following example files:

- `opendj/config/audit-handlers/mysql_tables-example.sql`
- `opendj/config/audit-handlers/oracle_tables-example.sql`
- `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, refer to 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, refer to `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, refer to 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. Refer to the provider documentation for details.
            }
        }
    }
}
```

For a sample configuration, refer to `opendj/config/audit-handlers/jms-config.json-example`.

Syslog

A Syslog handler sends messages to the Linux 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, refer to 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, refer to `opendj/config/audit-handlers/syslog-config.json-example`.

For additional details, refer to 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 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, refer to 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, refer to `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:notice \
 --set override-severity:SYNC=INFO \
 --offline \
 --configFile /path/to/opendj/config/config.ldif \
 --no-prompt

$ start-ds --noDetach
```

Log error messages as JSON

By default, error log messages start with a timestamp, followed by space-separated `field=value` pairs, where the last `msg` field contains free-form text. The format is human-readable, but can be less convenient to parse than standard JSON.

You can set the error log publisher property `json-output:true` to cause the publisher to generate JSON messages:

1. Stop DS.
2. Move existing, free-form text error log files.
3. Change to JSON output.

   The following command changes the default error log publisher configuration while DS is stopped:

```
$ dsconfig \
 set-log-publisher-prop \
 --publisher-name "File-Based Error Logger" \
 --set json-output:true \
 --offline \
 --configFile /path/to/opendj/config/config.ldif \
 --no-prompt
```

4. Start DS.

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 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 noticing 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: editable:dsAdminDN["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.

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 notice 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

> **Tip**
>
> This page covers the HTTP interfaces for monitoring DS servers. For the same capabilities over LDAP, refer to LDAP-based monitoring.

DS servers publish monitoring information at these HTTP endpoints:

`/alive`

Whether the server is currently *alive*, meaning its internal checks have not found any errors that would require administrative action.

`/healthy`

Whether the server is currently *healthy*, meaning it is alive, the replication server is accepting connections on the configured port, and any replication delays are below the configured threshold.

`/metrics/prometheus`

Monitoring information in Prometheus monitoring software format. For details, refer to 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, refer to Monitor privilege.

Basic availability

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
```

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`.

Activity

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]"
```

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, refer to the settings for Traditional Work Queue.

Counts

ACIs (Prometheus)

DS maintains counts of ACIs:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
_aci
```

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
```

Entry caches (Prometheus)

DS servers maintain entry cache statistics:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
entry_cache
```

Groups (Prometheus)

The following example reads counts of static, dynamic, and virtual static groups, and statistics on the distribution of static group size:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep -i
group
```

At startup time, DS servers log a message showing the number of different types of groups and the memory allocated to cache static groups.

Subentries (Prometheus)

DS maintains counts of LDAP subentries:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
subentries
```

Indexing

Index cost (Prometheus)

DS maintains metrics about index cost. The metrics count the number of updates and how long they took 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_cost
```

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
```

Replication

Monitor the following to ensure replication runs smoothly. Take action as described in these sections and in the troubleshooting documentation for replication problems.

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, refer to Replication status.

Replication status (Prometheus)

The following example checks the replication status metrics:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
replica_status
```

The effective replica status is the gauge whose value is `1.0`. For example, this output shows normal status:

```
ds_replication_replica_status{domain_name="dc=example,dc=com",server_id="evaluation-only",status="BAD_DATA",} 0.0
ds_replication_replica_status{domain_name="dc=example,dc=com",server_id="evaluation-only",status="DEGRADED",} 0.0
ds_replication_replica_status{domain_name="dc=example,dc=com",server_id="evaluation-only",status="FULL_UPDATE",} 0.0
ds_replication_replica_status{domain_name="dc=example,dc=com",server_id="evaluation-only",status="INVALID",} 0.0
ds_replication_replica_status{domain_name="dc=example,dc=com",server_id="evaluation-only",status="NORMAL",} 1.0
ds_replication_replica_status{domain_name="dc=example,dc=com",server_id="evaluation-only",status="NOT_CONNECTED",}
0.0
ds_replication_replica_status{domain_name="dc=example,dc=com",server_id="evaluation-only",status="TOO_LATE",} 0.0
```

Note

The `DEGRADED` status is for backwards compatibility only.

If the status is not `Normal`, how you react depends on the value of the `ds-mon-status` attribute for LDAP, or `ds_replication_replica_status{status}` for Prometheus:

Status	Explanation	Actions to take
`Bad data`	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, combined with the total number of entries. When the replicas' generation IDs match, the servers can replicate data without user intervention. When the replicas' generation IDs don't match for a given backend, the servers can't replicate the data. 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 began with different data than its peer replicas. • The fractional replication configuration for this replica doesn't match the backend data. For example, you reconfigured fractional replication to include or exclude different attributes, or you configured fractional replication incompatibly on different peer replicas. You must intervene to make sure the replicas with bad data start from the same initial state as their peers. Follow the suggested actions to take. Don't replace or reinitialize the backend data alone. DS stores the generation ID in the backend and in the changelog. The generation IDs in the backend and in the changelog must match on all peer replicas. DS 7.3 introduced this status. Earlier releases included this state as part of the `Bad generation id` status.	Whenever this status displays: 1. If fractional replication is configured, make sure the configuration is compatible on all peer replicas. Learn more in Fractional replication (advanced). 2. Initialize the replica with `Bad data` online from a replica with good data. Use the `dsrepl initialize` command to initialize the single bad replica. This fixes the bad generation IDs, correcting the problem in the backend and changelog data. Find an example in Initialize over the network. If you can't initialize the replica with `Bad data` online, remove it and replace it with a new replica.
`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.

Status	Explanation	Actions to take
`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.
`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.
`Too late`	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 replica no longer receives updates from replication servers. Other replicas that recognize this status stop returning referrals to this replica. DS 7.3 introduced this status. Earlier releases included this state as part of the `Bad generation id` status.	Whenever this status displays: 1. Reinitialize replication for the replica that is too late. For details, refer to Manual initialization.

Change number indexing (Prometheus)

DS replication servers maintain a changelog database to record updates to directory data. The changelog database serves to:

- Replicate changes, synchronizing data between replicas.
- Let client applications get change notifications.

DS replication servers purge historical changelog data after the `replication-purge-delay` in the same way replicas purge their historical data.

Client applications can get changelog notifications using cookies (recommended) or change numbers.

To support change numbers, the servers maintain a change number index to the replicated changes. A replication server maintains the index when its configuration properties include `changelog-enabled:enabled`. (Cookie-based notifications do not require a change number index.)

The change number indexer must not be interrupted for long. Interruptions can arise when, for example, a DS server:

- Stays out of contact, not sending any updates or heartbeats.
- Gets removed without being shut down cleanly.
- Gets lost in a system crash.

Interruptions prevent the change number indexer from advancing. When a change number indexer cannot advance for almost as long as the purge delay, it may be unable to recover as the servers purge historical data needed to determine globally consistent change numbers.

The following example checks the state of change number indexing:

```
$ curl --cacert ca-cert.pem --user monitor:password https://localhost:8443/metrics/prometheus 2>/dev/null | grep
change_number_
# HELP ds_change_number_indexing_state Automatically generated
# TYPE ds_change_number_indexing_state gauge
ds_change_number_indexing_state{indexing_state="BLOCKED_BY_REPLICA_NOT_IN_TOPOLOGY",} 0.0
ds_change_number_indexing_state{indexing_state="INDEXING",} 1.0
ds_change_number_indexing_state{indexing_state="WAITING_ON_UPDATE_FROM_REPLICA",} 0.0
# HELP ds_change_number_time_since_last_indexing_seconds Duration since the last time a change was indexed
# TYPE ds_change_number_time_since_last_indexing_seconds gauge
ds_change_number_time_since_last_indexing_seconds 0.0
# HELP ds_replication_changelog_purge_waiting_for_change_number_indexing Indicates whether changelog purging is
waiting for change number indexing to advance. If true, check the ds-mon-indexing-state and ds-mon-replicas-
preventing-indexing metrics
# TYPE ds_replication_changelog_purge_waiting_for_change_number_indexing gauge
ds_replication_changelog_purge_waiting_for_change_number_indexing 0.0
```

When `ds_change_number_indexing_state` has `BLOCKED_BY_REPLICA_NOT_IN_TOPOLOGY` or `WAITING_ON_UPDATE_FROM_REPLICA` greater than 0, refer to `ds_change_number_time_since_last_indexing_seconds` for the wait time in seconds and to the LDAP `ds-mon-replicas-preventing-indexing` metric for the list of problem servers.

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

> **Tip**
>
> This page covers the LDAP interfaces for monitoring DS servers. For the same capabilities over HTTP, refer to HTTP-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 types of metrics are exposed. For details, refer to LDAP metrics reference.

Basic availability

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.

Activity

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, refer to Metric types reference.

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, refer to 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, refer to Metric types reference. To adjust the number of worker threads, refer to the settings for Traditional Work Queue.

Counts

ACIs (LDAP)

DS maintains counts of ACIs:

```
$ 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-aci)" \
 ds-mon-entries-acis-count ds-mon-entries-with-aci-attributes-count ds-mon-global-acis-count
```

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
```

Entry caches (LDAP)

DS servers maintain entry cache statistics:

```
$ 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-entry-cache)" \
```

Entry caches for groups have their own monitoring entries.

Groups (LDAP)

The following example reads counts of static, dynamic, and virtual static groups, and statistics on the distribution of static group size:

```
$ 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-groups)" \
 ds-mon-dynamic-groups-count ds-mon-static-groups-count ds-mon-virtual-static-groups-count \
 ds-mon-static-group-size-less-or-equal-to-100 \
 ds-mon-static-group-size-less-or-equal-to-1000 \
 ds-mon-static-group-size-less-or-equal-to-10000 \
 ds-mon-static-group-size-less-or-equal-to-100000 \
 ds-mon-static-group-size-less-or-equal-to-1000000 \
 ds-mon-static-group-size-less-or-equal-to-inf
```

At startup time, DS servers log a message showing the number of different types of groups and the memory allocated to cache static groups.

Subentries (LDAP)

DS maintains counts of LDAP subentries:

```
$ 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-subentries)" \
 ds-mon-collective-attribute-subentries-count \
 ds-mon-password-policy-subentries-count
```

Indexing

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
```

Index cost (LDAP)

DS maintains metrics about index cost. The metrics count the number of updates and how long they took 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-cost
```

Logging

DS maintains a list of supported logging categories. The following example reads the list:

```
$ 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-logging)"
```

Replication

Monitor the following to ensure replication runs smoothly. Take action as described in these sections and in the troubleshooting documentation for replication problems.

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:

- 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, refer to 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 for LDAP, or `ds_replication_replica_status{status}` for Prometheus:

Status	Explanation	Actions to take
`Bad data`	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, combined with the total number of entries. When the replicas' generation IDs match, the servers can replicate data without user intervention. When the replicas' generation IDs don't match for a given backend, the servers can't replicate the data. 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 began with different data than its peer replicas. • The fractional replication configuration for this replica doesn't match the backend data. For example, you reconfigured fractional replication to include or exclude different attributes, or you configured fractional replication incompatibly on different peer replicas. You must intervene to make sure the replicas with bad data start from the same initial state as their peers. Follow the suggested actions to take. Don't replace or reinitialize the backend data alone. DS stores the generation ID in the backend and in the changelog. The generation IDs in the backend and in the changelog must match on all peer replicas. DS 7.3 introduced this status. Earlier releases included this state as part of the `Bad generation id` status.	Whenever this status displays: 1. If fractional replication is configured, make sure the configuration is compatible on all peer replicas. Learn more in Fractional replication (advanced). 2. Initialize the replica with `Bad data` online from a replica with good data. Use the `dsrepl initialize` command to initialize the single bad replica. This fixes the bad generation IDs, correcting the problem in the backend and changelog data. Find an example in Initialize over the network. If you can't initialize the replica with `Bad data` online, remove it and replace it with a new replica.
`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.

Status	Explanation	Actions to take
`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.
`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.
`Too late`	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 replica no longer receives updates from replication servers. Other replicas that recognize this status stop returning referrals to this replica. DS 7.3 introduced this status. Earlier releases included this state as part of the `Bad generation id` status.	Whenever this status displays: 1. Reinitialize replication for the replica that is too late. For details, refer to Manual initialization.

Change number indexing (LDAP)

DS replication servers maintain a changelog database to record updates to directory data. The changelog database serves to:

- Replicate changes, synchronizing data between replicas.
- Let client applications get change notifications.

DS replication servers purge historical changelog data after the `replication-purge-delay` in the same way replicas purge their historical data.

Client applications can get changelog notifications using cookies (recommended) or change numbers.

To support change numbers, the servers maintain a change number index to the replicated changes. A replication server maintains the index when its configuration properties include `changelog-enabled:enabled`. (Cookie-based notifications do not require a change number index.)

The change number indexer must not be interrupted for long. Interruptions can arise when, for example, a DS server:

- Stays out of contact, not sending any updates or heartbeats.
- Gets removed without being shut down cleanly.
- Gets lost in a system crash.

Interruptions prevent the change number indexer from advancing. When a change number indexer cannot advance for almost as long as the purge delay, it may be unable to recover as the servers purge historical data needed to determine globally consistent change numbers.

The following example uses the default monitor user account to check the state of change number indexing:

```
$ 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=changelog,cn=replication,cn=monitor" \
 "(objectClass=ds-monitor-change-number-indexing)" \
 ds-mon-indexing-state ds-mon-time-since-last-indexing ds-mon-replicas-preventing-indexing

dn: cn=change number indexing,cn=changelog,cn=replication,cn=monitor
ds-mon-indexing-state: INDEXING
ds-mon-time-since-last-indexing: 0
```

When `ds-mon-indexing-state: BLOCKED_BY_REPLICA_NOT_IN_TOPOLOGY` or `ds-mon-indexing-state: WAITING_ON_UPDATE_FROM_REPLICA`, refer to `ds-mon-time-since-last-indexing` for the wait time in milliseconds and to `ds-mon-replicas-preventing-indexing` for the list of problem servers.

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-indexed: <number>
ds-mon-backend-filter-unindexed: <number>
ds-mon-backend-filter-use-start-time: <timestamp>
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: <number>
ds-mon-db-log-size-total: <number>
ds-mon-db-log-utilization-max: <number>
ds-mon-db-log-utilization-min: <number>
ds-mon-db-version: <version>
```

JMX-based monitoring

> **Note**
>
> The interface stability of the JMX connection handler is *Deprecated*.
> JMX MBeans remain supported.

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`.

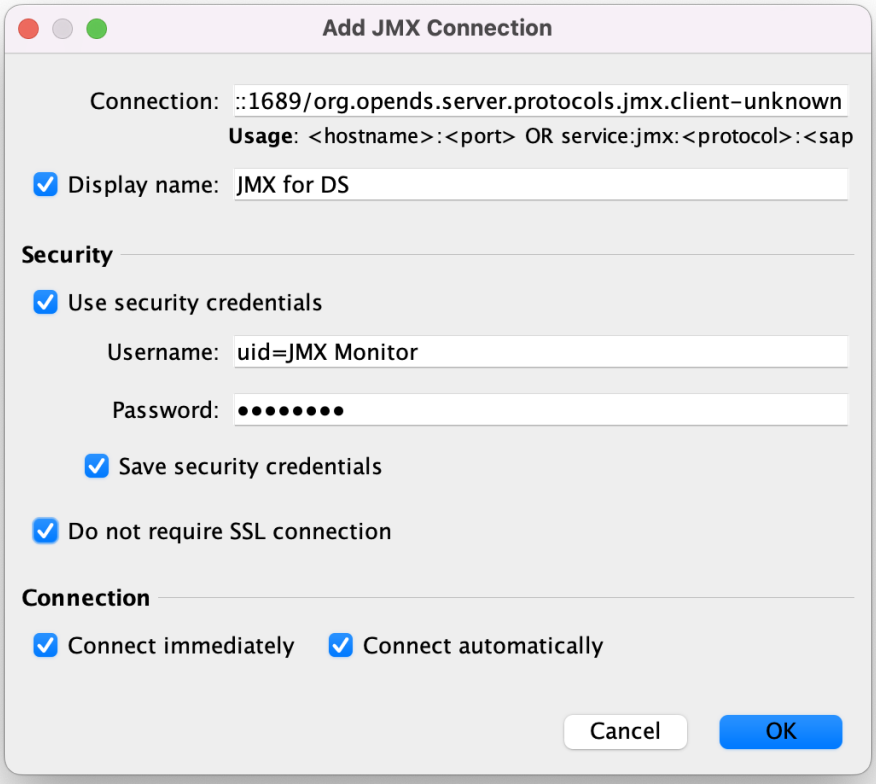
Password

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, refer to 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 scheduled server tasks, 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. Refer to 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.
	LDAP metrics 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. LDAP metrics 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.
	LDAP metrics 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, refer to 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, refer to the About This Reference.

For examples of common monitoring requests, refer to 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-connector-connections`	Integer	Number of connections currently established on the Administration Connector
`ds-mon-admin-hostport`	Host port	The administrative host and port
`ds-mon-alias`	Directory String	Certificate alias
`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`	Directory String	Backend degraded index
`ds-mon-backend-degraded-index-count`	Integer	Number of degraded indexes in the backend
`ds-mon-backend-entry-count`	Integer	Number of entries contained in the backend
`ds-mon-backend-entry-size-read`	Summary metric	Histogram of entry sizes being read from the underlying storage
`ds-mon-backend-entry-size-written`	Summary metric	Histogram of entry sizes being written to the underlying storage

Name	Syntax	Description
`ds-mon-backend-filter-indexed`	Integer	Number of indexed searches performed against the backend
`ds-mon-backend-filter-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-filter-use-start-time`	Generalized Time	Time the server started recording statistical information about the simple search filters 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 time 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`	DN	Base DN handled by a backend
`ds-mon-base-dn-entry-count`	Integer	Number of subordinate entries of the base DN, including the base DN
`ds-mon-build-number`	Integer	Build number of the Directory Server
`ds-mon-build-time`	Directory String	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

Name	Syntax	Description
`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
`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-size-bytes`	Integer	Total memory in bytes used by 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	Time the certificate expires
`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-file-count`	Integer	The number of changelog files containing updates generated by this replica. A value of zero indicates the replica did not generate any updates during the last purge delay interval
`ds-mon-changelog-hostport`	Host port	The host and port of the changelog server
`ds-mon-changelog-id`	Directory String	Changelog identifier
`ds-mon-changelog-purge-delay`	Duration in milli-seconds	The purge delay of the changelog
`ds-mon-collective-attribute-subentries-count`	Integer	Total number of collective attribute subentries
`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`	Directory String	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

Name	Syntax	Description
`ds-mon-current-connections`	Integer	Number of client connections currently established except on the Administration Connector
`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
`ds-mon-current-time`	Generalized Time	Current 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

Name	Syntax	Description
`ds-mon-db-log-utilization-max`	Integer	Current maximum (upper bound) log utilization as a percentage
`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-dynamic-groups-count`	Integer	Total number of dynamic groups
`ds-mon-entries-acis-count`	Integer	Total number of entries ACIs
`ds-mon-entries-awaiting-updates-count`	Integer	Number of entries for which an update operation has been received but not replayed yet by this replica
`ds-mon-entries-with-aci-attributes-count`	Integer	Total number of entries with ACI attributes
`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-global-acis-count`	Integer	Total number of global ACIs
`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

Name	Syntax	Description
`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-cost`	Timer metric	Number of index updates and their time cost
`ds-mon-indexing-state`	Directory String	Change number indexing state, can be one of "INDEXING", "BLOCKED_BY_REPLICA_NOT_IN_TOPOLOGY" or "WAITING_ON_UPDATE_FROM_REPLICA"
`ds-mon-index-uses`	Summary metric	Number of accesses of this index. For attribute indexes it represents the number of search operations that have used this index, for system indexes it represents the number of key lookups.
`ds-mon-install-path`	Filesystem path	Directory Server root installation path
`ds-mon-instance-path`	Filesystem path	Directory Server instance path
`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-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-class-path`	Filesystem path	Path used to find directories and JAR archives containing Java class files
`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

Name	Syntax	Description
`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
`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

Name	Syntax	Description
`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-received-update`	Directory String	The CSN of the last received update originating from the remote replica
`ds-mon-last-replayed-update`	Directory String	The CSN of the last replayed update originating from the remote replica
`ds-mon-last-seen`	Generalized Time	Time 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-ldaps-hostport`	Host port	The host and port to connect using LDAPS
`ds-mon-ldap-starttls-hostport`	Host port	The host and port to connect using LDAP (with support for start TLS)
`ds-mon-listen-address`	Directory String	Host and port
`ds-mon-lost-connections`	Integer	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-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`	CSN (Change Sequence Number)	Newest CSN present in the replica database
`ds-mon-newest-csn-timestamp`	Generalized Time	Time of the newest CSN present in the replica database

Name	Syntax	Description
`ds-mon-oldest-change-number`	Integer	Oldest change number present in the change number index database
`ds-mon-oldest-csn`	CSN (Change Sequence Number)	Oldest CSN present in the replica database
`ds-mon-oldest-csn-timestamp`	Generalized Time	Time of the 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
`ds-mon-password-policy-subentries-count`	Integer	Total number of password policy subentries
`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-purge-waiting-for-change-number-indexing`	Boolean	Indicates whether changelog purging is waiting for change number indexing to advance. If true, check the ds-mon-indexing-state and ds-mon-replicas-preventing-indexing metrics
`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-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	Replay etime for updates that have been replayed on this replica
`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

Name	Syntax	Description
`ds-mon-replicas-preventing-indexing`	Directory String	Lists the replicas preventing external changelog change numbers from incrementing
`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
`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))

Name	Syntax	Description
`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`	Timer metric	Modify request timer
`ds-mon-requests-modify-dn`	Timer metric	Modify DN request timer
`ds-mon-requests-patch`	Timer metric	PATCH request timer
`ds-mon-requests-post`	Timer metric	POST request timer
`ds-mon-requests-psearch`	Timer metric	Persistent search request timer
`ds-mon-requests-put`	Timer metric	PUT request timer
`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-id`	Directory String	Server identifier
`ds-mon-server-is-local`	Boolean	Indicates whether this is the topology server that has handled the monitoring request

Name	Syntax	Description
`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	Time the Directory Server started
`ds-mon-static-groups-count`	Integer	Total number of static groups
`ds-mon-static-group-size-less-or-equal-to-100`	Integer	Number of static groups with at most 100 members
`ds-mon-static-group-size-less-or-equal-to-1000`	Integer	Number of static groups with at most 1000 members
`ds-mon-static-group-size-less-or-equal-to-10000`	Integer	Number of static groups with at most 10000 members
`ds-mon-static-group-size-less-or-equal-to-100000`	Integer	Number of static groups with at most 100000 members
`ds-mon-static-group-size-less-or-equal-to-1000000`	Integer	Number of static groups with at most 1000000 members
`ds-mon-static-group-size-less-or-equal-to-inf`	Integer	Total number of static groups
`ds-mon-status`	Directory String	Replication status of the local replica, can either be "Invalid", "Not connected", "Normal", "Too late", "Full update", "Bad data"
`ds-mon-status-last-changed`	Generalized Time	Last time the replication status of the local replica changed
`ds-mon-supported-log-category`	Directory String	Supported server log categories
`ds-mon-system-name`	Directory String	Fully qualified domain name of the system where the Directory Server is running
`ds-mon-time-since-last-indexing`	Duration in milli-seconds	Duration since the last time a change was indexed
`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-total-update`	Directory String	The type of total update when it is in progress. Possible values: import or export
`ds-mon-total-update-entry-count`	Integer	The total number of entries to be processed when a total update is in progress
`ds-mon-total-update-entry-left`	Integer	The number of entries still to be processed when a total update is in progress
`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-in-progress`	Integer	Number of remote updates received in the queue waiting to be replayed
`ds-mon-updates-in-queue`	Integer	Number of remote updates received in the queue
`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-virtual-static-groups-count`	Integer	Total number of virtual static groups
`ds-mon-working-directory`	Filesystem path	Current working directory of the user running the Directory Server

Prometheus metrics reference

This page lists Prometheus metrics:

- This page shows labels in braces; for example, the labels in `ds_backend_db_cache_misses_internal_nodes{backend,type}` are `backend` and `type`.
- Time gauges whose names end in `_seconds` indicate seconds since 1 Jan 1970 UTC; for example `ds_current_time_seconds 1679472039` means Wed, 22 Mar 2023 08:00:39.

For examples of common monitoring requests, refer to 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_acis_count{type}`	Gauge	Total number of ACIs of the specified type
`ds_admin_connector_connections`	Gauge	Number of connections currently established on the Administration Connector
`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_sum`	Summary	Number of attempts to retrieve an entry that was not held in this cache
`ds_all_entry_caches_cache_size_bytes`	Gauge	Total memory in bytes used by this cache
`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_sum`	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

Name	Type	Description
`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_opened{backend,type}`	Gauge	Number of times a log file has been opened
`ds_backend_db_log_files_open{backend,type}`	Gauge	Number of files currently open in the database file cache
`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)
`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,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_read_count{backend,type}`	Histogram	Histogram of entry sizes being read from the underlying storage
`ds_backend_entry_size_read_sum{backend,type}`	Histogram	Histogram of entry sizes being read from the underlying storage

Name	Type	Description
`ds_backend_entry_size_written_bucket{backend,type,le}`	Histogram	Histogram of entry sizes being written to the underlying storage
`ds_backend_entry_size_written_count{backend,type}`	Histogram	Histogram of entry sizes being written to the underlying storage
`ds_backend_entry_size_written_sum{backend,type}`	Histogram	Histogram of entry sizes being written to the underlying storage
`ds_backend_filter_indexed{backend,type}`	Gauge	Number of indexed searches performed against the backend
`ds_backend_filter_unindexed{backend,type}`	Gauge	Number of unindexed searches performed against the backend
`ds_backend_filter_use_start_time_seconds{backend,type}`	Gauge	Time the server started recording statistical information about the simple search filters processed against the backend
`ds_backend_index_cost_seconds_count{backend,base_dn,index,type}`	Summary	Number of index updates and their time cost
`ds_backend_index_cost_seconds_sum{backend,base_dn,index,type}`	Summary	Number of index updates and their time cost
`ds_backend_index_cost_seconds{backend,base_dn,index,type,quantile}`	Summary	Number of index updates and their time cost
`ds_backend_index_uses_count{backend,base_dn,index,type}`	Summary	Number of accesses of this index. For attribute indexes it represents the number of search operations that have used this index, for system indexes it represents the number of key lookups.
`ds_backend_index_uses_sum{backend,base_dn,index,type}`	Summary	Number of accesses of this index. For attribute indexes it represents the number of search operations that have used this index, for system indexes it represents the number of key lookups.
`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_sum{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 time 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	Time the certificate expires
`ds_change_number_indexing_state{indexing_state}`	Gauge	Refer to the indexing_state dimension for details
`ds_change_number_time_since_last_indexing_seconds`	Gauge	Duration since the last time a change was indexed
`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_sum{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_sum{http_handler}`	Summary	Network bytes written summary
`ds_connection_handlers_http_requests_failure_seconds_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_seconds_sum{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)

Name	Type	Description
`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_seconds_count{http_handler,type}`	Summary	Timer for the specified request type
`ds_connection_handlers_http_requests_seconds_sum{http_handler,type}`	Summary	Timer for the specified request type
`ds_connection_handlers_http_requests_seconds{http_handler,type,quantile}`	Summary	Timer for the specified request type
`ds_connection_handlers_ldap_abandoned_requests_total{ldap_handler}`	Counter	Total number of abandoned operations since startup
`ds_connection_handlers_ldap_abandoned_requests{ldap_handler}` (deprecated)	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_sum{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_sum{ldap_handler}`	Summary	Network bytes written summary
`ds_connection_handlers_ldap_connections_count{ldap_handler}`	Summary	Connection summary
`ds_connection_handlers_ldap_connections_sum{ldap_handler}`	Summary	Connection summary

Name	Type	Description
`ds_connection_handlers_ldap_requests_failure_seconds_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_seconds_sum{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{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_seconds_count{ldap_handler,scope,type}`	Summary	Timer for the specified request type
`ds_connection_handlers_ldap_requests_seconds_count{ldap_handler,type}`	Summary	Timer for the specified request type
`ds_connection_handlers_ldap_requests_seconds_sum{ldap_handler,scope,type}`	Summary	Timer for the specified request type
`ds_connection_handlers_ldap_requests_seconds_sum{ldap_handler,type}`	Summary	Timer for the specified request type
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,scope,type,quantile}`	Summary	Timer for the specified request type
`ds_connection_handlers_ldap_requests_seconds{ldap_handler,type,quantile}`	Summary	Timer for the specified request type
`ds_current_connections`	Gauge	Number of client connections currently established except on the Administration Connector
`ds_current_time_seconds`	Gauge	Current time
`ds_disk_free_space_bytes{disk}`	Gauge	Amount of free disk space

Name	Type	Description
`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_entries_with_aci_attributes_count`	Gauge	Total number of entries with ACI attributes
`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_sum{cache}`	Summary	Number of attempts to retrieve an entry that was not held in this cache
`ds_entry_cache_size_bytes{cache}`	Gauge	Total memory in bytes used by 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_sum{cache}`	Summary	Number of attempts to retrieve an entry from this cache
`ds_groups_count{type}`	Gauge	Total number of groups of the specified type
`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}`	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_domain_generation_id{changelog_id,domain_name}`	Gauge	Replication domain generation identifier
`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,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_domain_generation_id{domain_name,server_id}`	Gauge	Replication domain generation identifier
`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}`	Gauge	Replication domain generation identifier
`ds_replication_changelog_newest_change_number`	Gauge	Newest change number present in the change number index database
`ds_replication_changelog_oldest_change_number`	Gauge	Oldest change number present in the change number index database
`ds_replication_changelog_purge_waiting_for_change_number_indexing`	Gauge	Indicates whether changelog purging is waiting for change number indexing to advance. If true, check the ds-mon-indexing-state and ds-mon-replicas-preventing-indexing metrics
`ds_replication_changelog_replica_dbs_changelog_file_count{domain_name,server_id}`	Gauge	The number of changelog files containing updates generated by this replica. A value of zero indicates the replica did not generate any updates during the last purge delay interval

Name	Type	Description
`ds_replication_changelog_replica_dbs_newest_csn_timestamp_seconds{domain_name,server_id}`	Gauge	Time of the newest CSN present in the replica database
`ds_replication_changelog_replica_dbs_oldest_csn_timestamp_seconds{domain_name,server_id}`	Gauge	Time of the oldest CSN present in the replica database
`ds_replication_replica_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_replica_domain_generation_id{domain_name,server_id}`	Gauge	Replication domain generation identifier
`ds_replication_replica_entries_awaiting_updates_count{domain_name,server_id}`	Gauge	Number of entries for which an update operation has been received but not replayed yet by this replica
`ds_replication_replica_lost_connections{domain_name,server_id}`	Gauge	Number of times the replica lost its connection to the replication server
`ds_replication_replica_remote_replicas_receive_delay_seconds{domain_name,remote_server_id,server_id}`	Gauge	Current local delay in receiving replicated operations
`ds_replication_replica_remote_replicas_replay_delay_seconds{domain_name,remote_server_id,server_id}`	Gauge	Current local delay in replaying replicated operations
`ds_replication_replica_remote_replicas_replayed_updates_seconds_count{domain_name,remote_server_id,server_id}`	Summary	Replay etime for updates that have been replayed on this replica
`ds_replication_replica_remote_replicas_replayed_updates_seconds_sum{domain_name,remote_server_id,server_id}`	Summary	Replay etime for updates that have been replayed on this replica
`ds_replication_replica_remote_replicas_replayed_updates_seconds{domain_name,remote_server_id,server_id,quantile}`	Summary	Replay etime for updates that have been replayed on this replica

Name	Type	Description
`ds_replication_replica_remote_replicas_updates_in_progress{domain_name,remote_server_id,server_id}`	Gauge	Number of remote updates received in the queue waiting to be replayed
`ds_replication_replica_remote_replicas_updates_in_queue{domain_name,remote_server_id,server_id}`	Gauge	Number of remote updates received in the queue
`ds_replication_replica_replayed_internal_updates_total{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_internal_updates{domain_name,server_id}` (deprecated)	Counter	Number of updates replayed on this replica which modify the internal state but not user data
`ds_replication_replica_replayed_updates_conflicts_resolved_total{domain_name,server_id}`	Counter	Number of updates replayed on this replica for which replication naming conflicts have been resolved
`ds_replication_replica_replayed_updates_conflicts_resolved{domain_name,server_id}` (deprecated)	Counter	Number of updates replayed on this replica for which replication naming conflicts have been resolved
`ds_replication_replica_replayed_updates_conflicts_unresolved_total{domain_name,server_id}`	Counter	Number of updates replayed on this replica for which replication naming conflicts have not been resolved
`ds_replication_replica_replayed_updates_conflicts_unresolved{domain_name,server_id}` (deprecated)	Counter	Number of updates replayed on this replica for which replication naming conflicts have not been resolved
`ds_replication_replica_replayed_updates_seconds_count{domain_name,server_id}`	Summary	Replay etime for updates that have been replayed on this replica
`ds_replication_replica_replayed_updates_seconds_sum{domain_name,server_id}`	Summary	Replay etime for updates that have been replayed on this replica
`ds_replication_replica_replayed_updates_seconds{domain_name,server_id,quantile}`	Summary	Replay etime for updates that have been replayed on this replica
`ds_replication_replica_sent_updates_total{domain_name,server_id}`	Counter	Number of replication updates sent by this replica

Name	Type	Description
`ds_replication_replica_ssl_encryption{domain_name,server_id}`	Gauge	Whether SSL encryption is used when exchanging messages with this server
`ds_replication_replica_status_last_changed_seconds{domain_name,server_id}`	Gauge	Last time the replication status of the local replica changed
`ds_replication_replica_status{domain_name,server_id,status}`	Gauge	Status of the specified replica
`ds_replication_replica_total_update_entry_count{domain_name,server_id}`	Gauge	The total number of entries to be processed when a total update is in progress
`ds_replication_replica_total_update_entry_left{domain_name,server_id}`	Gauge	The number of entries still to be processed when a total update is in progress
`ds_replication_replica_updates_already_in_progress_total{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_already_in_progress{domain_name,server_id}` (deprecated)	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{domain_name,server_id}`	Gauge	Number of remote updates received from the replication server but not replayed yet on this replica
`ds_replication_replica_updates_outbound_queue{domain_name,server_id}`	Gauge	Number of local updates that are waiting to be sent to the replication server once they complete
`ds_start_time_seconds`	Gauge	Time the Directory Server started
`ds_static_group_size_bucket{le}`	Gauge	Number of static groups with at most the specified number of members
`ds_subentries_count{type}`	Gauge	Total number of LDAP subentries of the specified type
`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

Name	Type	Description
`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_sum`	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.

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.
`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 refer to makeldif-template.
`modrate`	Measure modification throughput and response time.

Command[1]	Description
`searchrate`	Measure search throughput and response time.

[1] Linux 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 decide whether it does trust the server certificate and continues to negotiate a secure connection, or doesn't 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 `%USERPROFILE%` on Windows. The keystore password is `OpenDJ`. Do not change the file name or the password. • 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}`	DS only uses the specified truststore. The <*Type*> in the option name reflects the trust store type. The tool fails with an error if it can't 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
bindPassword\:file=/path/to/.pwd
useSsl=true
```

When you use an option with a colon, such as `bindPassword:file`, escape the colon with a backslash (`\:`) in the properties file.

The file location on Windows is `%UserProfile%\.opendj\tools.properties`.

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, refer to Learn about the evaluation setup profile.
The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
```

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, refer to 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))`.

Substring searches

Sometimes you only have part of the value that the search must match. Use a *substring search* in this case.

The following example returns entries with phone numbers that contain `5551212`:

```
$ 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 \
 "(telephoneNumber=*5551212*)"
```

The following example returns entries where the full name (common name) starts with `Barb`:

```
$ 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=barb*)"
```

The filter has `barb` in lower case because the `cn` attribute is case-insensitive. Many standard LDAP attributes are case-insensitive.

The following example returns entries where the full name ends with `ensen`:

```
$ 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=*ensen)"
```

Tip

If you had the whole last name, such as `Jensen`, then you would use `(sn=jensen)` as the search filter. Substring searches are useful, but they are also more expensive than exact searches.

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.

With the setting shown in the example, `ds-pwp-last-login-time-format: yyyyMMddHH'Z'`, DS records last login time to the nearest hour. For each bind where the timestamp changes, DS updates the timestamp on the entry. So this recommended setting avoids updating entries often for users who bind repeatedly over a short period. If the deployment requires a fine-grained last login timestamp, use a format that includes minutes or seconds. For example, to get last login times that are accurate to the second, use `ds-pwp-last-login-time-format:"yyyyMMddHHmmss'Z'"`.

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 HDAP API reference.

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, refer to 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, refer to 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, refer to Learn about the evaluation setup profile.
> The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
> ```

For details on the LDIF format shown in the examples that follow, refer to 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 observe it, and updates profiles exactly as users observe 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, refer to 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, refer to 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 observe `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 the 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, refer to Learn about the evaluation setup profile.
> The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
> ```

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 get the changes:

```
$ 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, refer to 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 stops returning results. 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, refer to 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 observed. 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 directory superuser (`uid=admin`) to read the change log. Other accounts require sufficient access and privileges to read the change log. For instructions, refer to 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>
-
```

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
cmVwbGFjZTogZGVzY3JpcHRpb246IE5ldywgaW1wcm92ZWQgZGVzY3JpcHRpb24KLQpyZXBsYWNlOiBtb2RpZmllcnNOYW
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 HDAP

This guide shows you how to use the DS **H**TTP **D**irectory **A**ccess **P**rotocol (HDAP) APIs for REST-based HTTP access to directory services. HDAP transforms HTTP operations into LDAP operations and maps JSON resources to LDAP entries.

The interface stability for HDAP is *Evolving*.

> **Note**
>
> REST to LDAP remains supported as documented for DS 7.3. The interface stability for REST to LDAP is *Deprecated* in favor of HDAP for future applications.

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.

HDAP API reference

HDAP APIs offer HTTP access to directory data as JSON resources. DS software maps JSON resources to LDAP entries.

Note

Examples in this documentation depend on features activated in the `ds-evaluation` setup profile. For details, refer to Learn about the evaluation setup profile.
The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
```

Authentication

Anonymous

When you perform an operation without authenticating, HDAP responds like LDAP would:

- If DS access control lets anonymous users perform the operation, the operation returns the expected result.

  For example, the `ds-evaluation` profile lets anonymous users read and search many attributes.

- If anonymous users can't perform the operation, HDAP returns HTTP 401 Unauthorized.

  HTTP status code 401 means the user must authenticate.

Curl

```
$ curl \
 --cacert ca-cert.pem \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen'
```

JavaScript

```
const { doRequest, getOptions } = require('./utils')

const options = getOptions({
    path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers: { 'Content-Type': 'application/json' }
})

doRequest('HDAP: anonymous read', options)
    .then(response => { console.log(response) })
    .catch(error => { console.error(error) })
```

Source files for this sample: anonymous-read.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, anonymous-read.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('', '')
options = { ca_file: utils.ca_pem }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
end
response = hdap.get('dc=com/dc=example/ou=People/uid=bjensen')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, anonymous-read.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Basic auth

HDAP supports HTTP Basic authorization with the `_id` of resource as the HTTP username. The `_id` matches the suffix of the path to the resource. For example, when you set up DS with the `ds-evaluation` profile, Babs Jensen's `_id` is `dc=com/dc=example/ou=People/uid=bjensen`.

Use HTTPS to avoid sending the password over an insecure connection.

Curl

HTTP Basic authorization

```
$ curl \
 --user dc=com/dc=example/ou=People/uid=bjensen:hifalutin \
 --cacert ca-cert.pem \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=cn'
{"_id":"dc=com/dc=example/ou=People/uid=bjensen","_rev":"<revision>","cn":["Barbara Jensen","Babs Jensen"]}
```

HTTP Basic with credentials in the URL

```
$ curl \
 --cacert ca-cert.pem \
 'https://dc=com%2Fdc=example%2Fou=People%2Fuid=bjensen:hifalutin@localhost:8443/hdap/dc=com/dc=example/
ou=People/uid=bjensen?_fields=cn'
{"_id":"dc=com/dc=example/ou=People/uid=bjensen","_rev":"<revision>","cn":["Barbara Jensen","Babs Jensen"]}
```

JavaScript

```
const { doRequest, getOptions } = require('./utils')

const options = getOptions({
    path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=cn',
    credentials: 'dc=com/dc=example/ou=People/uid=bjensen:hifalutin'
})

doRequest('HDAP: Basic auth', options)
    .then(response => { console.log(response) })
    .catch(error => { console.error(error) })
```

Source files for this sample: basic-auth.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
from requests.auth import HTTPBasicAuth
import utils

response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    auth=HTTPBasicAuth('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin'),
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, basic-auth.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('', '')
options = { ca_file: utils.ca_pem }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: { '_fields': 'cn' }, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, :basic, 'dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin'
end
response = hdap.get('dc=com/dc=example/ou=People/uid=bjensen')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, basic-auth.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Bearer auth

HDAP supports HTTP Bearer authorization using an access token. The access token is a JSON Web Token (JWT) DS returns for a successful authentication action.

Tip

Using a JWT bearer token for authorization means DS only has to authenticate the user once during the lifetime of the token.
This is especially good when performing multiple operations as a user with a strong password storage scheme. The computational cost to validate the password is high, but you only pay the cost once. The cost to validate a JWT token is low for each subsequent operation.

Use HTTPS POST to the path for the account to authenticate with `_action=authenticate` in the query string and a JSON object `{ "password": "<password>" }` as the request payload.

On success, the HTTP status code is 200 OK, and the response body is a JSON resource with an `access_token`. Use the bearer token to authorize subsequent HTTP requests:

Curl

HTTP Bearer authorization

```
$ curl \
 --request POST \
 --header 'Content-Type: application/json' \
 --data '{ "password": "hifalutin" }' \
 --cacert ca-cert.pem \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_action=authenticate'
{"access_token":"<jwt>","expires_in":"299","token_type":"Bearer"}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const resource = '/hdap/dc=com/dc=example/ou=People/uid=bjensen'
    const options = getOptions({ path: resource })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: JWT auth', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: basic-auth.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, basic-auth.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: { '_fields': 'cn' }, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People/uid=bjensen')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, basic-auth.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

For alternative authentication options, refer to:

- Configure HTTP authorization
- Identity mappers

Resources

HDAP resources are JSON objects whose top-level elements are fields with string identifiers; for example:

```
{
  "_id" : "dc=com/dc=example/ou=People/uid=user.0",
  "_rev" : "0d3ce3bf-4107-3b34-9e5a-fa71deb8b504",
  "objectClass" : [ "top", "person", "organizationalPerson", "inetOrgPerson" ],
  "cn" : [ "Aaccf Amar" ],
  "description" : [ "This is the description for Aaccf Amar." ],
  "employeeNumber" : "0",
  "givenName" : [ "Aaccf" ],
  "homePhone" : [ "+1 720 204 9090" ],
  "initials" : [ "AAA" ],
  "l" : [ "Harrisonburg" ],
  "mail" : [ "user.0@example.com" ],
  "mobile" : [ "+1 412 030 0042" ],
  "pager" : [ "+1 439 180 6470" ],
  "postalAddress" : [ "Aaccf Amar$31206 Spring Street$Harrisonburg, IL  04284" ],
  "postalCode" : [ "04284" ],
  "sn" : [ "Amar" ],
  "st" : [ "IL" ],
  "street" : [ "31206 Spring Street" ],
  "telephoneNumber" : [ "+1 485 381 2060" ],
  "uid" : [ "user.0" ]
}
```


HDAP resources have the following special fields:

`_id`

Unique identifier.

The resource `_id` matches the trailing components of its path. For example, the resource at `/hdap/dc=com/dc=example/ou=People/uid=bjensen` has `"_id":"dc=com/dc=example/ou=People/uid=bjensen"`.

The path elements of an `_id` are URL path-encoded RDN strings. When you create a resource:

- Percent-encode each path element.
- If the path element includes characters to escape in LDAP, `" # + , ; < = > \`, escape those characters in the LDAP RDN string before percent-encoding the result for HDAP.

CN value	LDAP RDN	HDAP path element
`Babs Jensen`	`cn=Babs Jensen`	`cn=Babs%20Jensen`
`Babs/Jensen`	`cn=Babs/Jensen`	`cn=Babs%2FJensen`
`Babs\Jensen`	`cn=Babs\\Jensen`	`cn=Babs%5C%5CJensen`
`Babs,Jensen`	`cn=Babs\2CJensen`	`cn=Babs%5C2Jensen`

`_rev`

The resource revision.

HDAP versions individual resources with revision numbers corresponding to the LDAP `ETag` operational attribute. HDAP uses the `If-Match: <revision>` header to determine whether to apply changes to a resource.

The other JSON fields have the same names as the LDAP attributes they represent.

Fields

Multivalued LDAP attributes correspond to arrays in JSON. Unlike arrays in JavaScript, these arrays have set semantics. No duplicates are allowed and the element order is arbitrary.

By default, HDAP behaves like a normal LDAP client application, returning JSON fields for all non-empty LDAP user attributes.

- To return JSON fields for operational LDAP attributes, request them specifically with the `_fields` parameter.

  Use `+`, which encodes to `%2B`, to return fields for all operational attributes.

- To return empty fields, set the advanced HDAP endpoint configuration property `return-null-for-missing-properties:true`.

  With the feature enabled, HDAP returns empty single-valued fields as `null` and empty multivalued fields as `[]`, except for password fields, which HDAP returns as an array even if they are constrained to being single-valued.

  ACI `deny` rules can cause misleading results, where although an "empty" attribute is set you cannot read it. Check the applicable ACIs when HDAP does not return an expected field.

Values

HDAP derives its resource field values from LDAP attribute values based on the attribute syntax.

LDAP attribute type	LDAP example	JSON field type	JSON example
ACI	`(targetattr="userPassword")(version 3.0; acl "Read own password";allow (read,search,compare) userdn="ldap:///self";)`	ACI string[1]	`(targetattr=\"userPassword\")(version 3.0; acl \"Read own password\";allow (read,search,compare) userdn=\"ldap:///self\";)`
Binary	A binary photo or a certificate	Base64-encoded string	A base64-encoded photo or certificate
Boolean	`true`	Boolean	`true`
DN	`uid=bjensen,ou=People,dc=example,dc=com`	`_id` string	`dc=com/dc=example/ou=People/uid=bjensen`
JSON	`{"array":[{"x":1,"y":2},{"x":3,"y":4}]}`	JSON	`{"array":[{"x":1,"y":2},{"x":3,"y":4}]}`
Number	`42`	Number	`42`

LDAP attribute type	LDAP example	JSON field type	JSON example
Password	A password hash	Password string	The same password hash
Postal address	`1234 Main St.$Anytown, CA 12345$USA`	Array of strings[1]	`["1234 Main St.","Anytown, CA 12345","USA"]`
String	`Hello world!`	String[1]	`Hello world!`
Subtree specification	`{ base "ou=people", specificationFilter "(uid=bjensen)" }`	Subtree specification object[1], [2]	`{ "base": "ou=people", "filter": "uid eq \"bjensen\"" }`
Telephone number	`+1 408 555 1212`	Telephone number string	`+1 408 555 1212`
Time	`20230622065924Z`	ISO 8601 string	`2023-06-22T06:59:24Z`
UUID	`597ae2f6-16a6-1027-98f4-d28b5365dc14`	UUID string	`597ae2f6-16a6-1027-98f4-d28b5365dc14`

[1] JSON strings are enclosed in double quotes, so double quotes are escaped with a backslash `\"`.

[2] DS supports more subtree specification features than demonstrated in this simple example. For complex subtree specifications, add an example from LDIF, read the resource, and review the resulting JSON.

Schema

HDAP provides two ways to read JSON schema for resources:

- Read the schema for an existing resource or a resource to create with the `schema` action.
- Read the schema for an individual field or object class directly.

  The fields map to LDAP attribute types. A resource's `objectClass` values map to LDAP object classes.

To read the schema for	Use

An individual field.	HTTP GET on `schemas/attributeTypes/type`.
	Example:
	`GET /hdap/schemas/attributeTypes/cn HTTP/1.1`
	`Host: example.com`
	`Accept: application/json`

To read the schema for	Use

An object class.	HTTP GET on `schemas/objectClasses/class`.
	Example:
	`GET /hdap/schemas/objectClasses/person HTTP/1.1`
	`Host: example.com`
	`Accept: application/json`

Operations

HDAP APIs support the following operations:

HDAP operation	Description	HTTP method
Create	Add a new resource	PUT or POST
Read	Retrieve a single resource	GET
Update	Replace content in an existing resource	PUT
Delete	Remove an existing resource	DELETE
Patch	Modify an existing resource	PATCH
Action	Perform a predefined action	POST
Query	List resources	GET

Create

Use either HTTP POST or HTTP PUT.

Use HTTP POST with the query string parameter `_action=create` and the JSON resource as a payload. Accept a JSON response. HDAP builds the `_id` using the parent resource path and the field in the resource corresponding to the LDAP RDN:

```
POST /hdap/dc=com/dc=example/ou=People?_action=create HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
{ JSON resource }
```

Use HTTP PUT with the `_id` in the resource and the URL path. Use the `If-None-Match: *` header. Accept a JSON response:

```
PUT /hdap/dc=com/dc=example/ou=People/uid=newuser HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
If-None-Match: *
{ JSON resource }
```

The response indicates the resource location in 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: *`, HDAP 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.

Supported parameters:

- `_action=create`
- `dryRun=true`
- `_fields`
- `manageDsaIT=true`
- `passwordQualityAdvice=true`
- `_prettyPrint=true`
- `relax=true`

For examples, refer to Create.

Read

Read a resource with HTTP GET and accept a JSON response:

```
GET /hdap/dc=com/dc=example/ou=People/uid=newuser HTTP/1.1
Host: example.com
Accept: application/json
```

Supported parameters:

- `_fields`
- `manageDsaIT=true`
- `_prettyPrint=true`

For examples, refer to Read.

Update

Update a resource with HTTP PUT and the JSON resource as the payload. Accept a JSON response.

- Unlike other Common REST applications, HDAP does not require the full JSON resource.

  HDAP replaces the fields in the resource with the fields in the payload, operating like a patch Replace. Other fields retain their existing values.

- Use the `If-Match: _rev` header to update the resource only if the revision matches.

  Use `If-Match: *` to apply the update irrespective of the revision.

```
PUT /hdap/dc=com/dc=example/ou=People/uid=newuser HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
If-Match: _rev
{ JSON resource }
```

Supported parameters:

- `dryRun=true`
- `_fields`
- `manageDsaIT=true`
- `passwordQualityAdvice=true`
- `_prettyPrint=true`
- `relax=true`

For examples, refer to Update.

Delete

Use HTTP DELETE and accept a JSON response:

```
DELETE /hdap/dc=com/dc=example/ou=People/uid=newuser HTTP/1.1
Host: example.com
Accept: application/json
```

Supported parameters:

- `dryRun=true`
- `_fields`
- `manageDsaIT=true`
- `_prettyPrint=true`

- `subtreeDelete=true`

For examples, refer to Delete.

Patch

Use HTTP PATCH request with a JSON array of patch objects appropriate to the operation. Each patch object in the payload can have these fields:

`operation`

HDAP supports these patch operations:

- `add`
- `remove`
- `replace`
- `increment`

HDAP does not support the common REST `copy`, `move`, or `transform` operations.

`field`

A JSON pointer to the target field

`value`

The value for the patch

The patch request has the following format:

```
PATCH /hdap/dc=com/dc=example/ou=People/uid=newuser 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 these types of target fields:

- **Single-valued**, such as an object, string, boolean, or number.
- **Set semantics array**, where the elements are not ordered and duplicates are not allowed.

  If you reference array values by index, HDAP returns an error.

Supported parameters:

- `dryRun=true`
- `_fields`
- `manageDsaIT=true`

- `passwordQualityAdvice=true`
- `_prettyPrint=true`
- `relax=true`

For examples, refer to Patch.

Add

Patch `add` ensures the target field contains the value provided, creating parent fields as necessary.

- If the target field is single-valued and a value already exists, HDAP replaces the value with the value you provide.

  HDAP does not return an error when adding a value to a single-valued field with an existing value.

- If the target field is multivalued (an array), HDAP merges the array of values you provide with the existing values.

  For arrays, HDAP adds new values and ignores duplicates.

Add a telephone number

```
[{
  "operation" : "add",
  "field" : "/telephoneNumber",
  "value" : "+1 408 555 1212"
}]
```

Remove

Patch `remove` ensures the target field does not contain the value provided.

- If you do not provide a value and the target field exists, HDAP removes the entire field.
- If you provide a value and the target field is single-valued, the value must match the existing value to remove; otherwise, HDAP does not change the field.
- If the target field is multivalued, HDAP removes the values in the array you provide from the existing set of values.

Remove all telephone numbers

```
[{
  "operation" : "remove",
  "field" : "/telephoneNumber"
}]
```

Remove a specific telephone number

```
[{
  "operation" : "remove",
  "field" : "/telephoneNumber",
  "value" : "+1 408 555 1212"
}]
```

Replace

Patch `replace` removes existing values on the target field and replaces them with the values you provide. It is equivalent to a `remove` of the field followed by an `add`.

Reset the telephone number

```
[{
  "operation" : "replace",
  "field" : "/telephoneNumber",
  "value" : "+1 408 555 1212"
}]
```

Increment

Patch `increment` changes the value(s) in the target field by the amount you specify.

- Use a positive number to increment or a negative number to decrement the value(s).
- The target field must hold a number or a set of numbers.
- The value you provide must be a single number.

Lower the temperature

```
[{
  "operation" : "increment",
  "field" : "/temperature",
  "value" : -2
}]
```

Action

Use HTTP POST with the `_action` parameter to request a predefined action.

- `_action=accountUsability`: Determine whether the user can authenticate to the directory.
- `_action=create`: Create a resource with HTTP POST.
- `_action=modifyPassword`: Change your password.
- `_action=rename`: Rename a resource, changing its `_id`.

- `_action=resetPassword`: Change another user's password.
- `_action=schema`: Get the JSON schema for an object.

Supported parameters:

- `_action`
- `deleteOldRdn=true` (rename operation)
- `dryRun=true` (create, password, and rename operations)
- `_fields`
- `manageDsaIT=true` (rename operation)
- `objectClasses` (schema operation)
- `passwordQualityAdvice=true` (password operations)
- `_prettyPrint=true`
- `relax=true` (rename operation)

The request and response depend on the action. For details, refer to Actions.

Query

Use HTTP GET with the `_queryFilter` parameter to list resources at or under a target resource and matching a query filter. Accept a JSON response.

```
GET /hdap/dc=com/dc=example/ou=People?_queryFilter=... HTTP/1.1
Host: example.com
Accept: application/json
```

HDAP returns the result as a JSON object including a `results` array. Other response fields depend on the parameters.

Supported parameters:

- `_countOnly=true`
- `_fields`
- `_pagedResultsCookie`
- `_pageSize`
- `_prettyPrint=true`
- `_queryFilter`
- `scope`
- `_sortKeys`
- `subentries=true`

- `_totalPagedResultsPolicy`

For examples, refer to Query.

Headers

In addition to the headers described for HDAP Operations, HDAP APIs support these Common REST headers.

Accept-API-Version

The `Accept-API-Version` header specifies 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.

The `Content-API-Version` response header specifies the protocol and resource versions for the operation. The default HDAP settings are equivalent to:

```
Accept-API-Version: protocol=2.1,resource=1.0
```

X-ForgeRock-TransactionId

The optional `X-ForgeRock-TransactionId` header helps to track related requests through the Ping 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.

You configure platform products to trust transaction IDs and let them propagate for audit purposes. For DS, refer to Log LDAP access to files or Log HTTP access to files.

Query parameters

HDAP supports the following query string parameters.

> **Note**
>
> Some parameter values are not safe for URLs.
> URL-encode parameter values as necessary.

`_action=<action>`

Perform an extended action as part of an HTTP POST.

The <action> is one of:

- `accountUsability`
- `create`
- `modifyPassword`
- `rename`
- `resetPassword`
- `schema`

`_countOnly=true`

Return a count of query results without returning the resources.

This parameter requires protocol version 2.2 or later in the `Accept-API-Version` request header:

```
Accept-API-Version: protocol=2.2,resource=1.0
```

`deleteOldRdn=true`

Delete the old RDN value when renaming a resource.

`dryRun=true`

Discover how a server reacts to an 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`.

`_fields=<field>[,<field>…]`

Return only the specified fields in the body of the response.

The <field> values are JSON pointers. In `{"parent":{"child":"value"}}`, `parent/child` refers to `"child": "value"`.

When the request omits the `_field` parameter, HDAP returns fields for all LDAP user attributes.

HDAP returns fields for operational attributes only when specifically requested. Use `+`, which encodes to `%2B`, to return fields for all operational attributes.

`manageDsaIT=true`

Manage referrals.

This parameter relies on the LDAP manage DSAIT request control, OID `2.16.840.1.113730.3.4.2`.

`objectClasses=<objectClass>[,<objectClass>…]`

Return JSON schema for a resource to create based on the LDAP object classes and the parent resource.

`_pagedResultsCookie=<cookie>`

The <cookie> is an opaque string HDAP uses when paging to keep track of the position in the query results:

1. Set the `_pageSize` in the request to a non-zero number.

   HDAP returns the cookie with the first request.

2. Supply the cookie in subsequent requests until HDAP returns a `null` cookie when it reaches the final page.

`_pageSize=<number>`

Return query results in pages of this size.

After the initial request, use `_pagedResultsCookie` or `_pageResultsOffset` to page through the results.

`passwordQualityAdvice=true`

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`.

`_prettyPrint=true`

Format the body of the response.

`_queryFilter=filter-expression`

Query resources matching the filter expression.

You must URL-escape the filter-expression.

`relax=true`

Relax data and service rules temporarily for the requested update.

This parameter relies on the LDAP relax rules control, OID `1.3.6.1.4.1.4203.666.5.12`.

`scope=<scope>`

Scope of the query; one of:

- `base`: Query only the target resource.
- `one` (default): Query direct child resources of the target resource.
- `sub`: Query the target resource and all child resources recursively.
- `subordinates`: Query all child resources recursively but do not include the target resource.

`_sortKeys=(|-)[.var]##<field>##[,(|-)<field>…]`

Sort the query results based on the specified field(s).

- Use `+` for ascending order (default, encoded as `%2B`).
- Use `-` for descending order.

`subentries=true`

Return resources corresponding to LDAP subentries. Subentries aren't returned by default.

This parameter relies on the LDAP subentries request control, OID `1.3.6.1.4.1.4203.1.10.1`.

`subtreeDelete=true`

Delete an entire subtree of resources.

This parameter relies on the LDAP subtree delete control, OID `1.2.840.113556.1.4.805`.

`_totalPagedResultsPolicy=<policy>`

When a `_pageSize` is non-zero, HDAP can calculate `totalPagedResults`. It returns the `totalPagedResults` in the response depending on the <policy>:

- `"totalPagedResults": -1` by default, when `_pageSize` is not set, or when `_totalPagedResultsPolicy=NONE`.
- An estimated count when `_totalPagedResultsPolicy=ESTIMATE`.
- The exact count when `_totalPagedResultsPolicy=EXACT`, if possible.

Filter expressions

Query filters request entries matching the filter expression. You must URL-escape the filter expression.

The string representation of the filter expression is:

```
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 and requires URL encoding. For details, refer to RFC 3986: Uniform Resource Identifier (URI): Generic Syntax. For example, spaces, double quotes, parentheses, and exclamation characters need URL encoding. 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  = "!" / "$" / "&" / "'" / "(" / ")"
                 / "*" / "+" / "," / ";" / "="
```

`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"
```

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 `uid eq 'test\\'`, use `uid+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.
- 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.

HTTP status codes

When working with a HDAP APIs, client applications should expect at least these HTTP status codes:

200 OK

The request succeeded and HDAP returned a resource, depending on the request.

201 Created

The request succeeded and the HDAP created the resource.

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 unacceptable parameters, such as an unsupported resource or protocol version.

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 and no version was supplied in the request.

500 Internal Server Error

HDAP encountered an unexpected condition preventing 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.

Create

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile. For details, refer to Learn about the evaluation setup profile.
> The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
> ```

Before creating a resource, you can get its JSON schema if you know its LDAP object classes.

Create (HTTP PUT)

Use HTTP PUT with the headers `Content-Type: application/json` and `If-None-Match: *`. The payload is the JSON resource:

Curl

```
$ curl \
 --request PUT \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --header 'If-None-Match: *' \
 --data '{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
 }' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser?_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=newuser',
        method: 'PUT',
        body: {
            "_id": "dc=com/dc=example/ou=People/uid=newuser",
            "objectClass": ["person", "inetOrgPerson", "organizationalPerson", "top"],
            "cn": ["New User"],
            "givenName": ["New"],
            "mail": ["newuser@example.com"],
            "manager": ["dc=com/dc=example/ou=People/uid=bjensen"],
            "sn": ["User"],
            "telephoneNumber": ["+1 408 555 1212"],
            "uid": ["newuser"]
        }
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    options.headers['If-None-Match'] = '*'
    const response = await doRequest('HDAP: create with PUT', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: create.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

body = {
    '_id': 'dc=com/dc=example/ou=People/uid=newuser',
    'objectClass': ['person', 'inetOrgPerson', 'organizationalPerson', 'top'],
    'cn': ['New User'],
    'givenName': ['New'],
    'mail': ['newuser@example.com'],
    'manager': ['dc=com/dc=example/ou=People/uid=bjensen'],
    'sn': ['User'],
    'telephoneNumber': ['+1 408 555 1212'],
    'uid': ['newuser']
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = {
    'Authorization': f'Bearer {jwt}',
    'Content-Type': 'application/json',
    'If-None-Match': '*'
}
response = requests.put(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=newuser',
    headers=headers,
    json=body,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, create.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = {
    '_id' => "dc=com/dc=example/ou=People/uid=newuser",
    'objectClass' => ["person", "inetOrgPerson", "organizationalPerson", "top"],
    'cn' => ["New User"],
    'givenName' => ["New"],
    'mail' => ["newuser@example.com"],
    'manager' => ["dc=com/dc=example/ou=People/uid=bjensen"],
    'sn' => ["User"],
    'telephoneNumber' => ["+1 408 555 1212"],
    'uid' => ["newuser"]
}
response = hdap.put do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=newuser'
    h.body = JSON.generate(body)
    h.headers['If-None-Match'] = '*'
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, create.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Create (HTTP POST)

Use HTTP POST with the header `Content-Type: application/json` and, optionally, the parameter `_action=create`. The payload is the JSON resource:

Curl

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user uid=admin:password \
 --header 'Content-Type: application/json' \
 --data '{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
 }' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People?_action=create&_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People?_action=create',
        method: 'POST',
        body: {
            "_id": "dc=com/dc=example/ou=People/uid=newuser",
            "objectClass": ["person", "inetOrgPerson", "organizationalPerson", "top"],
            "cn": ["New User"],
            "givenName": ["New"],
            "mail": ["newuser@example.com"],
            "manager": ["dc=com/dc=example/ou=People/uid=bjensen"],
            "sn": ["User"],
            "telephoneNumber": ["+1 408 555 1212"],
            "uid": ["newuser"]
        }
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: create with POST', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: create-post.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

body = {
    '_id': 'dc=com/dc=example/ou=People/uid=newuser',
    'objectClass': ['person', 'inetOrgPerson', 'organizationalPerson', 'top'],
    'cn': ['New User'],
    'givenName': ['New'],
    'mail': ['newuser@example.com'],
    'manager': ['dc=com/dc=example/ou=People/uid=bjensen'],
    'sn': ['User'],
    'telephoneNumber': ['+1 408 555 1212'],
    'uid': ['newuser']
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    json=body,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, create-post.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = {
    '_id' => "dc=com/dc=example/ou=People/uid=newuser",
    'objectClass' => ["person", "inetOrgPerson", "organizationalPerson", "top"],
    'cn' => ["New User"],
    'givenName' => ["New"],
    'mail' => ["newuser@example.com"],
    'manager' => ["dc=com/dc=example/ou=People/uid=bjensen"],
    'sn' => ["User"],
    'telephoneNumber' => ["+1 408 555 1212"],
    'uid' => ["newuser"]
}
response = hdap.post do |h|
    h.path = 'dc=com/dc=example/ou=People'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, create-post.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Read

Note

Examples in this documentation depend on features activated in the `ds-evaluation` setup profile. For details, refer to Learn about the evaluation setup profile.
The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
```

Read a resource

Read with HTTP GET:

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --data '_prettyPrint=true' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen'
{
  "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
  "_rev" : "<revision>",
  "objectClass" : [ "person", "cos", "oauth2TokenObject", "inetOrgPerson", "organizationalPerson",
"posixAccount", "top" ],
  "classOfService" : "bronze",
  "cn" : [ "Barbara Jensen", "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" : [ "dc=com/dc=example/ou=People/uid=trigden" ],
  "oauth2Token" : [ {
    "access_token" : "123",
    "expires_in" : 59,
    "token_type" : "Bearer",
    "refresh_token" : "456"
  } ],
  "ou" : [ "Product Development", "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,
  "userPassword" : [ "<hashed-password>" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen'
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: read with GET', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: read.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, read.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People/uid=bjensen')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, read.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Read specific fields

HDAP can return only specified fields in the resource. Use the `_fields` parameter:

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --data '_fields=cn,mail' \
 --data '_prettyPrint=true' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen'
{
  "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
  "_rev" : "<revision>",
  "mail" : [ "bjensen@example.com" ],
  "cn" : [ "Barbara Jensen", "Babs Jensen" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=cn,mail'
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: read specific fields', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: read-fields.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = { '_fields': 'cn,mail' }
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, read-fields.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
fields = { '_fields': 'cn,mail' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: fields, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People/uid=bjensen')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, read-fields.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Update

Note

Examples in this documentation depend on features activated in the `ds-evaluation` setup profile. For details, refer to Learn about the evaluation setup profile.
The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
```

Update a resource

Use HTTP PUT to replace any and all writable fields in a resource.

The effect is the same as a patch replace operation.

The following example updates Sam Carter's telephone number regardless of the revision:

Curl

```
$ curl \
 --request PUT \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --data '{"telephoneNumber": "+1 408 555 1212"}' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=scarter?
_fields=telephoneNumber&_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=scarter",
  "telephoneNumber" : [ "+1 408 555 1212" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=scarter?_fields=telephoneNumber',
        method: 'PUT',
        body: { "telephoneNumber": "+1 408 555 1212" }
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: update telephone number', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: update.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

body = { 'telephoneNumber': '+1 408 555 1212' }
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = { '_fields': 'telephoneNumber' }
response = requests.put(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=scarter',
    headers=headers,
    json=body,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, update.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
fields = { '_fields': 'telephoneNumber' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: fields, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = { "telephoneNumber" => "+1 408 555 1212" }
response = hdap.put do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=scarter'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, update.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Update a specific revision

To update a resource only if the resource matches a particular version, use an `If-Match: <revision>` header:

Curl

```
$ export JWT=$(echo $(curl \
 --request POST \
 --cacert ca-cert.pem \
 --header 'Content-Type: application/json' \
 --data '{ "password": "bribery" }' \
 --silent \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=kvaughan?_action=authenticate') | jq -
r .access_token)

$ export REVISION=$(cut -d \" -f 8 <(curl \
 --get \
 --cacert ca-cert.pem \
 --header "Authorization: Bearer $JWT" \
 --header 'Content-Type: application/json' \
 --data '_fields=_rev' \
 --silent \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=scarter'))

$ curl \
 --request PUT \
 --cacert ca-cert.pem \
 --header "Authorization: Bearer $JWT" \
 --header 'Content-Type: application/json' \
 --header "If-Match: $REVISION" \
 --data '{"telephoneNumber": "+1 408 555 1213"}' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=scarter?
_fields=telephoneNumber&_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=scarter",
  "telephoneNumber" : [ "+1 408 555 1213" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=scarter?_fields=telephoneNumber',
        body: { "telephoneNumber": "+1 408 555 1213" }
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    let response = await doRequest('HDAP: read scarter _rev', options)
    console.log(response)
    options.headers['If-Match'] = JSON.parse(response.data)._rev
    options.method = 'PUT'
    response = await doRequest('HDAP: update specific revision', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: update-rev.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
rev = requests.get(f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=scarter',
    headers=headers,
    verify=utils.ca_pem).json()['_rev']

headers['If-Match'] = rev
body = { 'telephoneNumber': '+1 408 555 1213' }
params = { '_fields': 'telephoneNumber' }
response = requests.put(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=scarter',
    headers=headers,
    json=body,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, update-rev.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
fields = { '_fields': 'telephoneNumber' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: fields, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
resource = 'dc=com/dc=example/ou=People/uid=scarter'
rev = JSON.parse(hdap.get(resource).body, symbolize_names: true)[:_rev]

body = { "telephoneNumber" => "+1 408 555 1213" }
response = hdap.put do |h|
    h.path = resource
    h.body = JSON.generate(body)
    h.headers['If-Match'] = rev
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, update-rev.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Rename a resource

Refer to the rename action.

Delete

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile. For details, refer to Learn about the evaluation setup profile.
> The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
> ```

Delete a resource

Use HTTP DELETE on the resource URL. HDAP returns the resource you deleted:

Curl

```
$ curl \
 --request DELETE \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser?_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=newuser',
        method: 'DELETE'
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: delete newuser', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: delete.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.delete(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=newuser',
    headers=headers,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, delete.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.delete('dc=com/dc=example/ou=People/uid=newuser')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, delete.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Delete a specific revision

To delete a resource only if the resource matches a particular version, use an `If-Match: <revision>` header:

Curl

```
$ export JWT=$(echo $(curl \
 --request POST \
 --cacert ca-cert.pem \
 --header 'Content-Type: application/json' \
 --data '{ "password": "bribery" }' \
 --silent \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=kvaughan?_action=authenticate') | jq -
r .access_token)

$ export REVISION=$(cut -d \" -f 8 <(curl \
 --get \
 --cacert ca-cert.pem \
 --header "Authorization: Bearer $JWT" \
 --header 'Content-Type: application/json' \
 --data '_fields=_rev' \
 --silent \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser'))

$ curl \
 --request DELETE \
 --cacert ca-cert.pem \
 --header "Authorization: Bearer $JWT" \
 --header "If-Match: $REVISION" \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=newuser?_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=newuser",
  "objectClass" : [ "person", "inetOrgPerson", "organizationalPerson", "top" ],
  "cn" : [ "New User" ],
  "givenName" : [ "New" ],
  "mail" : [ "newuser@example.com" ],
  "manager" : [ "dc=com/dc=example/ou=People/uid=bjensen" ],
  "sn" : [ "User" ],
  "telephoneNumber" : [ "+1 408 555 1212" ],
  "uid" : [ "newuser" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=newuser'
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    let response = await doRequest('HDAP: read newuser _rev', options)
    console.log(response)
    options.headers['If-Match'] = JSON.parse(response.data)._rev
    options.method = 'DELETE'
    response = await doRequest('HDAP: delete specific revision', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: delete-rev.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
rev = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=newuser',
    headers=headers,
    verify=utils.ca_pem).json()['_rev']

headers['If-Match'] = rev
response = requests.delete(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=newuser',
    headers=headers,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, delete-rev.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
resource = 'dc=com/dc=example/ou=People/uid=newuser'
rev = JSON.parse(hdap.get(resource).body, symbolize_names: true)[:_rev]

response = hdap.delete do |h|
    h.path = resource
    h.headers['If-Match'] = rev
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, delete-rev.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Delete a subtree

Note

- Only users granted access to perform a subtree delete can remove a resource with children.
- This can be a resource-intensive operation.
  The resources required to remove a branch depend on the number of LDAP entries to delete.

To delete a resource and all of its children, follow these high-level steps:

- When configuring the gateway, make sure `"useSubtreeDelete": "true"` (default).
- Grant the user access to 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
```

- Delete the base resource as a user with access to perform a subtree delete.

  Include the `subtreeDelete=true` query string parameter in the delete request.

Patch

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile. For details, refer to Learn about the evaluation setup profile.
> The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
> ```

The patch operation updates one or more fields of a resource. Use it when you must make fine-grained changes to a resource; for example:

- Add a member to a static group.
- Remove a member from a static group.
- Add or remove a single mail address or telephone number.

If you intend only to *replace* fields' values, update the resource instead with HTTP PUT and a partial resource including just the fields to replace.

Add a member to a group

The following example adds Babs to a static group:

Curl

```
$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --data '[{
  "operation": "add",
  "field": "uniqueMember",
  "value": "dc=com/dc=example/ou=People/uid=bjensen"
  }]' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=Groups/cn=Directory%20Administrators?_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=Groups/cn=Directory%20Administrators",
  "objectClass" : [ "groupofuniquenames", "top" ],
  "cn" : [ "Directory Administrators" ],
  "ou" : [ "Groups" ],
  "uniqueMember" : [ "dc=com/dc=example/ou=People/uid=kvaughan", "dc=com/dc=example/ou=People/
uid=rdaugherty", "dc=com/dc=example/ou=People/uid=hmiller", "dc=com/dc=example/ou=People/uid=bjensen" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=Groups/cn=Directory%20Administrators',
        method: 'PATCH',
        body: [{
            "operation": "add",
            "field": "uniqueMember",
            "value": "dc=com/dc=example/ou=People/uid=bjensen"
        }]
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: add group member', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: patch-group-add.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

patch = [{
    'operation': 'add',
    'field': 'uniqueMember',
    'value': 'dc=com/dc=example/ou=People/uid=bjensen'
}]
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.patch(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=Groups/cn=Directory%20Administrators',
    headers=headers,
    json=patch,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, patch-group-add.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
patch = [{
    "operation" => "add",
    "field" => "uniqueMember",
    "value" => "dc=com/dc=example/ou=People/uid=bjensen"
}]
response = hdap.patch do |h|
    h.path = 'dc=com/dc=example/ou=Groups/cn=Directory%20Administrators'
    h.body = JSON.generate(patch)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, patch-group-add.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Remove a member from a group

The following example removes Babs from the group:

Curl

```
$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --data '[{
  "operation": "remove",
  "field": "uniqueMember",
  "value": "dc=com/dc=example/ou=People/uid=bjensen"
  }]' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=Groups/cn=Directory%20Administrators?_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=Groups/cn=Directory%20Administrators",
  "objectClass" : [ "groupofuniquenames", "top" ],
  "cn" : [ "Directory Administrators" ],
  "ou" : [ "Groups" ],
  "uniqueMember" : [ "dc=com/dc=example/ou=People/uid=kvaughan", "dc=com/dc=example/ou=People/
uid=rdaugherty", "dc=com/dc=example/ou=People/uid=hmiller" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=Groups/cn=Directory%20Administrators',
        method: 'PATCH',
        body: [{
            "operation": "remove",
            "field": "uniqueMember",
            "value": "dc=com/dc=example/ou=People/uid=bjensen"
        }]
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: remove group member', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: patch-group-remove.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

patch = [{
    'operation': 'remove',
    'field': 'uniqueMember',
    'value': 'dc=com/dc=example/ou=People/uid=bjensen'
}]
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.patch(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=Groups/cn=Directory%20Administrators',
    headers=headers,
    json=patch,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, patch-group-remove.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
patch = [{
    "operation" => "remove",
    "field" => "uniqueMember",
    "value" => "dc=com/dc=example/ou=People/uid=bjensen"
}]
response = hdap.patch do |h|
    h.path = 'dc=com/dc=example/ou=Groups/cn=Directory%20Administrators'
    h.body = JSON.generate(patch)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, patch-group-remove.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Add multiple values

To change multiple fields, include multiple operation objects in the patch payload:

Curl

```
$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --data '[{
  "operation": "add",
  "field": "telephoneNumber",
  "value": "+1 408 555 9999"
 }, {
  "operation": "add",
  "field": "mail",
  "value": "barbara.jensen@example.com"
 }]' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?
_fields=mail,telephoneNumber&_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
  "telephoneNumber" : [ "+1 408 555 1862", "+1 408 555 9999" ],
  "mail" : [ "bjensen@example.com", "barbara.jensen@example.com" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=mail,telephoneNumber',
        method: 'PATCH',
        body: [{
            "operation": "add",
            "field": "telephoneNumber",
            "value": "+1 408 555 9999"
        }, {
            "operation": "add",
            "field": "mail",
            "value": "barbara.jensen@example.com"
        }]
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: patch multiple fields', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: patch-multiple.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

patch = [{
    'operation': 'add',
    'field': 'telephoneNumber',
    'value': '+1 408 555 9999'
}, {
    'operation': 'add',
    'field': 'mail',
    'value': 'barbara.jensen@example.com'
}]
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = { '_fields': 'mail,telephoneNumber' }
response = requests.patch(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    json=patch,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, patch-multiple.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
fields = { '_fields': 'mail,telephoneNumber' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: fields, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
patch = [{
    "operation" => "add",
    "field" => "telephoneNumber",
    "value" => "+1 408 555 9999"
}, {
    "operation" => "add",
    "field" => "mail",
    "value" => "barbara.jensen@example.com"
}]
response = hdap.patch do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=bjensen'
    h.body = JSON.generate(patch)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, patch-multiple.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

For a multivalued 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.

Patch a specific revision

Use an `If-Match: <revision>` header to patch only a specific revision of a resource:

Curl

```
$ export JWT=$(echo $(curl \
 --request POST \
 --cacert ca-cert.pem \
 --header 'Content-Type: application/json' \
 --data '{ "password": "bribery" }' \
 --silent \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=kvaughan?_action=authenticate') | jq -
r .access_token)

$ export REVISION=$(cut -d \" -f 8 <(curl \
 --get \
 --cacert ca-cert.pem \
 --header "Authorization: Bearer $JWT" \
 --header 'Content-Type: application/json' \
 --data '_fields=_rev' \
 --silent \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen'))

$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --header "Authorization: Bearer $JWT" \
 --header 'Content-Type: application/json' \
 --header "If-Match: $REVISION" \
 --data '[{
  "operation": "remove",
  "field": "telephoneNumber",
  "value": "+1 408 555 9999"
 }, {
  "operation": "remove",
  "field": "mail",
  "value": "barbara.jensen@example.com"
 }]' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?
_fields=mail,telephoneNumber&_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
  "telephoneNumber" : [ "+1 408 555 1862" ],
  "mail" : [ "bjensen@example.com" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen',
        body: [{
            "operation": "add",
            "field": "telephoneNumber",
            "value": "+1 408 555 9999"
        }, {
            "operation": "add",
            "field": "mail",
            "value": "barbara.jensen@example.com"
        }]
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    let response = await doRequest('HDAP: read bjensen _rev', options)
    console.log(response)
    options.headers['If-Match'] = JSON.parse(response.data)._rev
    options.method = 'PATCH'
    response = await doRequest('HDAP: patch specific revision', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: patch-rev.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
rev = requests.get(f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    verify=utils.ca_pem).json()['_rev']

headers['If-Match'] = rev
patch = [{
    'operation': 'add',
    'field': 'telephoneNumber',
    'value': '+1 408 555 9999'
}, {
    'operation': 'add',
    'field': 'mail',
    'value': 'barbara.jensen@example.com'
}]
params = { '_fields': 'mail,telephoneNumber' }
response = requests.patch(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    json=patch,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, patch-rev.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
fields = { '_fields': 'mail,telephoneNumber' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: fields, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
patch = [{
    "operation" => "add",
    "field" => "telephoneNumber",
    "value" => "+1 408 555 9999"
}, {
    "operation" => "add",
    "field" => "mail",
    "value" => "barbara.jensen@example.com"
}]
resource = 'dc=com/dc=example/ou=People/uid=bjensen'
rev = JSON.parse(hdap.get(resource).body, symbolize_names: true)[:_rev]

response = hdap.patch do |h|
    h.path = resource
    h.body = JSON.generate(patch)
    h.headers['If-Match'] = rev
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, patch-rev.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

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, refer to Learn about the evaluation setup profile.
> The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
> ```

Authenticate

You can use `_action=authenticate` to get a bearer token valid for multiple HDAP requests.

For details, refer to Bearer auth.

Change your password

> **Note**
>
> This action *requires HTTPS* to avoid sending the password over an insecure connection.

Use HTTPS POST with `_action=modifyPassword` in the query string and a JSON object with the old and new passwords in 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

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=bjensen:hifalutin \
 --header 'Content-Type: application/json' \
 --data '{"oldPassword": "hifalutin", "newPassword": "chngthspwd"}' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_action=modifyPassword'
{}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen?_action=modifyPassword&dryRun=true',
        method: 'POST',
        credentials: 'dc=com/dc=example/ou=People/uid=bjensen:hifalutin',
        body: { "oldPassword": "hifalutin", "newPassword": "chngthspwd" }
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: dry-run change password', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: action-change-password.js, utils.js

Remove the `dry-run` parameter to perform the operation.

Python

```
#!/usr/bin/env python3

import requests
import utils

body = { 'oldPassword': 'hifalutin', 'newPassword': 'chngthspwd' }
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = { '_action': 'modifyPassword', 'dryRun': True }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    json=body,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, action-change-password.py

Remove the `dryRun` parameter to perform the operation.

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_action': 'modifyPassword', 'dryRun': true }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = { "oldPassword" => "hifalutin", "newPassword" => "chngthspwd" }
response = hdap.post do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=bjensen'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, action-change-password.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Check password quality

Use the `passwordQualityAdvice` and `dryRun` query string parameters to get details when a password update fails, to test passwords, and to test password policies:

- The `passwordQualityAdvice` parameter relies on the LDAP password quality advice control, OID `1.3.6.1.4.1.36733.2.1.5.5`. Users modifying their password must have access to request the control.

  > **Note**
  >
  > The password quality advice control and the `passwordQualityAdvice` parameter have interface stability: *Evolving*.

- The `dryRun` parameter relies on the LDAP no-op control, OID `1.3.6.1.4.1.4203.1.10.2`.

The following example shows a password update failure. The status code is HTTP 400 Bad Request and the response JSON describes what passed and what failed:

Curl

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=bjensen:hifalutin \
 --header 'Content-Type: application/json' \
 --data '{"oldPassword": "hifalutin", "newPassword": "t00shrt"}' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?
_action=modifyPassword&passwordQualityAdvice=true&dryRun=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" : {
      "attributeType" : "userPassword",
      "failingCriteria" : [ {
        "parameters" : {
          "max-password-length" : 0,
          "min-password-length" : 8
        },
        "type" : "length-based"
      } ],
      "passingCriteria" : [ {
        "parameters" : {
          "case-sensitive-validation" : false,
          "check-substrings" : true,
          "min-substring-length" : 5,
          "test-reversed-password" : false
        },
        "type" : "dictionary"
      } ]
    }
  }
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen?
_action=modifyPassword&passwordQualityAdvice=true&dryRun=true',
        method: 'POST',
        credentials: 'dc=com/dc=example/ou=People/uid=bjensen:hifalutin',
        body: { "oldPassword": "hifalutin", "newPassword": "t00shrt" }
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: dry-run check password quality', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: action-check-password-quality.js, utils.js

Remove the `dryRun` parameter to perform the operation.

Python

```
#!/usr/bin/env python3

import requests
import utils

body = { 'oldPassword': 'hifalutin', 'newPassword': 't00shrt' }
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = { '_action': 'modifyPassword', 'passwordQualityAdvice': True, 'dryRun': True }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    json=body,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, action-check-password-quality.py

Remove the `dryRun` parameter to perform the operation. Remove the `dry-run` parameter to perform the operation.

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_action': 'modifyPassword', 'passwordQualityAdvice': true, 'dryRun': true }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = { "oldPassword" => "hifalutin", "newPassword" => "t00shrt" }
response = hdap.post do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=bjensen'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, action-check-password-quality.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

`"max-password-length" : 0` means DS does not enforce an upper bound.

You can use `passwordQualityAdvice` without the `dryRun` parameter. On failure, you get diagnostic information as shown in the preceding example. On success, the HTTP status code is 200 OK and the response body is an empty JSON resource.

Reset a password

When one user changes another user's password, DS considers it a password reset. Password policies can require users to change their passwords again after a password reset.

> **Note**
>
> This action *requires HTTPS* to avoid sending the password over an insecure connection.

The example demonstrates a password administrator changing a user's password. The password administrator must have the `password-reset` privilege; otherwise, the reset fails due to insufficient access:

Curl

```
$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --user uid=admin:password \
 --header 'Content-Type: application/json' \
 --data '[{
  "operation": "add",
  "field": "ds-privilege-name",
  "value": "password-reset"
 }]' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=kvaughan?_prettyPrint=true'
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=kvaughan?_fields=_id,ds-privilege-name',
        method: 'PATCH',
        credentials: 'uid=admin:password',
        body: [{
            "operation": "add",
            "field": "ds-privilege-name",
            "value": "password-reset"
        }]
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: assign the password-reset privilege', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: action-password-reset.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

jwt = utils.authenticate('uid=admin', 'password')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
patch = [{
    'operation': 'add',
    'field': 'ds-privilege-name',
    'value': 'password-reset'
}]
response = requests.patch(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=kvaughan',
    headers=headers,
    json=patch,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, action-password-reset.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('uid=admin', 'password')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_fields': '_id,ds-privilege-name' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = [{
    "operation" => "add",
    "field" => "ds-privilege-name",
    "value" => "password-reset"
}]
response = hdap.patch do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=kvaughan'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, action-password-reset.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Use HTTPS POST with `_action=resetPassword` in the query string and an empty JSON document (`{}`) as the POST data.

On success, the HTTP status code is 200 OK. The response body is a JSON resource with a `generatedPassword` containing the new password:

Curl

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header "Content-Type: application/json" \
 --data '{}' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_action=resetPassword'
{"generatedPassword":"<new-password>"}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen?_action=resetPassword&dryRun=true',
        method: 'POST',
        body: {}
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: dry-run reset password', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: action-reset-password.js, utils.js

Remove the `dryRun` parameter to perform the operation.

Python

```
#!/usr/bin/env python3

import requests
import utils

body = {}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = { '_action': 'resetPassword', 'dryRun': True }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    json=body,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, action-reset-password.py

Remove the `dryRun` parameter to perform the operation.

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_action': 'resetPassword', 'dryRun': true }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.post do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=bjensen'
    h.body = JSON.generate('{}')
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, action-reset-password.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Remove the `dryRun` parameter to perform the operation.

The password administrator must communicate the generated password to the user.

Use this in combination with a password policy to force the user to change their password again after a reset.

Account usability action

Use the `accountUsability` action to get details about a user's ability to authenticate.

- The action depends on the LDAP Account usability control, which has OID `1.3.6.1.4.1.42.2.27.9.5.8`.
- The password administrator must have access to use the LDAP control.

Try the `accountUsability` action:

1. Grant the password administrator permission to use the 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:

Curl

```
$ curl \
 --request POST \
 --user uid=admin:password \
 --cacert ca-cert.pem \
 --header 'Content-Type: application/json' \
 --data '{
  "_id": "dc=com/dc=example/cn=Lockout%20with%20max%20age%20and%20grace%20logins",
  "objectClass": ["top", "subentry", "ds-pwp-password-policy"],
  "cn": ["Lockout with max age and grace logins"],
  "ds-pwp-default-password-storage-scheme": ["PBKDF2-HMAC-SHA256"],
  "ds-pwp-grace-login-count": 3,
  "ds-pwp-lockout-duration": "5 m",
  "ds-pwp-lockout-failure-count": 3,
  "ds-pwp-lockout-failure-expiration-interval": "10 m",
  "ds-pwp-max-password-age": "30 d",
  "ds-pwp-password-attribute": "userPassword",
  "subtreeSpecification": { "base": "ou=people", "filter": "/uid eq \"bjensen\"" }
 }' \
 'https://localhost:8443/hdap/dc=com/dc=example?_action=create'
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example?_action=create',
        credentials: 'uid=admin:password',
        method: 'POST',
        body: {
            "_id": "dc=com/dc=example/cn=Lockout%20with%20max%20age%20and%20grace%20logins",
            "objectClass": ["top", "subentry", "ds-pwp-password-policy"],
            "cn": ["Lockout with max age and grace logins"],
            "ds-pwp-default-password-storage-scheme": ["PBKDF2-HMAC-SHA256"],
            "ds-pwp-grace-login-count": 3,
            "ds-pwp-lockout-duration": "5 m",
            "ds-pwp-lockout-failure-count": 3,
            "ds-pwp-lockout-failure-expiration-interval": "10 m",
            "ds-pwp-max-password-age": "30 d",
            "ds-pwp-password-attribute": "userPassword",
            "subtreeSpecification": { "base": "ou=people", "filter": "/uid eq \"bjensen\"" }
        }
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: add password policy', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: action-add-password-policy.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

body = {
    '_id': 'dc=com/dc=example/cn=Lockout%20with%20max%20age%20and%20grace%20logins',
    'objectClass': ['top', 'subentry', 'ds-pwp-password-policy'],
    'cn': ['Lockout with max age and grace logins'],
    'ds-pwp-default-password-storage-scheme': ['PBKDF2-HMAC-SHA256'],
    'ds-pwp-grace-login-count': 3,
    'ds-pwp-lockout-duration': '5 m',
    'ds-pwp-lockout-failure-count': 3,
    'ds-pwp-lockout-failure-expiration-interval': '10 m',
    'ds-pwp-max-password-age': '30 d',
    'ds-pwp-password-attribute': 'userPassword',
    'subtreeSpecification': { "base": "ou=people", "filter": "/uid eq \"bjensen\"" }
}
jwt = utils.authenticate('uid=admin', 'password')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example',
    headers=headers,
    json=body,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, utils.py, action-add-password-policy.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('uid=admin', 'password')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = {
    "_id" => "dc=com/dc=example/cn=Lockout%20with%20max%20age%20and%20grace%20logins",
    "objectClass" => ["top", "subentry", "ds-pwp-password-policy"],
    "cn" => ["Lockout with max age and grace logins"],
    "ds-pwp-default-password-storage-scheme" => ["PBKDF2-HMAC-SHA256"],
    "ds-pwp-grace-login-count" => 3,
    "ds-pwp-lockout-duration" => "5 m",
    "ds-pwp-lockout-failure-count" => 3,
    "ds-pwp-lockout-failure-expiration-interval" => "10 m",
    "ds-pwp-max-password-age" => "30 d",
    "ds-pwp-password-attribute" => "userPassword",
    "subtreeSpecification" => { "base": "ou=people", "filter": "/uid eq \"bjensen\"" }
}
response = hdap.post do |h|
    h.path = 'dc=com/dc=example'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, action-add-password-policy.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

3. Produce some account usability information on a user account:

Curl

```
$ curl \
 --user dc=com/dc=example/ou=People/uid=bjensen:wrong-password \
 --cacert ca-cert.pem \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=_id'

$ curl \
 --user dc=com/dc=example/ou=People/uid=bjensen:wrong-password \
 --cacert ca-cert.pem \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=_id'

$ curl \
 --user dc=com/dc=example/ou=People/uid=bjensen:wrong-password \
 --cacert ca-cert.pem \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=_id'
```

JavaScript

```
const { doRequest, getOptions } = require('./utils')

const options = getOptions({
    path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=_id',
    credentials: 'dc=com/dc=example/ou=People/uid=bjensen:wrong-password'
})

doRequest('HDAP: Basic auth with wrong password', options)
    .then(response => { console.log(response) })
    .catch(error => { console.error(error) })
    .finally(() => {
        doRequest('HDAP: Basic auth with wrong password', options)
            .then(response => { console.log(response) })
            .catch(error => { console.error(error) })
            .finally(() => {
                doRequest('HDAP: Basic auth with wrong password', options)
                    .then(response => { console.log(response) })
                    .catch(error => { console.error(error) })
            })
    })
```

Source files for this sample: action-lock-bjensen.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
from requests.auth import HTTPBasicAuth
import utils

for i in range(3):
    response = requests.get(
        f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
        auth=HTTPBasicAuth('dc=com/dc=example/ou=People/uid=bjensen', 'wrong-password'),
        verify=utils.ca_pem)
    print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, action-lock-bjensen.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('', '')
options = { ca_file: utils.ca_pem }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, :basic, 'dc=com/dc=example/ou=People/uid=bjensen', 'wrong-password'
end
3.times do
    response = hdap.get('dc=com/dc=example/ou=People/uid=bjensen')
    puts "Status code: #{response.status}\nJSON: #{response.body}"
end
```

Source files for this sample: utils.rb, action-lock-bjensen.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

4. Use the action to get account usability information:

Curl

```
$ curl \
 --request POST \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --cacert ca-cert.pem \
 --data '{}' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_action=accountUsability'
{"status":"locked","unlockIn":299}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen?_action=accountUsability&dryRun=true',
        method: 'POST',
        body: {}
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: dry-run check account usability', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: action-account-usability.js, utils.js

Remove the `dryRun` parameter to perform the operation.

Python

```
#!/usr/bin/env python3

import requests
import utils

body = {}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = { '_action': 'accountUsability' }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    json=body,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, action-account-usability.py

Remove the `dryRun` parameter to perform the operation.

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_action': 'accountUsability' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |
f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.post do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=bjensen'
    h.body = JSON.generate('{}')
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, action-account-usability.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Remove the `dryRun` parameter to perform the operation.

The JSON response can contain the following fields. The `status` property is always present. The other fields are present if they apply:

```
{
  "status": "string",              // One of "disabled", "locked", "passwordExpired",
                                   // "mustChangePassword", or "valid"
  "unlockIn": number,              // Seconds until locked account is unlocked
  "graceLoginsRemaining": number,  // Number of remaining authentications allowed with
                                   // an expired password
  "passwordExpiresIn": number,     // Seconds until password expires
}
```

Get JSON schema

Use the `schema` action to get the JSON schema for a resource.

Perform an HTTP POST with `_action=schema` in the query string and an empty JSON document (`{}`) as the POST data:

Curl

```
$ curl \
 --request POST \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --cacert ca-cert.pem \
 --data '{}' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_action=schema&_prettyPrint=true'
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen?_action=schema',
        method: 'POST',
        body: JSON.stringify({})
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: get schema', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: action-get-schema.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

body = {}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = { '_action': 'schema' }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    json=body,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, action-get-schema.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_action': 'schema' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.post do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=bjensen'
    h.body = JSON.generate('{}')
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, action-get-schema.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

```
{
  "type" : "object",
  "properties" : {
    "objectClass" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      },
      "const" : [ "top", "person", "cos", "oauth2TokenObject", "organizationalPerson", "inetOrgPerson",
"posixAccount" ]
    },
    "sn" : {
      "supertype" : "name",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "cn" : {
      "supertype" : "name",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "telephoneNumber" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "seeAlso" : {
      "supertype" : "distinguishedName",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string",
        "format" : "json-pointer"
      }
    },
    "userPassword" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "description" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "classOfService" : {
      "type" : "string"
```

```
      },
      "diskQuota" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "mailQuota" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "oauth2Token" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "object"
        }
      },
      "telexNumber" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "teletexTerminalIdentifier" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "ou" : {
        "supertype" : "name",
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "internationaliSDNNumber" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "registeredAddress" : {
        "supertype" : "postalAddress",
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "array",
          "items" : {
            "type" : "string"
          }
        }
      },
```

```
      "title" : {
        "supertype" : "name",
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "facsimileTelephoneNumber" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "x121Address" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "postOfficeBox" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "street" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "physicalDeliveryOfficeName" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "destinationIndicator" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "postalAddress" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "array",
          "items" : {
            "type" : "string"
          }
        }
      },
      "st" : {
        "supertype" : "name",
```

```
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "preferredDeliveryMethod" : {
      "type" : "string"
    },
    "postalCode" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "l" : {
      "supertype" : "name",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "preferredLanguage" : {
      "description" : "preferred written or spoken language for a person",
      "type" : "string"
    },
    "departmentNumber" : {
      "description" : "identifies a department within an organization",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "o" : {
      "supertype" : "name",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "carLicense" : {
      "description" : "vehicle license or registration plate",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "secretary" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string",
        "format" : "json-pointer"
      }
    },
    "employeeType" : {
      "description" : "type of employment for a person",
```

```
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "homePhone" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "pager" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "employeeNumber" : {
        "description" : "numerically identifies an employee within an organization",
        "type" : "string"
      },
      "mobile" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "userPKCS12" : {
        "description" : "PKCS #12 PFX PDU for exchange of personal identity information",
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string",
          "contentEncoding" : "base64"
        }
      },
      "userCertificate" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string",
          "contentEncoding" : "base64"
        }
      },
      "businessCategory" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "userSMIMECertificate" : {
        "description" : "PKCS#7 SignedData used to support S/MIME",
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string",
          "contentEncoding" : "base64"
```

```
      }
    },
    "mail" : {
      "description" : "The email address, including internationalized addresses (changed from the standard which only
allowed ascii)",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "roomNumber" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "audio" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string",
        "contentEncoding" : "base64"
      }
    },
    "initials" : {
      "supertype" : "name",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "manager" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string",
        "format" : "json-pointer"
      }
    },
    "photo" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string",
        "contentEncoding" : "base64"
      }
    },
    "givenName" : {
      "supertype" : "name",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "homePostalAddress" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
```

```
        "type" : "array",
        "items" : {
          "type" : "string"
        }
      }
    },
    "displayName" : {
      "description" : "preferred name of a person to be used when displaying entries",
      "type" : "string"
    },
    "labeledURI" : {
      "description" : "Uniform Resource Identifier with optional label",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "jpegPhoto" : {
      "description" : "a JPEG image",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string",
        "contentEncoding" : "base64"
      }
    },
    "x500UniqueIdentifier" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "uid" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "homeDirectory" : {
      "description" : "The absolute path to the home directory",
      "type" : "string"
    },
    "gidNumber" : {
      "description" : "An integer uniquely identifying a group in an administrative domain",
      "type" : "integer"
    },
    "uidNumber" : {
      "description" : "An integer uniquely identifying a user in an administrative domain",
      "type" : "integer"
    },
    "loginShell" : {
      "description" : "The path to the login shell",
      "type" : "string"
    },
    "authPassword" : {
      "description" : "password authentication information",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
```

```
          "type" : "string"
        }
      },
      "gecos" : {
        "description" : "The GECOS field; the common name",
        "type" : "string"
      }
    },
    "requiredProperties" : [ "cn", "gidNumber", "homeDirectory", "objectClass", "sn", "uid", "uidNumber" ],
    "additionalProperties" : false
  }
```

If you haven't yet created a resource, you get the JSON schema for the resource to create from the parent resource by specifying the LDAP object classes for the new resource. Use the `schema` action and a comma-separated list of object classes as the value of an `objectClasses` parameter:

Curl

```
$ curl \
 --request POST \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --cacert ca-cert.pem \
 --data '{}' \
'https://localhost:8443/hdap/dc=com/dc=example/ou=People?
_action=schema&objectClasses=person,posixAccount&_prettyPrint=true'
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People?_action=schema&objectClasses=person,posixAccount',
        method: 'POST',
        body: JSON.stringify({})
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: get schema for new resource', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: action-get-new-schema.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

body = {}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = { '_action': 'schema', 'objectClasses': 'person,posixAccount' }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    json=body,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, action-get-new-schema.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_action': 'schema', 'objectClasses': 'person,posixAccount' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.post do |h|
    h.path = 'dc=com/dc=example/ou=People'
    h.body = JSON.generate('{}')
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, action-get-new-schema.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

```
{
  "type" : "object",
  "properties" : {
    "objectClass" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      },
      "const" : [ "top", "person", "posixAccount" ]
    },
    "sn" : {
      "supertype" : "name",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "cn" : {
      "supertype" : "name",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "telephoneNumber" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "seeAlso" : {
      "supertype" : "distinguishedName",
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string",
        "format" : "json-pointer"
      }
    },
    "userPassword" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "description" : {
      "type" : "array",
      "uniqueItems" : true,
      "items" : {
        "type" : "string"
      }
    },
    "homeDirectory" : {
      "description" : "The absolute path to the home directory",
      "type" : "string"
```

```
      },
      "gidNumber" : {
        "description" : "An integer uniquely identifying a group in an administrative domain",
        "type" : "integer"
      },
      "uidNumber" : {
        "description" : "An integer uniquely identifying a user in an administrative domain",
        "type" : "integer"
      },
      "uid" : {
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "loginShell" : {
        "description" : "The path to the login shell",
        "type" : "string"
      },
      "authPassword" : {
        "description" : "password authentication information",
        "type" : "array",
        "uniqueItems" : true,
        "items" : {
          "type" : "string"
        }
      },
      "gecos" : {
        "description" : "The GECOS field; the common name",
        "type" : "string"
      }
    },
    "requiredProperties" : [ "cn", "gidNumber", "homeDirectory", "objectClass", "sn", "uid", "uidNumber" ],
    "additionalProperties" : false
  }
```

You can also read the schema for an individual field or object class directly as described in the reference for HDAP schema.

Rename a resource

Use the `rename` action to change a resource's `_id`. This effectively moves the resource.

Perform an HTTP POST with `_action=rename` in the query string and the `newId` in the POST data:

Curl

```
$ curl \
 --request POST \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --cacert ca-cert.pem \
 --data '{"newId": "dc=com/dc=example/ou=People/uid=sjensen"}' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=scarter?_action=rename&_fields=uid'
{"uid":["scarter","sjensen"],"_id":"dc=com/dc=example/ou=People/uid=sjensen","_rev":"<revision>"}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=scarter?_action=rename&_fields=uid',
        method: 'POST',
        body: { "newId": "dc=com/dc=example/ou=People/uid=sjensen" }
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: rename a resource', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: action-rename.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

body = { 'newId': 'dc=com/dc=example/ou=People/uid=sjensen' }
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = { '_action': 'rename', '_fields': 'uid' }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=scarter',
    headers=headers,
    json=body,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, action-rename.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_action': 'rename', '_fields': 'uid' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = { "newId" => "dc=com/dc=example/ou=People/uid=sjensen" }
response = hdap.post do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=scarter'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, action-rename.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

To remove the existing RDN value, `uid=scarter` in this example, use the `deleteOldRdn=true` parameter. An LDAP 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 HDAP, the last path element of the `_id` holds the field value DS deletes when `deleteOldRdn=true`. In `dc=com/dc=example/ou=People/uid=sjensen`, it's `uid=sjensen`. In `dc=com/dc=example/ou=People`, it's `ou=People`.

Note

When you rename a resource with child resources, DS renames all the child resources, too.
For example, if you rename `dc=com/dc=example/ou=People` to `dc=com/dc=example/ou=Subscribers`, `dc=com/dc=example/ou=People/uid=sjensen` becomes `dc=com/dc=example/ou=Subscribers/uid=sjensen`.
DS directory servers support this operation only for moving resources in the same backend, under the same path.
Depending on the number of resources moved, this can be a resource-intensive operation.

Query

Note

Examples in this documentation depend on features activated in the `ds-evaluation` setup profile. For details, refer to Learn about the evaluation setup profile.
The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
```

To search, use HTTP GET with at least a `_queryFilter=<filter-expression>` parameter and other applicable parameters as necessary, such as `scope`. For reference details, refer to:

- Query
- Query parameters
- Filter expressions

Example query filters

The following table shows LDAP search filters and corresponding query filter expressions:

LDAP Filter	REST Filter
`(&)`	`_queryFilter=true`
`(uid=*)`	`_queryFilter=uid+pr`
`(uid=bjensen)`	`_queryFilter=uid+eq+'bjensen'`

LDAP Filter	REST Filter
`(uid=*jensen*)`	`_queryFilter=uid+co+'jensen'`
`(uid=jensen*)`	`_queryFilter=uid+sw+'jensen'`
`(&(uid=*jensen*)(cn=babs*))`	`_queryFilter=(uid+co+'jensen'and+cn+sw'babs')`
`(|(uid=*jensen*)(cn=sam*))`	`_queryFilter=(uid+co+'jensen'or+cn+sw'sam')`
`(!(uid=*jensen*))`	`_queryFilter=!(uid+co+'jensen')`
`(uid<=jensen)`	`_queryFilter=uid+le+'jensen'`
`(uid>=jensen)`	`_queryFilter=uid+ge+'jensen'`

For details on which index to configure for a specific filter, refer to Necessary indexes.

Build query filters

For query operations, the <filter-expression> has the following building blocks. You must URL-encode the filter expressions. This page shows them without URL-encoding to make them easier to read.

In filter expressions, the simplest <json-pointer> is a JSON field name. A <json-pointer> can also reference nested elements. For details, read RFC 6901, JavaScript Object Notation (JSON) Pointer.

Important

The `co` (contains) and `sw` (starts with) query filter expressions match part of a value.
HDAP does not support `co` and `sw` to match values for:

- Binary fields, such as photos or digital certificates (binary attributes in LDAP)
- `_id` and JSON fields whose values are the `_id`s of other resources, such as `manager` (DN attributes in LDAP)
- Time fields, such as a last login time (generalized time attributes in LDAP)

Comparison expressions

Build filters using the following comparison expressions:

`<json-pointer> eq <json-value>`

Matches when the pointer equals the value.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=mail+eq+'bjensen@example.com'" \
 --data '_fields=cn' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
    "_rev" : "<revision>",
    "cn" : [ "Barbara Jensen", "Babs Jensen" ]
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: "/hdap/dc=com/dc=example/ou=People?
_queryFilter=mail+eq+'bjensen@example.com'&_fields=cn&scope=sub"
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query eq', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-eq.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': 'mail eq "bjensen@example.com"',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-eq.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": "mail eq 'bjensen@example.com'",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-eq.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

<json-pointer> co <json-value>

Matches when the pointer contains the value.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=mail+co+'jensen'" \
 --data '_fields=cn' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=ajensen",
    "_rev" : "<revision>",
    "cn" : [ "Allison Jensen" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
    "_rev" : "<revision>",
    "cn" : [ "Barbara Jensen", "Babs Jensen" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=gjensen",
    "_rev" : "<revision>",
    "cn" : [ "Gern Jensen" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=jjensen",
    "_rev" : "<revision>",
    "cn" : [ "Jody Jensen" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=kjensen",
    "_rev" : "<revision>",
    "cn" : [ "Kurt Jensen" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=rjensen",
    "_rev" : "<revision>",
    "cn" : [ "Richard Jensen" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=tjensen",
    "_rev" : "<revision>",
    "cn" : [ "Ted Jensen" ]
  } ],
  "resultCount" : 7,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: "/hdap/dc=com/dc=example/ou=People?
_queryFilter=mail+co+'jensen'&_fields=cn&scope=sub"
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query co', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-co.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': 'mail co "jensen"',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-co.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": "mail co 'jensen'",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-co.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

`<json-pointer> sw <json-value>`

Matches when the pointer starts with the value.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=mail+sw+'ab'" \
 --data '_fields=cn' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=abarnes",
     "_rev" : "<revision>",
    "cn" : [ "Anne-Louise Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=abergin",
    "_rev" : "<revision>",
    "cn" : [ "Andy Bergin" ]
  } ],
  "resultCount" : 2,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: "/hdap/dc=com/dc=example/ou=People?_queryFilter=mail+sw+'ab'&_fields=cn&scope=sub"
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query sw', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-sw.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': 'mail sw "ab"',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-sw.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": "mail sw 'ab'",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-eq.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

`<json-pointer> lt <json-value>`

Matches when the pointer is less than the value.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=mail+lt+'ac'" \
 --data '_fields=cn' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=abarnes",
    "_rev" : "<revision>",
    "cn" : [ "Anne-Louise Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=abergin",
    "_rev" : "<revision>",
    "cn" : [ "Andy Bergin" ]
  } ],
  "resultCount" : 2,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: "/hdap/dc=com/dc=example/ou=People?_queryFilter=mail+lt+'ac'&_fields=cn&scope=sub"
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query lt', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-lt.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': 'mail lt "ac"',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-lt.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": "mail lt 'ac'",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-lt.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

`<json-pointer> le <json-value>`

Matches when the pointer is less than or equal to the value.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=mail+le+'ad'" \
 --data '_fields=cn' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=abarnes",
    "_rev" : "<revision>",
    "cn" : [ "Anne-Louise Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=abergin",
    "_rev" : "<revision>",
    "cn" : [ "Andy Bergin" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=achassin",
    "_rev" : "<revision>",
    "cn" : [ "Ashley Chassin" ]
  } ],
  "resultCount" : 3,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: "/hdap/dc=com/dc=example/ou=People?_queryFilter=mail+le+'ad'&_fields=cn&scope=sub"
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query le', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-le.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': 'mail le "ad"',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-le.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": "mail le 'ad'",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-le.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

`<json-pointer> gt <json-value>`

Matches when the pointer is greater than the value.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=mail+gt+'wa'" \
 --data '_fields=cn' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=wlutz",
    "_rev" : "<revision>",
    "cn" : [ "Wendy Lutz" ]
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: "/hdap/dc=com/dc=example/ou=People?_queryFilter=mail+gt+'wa'&_fields=cn&scope=sub"
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query gt', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-gt.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': 'mail gt "wa"',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-gt.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": "mail gt 'wa'",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-gt.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

`<json-pointer> ge <json-value>`

Matches when the pointer is greater than or equal to the value.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=mail+ge+'va'" \
 --data '_fields=cn' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=wlutz",
    "_rev" : "<revision>",
    "cn" : [ "Wendy Lutz" ]
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: "/hdap/dc=com/dc=example/ou=People?_queryFilter=mail+ge+'va'&_fields=cn&scope=sub"
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query ge', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-ge.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': 'mail ge "va"',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-ge.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": "mail ge 'va'",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-ge.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Presence expression

`<json-pointer> pr` matches a resource where the pointer is present.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=cn+pr" \
 --data '_fields=cn' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=Groups'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=groups/cn=Accounting%20Managers",
    "_rev" : "<revision>",
    "cn" : [ "Accounting Managers" ]
  }, {
    "_id" : "dc=com/dc=example/ou=Groups/cn=Directory%20Administrators",
    "_rev" : "<revision>",
    "cn" : [ "Directory Administrators" ]
  }, {
    "_id" : "dc=com/dc=example/ou=groups/cn=HR%20Managers",
    "_rev" : "<revision>",
    "cn" : [ "HR Managers" ]
  }, {
    "_id" : "dc=com/dc=example/ou=groups/cn=PD%20Managers",
    "_rev" : "<revision>",
    "cn" : [ "PD Managers" ]
  }, {
    "_id" : "dc=com/dc=example/ou=groups/cn=QA%20Managers",
    "_rev" : "<revision>",
    "cn" : [ "QA Managers" ]
  }, {
    "_id" : "dc=com/dc=example/ou=Groups/ou=Self%20Service/cn=Carpoolers",
    "_rev" : "<revision>",
    "cn" : [ "Carpoolers" ]
  } ],
  "resultCount" : 6,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=Groups?_queryFilter=cn+pr&_fields=cn&scope=sub'
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query pr', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-pr.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': 'cn pr',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=Groups',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-pr.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": "cn pr",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |
f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=Groups')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-pr.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Literal expressions

`true` matches any resource in scope.

`false` matches no resources.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=true" \
 --data '_fields=_id' \
 --data '_prettyPrint=true' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=Groups'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=groups/cn=Accounting%20Managers",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=Groups/cn=Directory%20Administrators",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=groups/cn=HR%20Managers",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=groups/cn=PD%20Managers",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=groups/cn=QA%20Managers",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=Groups/ou=Self%20Service",
    "_rev" : "<revision>"
  } ],
  "resultCount" : 6,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=Groups?_queryFilter=true&_fields=cn&scope=sub'
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query true', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-true.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': True,
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=Groups',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-true.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": true,
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |
f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=Groups')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-true.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Complex expressions

Combine expressions using boolean operators `and`, `or`, and `!` (not), and by using parentheses `(<filter-expression>)` with group expressions.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=mail+co+'jensen'+and+givenName+sw+'Al'" \
 --data '_fields=cn' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=ajensen",
    "_rev" : "<revision>",
    "cn" : [ "Allison Jensen" ]
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const filter = encodeURIComponent("mail co 'jensen' and givenName sw 'Al'")
    const options = getOptions({
        path: `/hdap/dc=com/dc=example/ou=People?_queryFilter=${filter}&_fields=cn&scope=sub`
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: complex query', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-complex.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': 'mail co "jensen" and givenName sw "Al"',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-complex.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": "mail co 'jensen' and givenName sw 'Al'",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |
f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-complex.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Graph-like queries

Collective attributes provide a mechanism to inherit LDAP attributes from other entries. HDAP relies on this mechanism for graph-like queries.

The `ds-evaluation` setup profile uses collective attributes to inherit LDAP attribute values for street address from location and quota settings from class of service.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=(mail+co+'jensen'+and+l+eq+'San+Francisco'+and+classOfService+eq+'bronze')" \
 --data '_fields=diskQuota,mailQuota,street' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
    "_rev" : "<revision>",
    "street" : [ "201 Mission Street Suite 2900" ],
    "mailQuota" : [ "1 GB" ],
    "diskQuota" : [ "10 GB" ]
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const filter = "mail+co+'jensen'+and+l+eq+'San+Francisco'+and+classOfService+eq+'bronze'"
    const options = getOptions({
        path: `/hdap/dc=com/dc=example/ou=People?_queryFilter=${filter}
&_fields=diskQuota,mailQuota,street&scope=sub`
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: graph query', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-graph.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'diskQuota,mailQuota,street',
    '_queryFilter': 'mail co "jensen" and l eq "San Francisco" and classOfService eq "bronze"',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-graph.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "diskQuota,mailQuota,street",
    "_queryFilter": "mail co 'jensen' and l eq 'San Francisco' and classOfService eq 'bronze'",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-graph.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Configure collective attributes as necessary for your graph-like queries.

Queries and JSON attributes

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}]}`. As a result, HDAP filter expressions can target JSON fields in indexed JSON attribute objects.

HDAP query filter expressions support the grouping operators described in RFC 7644, section *3.4.2.2. Filtering*, Table 5: *Grouping Operators*. In other words, HDAP query filter expressions can use *complex attribute filter grouping*, with brackets (`[]`) to group expressions in the filter.

Complex attribute filter grouping lets filter expressions target array objects. This search finds an entry with a `json` attribute containing an `array` of objects:

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data-urlencode "_queryFilter=(json/array[x eq 1] and json/array[y eq 4])" \
 --data '_fields=json' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=abarnes",
    "_rev" : "<revision>",
    "json" : [ {
      "array" : [ {
        "x" : 1,
        "y" : 2
      }, {
        "x" : 3,
        "y" : 4
      } ]
    } ]
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const filter = encodeURIComponent('(json/array[x eq 1] and json/array[y eq 4])')
    const options = getOptions({
        path: `/hdap/dc=com/dc=example/ou=People?_queryFilter=${filter}&_fields=json&scope=sub`
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: JSON query', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-json.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'json',
    '_queryFilter': '(json/array[x eq 1] and json/array[y eq 4])',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-json.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "json",
    "_queryFilter": "(json/array[x eq 1] and json/array[y eq 4])",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-json.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

- 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}`.

Count child resources

Use the `_countOnly=true` query string parameter to get the number of child resources directly beneath a resource. You must also set `_queryFilter=true` and leave `scope=one` (default):

Curl

```
$ curl \
 --get \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --cacert ca-cert.pem \
 --header 'Accept-API-Version: protocol=2.2,resource=1.0' \
 --data '_countOnly=true' \
 --data '_queryFilter=true' \
 --data '_prettyPrint=true' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ ],
  "resultCount" : 100153,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "ESTIMATE",
  "totalPagedResults" : 100153,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People?_queryFilter=true&_countOnly=true',
    })
    const jwt = await authenticate(options)
    options.headers['Accept-API-Version'] = 'protocol=2.2,resource=1.0'
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query for count', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-count.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_countOnly': True,
    '_queryFilter': True
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = {
    'Content-Type': 'application/json',
    'Authorization': f'Bearer {jwt}',
    'Accept-API-Version': 'protocol=2.2,resource=1.0'
}
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-count.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_countOnly": true,
    "_queryFilter": true
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.headers['Accept-API-Version'] = 'protocol=2.2,resource=1.0'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-count.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Notice the search returns an empty `result` array. This feature relies on the `numSubordinates` virtual attribute.

Paged results

Get results one page at a time with these query string parameters:

- `_pageSize=<number>`
- `_pagedResultsCookie=<cookie>`

Curl

```
# Request five results per page and retrieve the first page:
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=objectClass+eq+'posixAccount'" \
 --data '_pageSize=5' \
 --data '_fields=_id' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=abarnes",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=abergin",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=achassin",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=ahall",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=ahel",
    "_rev" : "<revision>"
  } ],
  "resultCount" : 5,
  "pagedResultsCookie" : "AAAAAAAAABE=",
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const filter = "objectClass+eq+'posixAccount'"
    const options = getOptions({
        path: `/hdap/dc=com/dc=example/ou=People?_queryFilter=${filter}&_pageSize=5&_fields=_id&scope=sub`
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query first five paged results', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-paged-first-five.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': '_id',
    '_queryFilter': 'objectClass eq "posixAccount"',
    '_pageSize': 5,
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-paged-first-five.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "_id",
    "_queryFilter": "objectClass eq 'posixAccount'",
    "_pageSize": 5,
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-paged-first-five.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Curl

```
# Provide the cookie to request the next five results:
$ export COOKIE=$(cut -d \" -f 4 <(grep pagedResultsCookie \
 <(curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=objectClass+eq+'posixAccount'" \
 --data '_pageSize=5' \
 --data '_fields=_id' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People')))
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=objectClass+eq+'posixAccount'" \
 --data '_pageSize=5' \
 --data "_pagedResultsCookie=$COOKIE" \
 --data '_fields=_id' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=ahunter",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=ajensen",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=aknutson",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=alangdon",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=alutz",
    "_rev" : "<revision>"
  } ],
  "resultCount" : 5,
  "pagedResultsCookie" : "AAAAAAAAABE=",
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const filter = "objectClass+eq+'posixAccount'"
    const options = getOptions({
        path: `/hdap/dc=com/dc=example/ou=People?_queryFilter=${filter}&_pageSize=5&_fields=_id&scope=sub`
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    let response = await doRequest('HDAP: query first five paged results', options)
    console.log(response)
    const cookie = JSON.parse(response.data).pagedResultsCookie
    options.path += `&_pagedResultsCookie=${cookie}`
    response = await doRequest('HDAP: query next five paged results', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-paged-next-five.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

resource = f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People'
params = {
    '_fields': '_id',
    '_queryFilter': 'objectClass eq "posixAccount"',
    '_pageSize': 5,
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    resource,
    headers=headers,
    params=params,
    verify=utils.ca_pem)

cookie = response.json()['pagedResultsCookie']
params['_pagedResultsCookie'] = cookie

response = requests.get(
    resource,
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-paged-next-five.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "_id",
    "_queryFilter": "objectClass eq 'posixAccount'",
    "_pageSize": 5,
    "scope": "sub"
}
resource = 'dc=com/dc=example/ou=People'
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
cookie = JSON.parse(hdap.get(resource).body, symbolize_names: true)[:pagedResultsCookie]

query['_pagedResultsCookie'] = cookie
hdap.params = query
response = hdap.get(resource)

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-paged-next-five.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

Notice the following features of the responses:

- `"totalPagedResultsPolicy" : "NONE"` means HDAP did not calculate the counts.
- `"remainingPagedResults" : -1` means HDAP did not count the remaining results.

  HDAP never counts `remainingPagedResults` because it would require a potentially costly calculation to determine the current position in the total result set.
- `"totalPagedResults" : -1` means HDAP did not count the total results.

When the query has the following characteristics, the response contains an estimated `totalPagedResults` count:

- The request specifies the policy using the parameter `_totalPagedResultsPolicy=ESTIMATE`.
- The `_pageSize` parameter is an integer greater than zero.
- The LDAP search for the query is indexed.

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=mail+co+'jensen'" \
 --data '_pageSize=2' \
 --data '_totalPagedResultsPolicy=ESTIMATE' \
 --data '_fields=_id' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=ajensen",
    "_rev" : "<revision>"
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
    "_rev" : "<revision>"
  } ],
  "resultCount" : 2,
  "pagedResultsCookie" : "AAAAAAAAAEI=",
  "totalPagedResultsPolicy" : "ESTIMATE",
  "totalPagedResults" : 7,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const filter = "mail+co+'jensen'"
    const pageParams = '_pageSize=2&_totalPagedResultsPolicy=ESTIMATE'
    const options = getOptions({
        path: `/hdap/dc=com/dc=example/ou=People?_queryFilter=${filter}&${pageParams}&_fields=_id&scope=sub`
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: paged query', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-paged.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': '_id',
    '_queryFilter': 'mail co "jensen"',
    '_pageSize': 2,
    '_totalPagedResultsPolicy': 'ESTIMATE',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-paged.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "_id",
    "_queryFilter": "mail co 'jensen'",
    "_pageSize": 2,
    "_totalPagedResultsPolicy": "ESTIMATE",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-paged.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

The estimated number of results can be useful, for example, when the LDAP search uses a big index or a VLV index and the total number of results is large.

Server-side sort

Use the `_sortKeys` parameter to have HDAP sort the query results based on one or more fields in the resources.

The following example sorts results in reverse order by given name (`-givenName`):

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --data "_queryFilter=sn+co+'barnes'" \
 --data '_sortKeys=-givenName' \
 --data '_fields=cn' \
 --data '_prettyPrint=true' \
 --data 'scope=sub' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People'
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const filter = "sn+co+'barnes'"
    const options = getOptions({
        path: `/hdap/dc=com/dc=example/ou=People?_queryFilter=${filter}&_sortKeys=-
givenName&_fields=cn&scope=sub`
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query with server-side sort', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: query-sss.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': 'cn',
    '_queryFilter': 'sn co "barnes"',
    '_sortKeys': '-givenName',
    'scope': 'sub'
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, query-sss.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "cn",
    "_queryFilter": "sn co 'barnes'",
    "_sortKeys": "-givenName",
    "scope": "sub"
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, query-sss.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

```
{
  "result" : [ {
    "_id" : "dc=com/dc=example/ou=People/uid=user.94561",
    "_rev" : "<revision>",
    "cn" : [ "Atsushi Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.81142",
    "_rev" : "<revision>",
    "cn" : [ "Atsuo Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.67723",
    "_rev" : "<revision>",
    "cn" : [ "Atmane Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.54304",
    "_rev" : "<revision>",
    "cn" : [ "Atlante Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.40885",
    "_rev" : "<revision>",
    "cn" : [ "Atlanta Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.27466",
    "_rev" : "<revision>",
    "cn" : [ "Atl-Sales Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.14047",
    "_rev" : "<revision>",
    "cn" : [ "Atl Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=user.628",
    "_rev" : "<revision>",
    "cn" : [ "Atique Barnes" ]
  }, {
    "_id" : "dc=com/dc=example/ou=People/uid=abarnes",
    "_rev" : "<revision>",
    "cn" : [ "Anne-Louise Barnes" ]
  } ],
  "resultCount" : 9,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```


- To specify multiple sort keys, use a comma-separated list of fields.
- The sort key fields you specify must exist in the result entries.

  You do not need to include the sort field(s) in the results.
- VLV for paged server-side sort shows an HDAP query using a browsing index.

HDAP stores the entire result set before sorting the results. If you expect a large result set for your search, use paged results to limit the performance cost and get results quickly.

Binary resources

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile. For details, refer to Learn about the evaluation setup profile.
> The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
> ```

Add a binary resource

1. Ensure the application has a resource to upload.

   For example, copy a JPEG photo `picture.jpg` to the current directory.

2. Encode binary resource as a base64-encoded JSON string:

   ```
   $ base64 encode --rawDataFile picture.jpg
   /9j/4AAQSkZJRgABAQEAYABgAAD/4QAWRXhpZgAASUkqAAgAAAAAAAAAAD/
   2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
   2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
   wAARCAABAAEDASIAAhEBAxEB/8QAFQABAQAAAAAAAAAAAAAAAAAAAAr/xAAUEAEAAAAAAAAAAAAAAAAAAAA/
   8QAFAEBAAAAAAAAAAAAAAAAAAAP/EABQRAQAAAAAAAAAAAAAAAAAAAD/2gAMAwEAAhEDEQA/AL+AAf/Z
   ```

3. Upload the binary resource as a base64-encoded JSON string.

   The following example updates Babs Jensen's resource to add or replace the profile photo:

   Curl

   ```
   $ curl \
    --request PUT \
    --cacert ca-cert.pem \
    --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
    --header 'Content-Type: application/json' \
    --data '{"photo": "/9j/4AAQSkZJRgABAQEAYABgAAD/4QAWRXhpZgAASUkqAAgAAAAAAAAAAD/
   2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
   2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
   wAARCAABAAEDASIAAhEBAxEB/8QAFQABAQAAAAAAAAAAAAAAAAAAAAr/xAAUEAEAAAAAAAAAAAAAAAAAAAA/
   8QAFAEBAAAAAAAAAAAAAAAAAAAP/EABQRAQAAAAAAAAAAAAAAAAAAAD/2gAMAwEAAhEDEQA/AL+AAf/Z"}' \
    'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen'
   ```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const photo = { "photo": "/9j/4AAQSkZJRgABAQEAYABgAAD/4QAWRXhpZgAASUkqAAgAAAAAAAAAAD/
2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
wAARCAABAAEDASIAAhEBAxEB/8QAFQABAQAAAAAAAAAAAAAAAAAAAAr/xAAUEAEAAAAAAAAAAAAAAAAAAAA/
8QAFAEBAAAAAAAAAAAAAAAAAAAAAP/EABQRAQAAAAAAAAAAAAAAAAAAAD/2gAMAwEAAhEDEQA/AL+AAf/Z" }
    const resource = '/hdap/dc=com/dc=example/ou=People/uid=bjensen'
    const options = getOptions({ method: 'PUT', path: resource, body: photo })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: add photo', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: binary-add.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

photo = { 'photo': '/9j/4AAQSkZJRgABAQEAYABgAAD/4QAWRXhpZgAASUkqAAgAAAAAAAAAAD/
2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
wAARCAABAAEDASIAAhEBAxEB/8QAFQABAQAAAAAAAAAAAAAAAAAAAAr/xAAUEAEAAAAAAAAAAAAAAAAAAAA/
8QAFAEBAAAAAAAAAAAAAAAAAAAAAP/EABQRAQAAAAAAAAAAAAAAAAAAAD/2gAMAwEAAhEDEQA/AL+AAf/Z' }
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.put(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    json=photo,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, binary-add.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
photo = { "photo" => "/9j/4AAQSkZJRgABAQEAYABgAAD/4QAWRXhpZgAASUkqAAgAAAAAAAAAD/
2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
wAARCAABAAEDASIAAhEBAxEB/8QAFQABAQAAAAAAAAAAAAAAAAAAAr/xAAUEAEAAAAAAAAAAAAAAAAAAA/
8QAFAEBAAAAAAAAAAAAAAAAAAAAP/EABQRAQAAAAAAAAAAAAAAAAAAD/2gAMAwEAAhEDEQA/AL+AAf/Z" }
response = hdap.put do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=bjensen'
    h.body = JSON.generate(photo)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, binary-add.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

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

```
$ curl \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=photo&_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
  "_rev" : "<revision>",
  "photo" : [ "/9j/4AAQSkZJRgABAQEAYABgAAD/4QAWRXhpZgAASUkqAAgAAAAAAAAAAD/
2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
2wBDAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/
wAARCAABAAEDASIAAhEBAxEB/8QAFQABAQAAAAAAAAAAAAAAAAAAAAr/xAAUEAEAAAAAAAAAAAAAAAAAAAAA/
8QAFAEBAAAAAAAAAAAAAAAAAAAAAP/EABQRAQAAAAAAAAAAAAAAAAAAAD/2gAMAwEAAhEDEQA/AL+AAf/Z" ]
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const resource = '/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=photo'
    const options = getOptions({ path: resource })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: read photo', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: binary-read.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = { '_fields': 'photo' }
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, binary-read.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
fields = { "_fields": "photo" }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: fields, ssl: options) do
|f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People/uid=bjensen')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, binary-read.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

HDAP and password policies

> **Note**
>
> Examples in this documentation depend on features activated in the `ds-evaluation` setup profile. For details, refer to Learn about the evaluation setup profile.
> The code samples demonstrate how to contact the server over HTTPS using the deployment CA certificate. Before trying the samples, 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
> ```

This example demonstrates how to add a subentry password policy with HDAP. Subentry password policies are replicated.

This example uses Kirsten Vaughan as a password administrator. Kirsten is a member of the `Directory Administrators` group.

1. Before trying this example, make sure the password administrator has the necessary access:
    1. Grant the `subentry-write` privilege to edit password policies:

Curl

```
$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --user uid=admin:password \
 --header 'Content-Type: application/json' \
 --data '[{
  "operation": "add",
  "field": "ds-privilege-name",
  "value": "subentry-write"
 }]' \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=kvaughan?_fields=_id,ds-privilege-
name'
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=kvaughan?_fields=_id,ds-privilege-name',
        method: 'PATCH',
        credentials: 'uid=admin:password',
        body: [{
            "operation": "add",
            "field": "ds-privilege-name",
            "value": "subentry-write"
        }]
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: assign the subentry-write privilege', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: pwp-subentry-write.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

patch = [{
    'operation': 'add',
    'field': 'ds-privilege-name',
    'value': 'subentry-write'
}]
jwt = utils.authenticate('uid=admin', 'password')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.patch(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=kvaughan',
    headers=headers,
    json=patch,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, pwp-subentry-write.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('uid=admin', 'password')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_fields': '_id,ds-privilege-name' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = [{
    "operation" => "add",
    "field" => "ds-privilege-name",
    "value" => "subentry-write"
}]
response = hdap.patch do |h|
    h.path = 'dc=com/dc=example/ou=People/uid=kvaughan'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, pwp-subentry-write.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

2. Grant access to manage password policies.

Curl

```
$ curl \
 --request PATCH \
 --cacert ca-cert.pem \
 --user uid=admin:password \
 --header 'Content-Type: application/json' \
 --url 'https://localhost:8443/hdap/dc=com/dc=example?_fields=_id,aci' \
 --data @- << JSON
[{
  "operation": "add",
  "field": "aci",
  "value": "(targetattr = \"pwdPolicySubentry||ds-pwp-password-policy-dn||ds-pwp-password-
validator||subtreeSpecification\")(version 3.0;acl \"Allow Administrators to manage user
password policies\";allow (all) (groupdn = \"ldap:///cn=Directory
Administrators,ou=Groups,dc=example,dc=com\");)"
}]
JSON
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example?_fields=_id,aci',
        method: 'PATCH',
        credentials: 'uid=admin:password',
        body: [{
            "operation": "add",
            "field": "aci",
            "value": "(targetattr = \"pwdPolicySubentry||ds-pwp-password-policy-dn||ds-pwp-
password-validator||subtreeSpecification\")(version 3.0;acl \"Allow Administrators to manage
user password policies\";allow (all) (groupdn = \"ldap:///cn=Directory
Administrators,ou=Groups,dc=example,dc=com\");)"
        }]
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: grant access to manage password policies', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: pwp-admin-access.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

patch = [{
    'operation': 'add',
    'field': 'aci',
    'value': '(targetattr = "pwdPolicySubentry||ds-pwp-password-policy-dn||ds-pwp-password-
validator||subtreeSpecification")(version 3.0;acl "Allow Administrators to manage user password
policies";allow (all) (groupdn = "ldap:///cn=Directory
Administrators,ou=Groups,dc=example,dc=com");)'
}]
jwt = utils.authenticate('uid=admin', 'password')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = {'_fields': '_id,aci' }
response = requests.patch(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example',
    headers=headers,
    json=patch,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, pwp-admin-access.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('uid=admin', 'password')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_fields': '_id,aci' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl:
options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = [{
    "operation" => "add",
    "field" => "aci",
    "value" => "(targetattr = \"pwdPolicySubentry||ds-pwp-password-policy-dn||ds-pwp-password-
validator||subtreeSpecification\")(version 3.0;acl \"Allow Administrators to manage user
password policies\";allow (all) (groupdn = \"ldap:///cn=Directory
Administrators,ou=Groups,dc=example,dc=com\");)"
}]
response = hdap.patch do |h|
    h.path = 'dc=com/dc=example'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, pwp-admin-access.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

2. Create and assign a subentry password policy as the password administrator:

Curl

```
$ curl \
 --request POST \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 --header 'Content-Type: application/json' \
 --url 'https://localhost:8443/hdap/dc=com/dc=example?
_action=create&_fields=*,subtreeSpecification&_prettyPrint=true' \
 --data @- << JSON
{
  "_id" : "dc=com/dc=example/cn=Replicated%20password%20policy",
  "objectClass" : [ "top", "subentry", "ds-pwp-password-policy", "ds-pwp-validator", "ds-pwp-length-
based-validator" ],
  "cn" : [ "Replicated password policy" ],
  "ds-pwp-default-password-storage-scheme" : [ "PBKDF2-HMAC-SHA512" ],
  "ds-pwp-length-based-min-password-length" : 8,
  "ds-pwp-password-attribute" : "userPassword",
  "subtreeSpecification": { "base": "ou=People", "filter": "/objectClass eq \"person\"" }
 }
JSON
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example?_action=create&_fields=*,subtreeSpecification',
        method: 'POST',
        body: {
            "_id": "dc=com/dc=example/cn=Replicated%20password%20policy",
            "objectClass": ["top", "subentry", "ds-pwp-password-policy", "ds-pwp-validator", "ds-pwp-
length-based-validator"],
            "cn": ["Replicated password policy"],
            "ds-pwp-default-password-storage-scheme": ["PBKDF2-HMAC-SHA512"],
            "ds-pwp-length-based-min-password-length": 8,
            "ds-pwp-password-attribute": "userPassword",
            "subtreeSpecification": { "base": "ou=People", "filter": "/objectClass eq \"person\"" }
        }
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: create password policy', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: pwp-add-policy.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

body = {
    '_id': 'dc=com/dc=example/cn=Replicated%20password%20policy',
    'objectClass': ['top', 'subentry', 'ds-pwp-password-policy', 'ds-pwp-validator', 'ds-pwp-length-
based-validator'],
    'cn': ['Replicated password policy'],
    'ds-pwp-default-password-storage-scheme': ['PBKDF2-HMAC-SHA512'],
    'ds-pwp-length-based-min-password-length': 8,
    'ds-pwp-password-attribute': 'userPassword',
    'subtreeSpecification': { "base": "ou=people", "filter": "/objectClass eq \"person\"" }
}
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
params = { '_fields': '*,subtreeSpecification' }
response = requests.post(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example',
    headers=headers,
    json=body,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, pwp-add-policy.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'
require 'json'

utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = { '_fields': '*,subtreeSpecification' }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |
f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
body = {
    "_id": "dc=com/dc=example/cn=Replicated%20password%20policy",
    "objectClass": ["top", "subentry", "ds-pwp-password-policy", "ds-pwp-validator", "ds-pwp-length-
based-validator"],
    "cn": ["Replicated password policy"],
    "ds-pwp-default-password-storage-scheme": ["PBKDF2-HMAC-SHA512"],
    "ds-pwp-length-based-min-password-length": 8,
    "ds-pwp-password-attribute": "userPassword",
    "subtreeSpecification": { "base": "ou=People", "filter": "/objectClass eq \"person\"" }
}
response = hdap.post do |h|
    h.path = 'dc=com/dc=example'
    h.body = JSON.generate(body)
end

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, pwp-add-policy.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

3. Verify the password administrator can view which password policy applies:

Curl

```
$ curl \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=kvaughan:bribery \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=kvaughan?
_fields=_id,pwdPolicySubentry&_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=kvaughan",
  "_rev" : "<revision>",
  "pwdPolicySubentry" : "dc=com/dc=example/cn=Replicated%20password%20policy"
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    // Kirsten Vaughan is the password administrator in this example.
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=kvaughan?_fields=_id,pwdPolicySubentry'
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: read pwp as admin', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: pwp-read-pol-as-admin.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = { '_fields': '_id,pwdPolicySubentry' }
# Kirsten Vaughan is the password administrator in this example.
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, pwp-read-pol-as-admin.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

# Kirsten Vaughan is the password administrator in this example.
utils = Utils.new('dc=com/dc=example/ou=People/uid=kvaughan', 'bribery')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
fields = { "_fields": "_id,pwdPolicySubentry" }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: fields, ssl: options) do
|f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People/uid=bjensen')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, pwp-read-pol-as-admin.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

4. Verify the field is not visible to regular users:

Curl

```
$ curl \
 --cacert ca-cert.pem \
 --user dc=com/dc=example/ou=People/uid=bjensen:hifalutin \
 'https://localhost:8443/hdap/dc=com/dc=example/ou=People/uid=bjensen?
_fields=_id,pwdPolicySubentry&_prettyPrint=true'
{
  "_id" : "dc=com/dc=example/ou=People/uid=bjensen",
  "_rev" : "<revision>"
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const options = getOptions({
        path: '/hdap/dc=com/dc=example/ou=People/uid=bjensen?_fields=_id,pwdPolicySubentry',
        credentials: 'dc=com/dc=example/ou=People/uid=bjensen:hifalutin'
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: read pwp as user', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: pwp-read-pol-as-user.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = { '_fields': '_id,pwdPolicySubentry' }
jwt = utils.authenticate('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example/ou=People/uid=bjensen',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, pwp-read-pol-as-user.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('dc=com/dc=example/ou=People/uid=bjensen', 'hifalutin')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
fields = { "_fields": "_id,pwdPolicySubentry" }
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: fields, ssl: options) do
|f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example/ou=People/uid=bjensen')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, pwp-read-pol-as-user.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

When listing subentry password policies, use the `subentries=true` parameter:

Curl

```
$ curl \
 --get \
 --cacert ca-cert.pem \
 --user uid=admin:password \
 --data "_queryFilter=/objectClass+eq+'ds-pwp-password-policy'" \
 --data '_fields=*,subtreeSpecification' \
 --data 'subentries=true' \
 --data '_prettyPrint=true' \
 'https://localhost:8443/hdap/dc=com/dc=example'
{
  "result" : [ {
    "_id" : "dc=com/dc=example/cn=Replicated%20password%20policy",
    "_rev" : "<revision>",
    "objectClass" : [ "top", "subentry", "ds-pwp-password-policy", "ds-pwp-validator", "ds-pwp-length-based-
validator" ],
    "cn" : [ "Replicated password policy" ],
    "ds-pwp-default-password-storage-scheme" : [ "PBKDF2-HMAC-SHA512" ],
    "ds-pwp-length-based-min-password-length" : 8,
    "ds-pwp-password-attribute" : "userPassword",
    "subtreeSpecification" : {
      "filter" : "/objectClass eq \"person\"",
      "base" : "ou=People"
    }
  } ],
  "resultCount" : 1,
  "pagedResultsCookie" : null,
  "totalPagedResultsPolicy" : "NONE",
  "totalPagedResults" : -1,
  "remainingPagedResults" : -1
}
```

JavaScript

```
(async () => {
    const { authenticate, doRequest, getOptions } = require('./utils')
    const filter = "/objectClass+eq+'ds-pwp-password-policy'"
    const parameters = 'subentries=true&_fields=*,subtreeSpecification'
    const options = getOptions({
        path: `/hdap/dc=com/dc=example?_queryFilter=${filter}&${parameters}`,
        credentials: 'uid=admin:password'
    })
    const jwt = await authenticate(options)
    options.headers['Authorization'] = 'Bearer ' + jwt
    const response = await doRequest('HDAP: query subentry password policies', options)
    console.log(response)
})().catch(error => { console.error(error) })
```

Source files for this sample: pwp-query.js, utils.js

Python

```
#!/usr/bin/env python3

import requests
import utils

params = {
    '_fields': '*,subtreeSpecification',
    '_queryFilter': '/objectClass eq "ds-pwp-password-policy"',
    'subentries': True
}
jwt = utils.authenticate('uid=admin', 'password')
headers = { 'Content-Type': 'application/json', 'Authorization': f'Bearer {jwt}' }
response = requests.get(
    f'https://{utils.host}:{utils.port}/hdap/dc=com/dc=example',
    headers=headers,
    params=params,
    verify=utils.ca_pem)
print('Status code: %d\nJSON: %s' % (response.status_code, response.json()))
```

Source files for this sample: utils.py, pwp-query.py

Ruby

```
require_relative 'utils.rb'
require 'faraday'

utils = Utils.new('uid=admin', 'password')
options = { ca_file: utils.ca_pem }
jwt = utils.authenticate
query = {
    "_fields": "*,subtreeSpecification",
    "_queryFilter": "/objectClass eq 'ds-pwp-password-policy'",
    "subentries": true
}
hdap = Faraday.new(url: "https://#{utils.host}:#{utils.port}/hdap/", params: query, ssl: options) do |f|
    f.headers['Content-Type'] = 'application/json'
    f.request :authorization, 'Bearer', jwt
end
response = hdap.get('dc=com/dc=example')

puts "Status code: #{response.status}\nJSON: #{response.body}"
```

Source files for this sample: utils.rb, pwp-query.rb

HDAP Ruby examples require Ruby 3.2 and the `faraday` and `json` gems.

For details about policy settings, refer to DS subentry password policies.

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

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
- Extended Operation Handler
  - Cancel Extended Operation Handler
  - Get Connection ID Extended Operation Handler
  - Get Symmetric Key Extended Operation Handler
  - Password Modify Extended Operation Handler
  - Password Policy State Extended Operation Handler
  - StartTLS Extended Operation Handler
  - Who Am I Extended Operation Handler
- Global Configuration
- HTTP Endpoint
  - Admin Endpoint
  - Alive HTTP endpoint

Caching and Backends

- Entry Cache
  - FIFO Entry Cache
  - Soft Reference Entry Cache
- Root DSE Backend

Logging

- Access Log Filtering Criteria
- 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
  - Error Log Publisher
    - Console Error Log Publisher
    - File Based Error Log Publisher
  - HTTP Access Log Publisher
    - CSV File HTTP Access Log Publisher
    - External HTTP Access Log Publisher
    - File Based HTTP Access Log Publisher
    - JSON File Based HTTP Access Log Publisher
- Log Retention Policy
  - File Count Log Retention Policy
  - Free Disk Space Log Retention Policy
  - Size Limit Log Retention Policy
- Log Rotation Policy
  - Fixed Time Log Rotation Policy
  - Size Limit Log Rotation Policy

- Time Limit Log Rotation Policy

Directory Proxy

- Service Discovery Mechanism
  - Replication Service Discovery Mechanism
  - Static Service Discovery Mechanism

Replication

- Replication Domain
- Replication Server
- Synchronization Provider
  - Replication Synchronization Provider

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
  - HDAP 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

Service Discovery Mechanism

Properties index

A

alt-authentication-enabled [HTTP Basic Authorization Mechanism]

alt-password-header [HTTP Basic Authorization Mechanism]

alt-username-header [HTTP Basic Authorization Mechanism]

api-descriptor-enabled [HTTP Connection Handler]

append [File Based Access Log Publisher]

append [File Based Audit Log Publisher]

append [File Based Error Log Publisher]

append [File Based HTTP Access Log Publisher]

argon2-iterations [Argon2 Password Storage Scheme]

argon2-length [Argon2 Password Storage Scheme]

argon2-memory [Argon2 Password Storage Scheme]

argon2-memory-pool-size [Argon2 Password Storage Scheme]

argon2-parallelism [Argon2 Password Storage Scheme]

argon2-salt-length [Argon2 Password Storage Scheme]

argon2-variant [Argon2 Password Storage Scheme]

asynchronous [CSV File Access Log Publisher]

asynchronous [CSV File HTTP Access Log Publisher]

asynchronous [File Based Access Log Publisher]

asynchronous [File Based Audit Log Publisher]

asynchronous [File Based Error Log Publisher]

asynchronous [File Based HTTP Access Log Publisher]

attribute [Backend Index]

attribute-type [Collective Attribute Subentries Virtual Attribute]

attribute-type [Entity Tag Virtual Attribute]

attribute-type [entryDN Virtual Attribute]

attribute-type [entryUUID Virtual Attribute]

attribute-type [Governing Structure Rule Virtual Attribute]

attribute-type [Has Subordinates Virtual Attribute]

attribute-type [Is Member Of Virtual Attribute]

attribute-type [Num Subordinates Virtual Attribute]

B

C

case-sensitive-strings [JSON Equality Matching Rule]

case-sensitive-strings [JSON Ordering Matching Rule]

case-sensitive-strings [JSON Query Equality Matching Rule]

case-sensitive-strings [Name And JSON Query Equality Matching Rule]

case-sensitive-validation [Dictionary Password Validator]

case-sensitive-validation [Repeated Characters Password Validator]

case-sensitive-validation [Unique Characters Password Validator]

certificate-attribute [External SASL Mechanism Handler]

certificate-mapper [External SASL Mechanism Handler]

certificate-validation-policy [External SASL Mechanism Handler]

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]

D

db-logging-file-handler-on [JE Backend]

db-logging-level [JE Backend]

db-num-cleaner-threads [JE Backend]

db-num-lock-tables [JE Backend]

db-run-cleaner [JE Backend]

db-run-log-verifier [JE Backend]

default-auth-password-storage-scheme [Password Policy Import Plugin]

default-password-policy [Global Configuration]

default-password-storage-scheme [Password Policy]

default-severity [Error Log Publisher]

default-user-password-storage-scheme [Password Policy Import Plugin]

denied-client [Administration Connector]

denied-client [Connection Handler]

denied-client [Global Configuration]

deprecated-password-storage-scheme [Password Policy]

dictionary-file [Dictionary Password Validator]

digest-algorithm [Crypto Manager]

disabled-alert-type [Alert Handler]

disabled-matching-rule [Core Schema]

disabled-privilege [Global Configuration]

disabled-syntax [Core Schema]

discovery-interval [Proxy Backend]

discovery-interval [Replication Service Discovery Mechanism]

discovery-interval [Static Service Discovery Mechanism]

disk-full-threshold [JE Backend]

disk-full-threshold [Replication Server]

disk-low-threshold [JE Backend]

disk-low-threshold [Replication Server]

disk-space-used [Size Limit Log Retention Policy]

E

F

G

H

I

J

java-class [Backend]

java-class [Base64 Password Storage Scheme]

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 [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 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 [HDAP Authorization Mechanism]

java-class [Hdap Endpoint]

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 [Name And JSON Query Equality Matching Rule]

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 [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]

java-class [Subject Equals DN Certificate Mapper]

java-class [Subschema Subentry Virtual Attribute]

java-class [Synchronization Provider]

java-class [Task Backend]

java-class [Time Limit Log Rotation Policy]

java-class [Traditional Work Queue]

java-class [Triple-DES Password Storage Scheme]

java-class [Trust Manager Provider]

java-class [Unique Attribute Plugin]

java-class [Unique Characters Password Validator]

java-class [User Defined Virtual Attribute]

java-class [User Template Virtual Attribute]

java-class [Virtual Attribute]

java-class [Who Am I Extended Operation Handler]

java-class [Work Queue]

je-backend-shared-cache-enabled [Global Configuration]

je-property [JE Backend]

json-keys [JSON Equality Matching Rule]

json-keys [JSON Ordering Matching Rule]

json-output [Error Log Publisher]

json-validation-policy [Core Schema]

jwt-algorithm [HDAP Authorization Mechanism]

jwt-key-alias [HDAP Authorization Mechanism]

jwt-key-manager-provider [HDAP Authorization Mechanism]

jwt-validity-period [HDAP Authorization Mechanism]

K

kdc-address [GSSAPI SASL Mechanism Handler]

keep-alive-interval [Proxy Backend]

keep-alive-search-request-base-dn [Proxy Backend]

keep-alive-search-request-filter [Proxy Backend]

keep-alive-timeout [Proxy Backend]

keep-stats [HTTP Connection Handler]

keep-stats [LDAP Connection Handler]

key-manager-provider [Administration Connector]

key-manager-provider [Crypto Manager]

key-manager-provider [HTTP Connection Handler]

key-manager-provider [HTTP OAuth2 OpenAM Authorization Mechanism]

key-manager-provider [HTTP OAuth2 Token Introspection (RFC 7662) Authorization Mechanism]

key-manager-provider [JMX Connection Handler]

key-manager-provider [LDAP Connection Handler]

key-manager-provider [Proxy Backend]

key-manager-provider [Replication Service Discovery Mechanism]

key-manager-provider [Replication Synchronization Provider]

key-manager-provider [Static Service Discovery Mechanism]

key-store-file [CSV File Access Log Publisher]

key-store-file [CSV File HTTP Access Log Publisher]

key-store-file [File Based Key Manager Provider]

key-store-pin [CSV File Access Log Publisher]

key-store-pin [CSV File HTTP Access Log Publisher]

key-store-pin [File Based Key Manager Provider]

key-store-pin [LDAP Key Manager Provider]

key-store-pin [PKCS#11 Key Manager Provider]

key-store-type [File Based Key Manager Provider]

key-store-type [PKCS#11 Key Manager Provider]

key-wrapping-mode [Crypto Manager]

key-wrapping-transformation [Crypto Manager]

keytab [GSSAPI SASL Mechanism Handler]

L

M

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]

mapped-search-filter-template [LDAP Pass Through Authentication Policy]

mapping-policy [LDAP Pass Through Authentication Policy]

master-key-alias [Crypto Manager]

match-attribute [Attribute Value Password Validator]

match-attribute [Exact Match Identity Mapper]

match-attribute [Regular Expression Identity Mapper]

match-base-dn [Exact Match Identity Mapper]

match-base-dn [Regular Expression Identity Mapper]

match-pattern [Regular Expression Identity Mapper]

matching-rule-name [JSON Equality Matching Rule]

matching-rule-name [JSON Ordering Matching Rule]

matching-rule-name [JSON Query Equality Matching Rule]

matching-rule-name [Name And JSON Query Equality Matching Rule]

matching-rule-oid [JSON Equality Matching Rule]

matching-rule-oid [JSON Ordering Matching Rule]

matching-rule-oid [JSON Query Equality Matching Rule]

matching-rule-oid [Name And JSON Query Equality Matching Rule]

max-allowed-client-connections [Global Configuration]

max-blocked-write-time-limit [HTTP Connection Handler]

max-blocked-write-time-limit [LDAP Connection Handler]

max-candidate-set-size [Global Configuration]

max-concurrent-ops-per-connection [HTTP Connection Handler]

max-consecutive-length [Repeated Characters Password Validator]

max-entries [FIFO Entry Cache]

max-internal-buffer-size [Global Configuration]

max-memory-percent [FIFO Entry Cache]

N

O

P

plugin-order-post-commit-modify [Plugin Root]

plugin-order-post-commit-modify-dn [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]

plugin-order-search-result-reference [Plugin Root]

plugin-order-shutdown [Plugin Root]

plugin-order-startup [Plugin Root]

plugin-order-subordinate-delete [Plugin Root]

plugin-order-subordinate-modify-dn [Plugin Root]

plugin-type [Attribute Cleanup Plugin]

plugin-type [Change Number Control Plugin]

plugin-type [ETag Plugin]

plugin-type [entryUUID Plugin]

plugin-type [Graphite Monitor Reporter Plugin]

plugin-type [Last Mod Plugin]

plugin-type [LDAP Attribute Description List Plugin]

plugin-type [Password Policy Import Plugin]

plugin-type [Plugin]

plugin-type [Referential Integrity Plugin]

Q

R

recipient-address [SMTP Account Status Notification Handler]

recipient-address [SMTP Alert Handler]

referrals-url [Replication Synchronization Provider]

rehash-policy [Argon2 Password Storage Scheme]

rehash-policy [Bcrypt Password Storage Scheme]

rehash-policy [PBKDF2 Password Storage Scheme]

remove-inbound-attributes [Attribute Cleanup Plugin]

rename-inbound-attributes [Attribute Cleanup Plugin]

replace-pattern [Regular Expression Identity Mapper]

replication-db-directory [Replication Server]

replication-port [Replication Server]

replication-purge-delay [Replication Synchronization Provider]

reporting-interval [Graphite Monitor Reporter Plugin]

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-etime-processing-greater-than [Access Log Filtering Criteria]

response-etime-processing-less-than [Access Log Filtering Criteria]

response-etime-queueing-greater-than [Access Log Filtering Criteria]

response-etime-queueing-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 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]

return-null-for-missing-properties [Hdap Endpoint]

return-null-for-missing-properties [Rest2LDAP Endpoint]

rmi-port [JMX Connection Handler]

rotation-interval [Time Limit Log Rotation Policy]

rotation-policy [CSV File Access Log Publisher]

rotation-policy [CSV File HTTP Access Log Publisher]

rotation-policy [File Based Access Log Publisher]

rotation-policy [File Based Audit Log Publisher]

rotation-policy [File Based Error Log Publisher]

rotation-policy [File Based HTTP Access Log Publisher]

rotation-policy [JSON File Based Access Log Publisher]

rotation-policy [JSON File Based HTTP Access Log Publisher]

route-all [Proxy Backend]

S

smtp-server [Mail Server]

solve-conflicts [Replication Synchronization Provider]

sort-order [Backend VLV Index]

source-address [LDAP Pass Through Authentication Policy]

source-address [Replication Synchronization Provider]

ssl-cert-nickname [Administration Connector]

ssl-cert-nickname [HTTP Connection Handler]

ssl-cert-nickname [HTTP OAuth2 OpenAM Authorization Mechanism]

ssl-cert-nickname [HTTP OAuth2 Token Introspection (RFC 7662) Authorization Mechanism]

ssl-cert-nickname [JMX Connection Handler]

ssl-cert-nickname [LDAP Connection Handler]

ssl-cert-nickname [Proxy Backend]

ssl-cert-nickname [Replication Service Discovery Mechanism]

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]

T

test-reversed-password [Dictionary Password Validator]

time-interval [File Based Access Log Publisher]

time-interval [File Based Audit Log Publisher]

time-interval [File Based Error Log Publisher]

time-interval [File Based HTTP Access Log Publisher]

time-limit [Global Configuration]

time-of-day [Fixed Time Log Rotation Policy]

token-info-url [HTTP OAuth2 OpenAM Authorization Mechanism]

token-introspection-url [HTTP OAuth2 Token Introspection (RFC 7662) Authorization Mechanism]

trust-manager-provider [Administration Connector]

trust-manager-provider [HTTP Connection Handler]

trust-manager-provider [HTTP OAuth2 OpenAM Authorization Mechanism]

trust-manager-provider [HTTP OAuth2 Token Introspection (RFC 7662) Authorization Mechanism]

trust-manager-provider [LDAP Connection Handler]

trust-manager-provider [LDAP Pass Through Authentication Policy]

trust-manager-provider [Mail Server]

trust-manager-provider [Replication Service Discovery Mechanism]

trust-manager-provider [Replication Synchronization Provider]

trust-manager-provider [Static Service Discovery Mechanism]

trust-store-file [File Based Trust Manager Provider]

trust-store-pin [File Based Trust Manager Provider]

trust-store-pin [LDAP Trust Manager Provider]

trust-store-pin [PKCS#11 Trust Manager Provider]

trust-store-type [File Based Trust Manager Provider]

trust-store-type [PKCS#11 Trust Manager Provider]

trust-transaction-ids [Global Configuration]

ttl-age [Backend Index]

ttl-enabled [Backend Index]

type [Unique Attribute Plugin]

U

user-dn-not-equal-to [Access Log Filtering Criteria]

user-dn-not-equal-to [Global Access Control Policy]

user-is-member-of [Access Log Filtering Criteria]

user-is-not-member-of [Access Log Filtering Criteria]

V

value [User Defined Virtual Attribute]

W

weight [Replication Server]

writability-mode [Global Configuration]

writability-mode [LDIF Backend]

writability-mode [Local Backend]

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

Supported LDAP controls

Extended Operations

Supported LDAP extended operations

Localization

Supported locales and languages

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 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.

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."

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.

Server controls

DS servers support the following 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.

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

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

Proprietary control that provides additional modifications to a request for internal operations.

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

Proprietary 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.

Relax Rules Control

Object Identifier: 1.3.6.1.4.1.4203.666.5.12

Experimental LDAP control allowing a directory client application to request temporary relaxation of data and service model rules.

This control is always critical and doesn't have a value.

Replication Context control

Object Identifier: 1.3.6.1.4.1.36733.2.1.5.4

Proprietary control 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

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

Proprietary control enabling the common audit framework 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 Ping Identity 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.

Client controls

The Java SDK supports the following additional controls:

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.

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

Load Balancer Connection Affinity control

Object Identifier: 1.3.6.1.4.1.36733.2.1.5.2

Proprietary 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.

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

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 (Macao 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: he
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: id
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, he
- Hindi, hi
- Hungarian, hu
- Icelandic, is
- Indonesian, id
- 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.

Result code	Name	Description
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.
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.

Result code	Name	Description
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.
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.

Result code	Name	Description
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.
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.

Result code	Name	Description
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.
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.

Result code	Name	Description
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.
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.

Result code	Name	Description
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.
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.

Result code	Name	Description
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.
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.

Result code	Name	Description
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.
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.

Result code	Name	Description
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.
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.

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

- boundDevices
- 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
- debugsearchindex
- 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-connector-connections
- 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-indexed
- ds-mon-backend-filter-unindexed
- ds-mon-backend-filter-use-start-time
- 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-size-bytes
- 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-file-count
- ds-mon-changelog-hostport
- ds-mon-changelog-id
- ds-mon-changelog-purge-delay
- ds-mon-collective-attribute-subentries-count
- 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-dynamic-groups-count
- ds-mon-entries-acis-count
- ds-mon-entries-awaiting-updates-count
- ds-mon-entries-with-aci-attributes-count
- ds-mon-fix-ids
- ds-mon-full-version
- ds-mon-global-acis-count
- ds-mon-group-id
- ds-mon-healthy-errors
- ds-mon-healthy
- ds-mon-index-cost

- ds-mon-index-uses
- ds-mon-index
- ds-mon-indexing-state
- 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-received-update
- ds-mon-last-replayed-update
- 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-password-policy-subentries-count
- ds-mon-point-version
- ds-mon-process-id
- ds-mon-product-name
- ds-mon-protocol
- ds-mon-purge-waiting-for-change-number-indexing
- 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-replicas-preventing-indexing
- 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-psearch
- 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-static-group-size-less-or-equal-to-100
- ds-mon-static-group-size-less-or-equal-to-1000
- ds-mon-static-group-size-less-or-equal-to-10000
- ds-mon-static-group-size-less-or-equal-to-100000
- ds-mon-static-group-size-less-or-equal-to-1000000
- ds-mon-static-group-size-less-or-equal-to-inf
- ds-mon-static-groups-count
- ds-mon-status-last-changed
- ds-mon-status
- ds-mon-supported-log-category

- ds-mon-system-name
- ds-mon-time-since-last-indexing
- 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-in-progress
- ds-mon-updates-in-queue
- 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-virtual-static-groups-count
- 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-assignment-condition
- fr-idm-cluster-json
- fr-idm-condition
- fr-idm-consentedMapping
- fr-idm-custom-attrs
- fr-idm-effectiveApplications
- 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-application-json
- fr-idm-managed-application-member
- fr-idm-managed-application-name
- fr-idm-managed-application-owner
- fr-idm-managed-assignment-json
- fr-idm-managed-assignment-member
- 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-applications
- 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-managed-user-task-principals
- 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-reference-1
- fr-idm-reference-2
- fr-idm-reference-3
- fr-idm-reference-4
- fr-idm-reference-5
- 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:

- 1.3.6.1.4.1.26027.1.4.8.1.3.6.1.4.1.26027.1.3.6
- authPasswordExactMatch
- authPasswordMatch
- bitStringMatch
- booleanMatch
- caseExactIA5Match
- caseExactIA5SubstringsMatch
- caseExactJsonIdMatch
- caseExactJsonQueryMatch
- caseExactMatch
- caseExactOrderingMatch
- caseExactSubstringsMatch
- caseIgnoreIA5Match
- caseIgnoreIA5SubstringsMatch
- caseIgnoreJsonIdMatch
- caseIgnoreJsonQueryMatch
- caseIgnoreJsonQueryMatchClusterObject
- caseIgnoreJsonQueryMatchManagedRole
- caseIgnoreJsonQueryMatchManagedUser
- caseIgnoreJsonQueryMatchRelationship

- caseIgnoreListMatch
- caseIgnoreListSubstringsMatch
- caseIgnoreMatch
- caseIgnoreOrderingMatch
- caseIgnoreSubstringsMatch
- certificateExactMatch
- ctsOAuth2GrantSetEqualityMatch
- 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
- passwordStorageSchemeEqualityMatch
- 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
- boundDevicesContainer
- 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-aci
- 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-change-number-indexing
- 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-groups
- ds-monitor-health-status
- ds-monitor-http-connection-handler
- ds-monitor-je-database
- ds-monitor-jvm
- ds-monitor-ldap-connection-handler
- ds-monitor-logging
- ds-monitor-raw-je-database-statistics

- ds-monitor-remote-replica
- ds-monitor-replica-db
- ds-monitor-replica
- ds-monitor-server
- ds-monitor-subentries
- 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-application
- 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
- RFC3672 Subtree Specification
- Size in bytes
- Substring Assertion
- 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 document covers server messages, such as those in logs/errors.

IDs: 1-500

ID: 1

Severity: WARNING
Message: The provided string "%s" could not be parsed as a valid Access Control Instruction (ACI) because it failed general ACI syntax evaluation.

ID: 2

Severity: WARNING
Message: The provided Access Control Instruction (ACI) version value "%s" is invalid, only the version 3.0 is supported.

ID: 3

Severity: WARNING
Message: The provided Access Control Instruction access type value "%s" is invalid. A valid access type value is either allow or deny.

ID: 4

Severity: WARNING
Message: The provided Access Control Instruction (ACI) rights values "%s" are invalid. The rights must be a list of 1 to 6 comma-separated keywords enclosed in parentheses.

ID: 5

Severity: WARNING
Message: The provided Access Control Instruction (ACI) rights keyword values "%s" are invalid. The valid rights keyword values are one or more of the following: read, write, add, delete, search, compare or the single value all.

ID: 6

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule value "%s" is invalid because it is missing a close parenthesis that corresponded to the initial open parenthesis.

ID: 7

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule value "%s" is invalid. A valid bind rule value must be in the following form: keyword operator "expression".

ID: 8

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule keyword value "%s" is invalid. A valid keyword value is one of the following: userdn, groupdn, roledn, userattr,ip, dns, dayofweek, timeofday or authmethod.

ID: 9

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule operator value "%s" is invalid. A valid bind rule operator value is either '=' or "!=".

ID: 10

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule expression value corresponding to the keyword value "%s" is missing an expression. A valid bind rule value must be in the following form: keyword operator "expression".

ID: 11

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule boolean operator value "%s" is invalid. A valid bind rule boolean operator value is either "OR" or "AND".

ID: 12

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule keyword string "%s" is invalid for the bind rule operator string "%s".

ID: 13

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule userdn expression failed to URL decode for the following reason: %s.

ID: 14

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule groupdn expression value "%s" is invalid. A valid groupdn keyword expression value requires one or more LDAP URLs in the following format: ldap:///groupdn [|| ldap:///groupdn] ... [|| ldap:///groupdn].

ID: 15

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule groupdn expression value failed to URL decode for the following reason: %s.

ID: 16

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule ip expression value "%s" is invalid. A valid ip keyword expression requires one or more comma-separated elements of a valid IP address list expression.

ID: 17

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule dns expression value "%s" is invalid. A valid dns keyword expression value requires a valid fully qualified DNS domain name.

ID: 18

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule dns expression value "%s" is invalid, because a wild-card pattern was found in the wrong position. A valid dns keyword wild-card expression value requires the '*' character only be in the leftmost position of the domain name.

ID: 19

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule dayofweek expression value "%s" is invalid, because of an invalid day of week value. A valid dayofweek value is one of the following English three-letter abbreviations for the days of the week: sun, mon, tue, wed, thu, fri, or sat.

ID: 20

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule timeofday expression value "%s" is invalid. A valid timeofday value is expressed as four digits representing hours and minutes in the 24-hour clock (0 to 2359).

ID: 21

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule timeofday expression value "%s" is not in the valid range. A valid timeofday value is expressed as four digits representing hours and minutes in the 24-hour clock (0 to 2359).

ID: 22

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule authmethod expression value "%s" is invalid. A valid authmethod value is one of the following: none, simple,SSL, or "sasl mechanism", where mechanism is one of the supported SASL mechanisms including CRAM-MD5, DIGEST-MD5, and GSSAPI.

ID: 23

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule userattr expression value "%s" is invalid.

ID: 24

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule userattr expression inheritance pattern value "%s" is invalid. A valid inheritance pattern value must have the following format: parent[inheritance_level].attribute#bindType.

ID: 25

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule userattr expression inheritance pattern value "%s" is invalid. The inheritance level value cannot exceed the max level limit of %s.

ID: 26

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule userattr expression inheritance pattern value "%s" is invalid because it is non-numeric.

ID: 27

Severity: WARNING
Message: The provided Access Control Instruction (ACI) target rule value "%s" is invalid. A valid target rule value must be in the following form: keyword operator "expression".

ID: 28

Severity: WARNING
Message: The provided Access Control Instruction (ACI) target keyword value "%s" is invalid. A valid target keyword value is one of the following: target, targetscope, targetfilter, targetattr or targattrfilters.

ID: 29

Severity: WARNING
Message: The provided Access Control Instruction (ACI) target operator value "%s" is invalid. The only valid target operator value for the "%s" keyword is '='.

ID: 30

Severity: WARNING
Message: The provided Access Control Instruction (ACI) target keyword value "%s" was seen multiple times in the ACI "%s".

ID: 31

Severity: WARNING
Message: The provided Access Control Instruction (ACI) target keyword operator value "%s" is invalid. A valid target keyword operator value is either '=' or "!=".

ID: 32

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targetscope expression operator value "%s" is invalid. A valid targetscope expression value is one of the following: one, onelevel, subtree or subordinate.

ID: 33

Severity: WARNING
Message: The provided Access Control Instruction (ACI) target expression value "%s" is invalid. A valid target keyword expression value requires a LDAP URL in the following format: ldap:///distinguished_name_pattern.

ID: 34

Severity: WARNING
Message: The provided Access Control Instruction (ACI) target expression DN value "%s" is invalid. The target expression DN value must be a descendant of the ACI entry DN "%s", if no wild-card is specified in the target expression DN.

ID: 35

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targetattr expression value "%s" is invalid. A valid targetattr keyword expression value requires one or more valid attribute type names in the following format: attribute1 [|| attribute2] ... [|| attributeN]. Attribute options are not supported.

ID: 36

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targetfilter expression value "%s" is invalid because it is not a valid LDAP filter.

ID: 37

Severity: INFO
Message: An attempt to add the entry "%s" containing an aci attribute type failed, because the authorization DN "%s" lacked modify-acl privileges.

ID: 38

Severity: INFO
Message: An attempt to modify an aci attribute type in the entry "%s" failed, because the authorization DN "%s" lacked modify-acl privileges.

ID: 39

Severity: WARNING
Message: An attempt to add the entry "%s" containing an aci attribute type failed because of the following reason: %s.

ID: 40

Severity: WARNING
Message: An attempt to modify an aci attribute type in the entry "%s" failed because of the following reason: %s.

ID: 41

Severity: WARNING
Message: "%s", located in the entry "%s", because of the following reason: %s.

ID: 42

Severity: INFO
Message: Added %s Access Control Instruction (ACI) attribute types found in context "%s" to the access control evaluation engine.

ID: 43

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targattrfilter expression value %s is invalid because it is not in the correct format. A valid targattrsfilters expression value must be in the following format: "add=attr1: F1 && attr2: F2 ... && attrN: FN,del= attr1: F1 && attr2: F2 ... && attrN: FN".

ID: 44

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targattrfilter expression value %s is invalid because the both operation tokens match in the two filter lists.

ID: 45

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targattrfilters expression value %s is invalid because there are more than two filter list statements.

ID: 46

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targattrfilters expression value %s is invalid because the provided filter list string is in the wrong format. A valid targattrfilters filter list must be in the following format: add=attr1: F1 && attr2: F2 ... && attrN: FN.

ID: 47

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targattrfilters expression value %s is invalid because one or more of the specified filters are invalid for the following reason: %s.

ID: 48

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targattrfilters expression value %s is invalid because one or more of the specified filters are invalid because of non-matching attribute type names in the filter.

ID: 49

Severity: WARNING
Message: The provided Access Control Instruction (ACI) attribute name value %s is invalid. A valid attribute type name must begin with an ASCII letter and must contain only ASCII letters,digits or the "-" character.

ID: 50

Severity: NOTICE
Message: The SASL mechanism "%s" provided in the Access Control Instruction (ACI) bind rule authmethod expression is not one of the currently registered mechanisms in the server.

ID: 51

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule dns expression value "%s" references hostname %s, but the canonical representation for that hostname is configured to be %s. The server will attempt to automatically interpret the correct localhost value.

ID: 52

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule dns expression value "%s" references hostname %s, which resolves to IP address %s, but the canonical hostname for that IP address is %s. This likely means that the provided hostname will never match any clients.

ID: 53

Severity: WARNING
Message: An error occurred while attempting to determine whether hostname %s referenced in dns expression bind rule "%s" used the correct canonical representation: %s. This likely means that the provided hostname will never match any clients.

ID: 54

Severity: INFO
Message: Added %s Global Access Control Instruction (ACI) attribute types to the access control evaluation engine.

ID: 55

Severity: INFO
Message: An unexpected error occurred while processing the ds-cfg-global-aci attribute in configuration entry %s: %s.

ID: 56

Severity: INFO
Message: An unexpected error occurred while processing the aci attributes in the configuration system: %s.

ID: 57

Severity: WARNING
Message: The pattern DN %s is not valid because it contains two consecutive wildcards in an attribute value.

ID: 58

Severity: WARNING
Message: The pattern DN %s is not valid because it uses wildcards for substring matching on an attribute type. A single wildcard is allowed in place of an attribute type.

ID: 59

Severity: WARNING
Message: The pattern DN %s is not valid because it contains a wildcard in an attribute type in a multi-valued RDN.

ID: 60

Severity: WARNING
Message: Selfwrite check skipped because an attribute "%s" with a distinguished name syntax was not a valid DN.

ID: 61

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targetattr expression value "%s" is invalid because the expression contains invalid or duplicate tokens.

ID: 62

Severity: WARNING
Message: The provided Access Control Instruction (ACI) expression value "%s" is invalid because it contains the roledn keyword, which is not supported, replace it with the groupdn keyword.

ID: 63

Severity: WARNING
Message: Failed to decode the Access Control Instruction (ACI)%s.

ID: 64

Severity: WARNING
Message: The server is being put into lockdown mode because invalid ACIs rules were detected either when the server was started or during a backend initialization.

ID: 65

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule userattr expression value failed to URL decode for the following reason: %s.

ID: 66

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule userattr expression value failed to parse because the ldap URL "%s" contains an empty base DN.

ID: 67

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule userattr expression value failed to parse because the attribute field of the ldap URL "%s" either contains more than one description or the field is empty.

ID: 68

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule IP address expression failed to parse because the prefix part of the expression "%s" has an invalid format.

ID: 69

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule IP address expression failed to parse because the prefix value of the expression "%s" was an invalid value. All values must greater than or equal to 0 and either less than or equal 32 for IPV4 addresses or less than or equal to 128 for IPV6 addresses.

ID: 70

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule IP address expression failed to parse because the prefix part of the expression "%s" has an non-numeric value.

ID: 71

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule IP address expression failed to parse because the IPv4 address expression "%s" format was invalid.

ID: 72

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule IP address expression failed to parse because the IPv4 address expression "%s" contains an invalid value. All values of the address must be between 0 and 255.

ID: 73

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule IP address expression failed to parse because the IPv4 address expression "%s" contains a non-numeric value.

ID: 74

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule IP address expression failed to parse because the IPv6 address expression "%s" contains an illegal wildcard character. Wildcards are not supported when using IPv6 addresses in a IP bind rule expression.

ID: 75

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule IP address expression "%s" failed to parse for the following reason: "%s".

ID: 76

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule IP address expression failed to parse because the netmask part of the expression "%s" has an invalid format.

ID: 77

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule IP address expression failed to parse because the netmask part of the expression "%s" has an invalid value.

ID: 78

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targetcontrol expression value "%s" is invalid. A valid targetcontrol keyword expression value requires one or more valid control OID strings in the following format: oid [|| oid1] ... [|| oidN].

ID: 79

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targetcontrol OID value "%s" could not be parsed because the value contained an illegal character %c at position %d.

ID: 80

Severity: WARNING
Message: The provided Access Control Instruction (ACI) targetcontrol OID value "%s" could not be parsed because the numeric OID contained two consecutive periods at position %d.

ID: 81

Severity: WARNING
Message: The provided Access Control Instruction (ACI) extop expression value "%s" is invalid. A valid extop keyword expression value requires one or more valid extended operation request OID strings in the following format: oid [|| oid1] ... [|| oidN].

ID: 82

Severity: WARNING
Message: Backend %s does not have a presence index defined for attribute type %s. Access control initialization may take a very long time to complete in this backend.

ID: 83

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule SSF expression "%s" failed to parse for the following reason: "%s".

ID: 84

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule ssf expression value "%s" is not in the valid range. A valid ssf value is in the range of 1 to 1024.

ID: 85

Severity: WARNING
Message: The provided Access Control Instruction (ACI) bind rule timeofday expression "%s" failed to parse for the following reason: "%s".

ID: 86

Severity: NOTICE
Message: The global access control policy '%s' has been added.

ID: 87

Severity: NOTICE
Message: The global access control policy '%s' has been removed.

ID: 88

Severity: NOTICE
Message: The global access control policy '%s' has been modified.

ID: 89

Severity: NOTICE
Message: The global access control engine has been initialized with %d policies.

ID: 90

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: 91

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: 92

Severity: ERROR
Message: The global access control policy defined in "%s" could not be parsed because it contains an unrecognized control alias "%s".

ID: 93

Severity: ERROR
Message: The global access control policy defined in "%s" could not be parsed because it contains an unrecognized extended operation alias "%s".

ID: 94

Severity: INFO
Message: This utility can be used to display basic server information.

ID: 95

Severity: INFO
Message: Adds the local server (with version 7.0 or more) to a topology with older server versions (prior to 7.0).

ID: 96

Severity: INFO
Message: Establishing connections.

ID: 97

Severity: INFO
Message: Checking registration information.

ID: 98

Severity: INFO
Message: %s entries processed (%s %% complete).

ID: 99

Severity: INFO
Message: %s entries processed.

ID: 100

Severity: INFO
Message: %s remaining to be processed.

ID: 101

Severity: ERROR
Message: A target server must be specified either by using LDAP connection options or the --%s option.

ID: 102

Severity: ERROR
Message: An error occurred while reading the server configuration: %s.

ID: 103

Severity: ERROR
Message: An error occurred while printing server status script friendly output: %s.

ID: 104

Severity: INFO
Message: Run status.

ID: 105

Severity: INFO
Message: Started.

ID: 106

Severity: INFO
Message: Stopped.

ID: 107

Severity: INFO
Message: Open connections.

ID: 108

Severity: INFO
Message: General details.

ID: 109

Severity: INFO
Message: Host name.

ID: 110

Severity: INFO
Message: Server ID.

ID: 111

Severity: INFO
Message: Administration port (LDAPS).

ID: 112

Severity: INFO
Message: Version.

ID: 113

Severity: INFO
Message: Installation path.

ID: 114

Severity: INFO
Message: Instance path.

ID: 115

Severity: INFO
Message: Installation and instance path.

ID: 116

Severity: INFO
Message: Running server Java details.

ID: 117

Severity: INFO
Message: Java version.

ID: 118

Severity: INFO
Message: Java vendor.

ID: 119

Severity: INFO
Message: JVM available CPUs.

ID: 120

Severity: INFO
Message: JVM max heap size.

ID: 121

Severity: INFO
Message: Connection handlers.

ID: 122

Severity: INFO
Message: Name.

ID: 123

Severity: INFO
Message: Port.

ID: 124

Severity: INFO
Message: Protocol.

ID: 125

Severity: INFO
Message: Load m1 rate.

ID: 126

Severity: INFO
Message: Load m5 rate.

ID: 127

Severity: INFO
Message: Local backends.

ID: 128

Severity: INFO
Message: Proxy backends.

ID: 129

Severity: INFO
Message: Backend.

ID: 130

Severity: INFO
Message: Type.

ID: 131

Severity: INFO
Message: Active cache.

ID: 132

Severity: INFO
Message: Base DN.

ID: 133

Severity: INFO
Message: Entries.

ID: 134

Severity: INFO
Message: Replication.

ID: 135

Severity: INFO
Message: Receive delay.

ID: 136

Severity: INFO
Message: Replay delay.

ID: 137

Severity: INFO
Message: Disk space.

ID: 138

Severity: INFO
Message: State.

ID: 139

Severity: INFO
Message: Free space.

ID: 140

Severity: INFO
Message: There are no connection handlers setup in the server.

ID: 141

Severity: INFO
Message: There are no local backends setup in the server.

ID: 142

Severity: INFO
Message: There are no proxy backends setup in the server.

ID: 143

Severity: INFO
Message: No disks are monitored by the server.

ID: 144

Severity: INFO
Message: Connect to the server to obtain JVM information.

ID: 145

Severity: INFO
Message: Connect to the server to obtain disk space information.

ID: 146

Severity: ERROR
Message: Unable to perform the search on the monitor backend. To display information, the status tool requires the remote server monitor backend to be enabled.

ID: 147

Severity: INFO
Message: The folder into which the files will be placed into.

ID: 148

Severity: INFO
Message: {directory}.

ID: 149

Severity: INFO
Message: The instance is running.

ID: 150

Severity: INFO
Message: The instance is not running.

ID: 151

Severity: INFO
Message: Password to use to bind to the server.

ID: 152

Severity: INFO
Message: {password}.

ID: 153

Severity: INFO
Message: Path to the JDK utility binaries directory such as jstack.

ID: 154

Severity: INFO
Message: {directory}.

ID: 155

Severity: INFO
Message: Specifies whether audit files are excluded.

ID: 156

Severity: INFO
Message: Specifies whether keystore files are excluded.

ID: 157

Severity: INFO
Message: Specifies that the tool should not interact with the server, that is no LDAP operation, and no jstack sampling.

ID: 158

Severity: INFO
Message: 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.

ID: 159

Severity: INFO
Message: Maximum number of log files to collect. Ignored if --logsAfterDate is provided.

ID: 160

Severity: INFO
Message: {number}.

ID: 161

Severity: INFO
Message: Collect log files after this date. Format "YYYYMMDDhhmmss" like "20161123143612" = 23 November 2016, 14:36 12s. Overrides --maxLogFiles.

ID: 162

Severity: INFO
Message: {date}.

ID: 163

Severity: INFO
Message: When the server is embedded in OpenAM, there is no PID file. Therefore this option indicates the server PID of the OpenAM application server.

ID: 164

Severity: INFO
Message: {pid}.

ID: 165

Severity: INFO
Message: This tool collects support data from the OpenDJ instance it is bound to.

ID: 166

Severity: NOTICE
Message: <xinclude:include href="description-supportextract.xml" />.

ID: 167

Severity: NOTICE
Message: extract support data.

ID: 168

Severity: ERROR
Message: Error: %s.

ID: 169

Severity: ERROR
Message: The output directory "%s" could not be found.

ID: 170

Severity: ERROR
Message: The number argument for "maxLogFiles" is invalid.

ID: 171

Severity: ERROR
Message: The date argument for "logsAfterDate" is invalid.

ID: 172

Severity: ERROR
Message: The serverPID argument is not a valid number.

ID: 173

Severity: ERROR
Message: The --bindDN and/or --bindPassword arguments are missing.

ID: 174

Severity: ERROR
Message: Could not collect monitoring data from the directory server: %s.

ID: 175

Severity: ERROR
Message: Could not collect all the files.

ID: 176

Severity: ERROR
Message: An error occurred while parsing the command-line arguments: The argument --outputDirectory is required to have a value but none was provided in the argument list and no default value is available.

ID: 177

Severity: INFO
Message: Continuing data gathering though there was an error: %s.

ID: 178

Severity: INFO
Message: The following archive has been created : %s.

ID: 179

Severity: NOTICE
Message: Could not find "java.properties".

ID: 180

Severity: NOTICE
Message: Error while loading "java.properties": %s.

ID: 181

Severity: INFO
Message: No monitoring data can be collected when the server is not running or no server interaction is allowed.

ID: 182

Severity: INFO
Message: No JVM stack dump can be produced when the server is not running or no server interaction is allowed.

ID: 183

Severity: ERROR
Message: Could not find the "%s" command in "%s".

ID: 184

Severity: ERROR
Message: Skipping log "%s" (unsupported config).

ID: 185

Severity: INFO
Message: Collecting the configuration files.

ID: 186

Severity: INFO
Message: Collecting process statistics.

ID: 187

Severity: INFO
Message: Making copy of the %s file before collecting process statistics.

ID: 188

Severity: INFO
Message: Removing file copy %s.

ID: 189

Severity: INFO
Message: Skipping backup of %s log (file not found).

ID: 190

Severity: INFO
Message: Collecting JVM heap dump : using %s.

ID: 191

Severity: INFO
Message: Collecting the log files.

ID: 192

Severity: INFO
Message: Collecting backend statistics.

ID: 193

Severity: INFO
Message: Collecting the GC log files.

ID: 194

Severity: INFO
Message: Skipping GC logs collection because GC logging is not enabled.

ID: 195

Severity: INFO
Message: Collecting ChangelogDb information.

ID: 196

Severity: INFO
Message: No changelogDb data found (is a DS or is not replicated).

ID: 197

Severity: INFO
Message: Collecting the monitoring info from cn=monitor.

ID: 198

Severity: INFO
Message: Collecting process thread information, sample number : %d.

ID: 199

Severity: INFO
Message: Generating stack dump, sample number : %d using %s for pid %d.

ID: 200

Severity: INFO
Message: Processor information.

ID: 201

Severity: INFO
Message: OS information.

ID: 202

Severity: INFO
Message: Extract environment variables.

ID: 203

Severity: INFO
Message: DS /proc files.

ID: 204

Severity: INFO
Message: VM environment information.

ID: 205

Severity: INFO
Message: Disk information.

ID: 206

Severity: INFO
Message: Network information.

ID: 207

Severity: INFO
Message: Linux release.

ID: 208

Severity: INFO
Message: Linux /proc info files.

ID: 209

Severity: INFO
Message: Collecting system node information.

ID: 210

Severity: INFO
Message: Listing the security stores.

ID: 211

Severity: INFO
Message: Adding %s.

ID: 212

Severity: INFO
Message: Skipping %s.

ID: 213

Severity: INFO
Message: Skipping java security stores.

ID: 214

Severity: INFO
Message: Skipping audit files.

ID: 215

Severity: ERROR
Message: There was at least one error during the tasks execution: %s.

ID: 216

Severity: INFO
Message: No value was provided for %s, JDK tool directory is set to %s.

ID: 217

Severity: INFO
Message: Cannot extract process statistics (by running "top" command) on OS '%s'. Only %s dump samples will be collected.

ID: 218

Severity: INFO
Message: 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 '%s' subcommand.

ID: 219

Severity: NOTICE
Message: Manages data synchronization between servers.

ID: 220

Severity: ERROR
Message: An unexpected error has been raised during execution of the tool: '%s'.

ID: 221

Severity: INFO
Message: Initialize replication data for the server.

ID: 222

Severity: INFO
Message: Starting initialization from '%s' to '%s' for base DNs: %s.

ID: 223

Severity: INFO
Message: Starting initialization from '%s' to all replicas for base DNs: %s.

ID: 224

Severity: INFO
Message: Starting initialization for base DN: '%s'.

ID: 225

Severity: NOTICE
Message: An error occurred when launching initialization: %s.

ID: 226

Severity: INFO
Message: Base DN(s) to use. Multiple base DNs can be provided by using this option multiple times.

ID: 227

Severity: INFO
Message: Server ID of the server containing the source data.

ID: 228

Severity: INFO
Message: {serverSource}.

ID: 229

Severity: INFO
Message: Server ID of the server to be initialized.

ID: 230

Severity: INFO
Message: {serverToInitialize}.

ID: 231

Severity: INFO
Message: Initialize all the other servers in the topology.

ID: 232

Severity: NOTICE
Message: A mandatory argument is missing. Choose one and only one argument from '--toServer', '--fromServer' and '--toAllServers'.

ID: 233

Severity: NOTICE
Message: Invalid combination of arguments. Choose one and only one argument from '--toServer', '--fromServer' and '--toAllServers'.

ID: 234

Severity: NOTICE
Message: An error occurred during initialization. Last log details: %s. Task state: %s.

ID: 235

Severity: NOTICE
Message: An error occurred during initialization. Task state: %s.

ID: 236

Severity: NOTICE
Message: Error during the processing of the reset generationID task. Last log details: %s. Task state: %s.

ID: 237

Severity: NOTICE
Message: Error during the processing of the reset generationID task. Task state: %s.

ID: 238

Severity: INFO
Message: Cannot extract the master key pair certificate with alias '%s' because the crypto-manager's key manager provider of the local server does not contain such key or does not support extracting certificates.

ID: 239

Severity: INFO
Message: Cannot find a CA certificate in the replication trust managers of the local server. Please verify the %s trust-manager-provider configuration value.

ID: 240

Severity: INFO
Message: Base DN(s) to replicate.

ID: 241

Severity: ERROR
Message: Inappropriate server ID '%s' for the local server: it does not represent a numeric value as required to communicate with pre 7.0 servers. Use an integer server ID without leading zeros.

ID: 242

Severity: ERROR
Message: Inappropriate group ID '%s' for the local server: it cannot be converted to an integer, but this is required to communicate with pre 7.0 servers. Use a group ID that can be converted into an integer, or use 'default' if you do not want to set a group ID.

ID: 243

Severity: ERROR
Message: Cannot use server ID '%s' for the local server because it is already used by server '%s' (configuration object %s).

ID: 244

Severity: ERROR
Message: Cannot read the configuration of the local server. The error was: %s.

ID: 245

Severity: ERROR
Message: Cannot read 'cn=admin data' in server '%s'. The error was: %s.

ID: 246

Severity: ERROR
Message: Error when reading the server IDs from all the servers in the old topology. The error was: %s.

ID: 247

Severity: ERROR
Message: Cannot discover the configuration of the local server. The error was: %s.

ID: 248

Severity: ERROR
Message: Cannot add an entry describing the new server to 'cn=admin data' in the remote server '%s' in the existing topology. The error was: %s.

ID: 249

Severity: ERROR
Message: Cannot add '%s' to the bootstrap replication server list on server '%s'. The error was: %s.

ID: 250

Severity: ERROR
Message: Cannot add '%s' to the configuration of replication server on server '%s'. The error was: %s.

ID: 251

Severity: ERROR
Message: Cannot add '%s' to the configuration of replication domain '%s' on server '%s'. The error was: %s.

ID: 252

Severity: ERROR
Message: Cannot retrieve the replication port from server '%s'. The error was: %s.

ID: 253

Severity: ERROR
Message: Cannot configure the local server so it can replicate with the servers in the existing topology. The error was: %s.

ID: 254

Severity: ERROR
Message: Cannot read server ID from server '%s' on DNs '%s' and '%s'. The errors were: %s.

ID: 255

Severity: ERROR
Message: Cannot determine server ID for server '%s' on DNs '%s' and '%s'.

ID: 256

Severity: ERROR
Message: Cannot obtain a connection to server '%s' from the existing topology. The error was: %s.

ID: 257

Severity: ERROR
Message: Cannot obtain a connection for one of the servers in the existing topology. The error was: %s.

ID: 258

Severity: ERROR
Message: Cannot configure all the servers in the existing topology to replicate with the new server. The error was: %s.

ID: 259

Severity: ERROR
Message: An error occurred opening (or closing) a connection to change the configuration of the local server. The error was: %s.

ID: 260

Severity: ERROR
Message: An error occurred opening connections to the server in the existing topology or the new server. The error was: %s.

ID: 261

Severity: ERROR
Message: Once done, restart the server for the changes to take effect.

ID: 262

Severity: ERROR
Message: Cannot read the server ID or the group ID for the local server. The error(s) are: %s.

ID: 263

Severity: ERROR
Message: Cannot add the entry '%s' to the server '%s'. The error was: %s.

ID: 264

Severity: ERROR
Message: Cannot read the entry '%s' on server '%s'. The error was: %s.

ID: 265

Severity: ERROR
Message: Cannot modify the entry '%s' on server '%s' to reconcile it with what is expected by the tool. The error was: %s.

ID: 266

Severity: INFO
Message: Adding server instances keys from the existing topology into 'cn=admin data' in the new server.

ID: 267

Severity: ERROR
Message: Cannot add server instance keys to the new server. The error was: %s.

ID: 268

Severity: INFO
Message: Enabling 'cn=admin data' backend.

ID: 269

Severity: INFO
Message: Creating the trust manager '%s' to be used by replication connections in the local server.

ID: 270

Severity: INFO
Message: Updating replication configuration on local server.

ID: 271

Severity: INFO
Message: Updating replication configuration for baseDN '%s' on local server.

ID: 272

Severity: INFO
Message: Configuring the servers in the topology to talk to the local server.

ID: 273

Severity: INFO
Message: Replication has been successfully configured on the local server. Note that for replication to work you must initialize the contents of the base DNs that are being replicated. Run the following command(s) to do so:.

ID: 274

Severity: INFO
Message: Start disaster recovery for all servers. Read the documentation on disaster recovery carefully before using this command.

ID: 275

Severity: INFO
Message: End disaster recovery for all servers. Read the documentation on disaster recovery carefully before using this command.

ID: 276

Severity: NOTICE
Message: An error occurred while attempting to monitor the task progress. The error was: %s.

ID: 289

Severity: INFO
Message: Check the server error logs for additional details.

ID: 290

Severity: INFO
Message: Purges old replication meta-data from application data.

ID: 291

Severity: INFO
Message: Maximum duration of the command in seconds.

ID: 292

Severity: INFO
Message: {maximum duration in seconds}.

ID: 293

Severity: INFO
Message: Purging old replication meta-data for server '%s' for base DNs: %s.

ID: 294

Severity: NOTICE
Message: An error occurred while purging old replication meta-data: %s.

ID: 295

Severity: NOTICE
Message: An error occurred while purging old replication meta-data. Last log details: %s. Task state: %s.

ID: 296

Severity: NOTICE
Message: An error occurred while purging old replication meta-data. Task state: %s.

ID: 297

Severity: INFO
Message: Re-synchronizes the change-log change number of the target server with the change-log change number of the source server.

ID: 298

Severity: INFO
Message: The change number to use as the basis for re-synchronization.

ID: 299

Severity: INFO
Message: {change number}.

ID: 300

Severity: INFO
Message: Start to re-synchronize the change-log change number of server '%s' from the source server '%s'.

ID: 301

Severity: INFO
Message: Re-synchronizing with change number '%s'.

ID: 302

Severity: INFO
Message: Change number corresponds to start CSN '%s' and base DN '%s'.

ID: 303

Severity: NOTICE
Message: Error when searching change number '%s' in the source server: %s.

ID: 304

Severity: NOTICE
Message: Unable to retrieve change number '%s' in the source server.

ID: 305

Severity: NOTICE
Message: No CSN found for the change number '%s' in the source server.

ID: 306

Severity: NOTICE
Message: No base DN found for the change number '%s' in the source server.

ID: 307

Severity: NOTICE
Message: Unable to retrieve the newest change number in the source server.

ID: 308

Severity: NOTICE
Message: An error occurred when re-synchronizing the change number: %s.

ID: 309

Severity: NOTICE
Message: An error occurred during the re-synchronization of the change number. Last log details: %s. Task state: %s.

ID: 310

Severity: NOTICE
Message: An error occurred during the re-synchronization of the change number. Task state: %s.

ID: 311

Severity: NOTICE
Message: The source and target server have distinct replication domains. As a consequence the change number from the target server can’t be reset from a change number of the source server.

ID: 312

Severity: NOTICE
Message: Unable to find a replicated domain in the target server.

ID: 313

Severity: NOTICE
Message: Unable to find a replicated domain matching the DN '%s' corresponding to the change number.

ID: 314

Severity: NOTICE
Message: An error occurred when searching replicated domains in server. Result code was: '%s'.

ID: 315

Severity: NOTICE
Message: An error occurred when searching replicated domains in server: %s.

ID: 316

Severity: INFO
Message: 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. The status of a server is one of the following.

- BAD - DATA MISMATCH: either the fractional replication configuration does not match the backend data, or the initial state of the replicated data does not match other servers and this server must be re-initialized;
- BAD - TOO LATE: the server has fallen further behind than the replication purge delay and must be re-initialized;
- GOOD: normal operation, nothing to do;
- SLOW: the server's replay delay is greater than five seconds;
- UNHEALTHY: read the server health errors in the server monitoring data for details.

ID: 317

Severity: INFO
Message: Base DN.

ID: 318

Severity: INFO
Message: Base DN / DS.

ID: 319

Severity: INFO
Message: Base DN / RS.

ID: 320

Severity: INFO
Message: Base DN / RS / DS.

ID: 321

Severity: INFO
Message: Status.

ID: 322

Severity: INFO
Message: Group.

ID: 323

Severity: INFO
Message: Receive delay (ms).

ID: 324

Severity: INFO
Message: Replay delay (ms).

ID: 325

Severity: INFO
Message: Entry count.

ID: 326

Severity: INFO
Message: Server / host port.

ID: 327

Severity: INFO
Message: Error or diagnostic.

ID: 328

Severity: INFO
Message: Displays individual replicas in the output.

ID: 329

Severity: INFO
Message: Displays individual changelog servers in the output.

ID: 330

Severity: INFO
Message: Display replication group information in the output.

ID: 331

Severity: ERROR
Message: An error occurred while reading cn=monitor on server %s: %s.

ID: 332

Severity: ERROR
Message: Error while reading cn=monitor: %s.

ID: 333

Severity: ERROR
Message: Cannot contact server: no admin port could be discovered in its entry %s.

ID: 334

Severity: ERROR
Message: Server is not healthy: %s.

ID: 335

Severity: ERROR
Message: General error: %s.

ID: 336

Severity: INFO
Message: 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.

ID: 337

Severity: INFO
Message: 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.

ID: 338

Severity: ERROR
Message: A problem occurred while clearing the changelog data: %s.

ID: 339

Severity: INFO
Message: Clearing all replication server changelog data for the local server.

ID: 340

Severity: INFO
Message: Nothing to do: this server does not keep any replication server changelog data.

ID: 341

Severity: ERROR
Message: Cannot obtain a connection for one of the servers in the topology. The error was: %s.

ID: 342

Severity: ERROR
Message: Cannot obtain a connection to server '%s' from the topology. The error was: %s.

ID: 343

Severity: ERROR
Message: An error occurred opening (or closing) a connection to read or change the configuration of the local server. The error was: %s.

ID: 344

Severity: INFO
Message: This server has already been cleaned, there is nothing left to do.

ID: 345

Severity: INFO
Message: The remaining cleanup on this server cannot be performed because the administrator user '%s' in the admin backend was used to bind to the server. You should run again this command using a bind DN, such as "uid=admin", which lies outside the "cn=admin data" backend to be able to complete the cleanup on this server.

ID: 346

Severity: ERROR
Message: Cannot disable admin backend on the local server. The error was: %s.

ID: 347

Severity: ERROR
Message: Cannot update configuration for server '%s'. The error was: %s.

ID: 348

Severity: ERROR
Message: Cannot read monitor entries under '%s' on the local server. The error was: %s.

ID: 349

Severity: ERROR
Message: Cannot delete entries under '%s' in admin backend. The error was: %s.

ID: 350

Severity: INFO
Message: All servers have been cleaned.

ID: 351

Severity: INFO
Message: All servers have been cleaned. The admin backend will remain enabled because it contains secret keys that may be used by reversible password storage schemes.

ID: 352

Severity: INFO
Message: The server %s has now been cleaned.

ID: 353

Severity: INFO
Message: Servers have been cleaned, however the administrator user '%s' in the admin backend was not removed because it was used to bind to the server. To remove it, you should run the following commands on all servers using a bind DN, such as "uid=admin", which lies outside the "cn=admin data" backend:.

ID: 354

Severity: INFO
Message: Removing servers from admin backend.

ID: 355

Severity: INFO
Message: Removing instance keys from admin backend.

ID: 356

Severity: INFO
Message: Removing administrators from admin backend.

ID: 357

Severity: INFO
Message: Cleaning and updating configuration of server %s.

ID: 358

Severity: INFO
Message: Set bootstrap servers configuration for server %s.

ID: 359

Severity: INFO
Message: Disable admin backend for server %s.

ID: 360

Severity: WARNING
Message: Some reversible password storage schemes are enabled in the configuration, note that it won’t be possible to use them in password policies since the admin data backend has been disabled.

ID: 361

Severity: INFO
Message: 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).

ID: 362

Severity: ERROR
Message: An attempt was made to configure the root DSE backend without providing a configuration entry. This is not allowed.

ID: 363

Severity: WARNING
Message: Base DN "%s" is configured as one of the subordinate base DNs to use for searches below the root DSE. However, this base DN is not handled by any suffix registered with the Directory Server and will therefore not be used.

ID: 364

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: 365

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: 366

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: 367

Severity: ERROR
Message: An unexpected error occurred while trying to open the LDIF writer for the root DSE backend: %s.

ID: 368

Severity: ERROR
Message: An unexpected error occurred while trying to export the root DSE entry to the specified LDIF target: %s.

ID: 369

Severity: INFO
Message: The root DSE configuration has been updated so that it will now use a new set of user-defined attributes.

ID: 370

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: 371

Severity: ERROR
Message: Unwilling to add entry "%s" because add operations are not supported in the "%s" backend.

ID: 372

Severity: ERROR
Message: Unwilling to remove entry "%s" because delete operations are not supported in the "%s" backend.

ID: 373

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: 374

Severity: ERROR
Message: Unwilling to rename entry "%s" because modify DN operations are not supported in the "%s" backend.

ID: 375

Severity: ERROR
Message: An error occurred while attempting to export the base monitor entry: %s.

ID: 376

Severity: ERROR
Message: An error occurred while attempting to export the monitor entry for monitor provider %s: %s.

ID: 377

Severity: ERROR
Message: The "%s" backend does not support LDIF import operations.

ID: 378

Severity: INFO
Message: The monitor configuration has been updated so that it will now use a new set of user-defined attributes.

ID: 379

Severity: ERROR
Message: Unable to retrieve the requested entry from the "%s" backend because the provided DN was null.

ID: 380

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: 381

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: 382

Severity: ERROR
Message: An error occurred while attempting to export the base schema entry: %s.

ID: 383

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: 384

Severity: ERROR
Message: An unexpected error occurred while trying to open the LDIF writer for the schema backend: %s.

ID: 386

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: 387

Severity: ERROR
Message: The task defined in entry %s is invalid because it has an invalid state %s.

ID: 388

Severity: ERROR
Message: An error occurred while trying to parse the scheduled start time value %s from task entry %s.

ID: 389

Severity: ERROR
Message: An error occurred while trying to parse the actual start time value %s from task entry %s.

ID: 390

Severity: ERROR
Message: An error occurred while trying to parse the completion time value %s from task entry %s.

ID: 391

Severity: ERROR
Message: Task entry %s is missing required attribute %s.

ID: 392

Severity: ERROR
Message: There are multiple instances of attribute %s in task entry %s.

ID: 393

Severity: ERROR
Message: There are multiple values for attribute %s in task entry %s.

ID: 394

Severity: ERROR
Message: An error occurred while executing the task defined in entry %s: %s.

ID: 395

Severity: ERROR
Message: The provided recurring task entry does not contain attribute %s which is needed to hold the recurring task ID.

ID: 396

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: 397

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: 398

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: 399

Severity: ERROR
Message: The provided recurring task entry does not contain attribute %s which is needed to specify recurring task schedule.

ID: 400

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: 401

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: 402

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: 403

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: 404

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: 405

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: 406

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: 407

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: 408

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: 409

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: 410

Severity: ERROR
Message: An error occurred while attempting to determine the new path to the task data backing file: %s.

ID: 411

Severity: INFO
Message: The completed task retention time has been updated to %d seconds. This will take effect immediately.

ID: 412

Severity: INFO
Message: The path to the task data backing file has been changed to %s. A snapshot of the current task configuration has been written to that file and it will continue to be used for future updates.

ID: 413

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: 414

Severity: INFO
Message: This file contains the data used by the Directory Server task scheduler backend. Do not edit this file directly, as there is a risk that those changes will be lost. Scheduled and recurring task definitions should only be edited using the administration utilities provided with the Directory Server.

ID: 415

Severity: ERROR
Message: Unable to add recurring task %s to the task scheduler because another recurring task already exists with the same ID.

ID: 416

Severity: ERROR
Message: Unable to schedule task %s because another task already exists with the same ID.

ID: 417

Severity: WARNING
Message: Unable to add completed task %s to the task scheduler because another task already exists with the same ID.

ID: 418

Severity: ERROR
Message: An error occurred while attempting to schedule the next iteration of recurring task %s: %s.

ID: 419

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: 420

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: 421

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: 422

Severity: ERROR
Message: An error occurred while attempting to parse entry %s as a recurring task and add it to the scheduler: %s.

ID: 423

Severity: ERROR
Message: An error occurred while attempting to parse entry %s as a task and add it to the scheduler: %s.

ID: 424

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: 425

Severity: ERROR
Message: An error occurred while attempting to read from the tasks data backing file %s: %s.

ID: 426

Severity: ERROR
Message: An error occurred while attempting to create a new tasks backing file %s for use with the task scheduler: %s.

ID: 427

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: 428

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: 429

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: 430

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: 431

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: 432

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: 433

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: 434

Severity: WARNING
Message: An error occurred while attempting to rename the current tasks backing file from %s to %s: %s. The previous task configuration (which does not reflect the latest update) may be lost.

ID: 435

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: 436

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: 437

Severity: INFO
Message: The tasks backend is being shut down.

ID: 438

Severity: INFO
Message: The root DSE configuration has been updated so that configuration attribute %s will now use a value of %s.

ID: 439

Severity: ERROR
Message: Unable to remove pending task %s because no such task exists.

ID: 440

Severity: ERROR
Message: Unable to remove pending task %s because the task is no longer pending.

ID: 441

Severity: ERROR
Message: Unable to remove completed task %s because no such task exists in the list of completed tasks.

ID: 442

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: 443

Severity: ERROR
Message: Unable to remove entry %s from the task backend because there is no scheduled task associated with that entry DN.

ID: 444

Severity: ERROR
Message: Unable to delete entry %s from the task backend because the associated task is currently running.

ID: 445

Severity: ERROR
Message: Unable to remove entry %s from the task backend because there is no recurring task associated with that entry DN.

ID: 446

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: 447

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: 448

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: 449

Severity: ERROR
Message: Unwilling to update entry "%s" because modify operations are not supported in the "%s" backend.

ID: 450

Severity: ERROR
Message: Exactly one base DN must be provided for use with the memory-based backend.

ID: 451

Severity: ERROR
Message: Entry %s already exists in the memory-based backend.

ID: 452

Severity: ERROR
Message: Entry %s does not belong in the memory-based backend.

ID: 453

Severity: ERROR
Message: Unable to add entry %s because its parent entry %s does not exist in the memory-based backend.

ID: 454

Severity: ERROR
Message: Entry %s does not exist in the "%s" backend.

ID: 455

Severity: ERROR
Message: Cannot delete entry %s because it has one or more subordinate entries.

ID: 456

Severity: ERROR
Message: Unable to create an LDIF writer: %s.

ID: 457

Severity: ERROR
Message: Cannot write entry %s to LDIF: %s.

ID: 458

Severity: ERROR
Message: Unable to create an LDIF reader: %s.

ID: 459

Severity: ERROR
Message: An unrecoverable error occurred while reading from LDIF: %s.

ID: 460

Severity: ERROR
Message: An unexpected error occurred while processing the import: %s.

ID: 461

Severity: ERROR
Message: Cannot rename entry %s because it has one or more subordinate entries.

ID: 462

Severity: ERROR
Message: Cannot rename entry %s because the target entry is in a different backend.

ID: 463

Severity: ERROR
Message: Cannot rename entry %s because the new parent entry %s doesn't exist.

ID: 464

Severity: ERROR
Message: An error occurred while attempting to register the base DNs %s in the Directory Server: %s.

ID: 465

Severity: ERROR
Message: The schema backend does not support the %s modification type.

ID: 466

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: 467

Severity: ERROR
Message: An error occurred while attempting to write the updated schema: %s.

ID: 468

Severity: ERROR
Message: The server will not allow removing all values for the %s attribute type in the server schema.

ID: 469

Severity: ERROR
Message: Circular reference detected for attribute type %s in which the superior type chain references the attribute type itself.

ID: 470

Severity: ERROR
Message: Circular reference detected for objectclass %s in which the superior class chain references the objectclass itself.

ID: 471

Severity: ERROR
Message: Circular reference detected for DIT structure rule %s in which the superior rule chain references the DIT structure rule itself.

ID: 472

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: 473

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: 474

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: 475

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: 476

Severity: ERROR
Message: Unable to remove attribute type %s from the server schema because no such attribute type is defined.

ID: 477

Severity: ERROR
Message: Unable to remove objectclass %s from the server schema because no such objectclass is defined.

ID: 478

Severity: ERROR
Message: Unable to remove name form %s from the server schema because no such name form is defined.

ID: 479

Severity: ERROR
Message: Unable to remove DIT content rule %s from the server schema because no such DIT content rule is defined.

ID: 480

Severity: ERROR
Message: Unable to remove DIT structure rule %s from the server schema because no such DIT structure rule is defined.

ID: 481

Severity: ERROR
Message: Unable to remove matching rule use %s from the server schema because no such matching rule use is defined.

ID: 482

Severity: ERROR
Message: You do not have sufficient privileges to modify the Directory Server schema.

ID: 483

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: 484

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: 485

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: 486

Severity: ERROR
Message: The Directory Server is not configured to allow task %s to be invoked.

ID: 488

Severity: ERROR
Message: Indexes are not supported in the "%s" backend.

ID: 489

Severity: ERROR
Message: LDIF import and export operations are not supported in the "%s" backend.

ID: 490

Severity: ERROR
Message: The root container for backend %s has not been initialized preventing this backend from processing the requested operation.

ID: 491

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: 492

Severity: ERROR
Message: Entry %s cannot be modified because it does not represent a valid task in the server.

ID: 493

Severity: ERROR
Message: Entry %s cannot be modified because the associated task has completed running. Completed tasks cannot be modified.

ID: 494

Severity: ERROR
Message: Entry %s cannot be modified because the server does not currently support modifying recurring task entries.

ID: 495

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: 496

Severity: INFO
Message: Task processing was interrupted by a modify request to cancel the task.

ID: 497

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: 498

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: 499

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: 500

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.

IDs: 501-1000

ID: 501

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: 502

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: 503

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: 504

Severity: ERROR
Message: Entry %s already exists in the LDIF backend.

ID: 505

Severity: ERROR
Message: The parent for entry %s does not exist.

ID: 506

Severity: ERROR
Message: Entry %s does not exist.

ID: 507

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: 508

Severity: ERROR
Message: Entry %s does not exist.

ID: 509

Severity: ERROR
Message: Source entry %s does not exist.

ID: 510

Severity: ERROR
Message: Target entry %s already exists.

ID: 511

Severity: ERROR
Message: The new parent DN %s does not exist.

ID: 512

Severity: ERROR
Message: Entry %s specified as the search base DN does not exist.

ID: 513

Severity: ERROR
Message: An error occurred while trying to create the writer for the LDIF export operation: %s.

ID: 514

Severity: ERROR
Message: An error occurred while trying to write entry %s during the LDIF export: %s.

ID: 515

Severity: ERROR
Message: An error occurred while trying to create the reader for the LDIF import operation: %s.

ID: 516

Severity: ERROR
Message: An unrecoverable error occurred while attempting to read data from the import file: %s. The LDIF import cannot continue.

ID: 517

Severity: INFO
Message: The change to the LDIF file path will not take effect until the backend is disabled and re-enabled.

ID: 518

Severity: INFO
Message: The change to the LDIF backend base DN will not take effect until the backend is disabled and re-enabled.

ID: 519

Severity: ERROR
Message: The target entry %s does not exist.

ID: 520

Severity: ERROR
Message: The target entry %s does not exist.

ID: 521

Severity: ERROR
Message: This backend does not provide support for the numSubordinates operational attribute.

ID: 522

Severity: NOTICE
Message: The backend %s is now taken offline.

ID: 523

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid number of tokens.

ID: 524

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid minute token.

ID: 525

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid hour token.

ID: 526

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid day of the month token.

ID: 527

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid month of the year token.

ID: 528

Severity: ERROR
Message: The provided recurring task entry attribute %s holding the recurring task schedule has invalid day of the week token.

ID: 529

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: 530

Severity: ERROR
Message: An error occurred while attempting to export task backend data: %s.

ID: 531

Severity: INFO
Message: JE backend '%s' does not specify the number of cleaner threads: defaulting to %d threads.

ID: 532

Severity: INFO
Message: JE backend '%s' does not specify the number of lock tables: defaulting to %d.

ID: 533

Severity: ERROR
Message: Unable to schedule task %s because its dependency task %s is missing.

ID: 534

Severity: NOTICE
Message: %s task %s started execution.

ID: 535

Severity: NOTICE
Message: %s task %s finished execution in the state %s.

ID: 536

Severity: ERROR
Message: Unable to remove ldap syntax description %s from the server schema because no such ldap syntax description is defined.

ID: 537

Severity: ERROR
Message: An error occurred while attempting to decode the ldapsyntax description "%s": %s.

ID: 538

Severity: ERROR
Message: The provided recurring task schedule value has an invalid number of tokens.

ID: 539

Severity: ERROR
Message: The provided recurring task schedule value has an invalid minute token.

ID: 540

Severity: ERROR
Message: The provided recurring task schedule value has an invalid hour token.

ID: 541

Severity: ERROR
Message: The provided recurring task schedule value has an invalid day of the month token.

ID: 542

Severity: ERROR
Message: The provided recurring task schedule value has an invalid month of the year token.

ID: 543

Severity: ERROR
Message: The provided recurring task schedule value has an invalid day of the week token.

ID: 544

Severity: ERROR
Message: The schema backend does not support the Replace modification type for the %s attribute type.

ID: 545

Severity: ERROR
Message: An error occurred while trying to close file %s for the LDIF backend defined in configuration entry %s: %s.

ID: 546

Severity: ERROR
Message: The file %s written for the LDIF backend defined in configuration entry %s is 0 bytes long and unusable.

ID: 547

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: 548

Severity: ERROR
Message: Configuration attribute ds-cfg-db-cache-percent has a value of %d%% but the JVM has only %d%% available.

ID: 549

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: 550

Severity: WARNING
Message: Disk free space of %d bytes for directory %s is now below low threshold of %d bytes. Backend %s is now locked down and will no longer accept any operations from clients until sufficient disk space is restored.

ID: 551

Severity: WARNING
Message: Disk free space of %d bytes for directory %s is now below disk low threshold of %d bytes. Backend %s is now offline and will no longer accept any operations until sufficient disk space is restored.

ID: 552

Severity: ERROR
Message: An error occurred while trying to list the files to backup for backend '%s': %s.

ID: 553

Severity: ERROR
Message: Insufficient free memory (%d bytes) to perform import. At least %d bytes of free memory is required.

ID: 554

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: 555

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: 556

Severity: ERROR
Message: The entry database does not contain a record for ID %s.

ID: 557

Severity: ERROR
Message: Execution error during backend operation: %s.

ID: 558

Severity: ERROR
Message: The backend database directory could not be created: %s.

ID: 559

Severity: WARNING
Message: This platform does not support setting file permissions %s to the database directory %s.

ID: 560

Severity: WARNING
Message: An error occurred while setting file permissions for the backend database directory %s: %s.

ID: 561

Severity: NOTICE
Message: The change to the DB directory will not take effect until the backend is restarted. The DB files from the previous directory %s must be moved to the new directory %s after shutting down the backend to retain the existing data.

ID: 562

Severity: ERROR
Message: The backend database directory '%s' is not a valid directory.

ID: 563

Severity: ERROR
Message: The entry '%s' cannot be added because an entry with that name already exists.

ID: 564

Severity: ERROR
Message: The entry '%s' cannot be added because its parent entry does not exist.

ID: 565

Severity: ERROR
Message: There is no index configured for attribute type '%s'.

ID: 566

Severity: ERROR
Message: An error occurred while attempting to decode an attribute description token from the compressed schema definitions: %s.

ID: 567

Severity: ERROR
Message: An error occurred while attempting to decode an object class set token from the compressed schema definitions: %s.

ID: 568

Severity: ERROR
Message: An error occurred while attempting to store compressed schema information in the database: %s.

ID: 569

Severity: ERROR
Message: An error occurred while parsing the search filter %s defined for VLV index %s: %s.

ID: 570

Severity: ERROR
Message: Sort attribute %s for VLV index %s is not defined in the server schema.

ID: 571

Severity: ERROR
Message: Database exception: %s.

ID: 572

Severity: ERROR
Message: A plugin caused the delete operation to be aborted while deleting a subordinate entry %s.

ID: 573

Severity: ERROR
Message: The entry '%s' cannot be removed because it has subordinate entries.

ID: 574

Severity: ERROR
Message: The entry '%s' cannot be removed because it does not exist.

ID: 575

Severity: ERROR
Message: An entry container named '%s' is already registered for base DN '%s'.

ID: 576

Severity: ERROR
Message: The entry database does not contain a valid record for ID %s.

ID: 577

Severity: ERROR
Message: I/O error occurred while exporting entry: %s.

ID: 578

Severity: ERROR
Message: The backend must be disabled before the import process can start.

ID: 579

Severity: ERROR
Message: Unable to create the temporary directory %s.

ID: 580

Severity: ERROR
Message: The import has been aborted because the entry '%s' does not have a parent entry.

ID: 581

Severity: ERROR
Message: Entry record is not compatible with this version of the backend database. Entry version: %x.

ID: 582

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: 583

Severity: ERROR
Message: The following paged results control cookie value was not recognized: %s.

ID: 584

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: 585

Severity: ERROR
Message: The entry cannot be renamed to '%s' because an entry with that name already exists.

ID: 586

Severity: ERROR
Message: The entry '%s' cannot be renamed because it does not exist.

ID: 587

Severity: ERROR
Message: The entry '%s' cannot be modified because it does not exist.

ID: 588

Severity: ERROR
Message: The entry cannot be moved because the new parent entry '%s' does not exist.

ID: 589

Severity: ERROR
Message: The database environment could not be opened: %s.

ID: 590

Severity: ERROR
Message: Rebuilding system index(es) must be done with the backend containing the base DN disabled.

ID: 591

Severity: ERROR
Message: The backend database files could not be removed: %s.

ID: 592

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: 593

Severity: ERROR
Message: The search results cannot be sorted because the given search request is not indexed.

ID: 594

Severity: ERROR
Message: The search base entry '%s' does not exist.

ID: 595

Severity: ERROR
Message: You do not have sufficient privileges to perform an unindexed search.

ID: 596

Severity: ERROR
Message: Unchecked exception during database transaction: %s.

ID: 597

Severity: ERROR
Message: There is no VLV index configured with name '%s'.

ID: 598

Severity: INFO
Message: The filter value exceeded the index entry limit for the %s index.

ID: 599

Severity: INFO
Message: %s index is invalid and needs to be rebuilt.

ID: 600

Severity: INFO
Message: %s index type is disabled for the %s attribute.

ID: 601

Severity: INFO
Message: Average number of entries referenced is %.2f/record.

ID: 602

Severity: INFO
Message: Free memory = %d MB, Cache miss rate = %.1f/record.

ID: 603

Severity: INFO
Message: Number of records that exceed the entry limit: %d.

ID: 604

Severity: INFO
Message: Statistics for records that have exceeded the entry limit:.

ID: 605

Severity: INFO
Message: File %s has %d such record(s) min=%d max=%d median=%d.

ID: 606

Severity: INFO
Message: Maximum number of entries referenced by any record is %d.

ID: 607

Severity: INFO
Message: Number of records referencing more than one entry: %d.

ID: 608

Severity: NOTICE
Message: The database backend %s containing %d entries has started.

ID: 609

Severity: NOTICE
Message: Some index keys have already exceeded the previous index entry limit in index %s. This index must be rebuilt before it can use the new limit.

ID: 610

Severity: NOTICE
Message: Exported %d entries and skipped %d in %d seconds (average rate %.1f/sec).

ID: 611

Severity: NOTICE
Message: Exported %d records and skipped %d (recent rate %.1f/sec).

ID: 612

Severity: NOTICE
Message: Flushing data to disk.

ID: 613

Severity: NOTICE
Message: Processed %d entries, imported %d, skipped %d and rejected %d in %d seconds (average rate %.1f/sec).

ID: 614

Severity: NOTICE
Message: Setting DB cache size to %d bytes and phase one buffer size to %d bytes.

ID: 615

Severity: NOTICE
Message: Index %s phase two processing completed.

ID: 616

Severity: NOTICE
Message: Index %s phase two started processing %d buffers in %d batches.

ID: 617

Severity: NOTICE
Message: Index %s %d%% complete: remaining = %d KB, rate = %d KB/s; batch %d/%d.

ID: 618

Severity: NOTICE
Message: Import LDIF environment close took %d seconds.

ID: 619

Severity: NOTICE
Message: The amount of free memory available to the import task is %d bytes. The number of phase one buffers required is %d buffers.

ID: 620

Severity: NOTICE
Message: Total import time was %d seconds. Phase one processing completed in %d seconds, phase two processing completed in %d seconds.

ID: 621

Severity: NOTICE
Message: Processed %d entries, skipped %d and rejected %d (recent rate %.1f/sec).

ID: 622

Severity: NOTICE
Message: %s starting import (build %s, R%s).

ID: 623

Severity: NOTICE
Message: Import Thread Count: %d threads.

ID: 624

Severity: NOTICE
Message: Due to changes in the configuration, index %s is currently operating in a degraded state and must be rebuilt before it can be used.

ID: 625

Severity: NOTICE
Message: Rebuild of all indexes started with %d total entries to process.

ID: 626

Severity: NOTICE
Message: Degraded state of index(es) %s has been cleared.

ID: 627

Severity: NOTICE
Message: Rebuild of all degraded indexes started with %d total entries to process.

ID: 628

Severity: NOTICE
Message: Rebuild complete. Processed %d entries in %d seconds (average rate %.1f/sec).

ID: 629

Severity: NOTICE
Message: %.1f%% Completed. Processed %d/%d entries. (recent rate %.1f/sec).

ID: 630

Severity: NOTICE
Message: Rebuild of index(es) %s started with %d total entries to process.

ID: 631

Severity: NOTICE
Message: A referral entry %s indicates that the operation must be processed at a different server.

ID: 632

Severity: NOTICE
Message: Checked %d records and found %d error(s) in %d seconds (average rate %.1f/sec).

ID: 633

Severity: NOTICE
Message: Checked %d entries and found %d error(s) in %d seconds (average rate %.1f/sec).

ID: 634

Severity: NOTICE
Message: Processed %d out of %d records and found %d error(s) (recent rate %.1f/sec).

ID: 635

Severity: WARNING
Message: The requested operation is not supported by this backend.

ID: 636

Severity: WARNING
Message: Unable to determine the total number of entries in the container: %s.

ID: 637

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: 638

Severity: ERROR
Message: Configuration attribute ds-cfg-db-cache-size has a value of %d which is less than the minimum: %d.

ID: 639

Severity: ERROR
Message: Missing ID %d%n%s.

ID: 640

Severity: ERROR
Message: Reference to unknown entry ID %s%n%s.

ID: 641

Severity: ERROR
Message: The entry with ID %s is associated with the wrong key%n%s.

ID: 642

Severity: ERROR
Message: Empty ID set: %n%s.

ID: 643

Severity: ERROR
Message: Duplicate reference to ID %d%n%s.

ID: 644

Severity: ERROR
Message: Reference to unknown ID %d%n%s.

ID: 645

Severity: ERROR
Message: Reference to entry <%s> which does not match the value%n%s.

ID: 646

Severity: ERROR
Message: File dn2id is missing key %s.

ID: 647

Severity: ERROR
Message: File dn2id has ID %d instead of %d for key %s.

ID: 648

Severity: ERROR
Message: File dn2id has DN <%s> referencing unknown ID %d.

ID: 649

Severity: ERROR
Message: File dn2id has DN <%s> referencing entry with wrong DN <%s>.

ID: 650

Severity: ERROR
Message: The stored entry count in id2entry (%d) does not agree with the actual number of entry records found (%d).

ID: 651

Severity: ERROR
Message: File id2childrenCount has wrong number of children for DN <%s> (got %d, expecting %d).

ID: 652

Severity: ERROR
Message: File id2ChildrenCount references non-existing EntryID <%d>.

ID: 653

Severity: NOTICE
Message: Rebuilding index finished: no indexes to rebuild.

ID: 654

Severity: NOTICE
Message: Setting DB cache size to %d MB. Using %d mb off-heap memory through %d phase one buffers of %d KB.

ID: 655

Severity: ERROR
Message: Ignoring schema definition '%s' because the following error occurred while it was being parsed: %s.

ID: 656

Severity: ERROR
Message: Schema definition could not be parsed as valid attribute value.

ID: 657

Severity: ERROR
Message: Attribute %s is set as confidential on a backend whose entries are still cleartext. Enable confidentiality on the backend first.

ID: 658

Severity: ERROR
Message: The attribute '%s' cannot enable confidentiality for keys and values at the same time.

ID: 659

Severity: ERROR
Message: Cannot encode entry for writing on storage: %s.

ID: 660

Severity: ERROR
Message: Input stream ended unexpectedly while decoding entry.

ID: 661

Severity: ERROR
Message: Confidentiality cannot be disabled on suffix '%s' because the following indexes have confidentiality still enabled: %s.

ID: 662

Severity: NOTICE
Message: Changing confidentiality for index '%s' requires the index to be rebuilt before it can be used again.

ID: 663

Severity: ERROR
Message: Error while enabling confidentiality with cipher %s, %d bits: %s.

ID: 664

Severity: ERROR
Message: The import has been aborted because the data to be imported contains duplicate copies of entry '%s'.

ID: 665

Severity: ERROR
Message: Proxy backend '%s' could not discover remote servers capabilities: %s.

ID: 666

Severity: INFO
Message: Proxy backend '%s' successfully configured failover via the service discovery mechanism '%s': primary servers are %s and secondary servers are %s.

ID: 667

Severity: WARNING
Message: Proxy backend '%s' cannot failover: only primary servers have been discovered via the service discovery mechanism '%s'. Primary servers are %s.

ID: 668

Severity: WARNING
Message: Proxy backend '%s' cannot failover: only secondary servers have been discovered via the service discovery mechanism '%s'. Secondary servers are %s.

ID: 669

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: 670

Severity: ERROR
Message: Proxy backend '%s' cannot find the configured service discovery mechanism '%s'.

ID: 671

Severity: NOTICE
Message: Connection options have changed for the proxy backend '%s'. The existing connections are being closed immediately. New ones are being opened.

ID: 672

Severity: NOTICE
Message: Connection pool options have changed for the proxy backend '%s'. The existing connections are being closed and re-opened.

ID: 673

Severity: NOTICE
Message: Service discovery mechanism has changed from '%s' to '%s' for proxy backend '%s'. The existing connections are being closed immediately.

ID: 674

Severity: NOTICE
Message: Remote servers changed for the proxy backend '%s'. The proxy was using: primary servers=%s, secondary servers=%s; and it will now be using: primary servers=%s, secondary servers=%s.

ID: 675

Severity: INFO
Message: Proxy backend '%s' automatically registered itself against the base DNs %s. It was previously registered against the base DNs %s.

ID: 676

Severity: ERROR
Message: No backend is associated with the base DN '%s'.

ID: 677

Severity: ERROR
Message: Proxy backend '%s' cannot register itself against base DN %s because this base DN is already registered against backend '%s'.

ID: 678

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: 679

Severity: INFO
Message: The change to the LDIF backend visibility will not take effect until the backend is disabled and re-enabled.

ID: 680

Severity: INFO
Message: The primary servers for proxy backend '%s' are now available and will be used for proxying requests.

ID: 681

Severity: INFO
Message: The secondary servers for proxy backend '%s' are now available and will be used if the primary servers are unavailable.

ID: 682

Severity: WARNING
Message: The primary servers for proxy backend '%s' are unavailable. The secondary servers, if available, will be used for processing requests until the primary servers become available again. The last failure that prevented the primary servers from being used was: %s.

ID: 683

Severity: WARNING
Message: The secondary servers for proxy backend '%s' are unavailable. If the primary servers are unavailable, or become unavailable, then it will no longer be possible to proxy requests. The last failure that prevented the secondary servers from being used was: %s.

ID: 684

Severity: INFO
Message: The server '%s' for proxy backend '%s' is available and will be used for proxying requests.

ID: 685

Severity: WARNING
Message: The server '%s' for proxy backend '%s' is unavailable. The last failure that prevented the server from being used was: %s.

ID: 686

Severity: ERROR
Message: The partition base DN '%s' should be subordinate to one of the base DNs %s of proxy backend '%s'.

ID: 687

Severity: ERROR
Message: Backend database cache preload for backend '%s' is not supported in this release.

ID: 688

Severity: INFO
Message: Configuration of proxy backend '%s' with service discovery mechanism '%s' is in progress. Requests will not be proxied until configuration completes.

ID: 689

Severity: WARNING
Message: 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 this backend's indexes are configured correctly for these search parameters.

ID: 690

Severity: NOTICE
Message: Initial result code was '%s', with diagnostic message: %s.

ID: 691

Severity: ERROR
Message: There are insufficient resources to perform the operation.

ID: 692

Severity: ERROR
Message: The time-to-live (TTL) feature can only be enabled for generalized time ordering indexes.

ID: 693

Severity: ERROR
Message: An unexpected error occurred while purging expired entries: %s.

ID: 694

Severity: WARNING
Message: Unable to locate all expired entries on backend '%s' because the ttl-enabled index '%s' has reached the configured index-entry-limit.

ID: 695

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: 696

Severity: WARNING
Message: The proxy backend '%s' will ignore the discovered servers '%s' from shard '%s' because they do not expose the required base DNs '%s'.

ID: 697

Severity: WARNING
Message: Property '%s' contains value '%s' which cannot be parsed as a long. Defaulting to '%s' bytes for small DB size.

ID: 698

Severity: ERROR
Message: An error occurred while trying to retrieve the key managers from the key manager provider `%s`.

ID: 699

Severity: ERROR
Message: Could not stop export-ldif threads after 30 seconds. Now forcing stop by interrupting them.

ID: 700

Severity: WARNING
Message: The file system used by '%s' backend does not support last file modification time. Backups will still work but will take longer and consume more space in the backup storage.

ID: 701

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: 702

Severity: ERROR
Message: VLV index '%s' must be configured with at least one sort attribute.

ID: 703

Severity: ERROR
Message: Missing entry %s in index %s.

ID: 704

Severity: ERROR
Message: Big index for attribute '%s' cannot be created for matching rule '%s' because it is not an equality matching rule.

ID: 705

Severity: WARNING
Message: The subtree delete against '%s' is being performed non-atomically because it is deleting more than %s subordinate entries. To prevent inconsistency across replicas, the client must retry this same delete operation until it succeeds.

ID: 706

Severity: ERROR
Message: A Server Side Sort control must be specified whenever a Virtual List View control is present.

ID: 707

Severity: ERROR
Message: Counter of %s reports wrong number of entries for key <%s> (got %d, expecting %d).

ID: 708

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: 709

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: 710

Severity: ERROR
Message: The VLV request cannot be processed because the search is not indexed. Configure a VLV index matching the request.

ID: 711

Severity: NOTICE
Message: Both equality and presence index types are defined for attribute '%s'. When no presence index exists, presence search filters are processed using the equality index. Therefore, consider deleting the presence index for attribute '%s'.

ID: 712

Severity: ERROR
Message: An internal error occurred when accessing backend '%s': %s.

ID: 713

Severity: ERROR
Message: An internal error was detected when accessing backend '%s'.

ID: 714

Severity: NOTICE
Message: Starting backup for backend '%s'.

ID: 715

Severity: NOTICE
Message: Starting restore for backend '%s' with backup ID '%s'.

ID: 716

Severity: NOTICE
Message: Backup completed for backend '%s' with backup ID '%s'.

ID: 717

Severity: NOTICE
Message: Restore completed for backend '%s' with backup ID '%s'.

ID: 718

Severity: NOTICE
Message: An error occurred while attempting to backup backend '%s': %s.

ID: 719

Severity: NOTICE
Message: The backup process failed with one or more errors.

ID: 720

Severity: NOTICE
Message: The restore process failed with one or more errors.

ID: 721

Severity: NOTICE
Message: An error occurred while attempting to restore backend '%s' with backup ID '%s': %s.

ID: 722

Severity: NOTICE
Message: There are no enabled backends that support backup operation.

ID: 723

Severity: NOTICE
Message: The backup command was interrupted.

ID: 724

Severity: NOTICE
Message: Cannot read backup directory content: %s.

ID: 725

Severity: NOTICE
Message: Backup still in progress, new files have been created.

ID: 726

Severity: NOTICE
Message: Backing up file (%d/%d) '%s'.

ID: 727

Severity: NOTICE
Message: Verifying file (%d/%d) '%s'.

ID: 728

Severity: NOTICE
Message: Restoring file (%d/%d) '%s'.

ID: 729

Severity: NOTICE
Message: Purging backup (%d/%d) '%s'.

ID: 730

Severity: NOTICE
Message: The following backends do not exist or are disabled: %s. Here is the list of enabled backends that support backups: %s.

ID: 731

Severity: NOTICE
Message: The following backends do not support backups: %s. Here is the list of enabled backends that support backups: %s.

ID: 732

Severity: NOTICE
Message: The backends %s cannot be restored because they do not exist or are disabled.

ID: 733

Severity: NOTICE
Message: Cannot find backup with ID %s.

ID: 734

Severity: NOTICE
Message: There are no backups for backends %s.

ID: 735

Severity: NOTICE
Message: The backends %s cannot be restored while the server is running. Please stop the server first and try again with the --offline option.

ID: 736

Severity: NOTICE
Message: '%s' is not a valid name for a directory.

ID: 737

Severity: NOTICE
Message: Found backup IDs that correspond to the same backend name: %s. Make sure each backup ID corresponds to a distinct backend.

ID: 738

Severity: NOTICE
Message: Either the --%s or --%s argument must be provided.

ID: 739

Severity: NOTICE
Message: Cannot delete corrupted file '%s'.

ID: 740

Severity: NOTICE
Message: Deleting corrupted file from backend directory.

ID: 741

Severity: NOTICE
Message: Deleting corrupted file from backup storage.

ID: 742

Severity: NOTICE
Message: signature does not match.

ID: 743

Severity: NOTICE
Message: unexpected content.

ID: 744

Severity: NOTICE
Message: Problem with file '%s' : %s.

ID: 745

Severity: NOTICE
Message: IO error: %s.

ID: 746

Severity: NOTICE
Message: Operation was interrupted.

ID: 747

Severity: NOTICE
Message: File '%s' is corrupted: %s.

ID: 748

Severity: NOTICE
Message: File '%s' is missing.

ID: 749

Severity: NOTICE
Message: The directory does not exist.

ID: 750

Severity: NOTICE
Message: Not a directory.

ID: 751

Severity: NOTICE
Message: Cannot read backend files in directory '%s': %s.

ID: 752

Severity: NOTICE
Message: Cannot delete backend files from directory '%s': %s.

ID: 753

Severity: NOTICE
Message: The '%s' backend files have been removed after restore failed. The backend is now empty.

ID: 754

Severity: NOTICE
Message: Cannot create the backend directory '%s' before restoring: %s.

ID: 755

Severity: NOTICE
Message: The attempt to remove backend files from directory '%s' has failed, backend '%s' may contain corrupted data, please remove the files manually.

ID: 756

Severity: NOTICE
Message: The backup ID '%s' already exists.

ID: 757

Severity: NOTICE
Message: Cannot compute the fingerprint for backend file '%s': %s.

ID: 758

Severity: NOTICE
Message: Unable to access the backup storage: %s.

ID: 759

Severity: NOTICE
Message: No plugin found to handle the storage scheme '%s'.

ID: 760

Severity: NOTICE
Message: Cannot delete the backup lock file '%s', please remove this file manually: %s.

ID: 761

Severity: NOTICE
Message: Cannot 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: 762

Severity: NOTICE
Message: Cannot release the shared lock for backend '%s': %s. This lock should automatically be cleared when the backup process exits, so no further action should be required.

ID: 763

Severity: NOTICE
Message: Cannot 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).

ID: 764

Severity: NOTICE
Message: Cannot release the exclusive lock for backend '%s': %s. This lock should automatically be cleared when the restore process exits, so no further action should be required.

ID: 765

Severity: NOTICE
Message: Cannot enable backend '%s' after restoring: %s.

ID: 766

Severity: NOTICE
Message: Backup ID: %s.

ID: 767

Severity: NOTICE
Message: Backup Date: %s.

ID: 768

Severity: NOTICE
Message: Backend name: %s.

ID: 769

Severity: NOTICE
Message: Server ID: %s.

ID: 770

Severity: NOTICE
Message: Status: %s.

ID: 771

Severity: NOTICE
Message: Error: %s.

ID: 772

Severity: NOTICE
Message: Verified.

ID: 773

Severity: NOTICE
Message: Bad.

ID: 774

Severity: NOTICE
Message: Found %s backup(s) with the requested characteristics.

ID: 775

Severity: NOTICE
Message: There are no backups with the requested characteristics.

ID: 776

Severity: NOTICE
Message: %s backup(s) cannot be restored.

ID: 777

Severity: NOTICE
Message: Found %s backup(s) that cannot be fully identified due to the following error(s):.

ID: 778

Severity: NOTICE
Message: Backup and restore backends, manage backup files.

ID: 779

Severity: NOTICE
Message: Backup and restore backends.

ID: 780

Severity: NOTICE
Message: {backup location}.

ID: 781

Severity: NOTICE
Message: 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 %1$s sub-command, the server must be stopped; also as the command will write to server files, you should run the command as a user having the same filesystem permissions as the user running the server. Using this option with the %2$s 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.

ID: 782

Severity: NOTICE
Message: 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.

ID: 783

Severity: NOTICE
Message: 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.

ID: 784

Severity: NOTICE
Message: {PROP:VALUE}.

ID: 785

Severity: NOTICE
Message: 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.

ID: 786

Severity: NOTICE
Message: List the backups at the specified location.

ID: 787

Severity: NOTICE
Message: Show only backups taken from the provided backend.

ID: 788

Severity: NOTICE
Message: Show only backups taken from the provided server.

ID: 789

Severity: NOTICE
Message: {server ID}.

ID: 790

Severity: NOTICE
Message: Show only the last backup for each backend.

ID: 791

Severity: NOTICE
Message: Verify backups completeness, integrity and whether they can be decrypted.

ID: 792

Severity: NOTICE
Message: Take encrypted and signed backups of individual backends and send them to the desired location.

ID: 793

Severity: NOTICE
Message: 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.

ID: 794

Severity: NOTICE
Message: {backup ID}.

ID: 795

Severity: NOTICE
Message: 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.

ID: 796

Severity: NOTICE
Message: Restore the backup having the provided ID. Specify this option multiple times to restore multiple backends.

ID: 797

Severity: NOTICE
Message: Restore the last backup of the provided backend. Specify this option multiple times to restore multiple backends.

ID: 798

Severity: NOTICE
Message: Delete one or more backups.

ID: 799

Severity: NOTICE
Message: The ID of the backup that should be deleted. Specify this option multiple times to purge multiple backups.

ID: 800

Severity: NOTICE
Message: At least one of these options must be declared: %s.

ID: 801

Severity: NOTICE
Message: The option '--%s' can only be used with the option '--%s'.

ID: 802

Severity: NOTICE
Message: {number of backups}.

ID: 803

Severity: NOTICE
Message: 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.

ID: 804

Severity: NOTICE
Message: {duration}.

ID: 805

Severity: NOTICE
Message: Delete backups that are older than the provided duration. The latest backup of each backend will always be kept unless the '--%s' option is also provided. Duration examples: '12 hours', '3 days', '1y'.

ID: 806

Severity: NOTICE
Message: Must be used with the '--%s' option, indicates that the last backup of each backend can be deleted if older than the provided duration.

ID: 807

Severity: NOTICE
Message: {backend name}.

ID: 808

Severity: NOTICE
Message: 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 '--%s' or '--%s'.

ID: 809

Severity: NOTICE
Message: '%d' is not allowed: the value must be equal or greater than 0.

ID: 810

Severity: NOTICE
Message: Invalid value for '%s': %s.

ID: 811

Severity: NOTICE
Message: An error occurred while attempting to purge the backup '%s': %s.

ID: 812

Severity: NOTICE
Message: An error occurred while attempting to purge backup files starting with '%s': %s.

ID: 813

Severity: NOTICE
Message: Backup purge process failed with one or more errors.

ID: 814

Severity: NOTICE
Message: Purge completed successfully.

ID: 815

Severity: NOTICE
Message: The storage property '%s' has several values while it can have only one. The provided properties string was: "%s".

ID: 816

Severity: ERROR
Message: Backup id '%s' is invalid: expected to find an underscore in it.

ID: 817

Severity: ERROR
Message: Backup id '%s' is invalid: cannot parse '%s' as a timestamp.

ID: 818

Severity: NOTICE
Message: Access denied to file '%s'.

ID: 819

Severity: ERROR
Message: Credential file '%s' not found.

ID: 820

Severity: ERROR
Message: Failed to parse the credential file '%s'.

ID: 821

Severity: ERROR
Message: Invalid plugin type '%s', a backup plugin must be of type '%s'.

ID: 822

Severity: ERROR
Message: A problem occurred while connecting to cloud provider '%s' using the provided credentials: %s.

ID: 823

Severity: ERROR
Message: No value found for property '%s'.

ID: 824

Severity: ERROR
Message: No environment value '%s' found for property '%s'. If you are running the tool in offline mode, make sure the command has access to this variable, or else make sure the directory server has access to it.

ID: 825

Severity: ERROR
Message: Could not find file '%s' in cloud bucket '%s'.

ID: 826

Severity: ERROR
Message: An error occurred while uploading the file to the cloud storage: the host '%s' is an unknown host.

ID: 827

Severity: ERROR
Message: An error occurred while uploading the file to the cloud storage. The cloud provider returned the following error message: '%s'.

ID: 828

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: 829

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: 830

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: 831

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: 832

Severity: ERROR
Message: The specified configuration file %s does not exist or is not readable.

ID: 833

Severity: ERROR
Message: An error occurred while attempting to open the configuration file %s for reading: %s.

ID: 834

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: 835

Severity: ERROR
Message: An unexpected error occurred while attempting to process the Directory Server configuration file %s: %s.

ID: 836

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: 837

Severity: ERROR
Message: An unexpected error occurred while trying to write configuration entry %s to LDIF: %s.

ID: 838

Severity: ERROR
Message: The Directory Server configuration may not be altered by importing a new configuration from LDIF.

ID: 839

Severity: WARNING
Message: There are no active access loggers defined in the Directory Server configuration. No access logging will be performed.

ID: 840

Severity: WARNING
Message: There are no active error loggers defined in the Directory Server configuration. No error logging will be performed.

ID: 841

Severity: ERROR
Message: An error occurred while attempting to create a Directory Server logger from the information in configuration entry %s: %s.

ID: 842

Severity: ERROR
Message: Configuration entry %s does not contain a valid objectclass for a Directory Server access, error, or debug logger definition.

ID: 843

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: 844

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: 845

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: 846

Severity: ERROR
Message: Configuration entry %s does not contain attribute %s (or that attribute exists but is not accessible using JMX).

ID: 847

Severity: ERROR
Message: There is no method %s for any invokable component registered with configuration entry %s.

ID: 848

Severity: ERROR
Message: The Directory Server could not register a JMX MBean for the component associated with configuration entry %s: %s.

ID: 849

Severity: ERROR
Message: An unexpected error occurred while trying to export the Directory Server configuration to LDIF: %s.

ID: 850

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: 851

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: 852

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: 853

Severity: INFO
Message: The backend defined in configuration entry %s is marked as disabled and therefore will not be used.

ID: 854

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: 855

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: 856

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: 857

Severity: NOTICE
Message: The requested change to configuration entry %s would cause the class for the associated backend to change from %s to %s. This change will not take effect until the backend is disabled and re-enabled, or until the Directory Server is restarted.

ID: 858

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: 859

Severity: ERROR
Message: Unable to read the Directory Server schema definitions because the schema directory %s does not exist.

ID: 860

Severity: ERROR
Message: Unable to read the Directory Server schema definitions because the schema directory %s exists but is not a directory.

ID: 861

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: 862

Severity: WARNING
Message: Schema configuration file %s in directory %s cannot be parsed because an unexpected error occurred while trying to open the file for reading: %s.

ID: 863

Severity: WARNING
Message: Schema configuration file %s in directory %s cannot be parsed because an unexpected error occurred while trying to read its contents as an LDIF entry: %s.

ID: 864

Severity: ERROR
Message: An unexpected error occurred that prevented the server from installing its default entry cache framework: %s.

ID: 865

Severity: WARNING
Message: The entry cache configuration entry "cn=Entry Caches,cn=config" does not exist in the Directory Server configuration. No entry cache will be available until this entry is created with a valid entry cache configuration.

ID: 866

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: 867

Severity: ERROR
Message: The configuration for the entry cache defined in configuration entry %s was not acceptable: %s.

ID: 868

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: 869

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: 870

Severity: ERROR
Message: Class %s specified in configuration entry %s does not contain a valid extended operation handler implementation: %s.

ID: 871

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: 872

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: 873

Severity: ERROR
Message: Entry %s cannot be removed from the Directory Server configuration because that DN does not have a parent.

ID: 874

Severity: ERROR
Message: Entry %s cannot be added to the Directory Server configuration because another configuration entry already exists with that DN.

ID: 875

Severity: ERROR
Message: Entry %s cannot be added to the Directory Server configuration because that DN does not have a parent.

ID: 876

Severity: ERROR
Message: Entry %s cannot be added to the Directory Server configuration because its parent entry %s does not exist.

ID: 877

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: 878

Severity: ERROR
Message: An unexpected error occurred while attempting to add configuration entry %s as a child of entry %s: %s.

ID: 879

Severity: ERROR
Message: Entry %s cannot be removed from the Directory Server configuration because the specified entry does not exist.

ID: 880

Severity: ERROR
Message: Entry %s cannot be removed from the Directory Server configuration because the specified entry has one or more subordinate entries.

ID: 881

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: 882

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: 883

Severity: ERROR
Message: An unexpected error occurred while attempting to remove configuration entry %s as a child of entry %s: %s.

ID: 884

Severity: ERROR
Message: Entry %s cannot be modified because the specified entry does not exist.

ID: 885

Severity: ERROR
Message: Entry %s cannot be modified because one of the configuration change listeners registered for that entry rejected the change: %s.

ID: 886

Severity: ERROR
Message: An unexpected error occurred while attempting to modify configuration entry %s as a child of entry %s: %s.

ID: 887

Severity: ERROR
Message: An error occurred while attempting to export the new Directory Server configuration to file %s: %s.

ID: 888

Severity: ERROR
Message: An error occurred while attempting to rename the new Directory Server configuration from file %s to %s: %s.

ID: 889

Severity: ERROR
Message: Modify DN operations are not allowed in the Directory Server configuration.

ID: 890

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: 891

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: 892

Severity: WARNING
Message: The backend defined in configuration entry %s has a backend ID of %s that conflicts with the backend ID for another backend in the server. The backend will be disabled.

ID: 893

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: 894

Severity: WARNING
Message: An error occurred while attempting to release a shared lock for backend %s: %s. This may interfere with operations that require exclusive access, including LDIF import and restoring a backup.

ID: 895

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: 896

Severity: ERROR
Message: An error occurred while attempting to instantiate class %s referenced in synchronization provider configuration entry %s: %s.

ID: 897

Severity: ERROR
Message: An error occurred while attempting to initialize the Directory Server synchronization provider referenced in configuration entry %s: %s.

ID: 898

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: 899

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: 900

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: 901

Severity: ERROR
Message: The password policy defined in configuration entry %s is invalid: %s.

ID: 902

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: 903

Severity: WARNING
Message: The specified entry %s is currently defined as the configuration entry for the default password policy. The default password policy configuration entry may not be removed.

ID: 904

Severity: INFO
Message: Password policy entry %s has been removed from the Directory Server configuration. Any user entries that explicitly reference this password policy will no longer be allowed to authenticate.

ID: 905

Severity: WARNING
Message: Access control has been disabled.

ID: 906

Severity: NOTICE
Message: Access control has been enabled and will use the %s implementation.

ID: 907

Severity: ERROR
Message: An error occurred while attempting to instantiate class %s referenced in the access control configuration entry %s: %s.

ID: 908

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: 909

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: 910

Severity: ERROR
Message: An error occurred while attempting to apply the changes contained in file %s to the server configuration at startup: %s.

ID: 911

Severity: ERROR
Message: One or more errors occurred while applying changes on server startup: %s.

ID: 912

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: 913

Severity: ERROR
Message: Invalid UNIX file permissions %s does not allow read and write access to the backend database directory by the backend.

ID: 914

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: 915

Severity: WARNING
Message: An error occurred while attempting to register backend %s with the Directory Server: %s.

ID: 916

Severity: ERROR
Message: An error occurred while trying to create the configuration archive directory %s: %s.

ID: 917

Severity: ERROR
Message: An error occurred while trying to write the current configuration to the configuration archive: %s.

ID: 918

Severity: ERROR
Message: You do not have sufficient privileges to perform add operations in the Directory Server configuration.

ID: 919

Severity: ERROR
Message: You do not have sufficient privileges to perform delete operations in the Directory Server configuration.

ID: 920

Severity: ERROR
Message: You do not have sufficient privileges to perform modify operations in the Directory Server configuration.

ID: 921

Severity: ERROR
Message: You do not have sufficient privileges to perform modify DN operations in the Directory Server configuration.

ID: 922

Severity: ERROR
Message: You do not have sufficient privileges to perform search operations in the Directory Server configuration.

ID: 923

Severity: ERROR
Message: You do not have sufficient privileges to change the set of default root privileges.

ID: 924

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: 925

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: 926

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: 927

Severity: ERROR
Message: Unable to retrieve JMX attribute %s associated with configuration entry %s: %s.

ID: 928

Severity: ERROR
Message: %s.%s returned a result of null for entry %s.

ID: 929

Severity: ERROR
Message: %s.%s failed for entry %s: result code=%s, admin action required=%b, messages="%s".

ID: 930

Severity: WARNING
Message: %s.%s indicated that administrative action is required for entry %s: messages="%s".

ID: 931

Severity: INFO
Message: %s.%s succeeded but generated the following messages for entry %s: %s.

ID: 932

Severity: ERROR
Message: Unable to parse value "%s" from config entry "%s" as a valid search filter: %s.

ID: 933

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: 934

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: 935

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: 936

Severity: ERROR
Message: Configuration entry %s cannot be modified because the change would alter its structural object class.

ID: 937

Severity: ERROR
Message: An error occurred while attempting to calculate a SHA-1 digest of file %s: %s.

ID: 938

Severity: WARNING
Message: The Directory Server has detected that one or more external changes have been made to the configuration file %s while the server was online, but another change has caused the server configuration to be overwritten. The manual changes have not been applied, but they have been preserved in file %s.

ID: 939

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: 940

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: 941

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: 942

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: 943

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: 944

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: 945

Severity: WARNING
Message: Schema configuration file %s in directory %s contains more than one entry. Only the first entry will be examined, and the additional entries will be ignored.

ID: 946

Severity: WARNING
Message: The plugin order definition for plugins of type %s contains an empty element. This may cause the plugin order to be evaluated incorrectly.

ID: 947

Severity: WARNING
Message: The plugin order definition for plugins of type %s contains multiple wildcard characters. All plugin definitions should contain exactly one wildcard element to indicate where unmatched plugins should be included in the order, and including multiple wildcards may cause the plugin order to be evaluated incorrectly.

ID: 948

Severity: WARNING
Message: The plugin order definition for plugins of type %s includes multiple references to the '%s' plugin. This may cause the plugin order to be evaluated incorrectly.

ID: 949

Severity: WARNING
Message: The plugin order definition for plugins of type %s does not include a wildcard element to indicate where unmatched plugins should be included in the order. The server will default to invoking all unnamed plugins after set of named plugins.

ID: 950

Severity: ERROR
Message: Unable to initialize an instance of class %s as a work queue as specified in configuration entry %s: %s.

ID: 951

Severity: INFO
Message: The class used to provide the Directory Server work queue implementation has been changed from %s to %s, but this change will not take effect until the server is restarted.

ID: 952

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: 953

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: 954

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: 955

Severity: ERROR
Message: The configuration for the key manager provider defined in configuration entry %s was not acceptable: %s.

ID: 956

Severity: ERROR
Message: The configuration for the trust manager provider defined in configuration entry %s was not acceptable: %s.

ID: 957

Severity: ERROR
Message: The configuration for the trust manager provider defined in configuration entry %s was not acceptable: %s.

ID: 958

Severity: ERROR
Message: The configuration for the account status notification handler defined in configuration entry %s was not acceptable: %s.

ID: 959

Severity: ERROR
Message: The configuration for the certificate mapper defined in configuration entry %s was not acceptable: %s.

ID: 960

Severity: ERROR
Message: The configuration for the identity mapper defined in configuration entry %s was not acceptable: %s.

ID: 961

Severity: ERROR
Message: The configuration for the password generator defined in configuration entry %s was not acceptable: %s.

ID: 962

Severity: ERROR
Message: The configuration for the password storage scheme defined in configuration entry %s was not acceptable: %s.

ID: 963

Severity: ERROR
Message: The configuration for the password validator defined in configuration entry %s was not acceptable: %s.

ID: 964

Severity: ERROR
Message: The configuration for the plugin defined in configuration entry %s was not acceptable: %s.

ID: 965

Severity: ERROR
Message: The configuration for the SASL mechanism handler defined in configuration entry %s was not acceptable: %s.

ID: 966

Severity: ERROR
Message: The configuration for the virtual attribute provider defined in configuration entry %s was not acceptable: %s.

ID: 967

Severity: ERROR
Message: The configuration for the alert handler defined in configuration entry %s was not acceptable: %s.

ID: 968

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: 969

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: 970

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: 971

Severity: ERROR
Message: An error occurred while attempting to rename file %s to %s for use as the ".startok" configuration file: %s.

ID: 972

Severity: NOTICE
Message: The Directory Server is starting using the last known good configuration file %s rather than the active configuration file %s.

ID: 973

Severity: WARNING
Message: No last known good configuration file %s exists. The server will attempt to start using the active configuration file %s.

ID: 974

Severity: ERROR
Message: An error occurred while trying to parse and validate Berkeley DB JE property %s: %s.

ID: 975

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: 976

Severity: ERROR
Message: An error occurred while trying to parse and validate Berkeley DB JE property %s: the property shadows configuration attribute %s.

ID: 977

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: 978

Severity: ERROR
Message: An error occurred while attempting to open the configured log file %s for logger %s: %s.

ID: 979

Severity: ERROR
Message: Invalid UNIX file permissions %s does not allow write access to the log file by the log publisher.

ID: 980

Severity: ERROR
Message: Invalid UNIX file permissions %s: %s.

ID: 981

Severity: ERROR
Message: The configuration entry '%s' is currently defined to be the default password policy, however it is not a password policy.

ID: 982

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: 983

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: 984

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: 985

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: 986

Severity: WARNING
Message: There are no active HTTP access loggers defined in the Directory Server configuration. No HTTP access logging will be performed.

ID: 987

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: 988

Severity: WARNING
Message: The log format for %s contains the following unsupported fields: %s. Their output will be replaced with a dash ("-") character.

ID: 989

Severity: ERROR
Message: An error occurred while attempting to update a Directory Server logger from the information in configuration entry %s: %s.

ID: 990

Severity: ERROR
Message: Cannot configure java.util.logging root logger level: %s. java.util.logging support is now disabled.

ID: 991

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: 992

Severity: ERROR
Message: An error occurred while starting the HTTP endpoint as defined in configuration entry %s: %s.

ID: 993

Severity: ERROR
Message: The HTTP endpoint configuration defined in %s is invalid: %s.

ID: 994

Severity: ERROR
Message: Cannot initialize the configuration framework: %s.

ID: 995

Severity: ERROR
Message: Unable to retrieve children of configuration entry with dn: %s.

ID: 996

Severity: ERROR
Message: Unable to load the configuration-enabled schema: %s.

ID: 997

Severity: ERROR
Message: Backend config error when trying to delete an entry: %s.

ID: 998

Severity: ERROR
Message: The HTTP endpoint configuration defined in %s is referencing a non existing authorization DN %s.

ID: 999

Severity: ERROR
Message: The HTTP endpoint configuration defined in %s is referencing mutually exclusive authorization DNs %s and %s.

ID: 1000

Severity: ERROR
Message: Unable to read the configuration from %s in the REST2LDAP endpoint configuration entry %s: %s.

IDs: 1001-1500

ID: 1001

Severity: ERROR
Message: Invalid JSON element %s from %s in the REST2LDAP endpoint configuration entry %s: %s.

ID: 1002

Severity: ERROR
Message: Invalid configuration element from %s in the REST2LDAP endpoint configuration entry %s: %s.

ID: 1003

Severity: ERROR
Message: The OAuth2 authorization mechanism defined in %s contains an invalid JSON Pointer %s: %s.

ID: 1004

Severity: ERROR
Message: The authorization mechanism defined in %s is referencing a non-existing or non-readable directory: %s.

ID: 1005

Severity: ERROR
Message: The authorization mechanism defined in %s is referencing a non existing DN: %s.

ID: 1006

Severity: ERROR
Message: The authorization mechanism defined in %s is referencing an invalid URL %s: %s.

ID: 1007

Severity: ERROR
Message: Unable to configure the authorization mechanism defined in %s: %s.

ID: 1008

Severity: ERROR
Message: The requested admin API version '%s' is unsupported. This endpoint only supports the following admin API version(s): %s.

ID: 1009

Severity: ERROR
Message: The configuration of schema provider '%s' is not acceptable for the following reasons: %s.

ID: 1010

Severity: ERROR
Message: The schema provider class '%s' could not be instantiated or initialized with the configuration '%s' : %s.

ID: 1011

Severity: ERROR
Message: The core schema provider defined by '%s' has been disabled. The core schema must always be enabled.

ID: 1012

Severity: WARNING
Message: The config schema file '%s' generated warning when trying to update schema with its content: %s.

ID: 1013

Severity: WARNING
Message: The config schema file '%s' generated warning when trying to update schema with its content, despite allowing to overwrite definitions: %s.

ID: 1014

Severity: ERROR
Message: Unable to configure the backend '%s' because one of its base DNs is the empty DN.

ID: 1015

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: 1016

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: 1017

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: 1018

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: 1019

Severity: ERROR
Message: The list of keys defined for the JSON matching rule contains an invalid JSON pointer : %s.

ID: 1020

Severity: ERROR
Message: Cannot create the property resolver due to the following error: '%s'.

ID: 1021

Severity: ERROR
Message: Error creating SSL socket factory: %s.

ID: 1022

Severity: ERROR
Message: The smtp-server value '%s' is invalid: %s.

ID: 1023

Severity: ERROR
Message: Unable to resolve the host for the listen address '%s' of the LDAP connection handler %s.

ID: 1024

Severity: WARNING
Message: The latest archived configuration file %s appears to be invalid. Forcing creation of a new one.

ID: 1025

Severity: WARNING
Message: '%s' is DEPRECATED for removal since %s. Its usage is highly discouraged.

ID: 1026

Severity: WARNING
Message: '%s' is LEGACY since %s. Its usage is highly discouraged.

ID: 1027

Severity: ERROR
Message: Abandon requests cannot be canceled.

ID: 1028

Severity: ERROR
Message: Bind requests cannot be canceled.

ID: 1029

Severity: ERROR
Message: Unbind requests cannot be canceled.

ID: 1030

Severity: INFO
Message: Client Unbind.

ID: 1031

Severity: INFO
Message: Client Disconnect.

ID: 1032

Severity: INFO
Message: Client Connection Rejected.

ID: 1033

Severity: INFO
Message: I/O Error.

ID: 1034

Severity: INFO
Message: Protocol Error.

ID: 1035

Severity: INFO
Message: Server Shutdown.

ID: 1036

Severity: INFO
Message: Administrative Termination.

ID: 1037

Severity: INFO
Message: Security Problem.

ID: 1038

Severity: INFO
Message: Maximum Request Size Exceeded.

ID: 1039

Severity: INFO
Message: Administrative Limit Exceeded.

ID: 1040

Severity: INFO
Message: Idle Time Limit Exceeded.

ID: 1041

Severity: INFO
Message: I/O Timeout.

ID: 1042

Severity: INFO
Message: Connection Closed by Plugin.

ID: 1043

Severity: INFO
Message: Unknown Closure Reason.

ID: 1044

Severity: INFO
Message: Operations Error.

ID: 1045

Severity: INFO
Message: Invalid Credentials.

ID: 1046

Severity: INFO
Message: Processing on this operation has been canceled because the Directory Server is shutting down.

ID: 1047

Severity: ERROR
Message: %s encountered an uncaught exception while processing operation %s: %s.

ID: 1048

Severity: WARNING
Message: %s is unexpectedly exiting when the Directory Server is not in the process of shutting down. This likely indicates that the thread encountered an unexpected error.

ID: 1049

Severity: WARNING
Message: The request to process this operation has been rejected because the Directory Server has already started its shutdown process.

ID: 1050

Severity: WARNING
Message: %s was interrupted while waiting for new work: %s. This should not happen, but the thread will resume waiting for new work so there should be no adverse effects.

ID: 1051

Severity: WARNING
Message: An unexpected exception was caught while %s was waiting for new work: %s. This should not happen, but the thread will resume waiting for new work so there should be no adverse effects.

ID: 1052

Severity: WARNING
Message: The work queue caught an exception while trying to cancel pending operation %s when the Directory Server was shutting down: %s.

ID: 1053

Severity: INFO
Message: Server Error.

ID: 1054

Severity: ERROR
Message: The Directory Server is currently running. The configuration may not be bootstrapped while the server is online.

ID: 1055

Severity: ERROR
Message: The Directory Server may not be started before the configuration has been bootstrapped.

ID: 1056

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: 1057

Severity: INFO
Message: The Directory Server is beginning the configuration bootstrapping process.

ID: 1058

Severity: NOTICE
Message: %s (build %s, revision number %s) starting up.

ID: 1059

Severity: NOTICE
Message: The Directory Server has started successfully.

ID: 1060

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: 1061

Severity: NOTICE
Message: The Directory Server has sent an alert notification generated by class %s (alert type %s, alert ID %s): %s.

ID: 1062

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: 1063

Severity: NOTICE
Message: The Directory Server has started the shutdown process. The shutdown was initiated by an instance of class %s and the reason provided for the shutdown was %s.

ID: 1064

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: 1065

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: 1066

Severity: WARNING
Message: Entry "%s" contains a value "%s" for attribute %s that is invalid according to the syntax for that attribute: %s.

ID: 1067

Severity: WARNING
Message: Entry "%s" does not contain any values for attribute "%s".

ID: 1068

Severity: WARNING
Message: Entry "%s" does not contain any values for attribute "%s" with the specified set of options.

ID: 1069

Severity: NOTICE
Message: The Directory Server is now stopped.

ID: 1070

Severity: INFO
Message: %s has been stopped because the total number of worker threads in the Directory Server was reduced.

ID: 1071

Severity: INFO
Message: Processing on this operation has been canceled because the Directory Server received a bind request on this connection, which requires that all operations in progress to be abandoned.

ID: 1072

Severity: ERROR
Message: Unable to bind to the Directory Server because no such user exists in the server.

ID: 1073

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: 1074

Severity: ERROR
Message: Unable to bind to the Directory Server using simple authentication because that user does not have a password.

ID: 1075

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: 1076

Severity: ERROR
Message: The specified entry %s does not exist in the Directory Server.

ID: 1077

Severity: INFO
Message: The operation was canceled because the client issued an abandon request (message ID %d) for this operation.

ID: 1078

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: 1079

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: 1080

Severity: ERROR
Message: Entry %s cannot be added because its parent entry %s does not exist in the server.

ID: 1081

Severity: ERROR
Message: Entry %s cannot be added because the server failed to obtain a write lock for this entry after multiple attempts.

ID: 1082

Severity: ERROR
Message: Entry %s cannot be removed because the server failed to obtain a write lock for this entry after multiple attempts.

ID: 1083

Severity: ERROR
Message: The maximum time limit of %d seconds for processing this search operation has expired.

ID: 1084

Severity: ERROR
Message: This search operation has sent the maximum of %d entries to the client.

ID: 1085

Severity: ERROR
Message: The entry %s specified as the search base does not exist in the Directory Server.

ID: 1086

Severity: ERROR
Message: Entry %s does not exist in the Directory Server.

ID: 1087

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: 1088

Severity: ERROR
Message: A modify DN operation cannot be performed on entry %s because the new RDN would not have a parent DN.

ID: 1089

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because no backend is registered to handle that DN.

ID: 1090

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: 1091

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: 1092

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: 1093

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: 1094

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because that entry does not exist in the server.

ID: 1095

Severity: ERROR
Message: Entry %s cannot be modified because the server failed to obtain a write lock for this entry after multiple attempts.

ID: 1096

Severity: ERROR
Message: Entry %s cannot be modified because no such entry exists in the server.

ID: 1097

Severity: ERROR
Message: Entry %s cannot be modified because the modification contained an add component for attribute %s but no values were provided.

ID: 1098

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: 1099

Severity: ERROR
Message: Entry %s cannot be modified because it would have resulted in one or more duplicate values for attribute %s: %s.

ID: 1100

Severity: ERROR
Message: Entry %s cannot be modified because the change to attribute %s would have removed a value used in the RDN.

ID: 1101

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: 1102

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: 1103

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: 1104

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: 1105

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: 1106

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: 1107

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: 1108

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: 1109

Severity: ERROR
Message: Entry %s cannot be modified because the resulting entry would have violated the server schema: %s.

ID: 1110

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: 1111

Severity: ERROR
Message: An unexpected error was encountered while processing a search in one of the Directory Server backends: %s.

ID: 1112

Severity: ERROR
Message: The modify DN operation for entry %s cannot be performed because the change would have violated the server schema: %s.

ID: 1113

Severity: ERROR
Message: Object class %s cannot be added to entry %s because that class is not defined in the Directory Server schema.

ID: 1114

Severity: ERROR
Message: The password provided by the user did not match any password(s) stored in the user’s entry.

ID: 1115

Severity: INFO
Message: Display general system information.

ID: 1116

Severity: ERROR
Message: An error occurred while attempting to parse the provided set of command line arguments: %s.

ID: 1117

Severity: ERROR
Message: An error occurred while attempting to bootstrap the Directory Server: %s.

ID: 1118

Severity: ERROR
Message: An error occurred while trying to start the Directory Server: %s.

ID: 1119

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: 1120

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: 1121

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: 1122

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: 1123

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: 1124

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: 1125

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: 1126

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: 1127

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: 1128

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: 1129

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: 1130

Severity: ERROR
Message: The attempt to release the exclusive lock held on %s failed: %s.

ID: 1131

Severity: ERROR
Message: The attempt to release the shared lock held on %s failed: %s.

ID: 1132

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: 1133

Severity: WARNING
Message: An error occurred while attempting to release a shared lock for backend %s: %s. This lock should be automatically cleaned when the Directory Server process exits, so no additional action should be necessary.

ID: 1134

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: 1135

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: 1136

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: 1137

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: 1138

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: 1139

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: 1140

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: 1141

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: 1142

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: 1143

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: 1144

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: 1145

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: 1146

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: 1147

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: 1148

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: 1149

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: 1150

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: 1151

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: 1152

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: 1153

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: 1154

Severity: INFO
Message: The add operation was not actually performed in the Directory Server backend because the LDAP no-op control was present in the request.

ID: 1155

Severity: INFO
Message: The delete operation was not actually performed in the Directory Server backend because the LDAP no-op control was present in the request.

ID: 1156

Severity: INFO
Message: The modify operation was not actually performed in the Directory Server backend because the LDAP no-op control was present in the request.

ID: 1157

Severity: INFO
Message: The modify DN operation was not actually performed in the Directory Server backend because the LDAP no-op control was present in the request.

ID: 1158

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: 1159

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: 1160

Severity: WARNING
Message: There are multiple user-specific size limit values contained in user entry %s. The default server size limit will be used.

ID: 1161

Severity: WARNING
Message: The user-specific size limit value %s contained in user entry %s could not be parsed as an integer. The default server size limit will be used.

ID: 1162

Severity: WARNING
Message: There are multiple user-specific time limit values contained in user entry %s. The default server time limit will be used.

ID: 1163

Severity: WARNING
Message: The user-specific time limit value %s contained in user entry %s could not be parsed as an integer. The default server time limit will be used.

ID: 1165

Severity: ERROR
Message: An error occurred during preoperation synchronization processing for the add operation with connection ID %d and operation ID %d: %s.

ID: 1166

Severity: ERROR
Message: An error occurred during postoperation synchronization processing for the add operation with connection ID %d and operation ID %d: %s.

ID: 1167

Severity: ERROR
Message: An error occurred during preoperation synchronization processing for the delete operation with connection ID %d and operation ID %d: %s.

ID: 1168

Severity: ERROR
Message: An error occurred during postoperation synchronization processing for the delete operation with connection ID %d and operation ID %d: %s.

ID: 1169

Severity: ERROR
Message: An error occurred during preoperation synchronization processing for the modify operation with connection ID %d and operation ID %d: %s.

ID: 1170

Severity: ERROR
Message: An error occurred during postoperation synchronization processing for the modify operation with connection ID %d and operation ID %d: %s.

ID: 1171

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: 1172

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: 1173

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: 1174

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: 1175

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: 1176

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: 1177

Severity: ERROR
Message: Unable to add entry %s because the Directory Server is configured in read-only mode.

ID: 1178

Severity: ERROR
Message: Unable to add entry %s because the backend that should hold that entry is configured in read-only mode.

ID: 1179

Severity: ERROR
Message: Unable to delete entry %s because the Directory Server is configured in read-only mode.

ID: 1180

Severity: ERROR
Message: Unable to delete entry %s because the backend that holds that entry is configured in read-only mode.

ID: 1181

Severity: ERROR
Message: Unable to modify entry %s because the Directory Server is configured in read-only mode.

ID: 1182

Severity: ERROR
Message: Unable to modify entry %s because the backend that holds that entry is configured in read-only mode.

ID: 1183

Severity: ERROR
Message: Unable to rename entry %s because the Directory Server is configured in read-only mode.

ID: 1184

Severity: ERROR
Message: Unable to rename entry %s because the backend that holds that entry is configured in read-only mode.

ID: 1185

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: 1186

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: 1187

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: 1188

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: 1189

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 'java.text.SimpleDateFormat' class.

ID: 1190

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 'java.text.SimpleDateFormat' class.

ID: 1191

Severity: ERROR
Message: Attribute options are not allowed for the password attribute %s.

ID: 1192

Severity: ERROR
Message: Only a single value may be provided for the password attribute %s.

ID: 1193

Severity: ERROR
Message: Pre-encoded passwords are not allowed for the password attribute %s.

ID: 1194

Severity: ERROR
Message: The password value for attribute %s was found to be unacceptable: %s.

ID: 1195

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: 1196

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: 1197

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: 1198

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: 1199

Severity: ERROR
Message: %s must change their password before it will be allowed to request any other operations.

ID: 1200

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: 1201

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: 1202

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: 1203

Severity: ERROR
Message: Unable to decode value "%s" for attribute %s in user entry %s as a Boolean value.

ID: 1204

Severity: ERROR
Message: The entry %s cannot be added due to insufficient access rights.

ID: 1205

Severity: ERROR
Message: The user cannot bind due to insufficient access rights.

ID: 1206

Severity: ERROR
Message: The entry %s cannot be compared due to insufficient access rights.

ID: 1207

Severity: ERROR
Message: The entry %s cannot be deleted due to insufficient access rights.

ID: 1208

Severity: ERROR
Message: The extended operation %s cannot be performed due to insufficient access rights.

ID: 1209

Severity: ERROR
Message: The entry %s cannot be renamed due to insufficient access rights.

ID: 1210

Severity: ERROR
Message: The entry %s cannot be modified due to insufficient access rights.

ID: 1211

Severity: ERROR
Message: The entry %s cannot be searched due to insufficient access rights.

ID: 1212

Severity: ERROR
Message: Rejecting a simple bind request because the password policy requires secure authentication.

ID: 1213

Severity: ERROR
Message: Rejecting a bind request because the account has been administratively disabled.

ID: 1214

Severity: ERROR
Message: Rejecting a bind request because the account has been locked due to too many failed authentication attempts.

ID: 1215

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: 1216

Severity: ERROR
Message: Rejecting a bind request because the account has been locked after remaining idle for too long.

ID: 1217

Severity: ERROR
Message: Rejecting a bind request because that user's password is expired.

ID: 1218

Severity: ERROR
Message: An error occurred while attempting to update password policy state information for user %s: %s.

ID: 1219

Severity: ERROR
Message: Rejecting a SASL %s bind request for user %s because the password policy requires secure authentication.

ID: 1220

Severity: ERROR
Message: Rejecting a bind request because the account has expired.

ID: 1221

Severity: ERROR
Message: Attributes used to hold user passwords are not allowed to have any attribute options.

ID: 1222

Severity: ERROR
Message: Users are not allowed to change their own passwords.

ID: 1223

Severity: ERROR
Message: Password changes must be performed over a secure authentication channel.

ID: 1224

Severity: ERROR
Message: The password cannot be changed because it has not been long enough since the last password change.

ID: 1225

Severity: ERROR
Message: Multiple password values are not allowed in user entries.

ID: 1226

Severity: ERROR
Message: User passwords may not be provided in pre-encoded form.

ID: 1227

Severity: ERROR
Message: Invalid modification type %s attempted on password attribute %s.

ID: 1228

Severity: ERROR
Message: The user entry does not have any existing passwords to remove.

ID: 1229

Severity: ERROR
Message: The provided user password does not match any password in the user's entry.

ID: 1230

Severity: ERROR
Message: The password policy requires that user password changes include the current password in the request.

ID: 1231

Severity: ERROR
Message: The password change would result in multiple password values in the user entry, which is not allowed.

ID: 1232

Severity: ERROR
Message: The provided password value was rejected by a password validator: %s.

ID: 1233

Severity: ERROR
Message: %s must change their password before it will be allowed to perform any other operations.

ID: 1234

Severity: WARNING
Message: The user password is about to expire (time to expiration: %s).

ID: 1235

Severity: ERROR
Message: The account has been locked as a result of too many failed authentication attempts (time to unlock: %s).

ID: 1236

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: 1237

Severity: INFO
Message: The user password has been changed.

ID: 1238

Severity: INFO
Message: The user password has been administratively reset.

ID: 1239

Severity: INFO
Message: The user account has been administratively enabled.

ID: 1240

Severity: INFO
Message: The user account has been administratively disabled.

ID: 1241

Severity: INFO
Message: The user account has been administratively unlocked.

ID: 1242

Severity: ERROR
Message: The specified password value already exists in the user entry.

ID: 1243

Severity: ERROR
Message: Entry %s cannot be updated because the request did not contain any modifications.

ID: 1244

Severity: INFO
Message: Do not detach from the terminal and continue running in the foreground. This option cannot be used with the -t, --timeout option.

ID: 1245

Severity: INFO
Message: 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.

ID: 1246

Severity: ERROR
Message: Unable to process the request for extended operation %s because it contained an unsupported critical control with OID %s.

ID: 1247

Severity: ERROR
Message: Unable to register backend %s with the Directory Server because another backend with the same backend ID is already registered.

ID: 1248

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: 1249

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: 1250

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: 1251

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: 1252

Severity: WARNING
Message: Backend %s already contains entry %s which has just been registered as the base DN for backend %s. These conflicting entries can cause unexpected or errant search results, and both backends should be reinitialized to ensure that each has the correct content.

ID: 1253

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: 1254

Severity: WARNING
Message: Base DN %s has been deregistered from the Directory Server for backend %s. This base DN had both superior and subordinate entries in other backends, and there might be inconsistent or unexpected behavior when accessing entries in this portion of the hierarchy because of the missing entries that had been held in the de-registered backend.

ID: 1255

Severity: ERROR
Message: Rejecting the requested operation because the connection has not been authenticated.

ID: 1256

Severity: WARNING
Message: Entry "%s" cannot be added because it contains attribute type %s which is declared OBSOLETE in the server schema.

ID: 1257

Severity: WARNING
Message: Entry "%s" cannot be added because it contains objectclass %s which is declared OBSOLETE in the server schema.

ID: 1258

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: 1259

Severity: ERROR
Message: Object class %s added to entry %s is marked OBSOLETE in the server schema.

ID: 1260

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: 1261

Severity: WARNING
Message: Terminating the client connection because its associated authentication or authorization entry %s has been deleted.

ID: 1262

Severity: ERROR
Message: You do not have sufficient privileges to reset user passwords.

ID: 1263

Severity: ERROR
Message: You do not have sufficient privileges to access the server configuration.

ID: 1264

Severity: ERROR
Message: You do not have sufficient privileges to add entries that include privileges.

ID: 1265

Severity: ERROR
Message: You do not have sufficient privileges to modify the set of privileges contained in an entry.

ID: 1266

Severity: ERROR
Message: You do not have sufficient privileges to use the proxied authorization control.

ID: 1267

Severity: ERROR
Message: OpenDJ is configured to run as a Windows service and it cannot run in no-detach mode.

ID: 1268

Severity: ERROR
Message: Unable to decode an entry because it had an unsupported entry version byte value of %s.

ID: 1269

Severity: ERROR
Message: Unable to decode an entry because an unexpected exception was caught during processing: %s.

ID: 1270

Severity: ERROR
Message: The request control with Object Identifier (OID) "%s" cannot be used due to insufficient access rights.

ID: 1271

Severity: ERROR
Message: The connection handler %s is trying to use the listener %s which is already in use by another connection handler.

ID: 1272

Severity: ERROR
Message: No enabled connection handler available.

ID: 1273

Severity: ERROR
Message: Could not start connection handlers.

ID: 1274

Severity: ERROR
Message: Unable to process the non-root bind because the server is in lockdown mode.

ID: 1275

Severity: WARNING
Message: The Directory Server is entering lockdown mode, in which clients will only be allowed to connect via a loopback address, and only root users will be allowed to process operations.

ID: 1276

Severity: NOTICE
Message: The Directory Server is leaving lockdown mode and will resume normal operation.

ID: 1277

Severity: NOTICE
Message: Rejecting the requested operation because the server is in lockdown mode and will only accept requests from root users over loopback connections.

ID: 1278

Severity: ERROR
Message: Unable to decode the provided attribute because it used an undefined attribute description token %s.

ID: 1279

Severity: ERROR
Message: Unable to decode the provided object class set because it used an undefined token %s.

ID: 1280

Severity: ERROR
Message: Unable to decode the provided entry encode configuration element because it has an invalid length.

ID: 1281

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: 1282

Severity: ERROR
Message: The provided new password was found in the password history for the user.

ID: 1283

Severity: WARNING
Message: There are multiple user-specific idle time limit values contained in user entry %s. The default server idle time limit will be used.

ID: 1284

Severity: WARNING
Message: The user-specific idle time limit value %s contained in user entry %s could not be parsed as an integer. The default server idle time limit will be used.

ID: 1285

Severity: INFO
Message: This connection has been terminated because it has remained idle for too long.

ID: 1286

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: 1287

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: 1288

Severity: ERROR
Message: An error occurred while attempting to disconnect client connection %d: %s.

ID: 1289

Severity: ERROR
Message: The Directory Server is currently running. Environment configuration changes are not allowed with the server running.

ID: 1290

Severity: ERROR
Message: The specified server root directory '%s' is invalid. The specified path must exist and must be a directory.

ID: 1291

Severity: ERROR
Message: The specified config file path '%s' is invalid. The specified path must exist and must be a file.

ID: 1292

Severity: ERROR
Message: The specified schema configuration directory '%s' is invalid. The specified path must exist and must be a directory.

ID: 1293

Severity: ERROR
Message: The Directory Server is currently running. The environment configuration can not be altered while the server is online.

ID: 1294

Severity: ERROR
Message: An error occurred while attempting to initialize a SSL context for server to server communication: %s.

ID: 1295

Severity: INFO
Message: 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.

ID: 1296

Severity: INFO
Message: Error while searching base %s to synchronize the trust store: %s.

ID: 1297

Severity: ERROR
Message: An error occurred in the trust store synchronization thread: %s.

ID: 1298

Severity: INFO
Message: Error while trying to add entry %s to the trust store: %s.

ID: 1299

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: 1300

Severity: ERROR
Message: Password policy configuration entry %s references deprecated password storage scheme DN %s which does not support the auth password syntax.

ID: 1301

Severity: WARNING
Message: The search filter "%s" used by group implementation %s is not indexed in backend %s. Backend initialization for this group implementation might take a very long time to complete.

ID: 1302

Severity: ERROR
Message: CryptoManager cannot get the requested digest %s: %s.

ID: 1303

Severity: ERROR
Message: CryptoManager cannot get the requested MAC engine %s: %s.

ID: 1304

Severity: ERROR
Message: CryptoManager cannot get the requested encryption cipher %s: %s.

ID: 1305

Severity: ERROR
Message: CryptoManager cannot get the preferred key wrapping cipher: %s.

ID: 1306

Severity: ERROR
Message: CryptoManager failed to retrieve the collection of instance-key-pair public-key-certificates from ADS container "%s": %s.

ID: 1307

Severity: ERROR
Message: CryptoManager failed to encode symmetric key attribute value: %s.

ID: 1308

Severity: ERROR
Message: CryptoManager symmetric key attribute value "%s" syntax is invalid: incorrect number of fields.

ID: 1309

Severity: ERROR
Message: CryptoManager symmetric key attribute value "%s" syntax is invalid. Parsing failed in field "%s" at offset %d.

ID: 1310

Severity: ERROR
Message: CryptoManager failed to decipher the wrapped secret-key value: %s.

ID: 1311

Severity: ERROR
Message: CryptoManager cannot find the public-key-certificate (identifier "%s") requested for symmetric key re-encoding.

ID: 1312

Severity: ERROR
Message: CryptoManager failed to decode the key entry identifier "%s": %s.

ID: 1313

Severity: ERROR
Message: CryptoManager passed invalid MAC algorithm "%s": %s.

ID: 1314

Severity: ERROR
Message: CryptoManager failed to initialize MAC engine: %s.

ID: 1315

Severity: ERROR
Message: CryptoManager passed invalid Cipher transformation "%s": %s.

ID: 1316

Severity: ERROR
Message: CryptoManager cannot initialize Cipher: %s.

ID: 1317

Severity: ERROR
Message: CryptoManager failed to write the stream prologue: %s.

ID: 1318

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: 1319

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: 1320

Severity: ERROR
Message: CryptoManager failed to decrypt the supplied data because it could not read the cipher initialization vector in the data prologue.

ID: 1321

Severity: ERROR
Message: CryptoManager failed to decrypt the supplied data because there was an error reading from the input stream: %s.

ID: 1322

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: 1323

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: 1324

Severity: ERROR
Message: CryptoManager failed to import the symmetric key entry "%s": %s.

ID: 1325

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: 1326

Severity: ERROR
Message: CryptoManager failed to instantiate a KeyGenerator for algorithm "%s": %s.

ID: 1327

Severity: ERROR
Message: CryptoManager failed to add locally produced symmetric key entry "%s": %s.

ID: 1328

Severity: ERROR
Message: CryptoManager cipher transformation specification "%s" is invalid: it must be of the form "algorithm/mode/padding".

ID: 1329

Severity: ERROR
Message: CryptoManager cipher transformation specification "%s" is invalid: it must be of the form "algorithm/mode/padding".

ID: 1330

Severity: ERROR
Message: CryptoManager failed to decrypt the supplied data because it could not read the version number in the data prologue: %s.

ID: 1331

Severity: ERROR
Message: CryptoManager failed to decrypt the supplied data because the version "%d" in the data prologue is unknown.

ID: 1332

Severity: ERROR
Message: CryptoManager failed to sign the wrapped key entry: %s.

ID: 1333

Severity: ERROR
Message: The wrapped key entry is missing the following attributes: %s.

ID: 1334

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: 1335

Severity: ERROR
Message: The wrapped key entry signature does not match the entry content.

ID: 1336

Severity: ERROR
Message: CryptoManager failed to verify the wrapped key entry signature: %s.

ID: 1337

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: 1338

Severity: NOTICE
Message: %s.

ID: 1339

Severity: NOTICE
Message: Build ID: %s.

ID: 1340

Severity: ERROR
Message: Start TLS extended operations cannot be canceled.

ID: 1341

Severity: ERROR
Message: Cancel extended operations can not be canceled.

ID: 1342

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: 1343

Severity: INFO
Message: Unable to process operation because this search scope is not allowed in this network group.

ID: 1344

Severity: ERROR
Message: Entry %s can not be added because BER encoding of %s attribute is not supported.

ID: 1345

Severity: INFO
Message: No worker queue thread pool size specified: sizing automatically to use %d threads.

ID: 1346

Severity: INFO
Message: 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.

ID: 1347

Severity: ERROR
Message: In no-detach mode, the 'timeout' option cannot be used.

ID: 1348

Severity: WARNING
Message: The search filter "%s" used by subentry manager is not indexed in backend %s. Backend initialization for subentry manager processing might take a very long time to complete.

ID: 1349

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: 1350

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: 1351

Severity: ERROR
Message: Failed to update free disk space for directory %s: %s.

ID: 1352

Severity: ERROR
Message: The directory server is not accepting a new persistent search request because the server has already reached its limit.

ID: 1353

Severity: ERROR
Message: This operation involves LDAP subentries which you do not have sufficient privileges to administer.

ID: 1354

Severity: INFO
Message: Invalid Credentials.

ID: 1355

Severity: WARNING
Message: Entry "%s" contains a value for attribute %s that is invalid according to the syntax for that attribute: %s.

ID: 1356

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: 1357

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: 1358

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: 1359

Severity: ERROR
Message: The password could not be validated because of misconfiguration. Please contact the administrator.

ID: 1360

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: 1361

Severity: ERROR
Message: Could not get filesystem for directory %s: %s.

ID: 1362

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: 1363

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: 1364

Severity: NOTICE
Message: The free space (%s) on the disk containing directory "%s" is now above the low threshold for the following subsystems: %s.

ID: 1365

Severity: ERROR
Message: Cannot properly use SHA-1 using the java provider. Verify java.security is properly configured.

ID: 1366

Severity: ERROR
Message: Cannot complete initialization of server’s backends because the root and administrative backends have not been initialized yet.

ID: 1367

Severity: ERROR
Message: An error occurred while adding Service Discovery Mechanism '%s': %s.

ID: 1368

Severity: ERROR
Message: Registering Service Discovery Manager’s listener failed : %s.

ID: 1369

Severity: ERROR
Message: Discovery mechanism '%s' initialization failed : %s.

ID: 1370

Severity: WARNING
Message: Replication server '%s' references server '%s' that could not be parsed correctly; the definition will be skipped.

ID: 1371

Severity: ERROR
Message: Error occurred while creating an SSL context for service discovery mechanism '%s' : %s.

ID: 1372

Severity: ERROR
Message: Could not retrieve the list of replicas from replication server '%s' for replication server group '%s'. Exception : %s.

ID: 1373

Severity: ERROR
Message: Could not retrieve auto-configuration data from directory server '%s' for replication server group '%s'. Exception : %s".

ID: 1374

Severity: ERROR
Message: Service discovery mechanism '%s' failed to refresh the partition information. Exception : %s",.

ID: 1375

Severity: WARNING
Message: Settings for Replica '%s' should provide a hostname.

ID: 1376

Severity: WARNING
Message: Cannot connect to replica '%s' for replication service discovery mechanism '%s'. The replica entry is: %s.

ID: 1377

Severity: WARNING
Message: Cannot gather naming contexts from server %s: %s.

ID: 1378

Severity: WARNING
Message: Scheduled discovery '%s' failed : %s.

ID: 1379

Severity: ERROR
Message: Service discovery mechanism '%s' failed to refresh the connection options. Exception : %s",.

ID: 1380

Severity: ERROR
Message: "%s" (low=%s, full=%s).

ID: 1381

Severity: ERROR
Message: You do not have sufficient privileges to read directory server monitoring information.

ID: 1382

Severity: WARNING
Message: There are multiple user-specific cursor entry limit values contained in user entry '%s'. The default server cursor entry limit will be used.

ID: 1383

Severity: WARNING
Message: The user-specific cursor entry limit value '%s' contained in user entry '%s' could not be parsed as an integer. The default server cursor entry limit will be used.

ID: 1384

Severity: ERROR
Message: Entry %s cannot be added because its parent entry %s is a subentry.

ID: 1385

Severity: INFO
Message: Server in lockdown mode.

ID: 1386

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: 1387

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: 1388

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: 1389

Severity: ERROR
Message: No enabled password storage schemes in '%s' in subentry '%s'.

ID: 1390

Severity: ERROR
Message: Cannot use both pwdValidatorPolicy and ds-pwp-validator in subentry '%s'.

ID: 1391

Severity: ERROR
Message: The dictionary data could not be decompressed: %s.

ID: 1392

Severity: ERROR
Message: The dictionary validator configuration is invalid.

ID: 1393

Severity: WARNING
Message: Found %d conflicting password policy subentries for user %s, used %s.

ID: 1394

Severity: ERROR
Message: Requested cipher for a non existing cipher key: cryptographic services were not properly initialized, programming error.

ID: 1395

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: 1396

Severity: ERROR
Message: The subentry %s cannot use both %s and %s objectclasses.

ID: 1397

Severity: ERROR
Message: The subentry %s using the %s objectclass cannot define validators using the old %s objectclass.

ID: 1398

Severity: ERROR
Message: The subentry %s using the %s objectclass cannot define validators using the new %s objectclass.

ID: 1399

Severity: ERROR
Message: The value for the '%s' attribute is not a valid duration.

ID: 1400

Severity: ERROR
Message: The value for the '%s' attribute is not a valid integer.

ID: 1401

Severity: ERROR
Message: The value for the '%s' attribute is not a valid boolean.

ID: 1402

Severity: ERROR
Message: The value for the '%s' attribute is not a valid time.

ID: 1403

Severity: ERROR
Message: The value for the '%s' attribute is not a valid string.

ID: 1404

Severity: ERROR
Message: The value for the '%s' attribute is not a valid attribute.

ID: 1405

Severity: ERROR
Message: The values for the '%s' attribute are not valid strings.

ID: 1406

Severity: ERROR
Message: The value for the '%s' attribute is not a valid state update failure policy.

ID: 1407

Severity: ERROR
Message: A values for the '%s' attribute is not a valid attribute name.

ID: 1408

Severity: ERROR
Message: Could not start connection handler %s with listen addresses "%s". The error was: %s.

ID: 1409

Severity: INFO
Message: Registered %d static groups, %d dynamic groups and %d virtual static groups. The static group cache is using %d bytes of memory.

ID: 1410

Severity: INFO
Message: The following logging categories are defined for the server: %s.

ID: 1411

Severity: ERROR
Message: An error occurred while attempting to initialize the message digest generator for the %s algorithm: %s.

ID: 1412

Severity: ERROR
Message: An error occurred while attempting to base64-decode the password value %s: %s.

ID: 1413

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: 1414

Severity: ERROR
Message: An error occurred while trying to register the JMX alert handler with the MBean server: %s.

ID: 1415

Severity: ERROR
Message: An unexpected error occurred while attempting to encode a password using the storage scheme defined in class %s: %s.

ID: 1416

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: 1417

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: 1418

Severity: ERROR
Message: A fatal error occurred while trying to initialize fifo entry cache: %s.

ID: 1419

Severity: ERROR
Message: A fatal error occurred while trying to initialize soft reference entry cache: %s.

ID: 1420

Severity: ERROR
Message: An unexpected error occurred while attempting to decode the password modify extended request sequence: %s.

ID: 1421

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: 1422

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: 1423

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: 1424

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: 1425

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: 1426

Severity: ERROR
Message: The password modify extended operation cannot be processed because the current password provided for the user is invalid.

ID: 1427

Severity: ERROR
Message: The keystore file %s specified in attribute ds-cfg-key-store-file of configuration entry %s does not exist.

ID: 1428

Severity: ERROR
Message: An error occurred while trying to load the keystore contents from file %s: %s.

ID: 1429

Severity: ERROR
Message: The keystore type %s specified in attribute ds-cfg-key-store-type of configuration entry %s is not valid: %s.

ID: 1430

Severity: ERROR
Message: An error occurred while trying to access the PKCS#11 key manager: %s.

ID: 1431

Severity: ERROR
Message: An error occurred while trying to create a key manager factory to access the contents of keystore file %s: %s.

ID: 1432

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: 1433

Severity: ERROR
Message: The trust store file %s specified in attribute ds-cfg-trust-store-file of configuration entry %s does not exist.

ID: 1434

Severity: ERROR
Message: An error occurred while trying to load the trust store contents from file %s: %s.

ID: 1435

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: 1436

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: 1437

Severity: ERROR
Message: Could not map the provided certificate chain to a user entry because no peer certificate was available.

ID: 1438

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: 1439

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: 1440

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: 1441

Severity: ERROR
Message: Could not map the provided certificate chain to a user because no user entry exists with a DN of %s.

ID: 1442

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: 1443

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: 1444

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: 1445

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: 1447

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: 1448

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: 1449

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: 1450

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: 1451

Severity: ERROR
Message: SASL PLAIN authentication requires that SASL credentials be provided but none were included in the bind request.

ID: 1452

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: 1453

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: 1454

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: 1455

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: 1456

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: 1457

Severity: ERROR
Message: The authentication ID in the SASL PLAIN bind request appears to be an empty DN. This is not allowed.

ID: 1458

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: 1459

Severity: ERROR
Message: The server was not able to find any user entries for the provided authentication ID of %s.

ID: 1460

Severity: ERROR
Message: The provided password is invalid.

ID: 1461

Severity: INFO
Message: An unsupported or unexpected callback was provided to the SASL server for use during %s authentication: %s.

ID: 1462

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: 1463

Severity: ERROR
Message: An unexpected error occurred while trying to create an %s context: %s.

ID: 1464

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: 1465

Severity: ERROR
Message: The username in the SASL %s bind request appears to be an empty DN. This is not allowed.

ID: 1466

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: 1467

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: 1468

Severity: ERROR
Message: The server was not able to find any user entries for the provided username of %s.

ID: 1469

Severity: ERROR
Message: The provided authorization ID %s contained an invalid DN: %s.

ID: 1470

Severity: ERROR
Message: The entry %s specified as the authorization identity does not exist.

ID: 1471

Severity: ERROR
Message: The entry %s specified as the authorization identity could not be retrieved: %s.

ID: 1472

Severity: ERROR
Message: The server was unable to find any entry corresponding to authorization ID %s.

ID: 1473

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: 1474

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: 1475

Severity: ERROR
Message: SASL %s protocol error: %s.

ID: 1476

Severity: ERROR
Message: The authenticating user %s does not have sufficient privileges to assume a different authorization identity.

ID: 1477

Severity: ERROR
Message: The authenticating user %s does not have sufficient access to assume a different authorization identity.

ID: 1478

Severity: ERROR
Message: The server was unable to find any entry corresponding to authentication ID %s.

ID: 1479

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: 1480

Severity: ERROR
Message: An error occurred while attempting to map authorization ID %s to a user entry: %s.

ID: 1481

Severity: ERROR
Message: An error occurred while attempting to write a temporary JAAS configuration file for use during GSSAPI processing: %s.

ID: 1482

Severity: ERROR
Message: An error occurred while attempting to create the JAAS login context for GSSAPI authentication: %s.

ID: 1483

Severity: INFO
Message: GSSAPI mechanism using a principal name of: %s.

ID: 1484

Severity: INFO
Message: GSSAPI SASL mechanism using a server fully qualified domain name of: %s.

ID: 1485

Severity: INFO
Message: DIGEST-MD5 SASL mechanism using a realm of: %s.

ID: 1486

Severity: NOTICE
Message: DIGEST-MD5 SASL mechanism using a server fully qualified domain name of: %s.

ID: 1487

Severity: ERROR
Message: You do not have sufficient privileges to use the proxied authorization control.

ID: 1488

Severity: ERROR
Message: ID string %s could not be mapped to exactly one user. It maps at least to both '%s' and '%s'.

ID: 1489

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: 1490

Severity: ERROR
Message: An internal failure occurred while attempting to resolve ID string %s to a user entry: %s.

ID: 1491

Severity: ERROR
Message: ID string %s mapped to multiple users.

ID: 1492

Severity: ERROR
Message: An error occurred while attempting to map username %s to a Directory Server entry: %s.

ID: 1493

Severity: ERROR
Message: An error occurred while attempting to map username %s to a Directory Server entry: %s.

ID: 1494

Severity: ERROR
Message: Unable to process the cancel request because the extended operation did not include a request value.

ID: 1495

Severity: ERROR
Message: An error occurred while attempting to decode the value of the cancel extended request: %s.

ID: 1496

Severity: INFO
Message: Processing on this operation was terminated as a result of receiving a cancel request (message ID %d).

ID: 1497

Severity: ERROR
Message: Password storage scheme %s does not support use with the authentication password attribute syntax.

ID: 1498

Severity: ERROR
Message: The configured minimum password length of %d characters is greater than the configured maximum password length of %d.

ID: 1499

Severity: ERROR
Message: The provided password is shorter than the minimum required length of %d characters.

ID: 1500

Severity: ERROR
Message: The provided password is longer than the maximum allowed length of %d characters.

IDs: 1501-2000

ID: 1501

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: 1502

Severity: ERROR
Message: Configuration entry "%s" contains multiple definitions for the %s character set.

ID: 1503

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: 1504

Severity: ERROR
Message: The password format string "%s" references an undefined character set "%s".

ID: 1505

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: 1506

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: 1507

Severity: ERROR
Message: An error occurred while attempting to get the password policy for user %s: %s.

ID: 1508

Severity: ERROR
Message: The current password must be provided for self password changes.

ID: 1509

Severity: ERROR
Message: Password modify operations that supply the user's current password must be performed over a secure communication channel.

ID: 1510

Severity: ERROR
Message: End users are not allowed to change their passwords.

ID: 1511

Severity: ERROR
Message: Password changes must be performed over a secure communication channel.

ID: 1512

Severity: ERROR
Message: The password cannot be changed because the previous password change was too recent.

ID: 1513

Severity: ERROR
Message: The password cannot be changed because it is expired.

ID: 1514

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: 1515

Severity: ERROR
Message: An error occurred while attempting to create a new password using the password generator: %s.

ID: 1516

Severity: ERROR
Message: The password policy does not allow users to supply pre-encoded passwords.

ID: 1517

Severity: ERROR
Message: The provided new password failed the validation checks defined in the server: %s.

ID: 1518

Severity: ERROR
Message: Unable to encode the provided password using the default scheme(s): %s.

ID: 1519

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: 1520

Severity: ERROR
Message: The provided authorization ID string "%s" could not be mapped to any user in the directory.

ID: 1521

Severity: ERROR
Message: An error occurred while attempting to map authorization ID string "%s" to a user entry: %s.

ID: 1522

Severity: NOTICE
Message: Account-Status-Notification type='%s' userdn='%s' id=%d msg='%s'.

ID: 1523

Severity: ERROR
Message: An error occurred while attempting to verify the password for user %s during SASL PLAIN authentication: %s.

ID: 1524

Severity: WARNING
Message: The password modify operation was not actually performed in the Directory Server because the LDAP no-op control was present in the request.

ID: 1525

Severity: ERROR
Message: The user account has been administratively disabled.

ID: 1526

Severity: ERROR
Message: The user account is locked.

ID: 1527

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: 1528

Severity: ERROR
Message: You do not have sufficient privileges to perform password reset operations.

ID: 1529

Severity: ERROR
Message: The provided authorization ID was empty, which is not allowed for DIGEST-MD5 authentication.

ID: 1530

Severity: ERROR
Message: The provided authorization ID %s contained an invalid DN: %s.

ID: 1531

Severity: ERROR
Message: The authenticating user %s does not have sufficient privileges to specify an alternate authorization ID.

ID: 1532

Severity: ERROR
Message: The entry corresponding to authorization DN %s does not exist in the Directory Server.

ID: 1533

Severity: ERROR
Message: An error occurred while attempting to retrieve entry %s specified as the authorization ID: %s.

ID: 1534

Severity: ERROR
Message: No entry corresponding to authorization ID %s was found in the server.

ID: 1535

Severity: ERROR
Message: An error occurred while attempting to map authorization ID %s to a user entry: %s.

ID: 1536

Severity: ERROR
Message: Could not map the provided certificate chain to a user entry because no peer certificate was available.

ID: 1537

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: 1538

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: 1539

Severity: ERROR
Message: Configuration entry %s has value '%s' which violates the format required for attribute mappings. The expected format is 'certattr:userattr'.

ID: 1540

Severity: ERROR
Message: Configuration entry %s contains multiple mappings for certificate attribute %s.

ID: 1541

Severity: ERROR
Message: Mapping %s in configuration entry %s references attribute %s which is not defined in the server schema.

ID: 1542

Severity: ERROR
Message: Configuration entry %s contains multiple mappings for user attribute %s.

ID: 1543

Severity: ERROR
Message: Could not map the provided certificate chain to a user entry because no peer certificate was available.

ID: 1544

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: 1545

Severity: ERROR
Message: Unable to decode peer certificate subject %s as a DN: %s.

ID: 1546

Severity: ERROR
Message: Peer certificate subject %s does not contain any attributes for which a mapping has been established.

ID: 1547

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: 1548

Severity: ERROR
Message: Could not map the provided certificate chain to a user entry because no peer certificate was available.

ID: 1549

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: 1550

Severity: ERROR
Message: An error occurred while attempting to calculate the fingerprint for the peer certificate with subject %s: %s.

ID: 1551

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: 1552

Severity: ERROR
Message: Unable to decode value "%s" in entry "%s" as an LDAP URL: %s.

ID: 1553

Severity: ERROR
Message: Dynamic groups do not support explicitly altering their membership.

ID: 1554

Severity: WARNING
Message: Base DN %s specified in dynamic group %s does not exist in the server.

ID: 1555

Severity: ERROR
Message: An error occurred while attempting to 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: 1556

Severity: ERROR
Message: The provided password differs less than the minimum required difference of %d characters.

ID: 1557

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: 1558

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: 1559

Severity: ERROR
Message: The %s attribute is not searchable and should not be included in otherwise unindexed search filters.

ID: 1560

Severity: ERROR
Message: The provided password contained a word from the server's dictionary.

ID: 1561

Severity: ERROR
Message: The specified dictionary file %s does not exist.

ID: 1562

Severity: ERROR
Message: An error occurred while attempting to load the dictionary from file %s: %s.

ID: 1563

Severity: ERROR
Message: The provided password was found in another attribute in the user entry.

ID: 1564

Severity: ERROR
Message: The provided password contained character '%s' which is not allowed for use in passwords.

ID: 1565

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: 1566

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: 1567

Severity: ERROR
Message: The provided character set definition '%s' is invalid because the provided character set is empty.

ID: 1568

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: 1569

Severity: ERROR
Message: The provided character set definition '%s' is invalid because it contains character '%s' which has already been used.

ID: 1570

Severity: ERROR
Message: The virtual static group defined in entry %s contains multiple target group DNs, but only one is allowed.

ID: 1571

Severity: ERROR
Message: Unable to decode "%s" as the target DN for group %s: %s.

ID: 1572

Severity: ERROR
Message: The virtual static group defined in entry %s does not contain a target group definition.

ID: 1573

Severity: ERROR
Message: Target group %s referenced by virtual static group %s does not exist.

ID: 1574

Severity: ERROR
Message: Altering membership for virtual static group %s is not allowed.

ID: 1575

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: 1576

Severity: ERROR
Message: You do not have sufficient privileges to use the password policy state extended operation.

ID: 1577

Severity: ERROR
Message: The provided password policy state extended request did not include a request value.

ID: 1578

Severity: ERROR
Message: An unexpected error occurred while attempting to decode password policy state extended request value: %s.

ID: 1579

Severity: ERROR
Message: An unexpected error occurred while attempting to decode an operation from the password policy state extended request: %s.

ID: 1580

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: 1581

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: 1582

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: 1583

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: 1584

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: 1585

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: 1586

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: 1587

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: 1588

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: 1589

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: 1590

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: 1591

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: 1592

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: 1593

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: 1594

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: 1595

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: 1596

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: 1597

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: 1598

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: 1599

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: 1600

Severity: ERROR
Message: The password policy state extended request included an operation with an invalid or unsupported operation type of %s.

ID: 1601

Severity: WARNING
Message: An error occurred while attempting to update the password policy state information for user %s as part of a password modify extended operation (result code='%s', error message='%s').

ID: 1602

Severity: ERROR
Message: The provided new password was already contained in the password history.

ID: 1603

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: 1604

Severity: WARNING
Message: An error occurred when trying to send an e-mail message for administrative alert with type %s and message %s: %s.

ID: 1605

Severity: ERROR
Message: The provided match pattern "%s" could not be parsed as a regular expression: %s.

ID: 1606

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: 1607

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: 1608

Severity: ERROR
Message: An internal failure occurred while attempting to resolve processed ID string %s to a user entry: %s.

ID: 1609

Severity: ERROR
Message: The processed ID string %s mapped to multiple users.

ID: 1610

Severity: ERROR
Message: Group instance with DN %s has been deleted and is no longer valid.

ID: 1611

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: 1612

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: 1613

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: 1614

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: 1615

Severity: ERROR
Message: The message subject definitions contained in configuration entry '%s' have multiple subjects defined for notification type %s.

ID: 1616

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: 1617

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: 1618

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: 1619

Severity: ERROR
Message: The message template file '%s' referenced in configuration entry '%s' does not exist.

ID: 1620

Severity: ERROR
Message: An unclosed token was found starting at column %d of line %d.

ID: 1621

Severity: ERROR
Message: The notification-user-attr token starting at column %d of line %d references undefined attribute type %s.

ID: 1622

Severity: ERROR
Message: The notification-property token starting at column %d of line %d references undefined notification property %s.

ID: 1623

Severity: ERROR
Message: An unrecognized token %s was found at column %d of line %d.

ID: 1624

Severity: ERROR
Message: An error occurred while attempting to parse message template file '%s' referenced in configuration entry '%s': %s.

ID: 1625

Severity: INFO
Message: Directory Account Status Notification.

ID: 1626

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: 1627

Severity: ERROR
Message: An error occurred while trying to encrypt a value using password storage scheme %s: %s.

ID: 1628

Severity: ERROR
Message: An error occurred while trying to decrypt a value using password storage scheme %s: %s.

ID: 1629

Severity: ERROR
Message: Cannot decode the provided symmetric key extended operation because it does not have a value.

ID: 1630

Severity: ERROR
Message: Cannot decode the provided symmetric key extended request: %s.

ID: 1631

Severity: ERROR
Message: An unexpected error occurred while attempting to decode the symmetric key extended request sequence: %s.

ID: 1632

Severity: WARNING
Message: The exact match identity mapper defined in configuration entry %s references attribute type %s which does not have an equality index defined in backend %s.

ID: 1633

Severity: WARNING
Message: The regular expression identity mapper defined in configuration entry %s references attribute type %s which does not have an equality index defined in backend %s.

ID: 1634

Severity: WARNING
Message: The subject attribute to user attribute certificate mapper defined in configuration entry %s references attribute type %s which does not have an equality index defined in backend %s.

ID: 1635

Severity: ERROR
Message: Failed to create a SASL server for SASL mechanism %s.

ID: 1636

Severity: ERROR
Message: GSSAPI SASL mechanism handler initialization failed because the keytab file %s does not exist.

ID: 1637

Severity: INFO
Message: The GSSAPI SASL mechanism handler initialization was successful.

ID: 1638

Severity: INFO
Message: The GSSAPI SASL mechanism handler has been stopped.

ID: 1639

Severity: ERROR
Message: The password value %s has been base64-decoded but is too short to be valid.

ID: 1640

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: 1641

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: 1642

Severity: ERROR
Message: The provided password did not contain characters from at least %d of the following character sets or ranges: %s.

ID: 1643

Severity: ERROR
Message: SASL %s authentication is not supported for user %s because the account is not managed locally.

ID: 1644

Severity: ERROR
Message: Password modification is not supported for user %s because the account is not managed locally.

ID: 1645

Severity: ERROR
Message: The password policy state extended operation is not supported for user %s because the account is not managed locally.

ID: 1646

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: 1647

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: 1648

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: 1649

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: 1650

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: 1651

Severity: ERROR
Message: The configuration of LDAP PTA policy "%s" is invalid because it does not specify a means for obtaining the mapped search bind password.

ID: 1652

Severity: ERROR
Message: The certificate with subject %s mapped to multiple users.

ID: 1653

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: 1654

Severity: ERROR
Message: An internal failure occurred while attempting to map the certificate with subject %s to a user entry: %s.

ID: 1655

Severity: ERROR
Message: The certificate with subject %s mapped to multiple users.

ID: 1656

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: 1657

Severity: ERROR
Message: An internal failure occurred while attempting to map the certificate with subject %s to a user entry: %s.

ID: 1658

Severity: ERROR
Message: The certificate with fingerprint %s mapped to multiple users.

ID: 1659

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: 1660

Severity: ERROR
Message: An internal failure occurred while attempting to map the certificate with fingerprint %s to a user entry: %s.

ID: 1661

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: 1662

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: 1663

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: 1664

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: 1665

Severity: ERROR
Message: The provided character range definition '%s' is invalid because the range '%s' is reversed.

ID: 1666

Severity: ERROR
Message: The provided character range definition '%s' is invalid because the range '%s' is missing the minus.

ID: 1667

Severity: ERROR
Message: The provided character range definition '%s' is invalid because the range '%s' is too short.

ID: 1668

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: 1669

Severity: ERROR
Message: An error occurred while attempting to match a bcrypt hashed password value: %s.

ID: 1670

Severity: ERROR
Message: The mapped search filter template "%s" could not be parsed as a valid LDAP filter.

ID: 1671

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: 1672

Severity: ERROR
Message: An error occurred while trying to create a trust manager factory to access the contents of LDAP keystore with base DN '%s': %s.

ID: 1673

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: 1674

Severity: ERROR
Message: An error occurred while trying to access the PKCS#11 trust manager: %s.

ID: 1675

Severity: ERROR
Message: Unable to load JVM default keystore from system properties: %s.

ID: 1676

Severity: ERROR
Message: An error occurred while reading information contained within key manager provider from configuration: "%s".

ID: 1677

Severity: ERROR
Message: Unable to get the JVM default truststore: %s.

ID: 1678

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: 1679

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: 1680

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: 1681

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: 1682

Severity: ERROR
Message: The processed ID string %s is mapped to multiple users.

ID: 1683

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: 1684

Severity: WARNING
Message: File based key manager provider '%s' has loaded multiple key manager from keystore file '%s'. Only one will be used for securing TLS connections which rely on this key manager provider. 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: 1685

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: 1686

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: 1687

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: 1688

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: 1689

Severity: WARNING
Message: File based trust manager provider '%s' has loaded multiple trust manager from truststore file '%s'. Only one will be used for securing TLS connections which rely on this trust manager provider. 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: 1690

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: 1691

Severity: ERROR
Message: The %s SCRAM password storage scheme could not be initialized because the algorithm is not supported by the JVM.

ID: 1692

Severity: ERROR
Message: An error occurred while attempting to decode the SCRAM credential value %s: %s.

ID: 1693

Severity: ERROR
Message: SASL %s authentication is not possible for user '%s' because the user entry does not contain any SCRAM credentials.

ID: 1694

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: 1695

Severity: ERROR
Message: The %s SCRAM SASL mechanism handler could not be initialized because the algorithm is not supported by the JVM.

ID: 1696

Severity: ERROR
Message: Error loading dictionary: %s.

ID: 1697

Severity: ERROR
Message: An error occurred while trying to create a trust manager factory to access the certificates in "cn=admin data": %s.

ID: 1698

Severity: ERROR
Message: '%s' cannot list the secret files in directory '%s', all the secrets will be ignored: %s.

ID: 1699

Severity: WARNING
Message: '%s' cannot read the secret file '%s': %s.

ID: 1700

Severity: ERROR
Message: The file '%s' exceeds max size '%s'.

ID: 1701

Severity: ERROR
Message: '%s' cannot decode the secret file '%s': %s.

ID: 1702

Severity: WARNING
Message: '%s' has ignored the file '%s' either because the certificate does not contain the key usage extension '%s', or because the file does not contain the appropriate key types.

ID: 1703

Severity: ERROR
Message: Invalid excluded file name pattern: %s.

ID: 1704

Severity: ERROR
Message: The Argon2 password storage scheme could not be configured because it requires %s kB of memory, exceeding the maximum available of %s kB. Configure a bigger heap or reduce the Argon2 memory requirements.

ID: 1705

Severity: ERROR
Message: An unexpected error occurred while accessing the instance keys in "cn=admin data": %s.

ID: 1706

Severity: ERROR
Message: An unexpected error occurred while attempting to read the instance key '%s' in "cn=admin data": %s.

ID: 1707

Severity: ERROR
Message: A new Argon2 password storage scheme can only be added by running dsconfig in offline mode.

ID: 1708

Severity: INFO
Message: %s.

ID: 1709

Severity: INFO
Message: %s with exception: %s.

ID: 1710

Severity: ERROR
Message: Error occurred while writing log record for logger %s: %s. Any further write errors will be ignored.

ID: 1711

Severity: ERROR
Message: Error occurred while opening log file %s for logger %s: %s.

ID: 1712

Severity: ERROR
Message: Error occurred while closing log file for logger %s: %s.

ID: 1713

Severity: ERROR
Message: Error occurred while flushing writer buffer for logger %s: %s.

ID: 1714

Severity: WARNING
Message: Invalid error log severity "%s".

ID: 1715

Severity: WARNING
Message: Invalid error log category "%s".

ID: 1716

Severity: WARNING
Message: Invalid override of severity level "%s".

ID: 1717

Severity: WARNING
Message: Error occurred while setting file permissions for the log file %s: %s.

ID: 1718

Severity: WARNING
Message: This platform does not support setting file permissions %s to the log file %s.

ID: 1719

Severity: ERROR
Message: Error occurred while listing log files named by policy with initial file name %s.

ID: 1720

Severity: ERROR
Message: Error occurred while obtaining free disk space in the partition containing log file %s: %s.

ID: 1721

Severity: ERROR
Message: Error occurred while enforcing retention policy %s for logger %s: %s.

ID: 1722

Severity: ERROR
Message: Error while creating or updating common audit log publisher %s: %s.

ID: 1723

Severity: ERROR
Message: Error while adding common audit log publisher %s, the publisher has an unsupported handler type.

ID: 1724

Severity: ERROR
Message: Error while reading JSON configuration file %s while creating common audit external log publisher %s: %s.

ID: 1725

Severity: ERROR
Message: Error while creating common audit external log publisher %s: %s.

ID: 1726

Severity: ERROR
Message: Error while creating common audit log publisher %s: %s.

ID: 1727

Severity: ERROR
Message: Error while adding common audit log publisher %s, the publisher defines an unsupported log rotation policy %s.

ID: 1728

Severity: ERROR
Message: Error while adding common audit log publisher %s, the publisher defines an unsupported log retention policy %s.

ID: 1729

Severity: ERROR
Message: Error while processing common audit log publisher %s, this type of log publisher is unsupported.

ID: 1730

Severity: ERROR
Message: Error while processing common audit log publisher %s, delimiter char '%s' should not contain more than one character.

ID: 1731

Severity: ERROR
Message: Error while processing common audit log publisher %s, quote char '%s' should not contain more than one character.

ID: 1732

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: 1733

Severity: ERROR
Message: Error while processing a log event for common audit: %s.

ID: 1734

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore pin file %s is missing.

ID: 1735

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore pin file %s could not be read: %s.

ID: 1736

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore pin file %s contains an empty pin.

ID: 1737

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore file %s is missing.

ID: 1738

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore file %s could not be read: %s.

ID: 1739

Severity: ERROR
Message: Error while processing common audit log publisher %s, the keystore file %s is empty.

ID: 1740

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: 1741

Severity: ERROR
Message: The message with ID %d has been generated %d times in the last second: %s.

ID: 1742

Severity: WARNING
Message: Invalid error log severity "%s", only one severity must be provided, the highest one at which logging should occur.

ID: 1743

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: 1744

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: 1745

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: 1746

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: 1747

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: 1748

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: 1749

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: 1750

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: 1751

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: 1752

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: 1753

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: 1754

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: 1755

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: 1756

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: 1757

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: 1758

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: 1759

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: 1760

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: 1761

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: 1762

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: 1763

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: 1764

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: 1765

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: 1766

Severity: ERROR
Message: The LDIF import plugin defined in configuration entry %s threw an exception when it was invoked on entry %s: %s.

ID: 1767

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: 1768

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: 1769

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: 1770

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: 1771

Severity: ERROR
Message: The plugin defined in configuration entry %s does not support the %s plugin type.

ID: 1772

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: 1773

Severity: ERROR
Message: Auth password storage scheme %s referenced by the password policy import plugin is not configured for use in the server.

ID: 1774

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: 1775

Severity: ERROR
Message: User password storage scheme %s referenced by the password policy import plugin is not configured for use in the server.

ID: 1776

Severity: WARNING
Message: Entry '%s' indicates that it uses custom password policy '%s', but no such policy is defined in the server. Any passwords contained in the entry will be encoded using the default storage schemes, but authentication as this user might not be possible.

ID: 1777

Severity: WARNING
Message: An error occurred while attempting to decode the value of the custom password policy attribute in entry '%s': %s. Any passwords contained in the entry will be encoded using the default storage schemes, but authentication as this user might not be possible.

ID: 1778

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: 1779

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: 1780

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: 1781

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: 1782

Severity: ERROR
Message: An error occurred during Referential Integrity plugin initialization because log file creation failed: %s.

ID: 1783

Severity: ERROR
Message: An error occurred closing the Referential Integrity plugin update log file: %s.

ID: 1784

Severity: ERROR
Message: An error occurred replacing the Referential Integrity plugin update log file: %s.

ID: 1785

Severity: INFO
Message: The file name that the Referential Integrity plugin logs changes to during background processing has been changed from %s to %s, but this change will not take effect until the server is restarted.

ID: 1786

Severity: INFO
Message: The Referential Integrity plugin background processing update interval has been changed from %s to %s, the new value will now be during background processing.

ID: 1787

Severity: INFO
Message: The Referential Integrity plugin background processing has been stopped.

ID: 1788

Severity: INFO
Message: The Referential Integrity plugin has started background processing using the update interval %s.

ID: 1789

Severity: ERROR
Message: The Referential Integrity plugin failed when performing an internal search: %s.

ID: 1790

Severity: ERROR
Message: The Referential Integrity plugin failed when performing an internal modify on entry %s: %s.

ID: 1791

Severity: ERROR
Message: The Referential Integrity plugin failed to decode a entry DN from the update log: %s.

ID: 1792

Severity: INFO
Message: The Referential Integrity plugin failed when performing a search because the base DN %s does not exist.

ID: 1793

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: 1794

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: 1795

Severity: ERROR
Message: The modify DN operation would have resulted in a value for attribute %s that was not 7-bit clean.

ID: 1796

Severity: ERROR
Message: The entry included a value for attribute %s that was not 7-bit clean.

ID: 1797

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: 1798

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: 1799

Severity: ERROR
Message: A unique attribute conflict was detected for attribute %s: value %s already exists in entry %s.

ID: 1800

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: 1801

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: 1802

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: 1803

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: 1804

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: 1805

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: 1806

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: 1807

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: 1808

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: 1809

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: 1810

Severity: ERROR
Message: The Samba password synchronization plugin could not encode a password for the following reasons: %s.

ID: 1811

Severity: ERROR
Message: The Samba password synchronization plugin could not process a modification for the following reason: %s.

ID: 1812

Severity: ERROR
Message: Invalid plugin type '%s' for the Attribute Cleanup plugin.

ID: 1813

Severity: ERROR
Message: Attribute '%s' is not defined in the directory schema.

ID: 1814

Severity: ERROR
Message: The attribute '%s' has already been defined in the configuration.

ID: 1815

Severity: ERROR
Message: The mapping '%s:%s' maps the attribute to itself.

ID: 1816

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: 1817

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: 1818

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: 1819

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: 1820

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: 1821

Severity: ERROR
Message: The operation could not be processed due to an unexpected exception: '%s'.

ID: 1822

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: 1823

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: 1824

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: 1825

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.

ID: 1826

Severity: ERROR
Message: The post-commit %s plugin defined in configuration entry %s threw an exception when it was invoked for connection %d operation %d: %s.

ID: 1827

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: 1828

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: 1829

Severity: INFO
Message: The Directory Server is closing the connection to this client.

ID: 1830

Severity: WARNING
Message: The Directory Server is currently in the process of closing this client connection.

ID: 1831

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: 1832

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: 1833

Severity: INFO
Message: An internal error prevented the Directory Server from properly registering the client connection from %s to %s with an appropriate request handler: %s.

ID: 1834

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: 1835

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: 1836

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: 1837

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: 1838

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: 1839

Severity: ERROR
Message: The original result code for this message was 10 but this result is not allowed for LDAPv2 clients.

ID: 1840

Severity: ERROR
Message: The response included one or more referrals, which are not allowed for LDAPv2 clients. The referrals included were: %s.

ID: 1841

Severity: ERROR
Message: The Directory Server has been configured to deny access to LDAPv2 clients. This connection will be closed.

ID: 1842

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: 1843

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: 1844

Severity: INFO
Message: You have to rebuild the '%s' index(es) manually to get a fully functional server.

ID: 1845

Severity: NOTICE
Message: Started listening for new connections on %s.

ID: 1846

Severity: NOTICE
Message: Stopped listening for new connections on %s.

ID: 1847

Severity: ERROR
Message: User %s specified in the proxied authorization V1 control does not exist in the Directory Server.

ID: 1848

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: 1849

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: 1850

Severity: ERROR
Message: User %s specified in the proxied authorization V2 control does not exist in the Directory Server.

ID: 1851

Severity: WARNING
Message: The Directory Server is already processing another request on the same client connection with the same message ID of %d.

ID: 1852

Severity: ERROR
Message: Use of the proxied authorization V1 control for user %s is not allowed by the password policy configuration.

ID: 1853

Severity: INFO
Message: Unable to process the provided server-side sort request control because it included attribute %s which does not have a default ordering matching rule and no ordering rule was specified in the sort key.

ID: 1854

Severity: ERROR
Message: LDAPv2 clients are not allowed to use request controls.

ID: 1855

Severity: ERROR
Message: You do not have sufficient privileges to perform search operations through JMX.

ID: 1856

Severity: ERROR
Message: You do not have sufficient privileges to establish the connection through JMX. At least JMX_READ privilege is required.

ID: 1857

Severity: ERROR
Message: User %s does not exist in the directory.

ID: 1858

Severity: WARNING
Message: The value %s specified as the LDIF directory path for the LDIF connection handler defined in configuration entry %s exists but is not a directory. The specified path must be a directory. The LDIF connection handler will start, but will not be able to proces any changes until this path is changed to a directory.

ID: 1859

Severity: WARNING
Message: The directory %s referenced by the LDIF connection handler defined in configuration entry %s does not exist. The LDIF connection handler will start, but will not be able to process any changes until this directory is created.

ID: 1860

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: 1861

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: 1862

Severity: INFO
Message: Result Code: %d (%s).

ID: 1863

Severity: INFO
Message: Additional Info: %s.

ID: 1864

Severity: INFO
Message: Matched DN: %s.

ID: 1865

Severity: INFO
Message: Referral URL: %s.

ID: 1866

Severity: ERROR
Message: An I/O error occurred while the LDIF connection handler was processing LDIF file %s: %s.

ID: 1867

Severity: ERROR
Message: An error occurred while the LDIF connection handler was attempting to rename partially-processed file from %s to %s: %s.

ID: 1868

Severity: ERROR
Message: An error occurred while the LDIF connection handler was attempting to delete processed file %s: %s.

ID: 1869

Severity: ERROR
Message: An error occurred while attempting to initialize the SSL context for use in the LDAP Connection Handler: %s.

ID: 1870

Severity: ERROR
Message: The Directory Server does not support LDAP protocol version %d. This connection will be closed.

ID: 1871

Severity: ERROR
Message: Cannot decode the provided control %s because an error occurred while attempting to decode the control value: %s.

ID: 1872

Severity: ERROR
Message: Cannot decode the provided entry changelog notification control because it does not have a value.

ID: 1873

Severity: ERROR
Message: Cannot decode the provided entry changelog notification control because an error occurred while attempting to decode the control value: %s.

ID: 1874

Severity: INFO
Message: Connection handler '%s' does not specify the number of request handler threads: defaulting to %d threads.

ID: 1875

Severity: ERROR
Message: The server received configuration changes that require a restart of the %s connection handler to take effect.

ID: 1876

Severity: ERROR
Message: Authorization as '%s' specified in the proxied authorization control is not permitted.

ID: 1877

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: 1878

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: 1879

Severity: ERROR
Message: Use of the proxied authorization V2 control for user %s is not allowed: the account is disabled.

ID: 1880

Severity: ERROR
Message: Use of the proxied authorization V2 control for user %s is not allowed: the account is expired.

ID: 1881

Severity: ERROR
Message: Use of the proxied authorization V2 control for user %s is not allowed: the account is locked.

ID: 1882

Severity: ERROR
Message: Use of the proxied authorization V2 control for user %s is not allowed: the account's password is expired.

ID: 1889

Severity: ERROR
Message: Unable to process the provided internal modifications request control because it did not contain an origin.

ID: 1890

Severity: ERROR
Message: Unable to process the provided internal modifications request control because it did not contain modifications.

ID: 1891

Severity: ERROR
Message: Unable to process the provided replication context request control because it did not contain a CSN.

ID: 1892

Severity: ERROR
Message: Unable to process the provided replication context request control because it did not contain an entry UUID.

ID: 1893

Severity: ERROR
Message: Unable to process request '%s' received for HTTP client connection: %s.

ID: 1894

Severity: ERROR
Message: Unable to process request '%s' received for internal client connection: %s.

ID: 1895

Severity: ERROR
Message: No result received after completion for request '%s' received for internal client connection.

ID: 1896

Severity: INFO
Message: Internal operations can’t be cancelled.

ID: 1897

Severity: ERROR
Message: Unable to process request '%s' received for JMX client because this type of request is not supported for JMX.

ID: 1898

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: 1899

Severity: ERROR
Message: No result received after completion for request '%s' received for JMX client connection.

ID: 1900

Severity: ERROR
Message: Unable to process request '%s' received for JMX client connection: %s.

ID: 1901

Severity: ERROR
Message: User authentication is mandatory to access server monitoring information.

ID: 1902

Severity: ERROR
Message: Could not write data to the client for %s.

ID: 1903

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: 1904

Severity: ERROR
Message: Unable to process the provided server-side sort request control: %s.

ID: 1905

Severity: ERROR
Message: An error occurred during multi-stage authentication: '%s'.

ID: 1906

Severity: ERROR
Message: Unable to process HTTP request '%s': %s.

ID: 1907

Severity: ERROR
Message: Unable to write HTTP response to the client '%s': %s.

ID: 1908

Severity: ERROR
Message: Error while starting the HTTP application: %s.

ID: 1909

Severity: ERROR
Message: cancel() invoked.

ID: 1910

Severity: ERROR
Message: JMX connection %s with JMX connection ID '%s' has been disconnected because it was finalized by GC.

ID: 1911

Severity: ERROR
Message: JMX connection %s with JMX connection ID '%s' has been disconnected because it received a '%s' notification.

ID: 1912

Severity: ERROR
Message: LDAP connection %s has been closed because the LDAP connection handler %s is shutting down.

ID: 1913

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: 1914

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.

ID: 1915

Severity: ERROR
Message: The connection attempt from client '%s' to '%s' has been rejected because the proxy protocol header indicated that the client used SSL but the header does not specify which SSL cipher was used. The proxy should be configured to transmit all SSL headers.

ID: 1916

Severity: ERROR
Message: The configured DN is already used by another domain.

ID: 1917

Severity: ERROR
Message: Replication Server failed to start : could not bind to the listen port : %d. Error : %s.

ID: 1918

Severity: ERROR
Message: Unknown operation type : %s.

ID: 1919

Severity: ERROR
Message: Internal Error : Operation %s change number %s was not found in local change list.

ID: 1920

Severity: ERROR
Message: Internal Error : Operation %s change number %s was not found in remote change list.

ID: 1921

Severity: ERROR
Message: The replication server failed to start because the database %s could not be read : %s.

ID: 1922

Severity: ERROR
Message: An Exception was caught while replaying operation %s %s (%s) : %s.

ID: 1923

Severity: NOTICE
Message: Error %s when updating server state %s : %s base dn : %s.

ID: 1924

Severity: ERROR
Message: Error %s when searching for server state %s : %s base dn : %s.

ID: 1925

Severity: ERROR
Message: Caught IOException while sending topology info (for update) on domain %s for %s server %s : %s.

ID: 1926

Severity: ERROR
Message: Error when searching old changes from the database for base DN %s: %s.

ID: 1927

Severity: ERROR
Message: Error trying to replay %s, operation could not be decoded: %s.

ID: 1928

Severity: ERROR
Message: Error during the Replication Server database trimming or flush process. The Changelog service is going to shutdown: %s.

ID: 1929

Severity: ERROR
Message: An unexpected error happened handling connection with %s. This connection is going to be closed.

ID: 1930

Severity: ERROR
Message: A loop was detected while replaying operation: %s %s (%s) error %s.

ID: 1931

Severity: ERROR
Message: An Exception was caught while testing existence or trying to create the directory for the Replication Server database : %s %s.

ID: 1932

Severity: ERROR
Message: The current request is rejected due to an import or an export already in progress for the same data.

ID: 1933

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: 1934

Severity: ERROR
Message: Invalid target for the export.

ID: 1935

Severity: ERROR
Message: Domain %s: the server with serverId=%s is unreachable.

ID: 1936

Severity: ERROR
Message: No domain matches the provided base DN '%s'.

ID: 1937

Severity: ERROR
Message: Multiple domains match the base DN provided.

ID: 1938

Severity: ERROR
Message: The provider class does not allow the operation requested.

ID: 1939

Severity: NOTICE
Message: Exception when reading messages from %s: %s.

ID: 1940

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: 1941

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: 1942

Severity: ERROR
Message: Entry %s was containing some unknown historical information, This may cause some inconsistency for this entry.

ID: 1943

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: 1944

Severity: ERROR
Message: An error happened trying to rename a conflicting entry. DN: %s, Operation: %s, Result: %s.

ID: 1945

Severity: ERROR
Message: The Replication is configured for suffix %s but was not able to connect to any Replication Server.

ID: 1946

Severity: ERROR
Message: An unexpected error occurred while sending an Error Message to %s. This connection is going to be closed and reopened.

ID: 1947

Severity: ERROR
Message: An unexpected error occurred while sending a Message to %s. This connection is going to be closed and reopened.

ID: 1948

Severity: ERROR
Message: Could not replay operation %s %s with ChangeNumber %s error %s %s.

ID: 1949

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: 1950

Severity: ERROR
Message: The Replication Server socket could not be closed : %s.

ID: 1951

Severity: ERROR
Message: The thread listening on the replication server port could not be stopped : %s.

ID: 1952

Severity: ERROR
Message: An unexpected error occurred when loading the generation id of domain "%s": %s.

ID: 1953

Severity: ERROR
Message: An unexpected error occurred when looking for the replicated backend : %s. It may be not configured or disabled.

ID: 1954

Severity: ERROR
Message: An unexpected error occurred when searching in %s for the generation ID : %s.

ID: 1955

Severity: ERROR
Message: An unexpected error occurred when updating generation ID for domain "%s": %s.

ID: 1956

Severity: ERROR
Message: The following error has been received : %s.

ID: 1957

Severity: ERROR
Message: Initialization cannot be done because import is not supported by the backend %s.

ID: 1958

Severity: ERROR
Message: Initialization cannot be done because export is not supported by the backend %s.

ID: 1959

Severity: ERROR
Message: Initialization cannot be done because the following error occurred while locking the backend %s : %s.

ID: 1960

Severity: ERROR
Message: Replication server caught exception while listening for client connections: %s.

ID: 1961

Severity: ERROR
Message: While clearing the database %s, the following error happened: %s.

ID: 1962

Severity: ERROR
Message: An unexpected error occurred when testing existence or creating the replication backend : %s.

ID: 1963

Severity: ERROR
Message: An error occurred when searching for %s : %s.

ID: 1964

Severity: ERROR
Message: The base DN %s is not stored by any of the Directory Server backend.

ID: 1965

Severity: ERROR
Message: An Exception was caught while replaying replication message : %s.

ID: 1966

Severity: ERROR
Message: Caught exception publishing fake operations for domain %s : %s.

ID: 1967

Severity: NOTICE
Message: ServerState recovery for domain %s, updated with changeNumber %s.

ID: 1968

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: 1969

Severity: ERROR
Message: DN sent by remote replication server: %s does not match local replication server one: %s.

ID: 1970

Severity: ERROR
Message: DN sent by replication server: %s does not match local directory server one: %s.

ID: 1971

Severity: ERROR
Message: Caught IOException while forwarding ResetGenerationIdMsg to peer replication servers for domain %s : %s.

ID: 1972

Severity: ERROR
Message: Replication server received invalid initial status: %s for replication domain %s from server id %s.

ID: 1973

Severity: ERROR
Message: Received invalid requested status %s in DS replication domain %s with server id %s.

ID: 1974

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: 1975

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: 1976

Severity: ERROR
Message: Caught IOException while changing status for domain %s and serverId: %s after reset for generation id: %s.

ID: 1977

Severity: ERROR
Message: Received change status message does not come from a directory server (dn: %s, server id: %s, msg: %s).

ID: 1978

Severity: ERROR
Message: Received invalid new status %s in RS for replication domain %s and directory server id %s.

ID: 1979

Severity: WARNING
Message: Connected to a replication server with wrong group id. We have group id %s and replication server id %s %s has group id %s. This is for domain %s in directory server %s.

ID: 1980

Severity: ERROR
Message: Replication broker with dn %s and server id %s failed to signal status change because of: %s.

ID: 1981

Severity: ERROR
Message: Caught IOException while changing status for domain %s and serverId: %s from status analyzer: %s.

ID: 1982

Severity: NOTICE
Message: Ignoring reset generation ID request from '%s' to '%s' for replica '%s' because the replica is being re-initialized.

ID: 1983

Severity: ERROR
Message: The generation ID could not be reset for domain %s.

ID: 1984

Severity: NOTICE
Message: Cannot change the configuration while a total update is in progress.

ID: 1985

Severity: ERROR
Message: The Replication was not started on base-dn %s : %s.

ID: 1986

Severity: ERROR
Message: Replication protocol error. Bad message type. %s received, %s required.

ID: 1987

Severity: NOTICE
Message: Wrong fractional replication configuration: could not find object class definition for %s in schema.

ID: 1988

Severity: NOTICE
Message: Wrong fractional replication configuration : could not find attribute type definition for %s in schema.

ID: 1989

Severity: NOTICE
Message: Wrong fractional replication configuration : attribute %s is not optional in class %s.

ID: 1990

Severity: NOTICE
Message: Wrong fractional replication configuration : wrong format : %s (need at least [<className>|*],attributeName).

ID: 1991

Severity: NOTICE
Message: Wrong fractional replication configuration : cannot use both exclusive and inclusive modes.

ID: 1992

Severity: NOTICE
Message: Wrong fractional replication configuration : prohibited attribute %s usage.

ID: 1993

Severity: NOTICE
Message: Fractional replication : exception for domain : %s : %s.

ID: 1994

Severity: NOTICE
Message: Warning : domain %s fractional replication configuration is inconsistent with backend data set : need resynchronization or fractional configuration to be changed.

ID: 1995

Severity: ERROR
Message: The fractional replication ldif import plugin is configured with invalid plugin type %s. Only the ldifImport plugin type is allowed.

ID: 1996

Severity: NOTICE
Message: The online full update for importing suffix %s data from remote directory server %s has been stopped due to fractional configuration inconsistency between destination and source server : imported data set has not the same fractional configuration.

ID: 1997

Severity: NOTICE
Message: The online full update for importing suffix %s data from remote directory server %s has been stopped due to fractional configuration inconsistency between destination and source server : imported data set has some fractional configuration but not destination server.

ID: 1998

Severity: NOTICE
Message: The following operation has been forbidden in suffix %s due to inconsistency with the fractional replication configuration : %s.

ID: 1999

Severity: NOTICE
Message: The export of domain %s from server %s to all other servers of the topology is forbidden as the source server has some fractional configuration : only fractional servers in a replicated topology does not make sense.

ID: 2000

Severity: ERROR
Message: An error occurred when accessing the change number database : %s.

IDs: 2001-2500

ID: 2001

Severity: ERROR
Message: The initialization failed because the domain %s is not connected to a replication server.

ID: 2002

Severity: ERROR
Message: Could not retrieve the configuration for a replication domain matching the entry %s.

ID: 2003

Severity: NOTICE
Message: The LDIF import for importing suffix %s data has been stopped due to fractional configuration inconsistency : imported data set has not the same fractional configuration as local server.

ID: 2004

Severity: NOTICE
Message: The LDIF import for importing suffix %s data has been stopped due to fractional configuration inconsistency : imported data set has some fractional configuration but not local server.

ID: 2005

Severity: WARNING
Message: A transient problem occurred while Directory Server %s was connecting to this Replication Server %s. The peer server may try to establish a connection again. The problem was: %s.

ID: 2006

Severity: WARNING
Message: A transient problem occurred while Replication Server %s was connecting to this Replication Server %s. The Replication Server should try to reconnect later. The problem was: %s.

ID: 2007

Severity: NOTICE
Message: Error when loading a virtual attribute for external change log: Attribute: %s , Error: %s.

ID: 2008

Severity: ERROR
Message: Full resync required. Reason: The provided cookie contains unknown replicated domain %s. Current starting cookie <%s>.

ID: 2009

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: 2010

Severity: ERROR
Message: Invalid syntax for the provided cookie '%s'.

ID: 2011

Severity: ERROR
Message: Domain %s (server id: %s) : remote exporter server disconnection (server id: %s) detected during initialization.

ID: 2012

Severity: ERROR
Message: During initialization from a remote server, the following error occurred : %s.

ID: 2013

Severity: ERROR
Message: Connection failure with Replication Server %s during import.

ID: 2014

Severity: ERROR
Message: Bad msg id sequence during import. Expected:%s Actual:%s.

ID: 2015

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: 2016

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: 2017

Severity: ERROR
Message: When initializing remote server(s), connection to Replication Server with serverId=%s is lost.

ID: 2018

Severity: ERROR
Message: When initializing remote server(s), the initialized server with serverId=%s is potentially stopped or too slow.

ID: 2019

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: 2020

Severity: NOTICE
Message: Resending a new initialization request for an initialization from a remote server due to the root error : %s.

ID: 2021

Severity: ERROR
Message: Error while trying to solve conflict with DN : %s ERROR : %s.

ID: 2022

Severity: NOTICE
Message: Replication server RS(%s) started listening for new connections on address %s port %d.

ID: 2023

Severity: INFO
Message: Replication server RS(%s) has connected to replication server RS(%s) for domain "%s" at %s.

ID: 2024

Severity: INFO
Message: Replication server RS(%s) has accepted a connection from replication server RS(%s) for domain "%s" at %s.

ID: 2025

Severity: INFO
Message: Replication server RS(%s) has accepted a connection from directory server DS(%s) for domain "%s" at %s.

ID: 2026

Severity: NOTICE
Message: Directory server DS(%s) has connected to replication server RS(%s) for domain "%s" at %s with generation ID %s.

ID: 2027

Severity: WARNING
Message: Directory server DS(%s) has connected to replication server RS(%s) for domain "%s" at %s, but the generation IDs do not match, indicating that a full re-initialization is required. The local (DS) generation ID is %s and the remote (RS) generation ID is %s.

ID: 2028

Severity: WARNING
Message: Directory server DS(%s) was unable to connect to any of the following replication servers for domain "%s": %s.

ID: 2029

Severity: WARNING
Message: Directory server DS(%s) was unable to connect to any replication servers for domain "%s".

ID: 2030

Severity: WARNING
Message: Replication server RS(%s) at %s has closed the connection to this directory server DS(%s). This directory server will now try to connect to another replication server in order to receive changes for the domain "%s".

ID: 2031

Severity: WARNING
Message: Directory server DS(%s) encountered an error while receiving changes for domain "%s" from replication server RS(%s) at %s. The connection will be closed, and this directory server will now try to connect to another replication server. The error was: %s.

ID: 2032

Severity: NOTICE
Message: Directory Server DS(%s) is switching from replication server RS(%s) at %s to RS(%s) for domain "%s" because it is more suitable. The previous replication server evaluation was: "%s", and the new replication server evaluation was: "%s".

ID: 2033

Severity: NOTICE
Message: Starting total update: importing domain "%s" from remote directory server DS(%s) to this directory server DS(%s).

ID: 2034

Severity: NOTICE
Message: Finished total update: imported domain "%s" from remote directory server DS(%s) to this directory server DS(%s). %s.

ID: 2035

Severity: NOTICE
Message: Starting total update: exporting %d entries in domain "%s" from this directory server DS(%s) to remote directory server DS(%s).

ID: 2036

Severity: NOTICE
Message: Starting total update: exporting %d entries in domain "%s" from this directory server DS(%s) to all remote directory servers.

ID: 2037

Severity: NOTICE
Message: Finished total update: exported domain "%s" from this directory server DS(%s) to remote directory server DS(%s). %s.

ID: 2038

Severity: NOTICE
Message: Finished total update: exported domain "%s" from this directory server DS(%s) to all remote directory servers. %s.

ID: 2039

Severity: NOTICE
Message: Directory server DS(%s) for domain "%s" has changed its status to %s.

ID: 2040

Severity: WARNING
Message: Directory server DS(%s) is closing its connection to replication server RS(%s) at %s for domain "%s" because it could not detect a heart beat.

ID: 2041

Severity: WARNING
Message: Replication server RS(%s) at %s presented generation ID %s for domain "%s", but the generation ID of this replication server RS(%s) is %s. This usually indicates that one or more directory servers in the replication topology have not been initialized with the same data, and re-initialization is required.

ID: 2042

Severity: NOTICE
Message: The generation ID for domain "%s" has been reset to %s.

ID: 2043

Severity: WARNING
Message: Timed out while trying to acquire the domain lock for domain "%s". The connection attempt from replication server RS(%s) at %s to this replication server RS(%s) will be aborted. This is probably benign and a result of a simultaneous cross connection attempt.

ID: 2044

Severity: WARNING
Message: Directory server DS(%s) at %s presented generation ID %s for domain "%s", but the generation ID of this replication server RS(%s) is %s. This usually indicates that one or more directory servers in the replication topology have not been initialized with the same data, and re-initialization is required.

ID: 2045

Severity: NOTICE
Message: Replication server RS(%s) ignoring update %s for domain "%s" from replication server RS(%s) at %s because its generation ID %s is different to the local generation ID %s.

ID: 2046

Severity: WARNING
Message: Replication server RS(%s) not sending update %s for domain "%s" to replication server RS(%s) at %s because its generation ID %s is different to the local generation ID %s.

ID: 2047

Severity: WARNING
Message: Replication server RS(%s) ignoring update %s for domain "%s" from directory server DS(%s) at %s because its generation ID %s is different to the local generation ID %s.

ID: 2048

Severity: WARNING
Message: Replication server RS(%s) not sending update %s for domain "%s" to directory server DS(%s) at %s because its generation ID %s is different to the local generation ID %s.

ID: 2049

Severity: WARNING
Message: Replication server RS(%s) ignoring update %s for domain "%s" from directory server DS(%s) at %s because it is currently performing a full update.

ID: 2050

Severity: WARNING
Message: Replication server RS(%s) not sending update %s for domain "%s" to directory server DS(%s) at %s because it is currently performing a full update.

ID: 2051

Severity: ERROR
Message: The connection from this replication server RS(%s) to replication server RS(%s) at %s for domain "%s" has failed.

ID: 2052

Severity: ERROR
Message: The connection from this replication server RS(%s) to directory server DS(%s) at %s for domain "%s" has failed.

ID: 2053

Severity: WARNING
Message: Directory server DS(%s) was unable to connect to replication server %s for domain "%s". Please check that there is a replication server listening at this address.

ID: 2054

Severity: WARNING
Message: Directory server DS(%s) timed out while connecting to replication server %s for domain "%s".

ID: 2055

Severity: ERROR
Message: Directory server DS(%s) encountered an unexpected error while connecting to replication server %s for domain "%s": %s.

ID: 2056

Severity: WARNING
Message: Directory server DS(%s) encountered a transient problem while connecting to replication server %s for domain "%s". Directory server will try to connect to a replication server again. The problem was: %s.

ID: 2057

Severity: INFO
Message: Replication server accepted a connection from %s to local address %s 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: %s.

ID: 2058

Severity: NOTICE
Message: Directory Server DS(%s) is disconnecting from replication server RS(%s) at %s for domain "%s" in order to find another replication server in the topology and distribute load more equally.

ID: 2059

Severity: WARNING
Message: The attribute value '%s' is not a valid synchronization history value.

ID: 2060

Severity: WARNING
Message: Cannot open database %s because shutdown was requested from replication server RS(%s).

ID: 2061

Severity: NOTICE
Message: RS(%s) has no generation Id, but at least one other RS has the same generation Id %s as DS(%s).

ID: 2062

Severity: NOTICE
Message: RS(%s) generation Id %s does not match DS(%s) generation Id %s, but at least another RS does.

ID: 2063

Severity: NOTICE
Message: RS(%s) groupId '%s' does not match DS(%s) groupId '%s', but at least one other RS does.

ID: 2064

Severity: NOTICE
Message: RS(%s) newest change %s is behind DS(%s) newest change %s, but at least another RS is at the same point or ahead of the DS.

ID: 2065

Severity: NOTICE
Message: RS(%s) newest change %s is behind another RS which is ahead of DS(%s) newest change %s.

ID: 2066

Severity: NOTICE
Message: RS(%s) is not on the same virtual machine as DS(%s) but another RS is.

ID: 2067

Severity: NOTICE
Message: RS(%s) is on a different host than DS(%s), but at least another RS is on the same host.

ID: 2068

Severity: NOTICE
Message: DS(%s) disconnected from overloaded RS(%s).

ID: 2069

Severity: NOTICE
Message: DS(%s) not disconnected from overloaded RS(%s), other DSs will disconnect.

ID: 2070

Severity: NOTICE
Message: DS(%s) not disconnected from current RS(%s), since there is no need to rebalance all directory servers to other replication servers in the topology.

ID: 2071

Severity: NOTICE
Message: DS(%s) not disconnected from current RS(%s), because RS is underloaded or its load goal is reached.

ID: 2072

Severity: NOTICE
Message: DS(%s) will connect to RS(%s) because it has the biggest weight among all the replication servers.

ID: 2073

Severity: NOTICE
Message: DS(%s) stayed connected to RS(%s) to avoid the yoyo effect.

ID: 2074

Severity: NOTICE
Message: RS(%s) has been evaluated to be the best replication server for DS(%s) to connect to because it was the only one standing after all tests.

ID: 2075

Severity: NOTICE
Message: RS(%s) could not be contacted by DS(%s).

ID: 2076

Severity: ERROR
Message: Could not create replica database because the changelog database is shutting down.

ID: 2077

Severity: ERROR
Message: Could not add change %s to replicaDB %s %s because flushing thread is shutting down.

ID: 2078

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: 2079

Severity: ERROR
Message: Could not get or create replica DB for base DN '%s', serverId '%s', generationId '%s': %s.

ID: 2080

Severity: ERROR
Message: Could not get or create change number index DB in root path '%s', using path '%s': %s.

ID: 2081

Severity: ERROR
Message: Could not delete generation id file '%s' for DN '%s': %s.

ID: 2082

Severity: ERROR
Message: Could not create directory '%s' for server id %s: %s.

ID: 2083

Severity: ERROR
Message: Could not create generation id file '%s': %s.

ID: 2084

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: 2085

Severity: ERROR
Message: Could not read generation id because it uses a wrong format, expecting a number but got '%s'.

ID: 2086

Severity: ERROR
Message: Could not open log file '%s' for write: %s.

ID: 2087

Severity: ERROR
Message: Could not open a reader on log file '%s': %s.

ID: 2088

Severity: ERROR
Message: Could not decode a record from data read in log file '%s'.

ID: 2089

Severity: ERROR
Message: Could not delete log files: %s.

ID: 2090

Severity: ERROR
Message: Could not create log file '%s': %s.

ID: 2091

Severity: WARNING
Message: The changelog '%s' has been opened in read-only mode, it is not enabled for write.

ID: 2092

Severity: ERROR
Message: Could not add record '%s' in log file '%s': %s.

ID: 2093

Severity: ERROR
Message: Could not synchronize written records to file system for log file '%s': %s.

ID: 2094

Severity: ERROR
Message: Could not seek to position %d for reader on log file '%s'.

ID: 2095

Severity: ERROR
Message: Could not create root directory '%s' for log file: %s.

ID: 2096

Severity: ERROR
Message: Could not decode DN from domain state file '%s', from line '%s'.

ID: 2097

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: 2098

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: 2099

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: 2100

Severity: ERROR
Message: Could not decode the key from string [%s].

ID: 2101

Severity: ERROR
Message: Could not initialize the log '%s' : %s.

ID: 2102

Severity: ERROR
Message: Could not retrieve key bounds from log file '%s': %s.

ID: 2103

Severity: ERROR
Message: Could not retrieve read-only log files from log '%s': %s.

ID: 2104

Severity: ERROR
Message: While purging log, could not delete log file(s): '%s'.

ID: 2105

Severity: ERROR
Message: The following log '%s' must be released but it is not referenced.

ID: 2106

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: 2107

Severity: ERROR
Message: Could not read content of replica offline state file '%s' for domain %s.

ID: 2108

Severity: ERROR
Message: Could not retrieve file length of log file '%s'.

ID: 2109

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: 2110

Severity: INFO
Message: Log file '%s' was successfully recovered by removing unreadable records. The file changed size from %d to %d bytes.

ID: 2111

Severity: NOTICE
Message: You do not have sufficient privileges to perform a search request on cn=changelog.

ID: 2112

Severity: ERROR
Message: An error occurred when searching base DN '%s' with filter '%s' in changelog backend : %s.

ID: 2113

Severity: ERROR
Message: An error occurred when retrieving attribute value for attribute '%s' for entry DN '%s' in changelog backend : %s.

ID: 2114

Severity: ERROR
Message: Could not create file '%s' to store last log rotation time %d.

ID: 2115

Severity: ERROR
Message: Could not delete file '%s' that stored the previous last log rotation time.

ID: 2116

Severity: ERROR
Message: Cursor on log '%s' has been aborted after a purge.

ID: 2117

Severity: ERROR
Message: Could not position and read newest record from log file '%s'. Current thread is '%s'. Error was: %s.

ID: 2118

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: 2119

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: 2120

Severity: ERROR
Message: Change number indexing is disabled for replication domain '%s'.

ID: 2121

Severity: ERROR
Message: Cannot decode change-log record with version %x.

ID: 2122

Severity: ERROR
Message: Cannot start total update in domain "%s" from this directory server DS(%s): no remote directory servers found.

ID: 2123

Severity: ERROR
Message: Cannot start total update in domain "%s" from this directory server DS(%s): cannot find remote directory server DS(%s).

ID: 2124

Severity: ERROR
Message: New replication connection from %s started with unexpected message %s and is being closed.

ID: 2125

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: 2126

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: 2127

Severity: WARNING
Message: Error while setting replication listener socket timeout to 1 second for domain '%s', server state may be reported as being a bit late with respect to other servers. Cause was: %s.

ID: 2128

Severity: ERROR
Message: Invalid operator '%s' specified in historicalCsnRangeMatch extensible matching rule assertion.

ID: 2129

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: 2130

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: 2131

Severity: ERROR
Message: Specified operators '%s' and '%s' do not specify a range for historicalCsnRangeMatch extensible matching rule.

ID: 2132

Severity: ERROR
Message: Could not restart the Replication Server, bind to listen port %d failed : %s.

ID: 2133

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: 2134

Severity: WARNING
Message: Disk space for the replication changelog is getting low. The replication server will continue to operate, but will stop in the event of a full disk.

ID: 2135

Severity: WARNING
Message: The replication server has detected that the file system containing the changelog has sufficient disk space to resume operation. It is now reconnecting to the rest of the replication topology.

ID: 2136

Severity: NOTICE
Message: An I/O error occurred while reading replication messages from %s. The connection will close and replication server (%s) will not process any more updates from it. Reported error is: %s.

ID: 2137

Severity: NOTICE
Message: %s is disconnecting from this replication server %s.

ID: 2138

Severity: ERROR
Message: The replication server connector thread could not be stopped : %s.

ID: 2139

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: 2140

Severity: ERROR
Message: Assured replication is not supported anymore, it should be disabled on the topology.

ID: 2141

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: 2142

Severity: WARNING
Message: Replication delay for '%s' is above %dms, current delay: %dms.

ID: 2143

Severity: WARNING
Message: The replication server has detected that the file system containing the changelog has sufficient disk space to resume operation.

ID: 2144

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: 2145

Severity: ERROR
Message: An error occurred while verifying integrity of log file '%s' : %s.

ID: 2146

Severity: ERROR
Message: Cannot enable replication to server '%s' as this server's ID '%s' is not a number between 1 and 32767.

ID: 2147

Severity: ERROR
Message: An error occurred in session '%s' when trying to send a message to the socket: %s.

ID: 2148

Severity: ERROR
Message: Could not convert value '%s' to long.

ID: 2149

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: 2150

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: 2151

Severity: ERROR
Message: Unexpected message type when trying to create changelog entry for dn %s: '%s'.

ID: 2152

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: 2153

Severity: ERROR
Message: %s message of type %s cannot be routed. Details: %s.

ID: 2154

Severity: ERROR
Message: Error when trying to publish a message in '%s'. The connection is going to be closed and reopened.

ID: 2155

Severity: ERROR
Message: Error when trying to publish a message in '%s'. The connection is going to be closed and reopened. Error: %s.

ID: 2156

Severity: ERROR
Message: Error when initializing publisher of messages in '%s'.

ID: 2157

Severity: ERROR
Message: Error when initializing publisher of messages in '%s'.

ID: 2158

Severity: ERROR
Message: Directory server %s was attempting to connect to replication server %s but an error occurred in handshake phase. Error: %s.

ID: 2159

Severity: ERROR
Message: Replication server %s was attempting to connect to replication server %s but an error occurred in handshake phase. Error: %s.

ID: 2160

Severity: ERROR
Message: Error when enabling external changelog: %s.

ID: 2161

Severity: WARNING
Message: Directory server %s stopped the handshake process with replication server %s because it received the unexpected message '%s'.

ID: 2162

Severity: INFO
Message: Cursor on replica '%s' invalidated.

ID: 2163

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: 2164

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: 2165

Severity: INFO
Message: Task '%s' to initialize domain '%s' failed to contact the source server. The task will be retried because the current server may be in the process of re-connecting to the best replication server for itself.

ID: 2166

Severity: INFO
Message: The server '%s' may retry the initialization process because the server '%s' in domain '%s' is temporarily unreachable.

ID: 2167

Severity: ERROR
Message: Error when purging historical information for entry %s: %s.

ID: 2168

Severity: ERROR
Message: Replication server failed to start because setting socket timeout on port %d caused error %s.

ID: 2169

Severity: WARNING
Message: A transient problem occurred while DS(%s) was connecting to RS(%s), Directory Server will try to connect again. Problem was: %s.

ID: 2170

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: 2171

Severity: ERROR
Message: %s for domain '%s' cannot route message of type %s to replica %s because it is unreachable. Reachable replicas: %s.

ID: 2172

Severity: ERROR
Message: Server %s should be initialized but is not connected to the topology.

ID: 2174

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: 2175

Severity: NOTICE
Message: An unresolved conflict was detected in CSN %s replaying a DELETE of "%s" as child entries exist (renaming child "%s" to "%s"). Consider restoring the parent entry, or deleting the renamed child entries.

ID: 2176

Severity: NOTICE
Message: An unresolved conflict was detected in CSN %s replaying an ADD of "%s" as the parent entry is missing (adding as "%s"). Consider restoring the parent entry, or deleting the new entry.

ID: 2177

Severity: NOTICE
Message: An unresolved conflict was detected in CSN %s replaying an ADD of "%s" which already exists (adding as "%s"). Consider reviewing both entries and either merge the contents manually or delete one.

ID: 2178

Severity: NOTICE
Message: An unresolved conflict was detected in CSN %s replaying a MODIFYDN of "%s" as the new parent entry "%s" does not exist. Consider restoring the parent entry, or deleting this entry.

ID: 2179

Severity: NOTICE
Message: An unresolved conflict was detected in CSN %s replaying a MODIFYDN of "%s" to "%s" which already exists (renaming as "%s"). Consider merging the two entries manually, or deleting one of them.

ID: 2181

Severity: NOTICE
Message: RS(%s) groupId '%s' has lower priority than groupId '%s' which matched at least one other RS.

ID: 2182

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: 2183

Severity: ERROR
Message: Replication peer certificate verification failed with error: %s.

ID: 2184

Severity: ERROR
Message: Digest algorithm '%s' of fingerprint '%s' is not supported by the JVM.

ID: 2185

Severity: ERROR
Message: The following replication server listener threads were unexpectedly stopped: %s. A server restart may be needed.

ID: 2186

Severity: ERROR
Message: Cursor on log '%s' has been aborted after a clear.

ID: 2187

Severity: ERROR
Message: Initialization of domain '%s' interrupted: no data received from the remote exporter '%s' for %s.

ID: 2188

Severity: NOTICE
Message: Recovery interrupted because shutdown of replication domain has been requested.

ID: 2189

Severity: WARNING
Message: The replication server domain connection for '%s' will be forcibly closed because the peer replication server '%s' is unresponsive.

ID: 2190

Severity: WARNING
Message: The replication domain connection for '%s' will be forcibly closed because the peer replication server '%s' is unresponsive.

ID: 2191

Severity: WARNING
Message: ChangeTime Heartbeat publisher '%s' for domain '%s' will be forcibly stopped because the peer server is unresponsive.

ID: 2192

Severity: ERROR
Message: Publish of a replication message to server '%s' was interrupted.

ID: 2193

Severity: ERROR
Message: Changelog on replication server '%s' 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.

ID: 2195

Severity: WARNING
Message: Replication server RS(%s) is not sending update %s for domain '%s' to directory server DS(%s) at %s because the replica is too late and must be re-initialized.

ID: 2196

Severity: WARNING
Message: Replication server RS(%s) is ignoring update %s for domain "%s" from directory server DS(%s) at %s because the replica is too late and must be re-initialized.

ID: 2197

Severity: WARNING
Message: Replication server RS(%s) is closing its connection to %s at %s for domain '%s' because it could not detect a heartbeat: time since last message (%s) is greater than the timeout interval %s ms (heartbeat interval is %s ms).

ID: 2198

Severity: WARNING
Message: Change number indexing cannot advance because some replicas do not seem to be part of the topology anymore. The changelog will not be able to purge old files until replicas reconnect or their reference is deleted from user data. The following replicas are referenced by replication, but they are not connected to the topology or they have not been properly stopped: %s.

ID: 2199

Severity: WARNING
Message: Change number indexing cannot advance because %s has not sent any messages since '%s'. The changelog is temporarily unable to purge old files. Verify whether the replica is still running or is connected to a replication server.

ID: 2200

Severity: ERROR
Message: Replica '%s' seems to have disconnected from the topology while it was being re-initialized, initialization will stop.

ID: 2201

Severity: ERROR
Message: Initialization of replication domain '%s' interrupted because the domain is shutting down. The domain still needs to be re-initialized.

ID: 2202

Severity: INFO
Message: Change number indexing will resume since %s is now sending messages again.

ID: 2203

Severity: WARNING
Message: Directory server DS(%s) was unable to connect to replication server at %s for domain '%s': unknown host.

ID: 2204

Severity: INFO
Message: Directory server DS(%s) did not find a generation ID for domain '%s'. A new generation ID will be computed by exporting the first %d entries in the domain.

ID: 2205

Severity: WARNING
Message: Replica '%s' for domain '%s' tried to connect but the replication server is still processing messages from the previous replication session and considers the replica already connected. The connection will be closed, the replica will connect again.

ID: 2206

Severity: INFO
Message: %s: received message <%s>.

ID: 2207

Severity: NOTICE
Message: This replica `%s` has received a notification from its replication server `%s` to stop processing updates for domain `%s` as part of a disaster recovery procedure. This replica must be reinitialized as part of the recovery procedure.

ID: 2209

Severity: ERROR
Message: The search cannot be processed because the external changelog is not configured to generate change numbers. Change the replication server configuration if change numbers are needed.

ID: 2210

Severity: INFO
Message: The purge delay has been set to: %s.

ID: 2211

Severity: INFO
Message: Purging is disabled (delay set to 0).

ID: 2212

Severity: ERROR
Message: An unexpected error occurred when loading the fractional replication configuration of domain "%s": %s.

ID: 2213

Severity: WARNING
Message: Replication server RS(%s) ignoring update %s for domain "%s" from directory server DS(%s) at %s because the peer DS reported to be in BAD_DATA status. The generation ID matches, (DS is %s, RS is %s), there may be a problem with fractional replication configuration. Check the DS error logs for more details.

ID: 2214

Severity: WARNING
Message: Replication server RS(%s) not sending update %s for domain "%s" to directory server DS(%s) at %s because the peer DS reported to be in BAD_DATA status. The generation ID matches, (DS is %s, RS is %s), there may be a problem with fractional replication configuration. Check the DS error logs for more details.

ID: 2215

Severity: NOTICE
Message: JAVA Version: %s.

ID: 2216

Severity: NOTICE
Message: JAVA Vendor: %s.

ID: 2217

Severity: NOTICE
Message: JVM Version: %s.

ID: 2218

Severity: NOTICE
Message: JVM Vendor: %s.

ID: 2219

Severity: NOTICE
Message: JAVA Home: %s.

ID: 2220

Severity: NOTICE
Message: Class Path: %s.

ID: 2221

Severity: NOTICE
Message: Current Directory: %s.

ID: 2222

Severity: NOTICE
Message: Operating System: %s.

ID: 2223

Severity: NOTICE
Message: JVM Architecture: %s.

ID: 2224

Severity: NOTICE
Message: System Name: %s.

ID: 2225

Severity: NOTICE
Message: Available Processors: %d.

ID: 2226

Severity: NOTICE
Message: Max Available Memory: %d.

ID: 2227

Severity: NOTICE
Message: Currently Used Memory: %d.

ID: 2228

Severity: NOTICE
Message: Currently Free Memory: %d.

ID: 2229

Severity: NOTICE
Message: JVM Information: %s by %s, %s architecture, %d bytes heap size.

ID: 2230

Severity: NOTICE
Message: JVM Host: %s, running %s, %d bytes physical memory size, number of processors available %d.

ID: 2231

Severity: NOTICE
Message: JVM Arguments: %s.

ID: 2232

Severity: NOTICE
Message: JVM Host: %s, running %s, unknown physical memory size, number of processors available %d.

ID: 2233

Severity: NOTICE
Message: Installation Directory: %s.

ID: 2234

Severity: NOTICE
Message: Installation Directory: unknown.

ID: 2235

Severity: NOTICE
Message: Instance Directory: %s.

ID: 2236

Severity: NOTICE
Message: Instance Directory: unknown.

ID: 2237

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: 2238

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: 2239

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: 2240

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: 2241

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: 2242

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: 2243

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: 2244

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: 2245

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: 2246

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: 2247

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: 2248

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: 2249

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: 2251

Severity: ERROR
Message: The provided value "%s" could not be parsed as a valid subtree specification.

ID: 2252

Severity: NOTICE
Message: A schema element could not be imported: %s, %s.

ID: 2253

Severity: ERROR
Message: There should be no warnings on the schema, but instead got %d warnings: %s.

ID: 2254

Severity: ERROR
Message: Unable to parse the OID from the provided definition of objectclass: '%s'.

ID: 2255

Severity: ERROR
Message: Unable to parse the OID from the provided definition of attribute type: '%s'.

ID: 2256

Severity: ERROR
Message: Unable to parse the OID from the provided definition of ldap syntax: '%s'.

ID: 2257

Severity: ERROR
Message: Unable to parse the OID from the provided definition of matching rule use: '%s'.

ID: 2258

Severity: ERROR
Message: Unable to parse the OID from the provided definition of name form: '%s'.

ID: 2259

Severity: ERROR
Message: Unable to parse the OID from the provided definition of DIT content rule: '%s'.

ID: 2260

Severity: ERROR
Message: Unable to parse the rule ID from the provided definition of DIT structure rule: '%s'.

ID: 2268

Severity: ERROR
Message: There is already a server instance configured in the directory %s. Please make sure there is no configuration (config) or data (db) directory in the server instance root path.

ID: 2269

Severity: INFO
Message: You have chosen to combine an evaluation profile with another profile that is intended for production. Applying an evaluation profile after others can compromise the security of your server. Review the server configuration carefully before deployment.

ID: 2270

Severity: ERROR
Message: The file '%s' is referring to an invalid instance (%s). You must remove this file before setting up a new instance in the directory %s. Please use either the --%s option in non interactive mode or the interactive mode to setup an instance elsewhere.

ID: 2271

Severity: ERROR
Message: A server instance (%s) has already been setup in the directory %s.

ID: 2272

Severity: WARNING
Message: Setup has failed, see the errors in the messages below. The process also failed to automatically remove the files that it created during setup. Please manually remove the 'db' and 'config' directories from the instance directory before trying to run the setup again.

ID: 2273

Severity: WARNING
Message: Setup has failed, see the errors in the messages below. The process also failed to automatically remove the files that it created during setup. Please manually remove the 'db' and 'config' directories from the instance directory and the '%s' file before trying to run the setup again.

ID: 2274

Severity: ERROR
Message: The file based keystore password cannot be empty.

ID: 2275

Severity: ERROR
Message: You must specify the keystore password of the file based keystore. You can use either --%s or --%s options to specify it.

ID: 2276

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

ID: 2277

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

ID: 2278

Severity: ERROR
Message: You must specify a valid port number.

ID: 2279

Severity: INFO
Message: Trust remote server certificates.

ID: 2280

Severity: INFO
Message: Use existing PKCS#12 truststore.

ID: 2281

Severity: INFO
Message: Use existing JCEKS truststore.

ID: 2282

Severity: INFO
Message: Use existing JKS truststore.

ID: 2283

Severity: INFO
Message: Truststore file.

ID: 2284

Severity: INFO
Message: Truststore password.

ID: 2285

Severity: INFO
Message: PKCS#12 truststore path:.

ID: 2286

Severity: INFO
Message: JCEKS truststore path:.

ID: 2287

Severity: INFO
Message: JKS truststore path:.

ID: 2288

Severity: INFO
Message: Truststore password (leave empty if the truststore has no password):.

ID: 2289

Severity: INFO
Message: How should the server trust certificates from SSL peers?.

ID: 2290

Severity: INFO
Message: Use the JVM truststore.

ID: 2291

Severity: INFO
Message: Use a PKCS#12 truststore.

ID: 2292

Severity: INFO
Message: Use a JCEKS truststore.

ID: 2293

Severity: INFO
Message: Use a JKS truststore.

ID: 2294

Severity: INFO
Message: Blindly trust all certificates (evaluation only - unsafe for production).

ID: 2295

Severity: INFO
Message: No password provided.

ID: 2296

Severity: INFO
Message: Use existing PKCS12 truststore file for validating peer SSL certificates.

ID: 2297

Severity: INFO
Message: Use existing JCEKS truststore file for validating peer SSL certificates.

ID: 2298

Severity: INFO
Message: Use existing JKS truststore file for validating peer SSL certificates.

ID: 2299

Severity: INFO
Message: Blindly trust peer SSL certificates.

ID: 2300

Severity: ERROR
Message: The provided password file '%s' must be an existing readable file.

ID: 2301

Severity: ERROR
Message: Unable to read password in file '%s'. A password file must contain a single line with the keystore password, details: %s.

ID: 2302

Severity: ERROR
Message: Invalid content for password file '%s'. A password file must contain a single line with the password.

ID: 2303

Severity: ERROR
Message: password is empty.

ID: 2304

Severity: ERROR
Message: An error occurred while storing provided password in the instance file '%s', details: %s.

ID: 2305

Severity: ERROR
Message: Could not access the %s truststore '%s'. Check that the content of the file corresponds to a valid truststore, that you have access rights to it and that a valid password has been provided if needed. Error details: %s.

ID: 2306

Severity: ERROR
Message: clear-text password cannot be empty.

ID: 2307

Severity: ERROR
Message: An error occurred while creating the file based trust manager provider: %s.

ID: 2308

Severity: ERROR
Message: An error occurred while enabling the %s trust manager provider: %s.

ID: 2309

Severity: ERROR
Message: The provided truststore file '%s' must be an existing readable file.

ID: 2310

Severity: ERROR
Message: An error occurred while creating the PKCS#11 key manager provider: %s.

ID: 2311

Severity: ERROR
Message: Could not read the %s keystore '%s' entries. Check that the content of the file corresponds to a valid keystore, that you have access rights to it and that a valid password has been provided if needed. Error details: %s.

ID: 2312

Severity: ERROR
Message: Could not access the PKCS#11 keystore. Check that the JVM security settings have been configured to use a PKCS#11 keystore and that a valid password has been provided if needed. Error details: %s.

ID: 2313

Severity: ERROR
Message: No private key entry found in keystore '%s'. Keystores used for securing client connections must contain at least one private key entry. Beware: you may need to provide the keystore password to access keystore private entries.

ID: 2314

Severity: INFO
Message: Provide the server’s fully qualified host name.

ID: 2315

Severity: INFO
Message: Welcome to the OpenDJ %s interactive setup.%n%nInteractive mode helps you quickly set up an OpenDJ server.%nProvide all required options to set up the server in non-interactive mode,%nor use the command shown at the end of your interactive session.

ID: 2316

Severity: ERROR
Message: License not accepted, %s will exit.

ID: 2317

Severity: ERROR
Message: Invalid input: keystore path cannot be empty.

ID: 2318

Severity: ERROR
Message: Invalid configuration: you must use at least one certificate nickname.

ID: 2319

Severity: INFO
Message: The keystore contains the following certificate nicknames: %s.%nYou will be prompted for the nickname(s) to use.

ID: 2320

Severity: ERROR
Message: The number of entries to import must be a positive integer.

ID: 2321

Severity: ERROR
Message: Value '%s' must be a valid number.

ID: 2322

Severity: ERROR
Message: Value '%s' must be a valid version (two or three digits separated with a dot).

ID: 2323

Severity: ERROR
Message: Please select at least version %s.

ID: 2324

Severity: INFO
Message: Listening on port %d.

ID: 2325

Severity: INFO
Message: Customize Server With Profiles.

ID: 2326

Severity: INFO
Message: Server security.

ID: 2327

Severity: INFO
Message: Keystore file.

ID: 2328

Severity: INFO
Message: Use existing PKCS#11 keystore.

ID: 2329

Severity: INFO
Message: Use existing PKCS#12 keystore.

ID: 2330

Severity: INFO
Message: Use existing JCEKS keystore.

ID: 2331

Severity: INFO
Message: Use existing JKS keystore.

ID: 2332

Severity: INFO
Message: Keystore password.

ID: 2333

Severity: INFO
Message: Certificate nickname(s) to use.

ID: 2334

Severity: INFO
Message: LDAP (cleartext).

ID: 2335

Severity: INFO
Message: LDAPS (secure).

ID: 2336

Severity: INFO
Message: HTTP (cleartext).

ID: 2337

Severity: INFO
Message: HTTPS (secure).

ID: 2338

Severity: INFO
Message: How would you like to secure your deployment?.

ID: 2339

Severity: INFO
Message: You already have a CA certificate for the deployment and an SSL key-pair for this server.

ID: 2340

Severity: INFO
Message: Provide the SSL certificate that will be used for securing server connections:.

ID: 2341

Severity: INFO
Message: Generate a CA certificate and SSL key-pair using the deployment ID (easier to deploy, but not suitable for public facing services).

ID: 2342

Severity: INFO
Message: Provide the deployment ID:.

ID: 2343

Severity: INFO
Message: Server instance path.

ID: 2344

Severity: INFO
Message: Provide the server instance path.

ID: 2345

Severity: INFO
Message: Provide the directory root user DN.

ID: 2346

Severity: INFO
Message: Provide the password for root user (leave empty to use the deployment ID password):.

ID: 2347

Severity: INFO
Message: Confirm the password for root user:.

ID: 2348

Severity: INFO
Message: Enable HTTP?.

ID: 2349

Severity: INFO
Message: HTTP port.

ID: 2350

Severity: INFO
Message: Enable HTTPS?.

ID: 2351

Severity: INFO
Message: HTTPS port.

ID: 2352

Severity: INFO
Message: Start server after setup.

ID: 2353

Severity: INFO
Message: Run as Windows service.

ID: 2354

Severity: INFO
Message: Server’s fully qualified host name.

ID: 2355

Severity: INFO
Message: Root user password.

ID: 2356

Severity: INFO
Message: Enable LDAP?.

ID: 2357

Severity: INFO
Message: LDAP port.

ID: 2358

Severity: INFO
Message: Enable LDAPS?.

ID: 2359

Severity: INFO
Message: LDAPS port.

ID: 2360

Severity: INFO
Message: Administration connector port.

ID: 2361

Severity: INFO
Message: Start the server when setup is complete? Note: starting with version 7.0, it is advised to only start the server once it is customized for the intended deployment. If more configuration is needed, then select 'no'.

ID: 2362

Severity: INFO
Message: Global Parameters.

ID: 2363

Severity: INFO
Message: Network Parameters.

ID: 2364

Severity: INFO
Message: All Server Parameters.

ID: 2365

Severity: INFO
Message: Set up the server with the specified settings.

ID: 2366

Severity: INFO
Message: Start over.

ID: 2367

Severity: INFO
Message: Quit.

ID: 2368

Severity: INFO
Message: Equivalent non-interactive command:.

ID: 2369

Severity: INFO
Message: Accept these choices and continue?.

ID: 2370

Severity: ERROR
Message: Unable to encode provided password because '%s'.

ID: 2371

Severity: ERROR
Message: The fully qualified host name must be a non blank string.

ID: 2372

Severity: ERROR
Message: The provided keystore file '%s' must be an existing readable file.

ID: 2373

Severity: INFO
Message: Launch external process: '%s'.

ID: 2374

Severity: ERROR
Message: External process contains errors: %s.

ID: 2375

Severity: INFO
Message: External process standard output: %s.

ID: 2376

Severity: ERROR
Message: External process has been interrupted: '%s'.

ID: 2377

Severity: ERROR
Message: Unable to display license because '%s'.

ID: 2378

Severity: ERROR
Message: Unable to enable windows service (return code '%d'): %s.

ID: 2379

Severity: INFO
Message: Enable windows service command (return code '%d') output: %s.

ID: 2380

Severity: ERROR
Message: Unable to create server instance directory '%s'.

ID: 2381

Severity: ERROR
Message: The instance parent (%s) must be a valid existing directory.

ID: 2382

Severity: ERROR
Message: The instance path must be a directory.

ID: 2383

Severity: ERROR
Message: An error occurred while trying to read instance.loc file: %s.

ID: 2384

Severity: ERROR
Message: An error occurred while trying to write instance.loc file: %s.

ID: 2385

Severity: ERROR
Message: You cannot provide a keystore password without providing an existing keystore.

ID: 2386

Severity: ERROR
Message: You cannot specify certificate aliases without providing an existing keystore.

ID: 2387

Severity: ERROR
Message: You cannot specify truststore without providing an existing keystore.

ID: 2389

Severity: ERROR
Message: Unable to enable the file based HTTP access logger because: %s.

ID: 2390

Severity: ERROR
Message: An error occurred while writing password in configuration file: '%s'.

ID: 2391

Severity: ERROR
Message: An error occurred while attempting to update the HTTP connector port: '%s'.

ID: 2392

Severity: ERROR
Message: An error occurred while attempting to update the HTTPS connector: '%s'.

ID: 2393

Severity: ERROR
Message: An error occurred while attempting to update the port on which to listen for LDAP communication: %s.

ID: 2394

Severity: ERROR
Message: Unable to write configuration file '%s' (%s).

ID: 2395

Severity: ERROR
Message: Unable to write the upgrade version in '%s' file (%s).

ID: 2396

Severity: ERROR
Message: Unable to parse the version in '%s' file in order to initialize the content of the '%s' file.

ID: 2397

Severity: ERROR
Message: Unable to write file '%s' because: '%s'.

ID: 2398

Severity: ERROR
Message: Unable to edit runtime settings file because '%s'.

ID: 2399

Severity: ERROR
Message: An error occurred while copying file '%s' into '%s': '%s'.

ID: 2400

Severity: ERROR
Message: An error occurred while creating directory '%s'.

ID: 2401

Severity: ERROR
Message: Unable to set access permissions to allow write operations on file '%s'. Ensure that you have enough rights to change the access permissions on instance files.

ID: 2402

Severity: INFO
Message: Configuring server.

ID: 2403

Severity: INFO
Message: Validating parameters.

ID: 2404

Severity: ERROR
Message: The server can only be enabled as a windows service on a Windows operating system.

ID: 2405

Severity: INFO
Message: Port on which the server should listen for HTTP communication.

ID: 2406

Severity: INFO
Message: Port on which the server should listen for HTTPS communication.

ID: 2407

Severity: INFO
Message: Path were the instance should be set up.

ID: 2408

Severity: NOTICE
Message: /path/to/opendj.

ID: 2409

Severity: ERROR
Message: The provided certificate nickname '%s' could not be found. The keystore contains the following certificate nicknames: %s.

ID: 2410

Severity: ERROR
Message: The keystore contains the following certificate nicknames: %s.%nYou have to provide the nickname of the certificate you want to use.

ID: 2411

Severity: ERROR
Message: Could not access the %s keystore '%s'. Check that the content of the file corresponds to a valid keystore, that you have access rights to it and that a valid password has been provided if needed. Error details: %s.

ID: 2412

Severity: ERROR
Message: You must accept the product license to run setup. Either use interactive mode or the '--%s' option.

ID: 2413

Severity: ERROR
Message: The provided server install path '%s' must reference an existing directory.

ID: 2414

Severity: ERROR
Message: An unexpected error has been raised during setup: '%s'.

ID: 2415

Severity: ERROR
Message: Administration connector port '%d' is not in allowed range %d, %d.

ID: 2416

Severity: ERROR
Message: LDAP port '%d' is not in allowed range %d, %d.

ID: 2417

Severity: ERROR
Message: LDAPS port '%d' is not in allowed range %d, %d.

ID: 2418

Severity: ERROR
Message: HTTP port '%d' is not in allowed range %d, %d.

ID: 2419

Severity: ERROR
Message: HTTPS port '%d' is not in allowed range %d, %d.

ID: 2420

Severity: ERROR
Message: Replication port '%d' is not in allowed range %d, %d.

ID: 2421

Severity: ERROR
Message: Port '%d' is not in allowed range %d, %d.

ID: 2422

Severity: ERROR
Message: Value '%d' has been specified for different ports.

ID: 2423

Severity: ERROR
Message: The administrator user bind DN '%s' is invalid. %s.

ID: 2424

Severity: ERROR
Message: The root user DN cannot be empty.

ID: 2425

Severity: ERROR
Message: DN for the monitor user cannot be empty.

ID: 2426

Severity: ERROR
Message: Monitor user DN value '%s' is invalid because: %s.

ID: 2427

Severity: NOTICE
Message: This utility sets up an OpenDJ server. Use the %s option to list available profiles.

ID: 2428

Severity: INFO
Message: Done.

ID: 2429

Severity: INFO
Message: Enabling Windows service.

ID: 2430

Severity: INFO
Message: Error.

ID: 2431

Severity: INFO
Message: .....

ID: 2432

Severity: INFO
Message: Starting directory server.

ID: 2433

Severity: INFO
Message: Configuring certificates.

ID: 2434

Severity: INFO
Message: An error occurred creating the temporary file.

ID: 2435

Severity: INFO
Message: Error Enabling Windows service.

ID: 2436

Severity: INFO
Message: Error Starting Directory Server.

ID: 2437

Severity: INFO
Message: Error Starting Directory Server. Error code: %s.

ID: 2438

Severity: INFO
Message: See %s for a detailed log of the failed operation.%nPlease report this error and provide the log file mentioned above.

ID: 2439

Severity: INFO
Message: An error occurred while configuring the certificates: %s.

ID: 2440

Severity: INFO
Message: Administration connector port.

ID: 2441

Severity: INFO
Message: Please read the License Agreement above.%nYou must accept the terms of the agreement before continuing with the installation.

ID: 2442

Severity: INFO
Message: Root user DN.

ID: 2443

Severity: ERROR
Message: The --%s option should only be provided when using a file based trust store option.

ID: 2444

Severity: INFO
Message: Provide the DN for the default monitor user.

ID: 2445

Severity: INFO
Message: Provide the password for the default monitor user:.

ID: 2446

Severity: INFO
Message: Confirm the password for the default monitor user:.

ID: 2447

Severity: INFO
Message: Define and enable a default user for querying monitoring information?.

ID: 2448

Severity: INFO
Message: Monitor user.

ID: 2449

Severity: INFO
Message: Monitor user password.

ID: 2450

Severity: INFO
Message: {monitorUserDn}.

ID: 2451

Severity: INFO
Message: {monitorUserPassword}.

ID: 2452

Severity: INFO
Message: DN of the default user allowed to query monitoring information.

ID: 2453

Severity: INFO
Message: Password of the default user allowed to query monitoring information.

ID: 2454

Severity: ERROR
Message: When specifying a default user for monitoring a password should also be provided.

ID: 2455

Severity: ERROR
Message: An error occurred while attempting to write the monitor user entry: %s.

ID: 2456

Severity: ERROR
Message: An error occurred while attempting to configure the backend for the monitor user: %s.

ID: 2457

Severity: ERROR
Message: Folder '%s' of the setup profile '%s' must be a readable existing folder.

ID: 2458

Severity: ERROR
Message: Last folder of profile '%s' path '%s' must be the profile version (two or three digits separated by either dot or hyphen, e.g /path/to/profile/6.5.0).

ID: 2459

Severity: ERROR
Message: Folder of the setup profile '%s' must contain a '%s' readable regular file.

ID: 2460

Severity: ERROR
Message: File '%s' from the setup profile '%s' must be a readable regular file.

ID: 2461

Severity: INFO
Message: Configuring profile %s.

ID: 2462

Severity: ERROR
Message: The '%s' setup profile failed: %s.

ID: 2463

Severity: INFO
Message: {name[:version]}.

ID: 2464

Severity: INFO
Message: 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. %s.

ID: 2465

Severity: INFO
Message: There are no setup profiles available for this OpenDJ version.

ID: 2466

Severity: INFO
Message: Available setup profiles: %s.

ID: 2467

Severity: INFO
Message: {[profileName/]parameterName:value}.

ID: 2468

Severity: INFO
Message: 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.

ID: 2469

Severity: INFO
Message: Display all available profiles.

ID: 2470

Severity: INFO
Message: Display profile parameters.

ID: 2471

Severity: INFO
Message: Profiles available with this OpenDJ version are:.

ID: 2472

Severity: INFO
Message: Use %s %s to list profile parameters.

ID: 2473

Severity: NOTICE
Message: Parameter(s) '%s' provided for profile '%s' is (are) not used in the profile.

ID: 2474

Severity: NOTICE
Message: Parameter '%s' is provided without any profile specification which is not allowed when multiple profiles are used.

ID: 2475

Severity: ERROR
Message: The parameter '%s' is required by profile '%s', but no value has been provided.

ID: 2476

Severity: ERROR
Message: Error in profile '%s': the default value for parameter '%s' cannot be derived from the SetupConfiguration method '%s' because no such method exists.

ID: 2477

Severity: ERROR
Message: Error in profile '%s': the default value for parameter '%s' cannot be derived from the SetupConfiguration method '%s' because the method invocation threw an error: %s.

ID: 2478

Severity: ERROR
Message: Invalid profile parameter value '%s'. Supported format is '%s'.

ID: 2479

Severity: ERROR
Message: Option to setup profile parameters have been provided but not the profile name. If you want to use a setup profile, please provide its name with the '%s' option.

ID: 2480

Severity: ERROR
Message: Setup profile '%s' cannot be found. %s.

ID: 2481

Severity: ERROR
Message: Profile '%s' has been specified multiple times. It is not possible to configure the server for multiple versions of the same profile. Please choose one version for this profile.

ID: 2482

Severity: ERROR
Message: Setup profile parameter '%s' has been provided multiple times with different values. Please either choose one value or prefix the option values with a profile name.

ID: 2483

Severity: ERROR
Message: Setup profile parameter '%s' references the profile '%s' which has not been provided on the command line. Setup profiles can be added on the command line with the '%s' option.

ID: 2484

Severity: ERROR
Message: Unable to read template file '%s'. Please ensure that the file exists and that you have appropriate permissions to read the file.

ID: 2485

Severity: ERROR
Message: Unable to read file '%s'. Please ensure that the file exists and that you have appropriate permissions to read the file.

ID: 2486

Severity: WARNING
Message: An error occurred while attempting to remove content of directory '%s': %s.

ID: 2487

Severity: ERROR
Message: An error occurred while substituting properties from profile '%s' resource file '%s': %s.

ID: 2488

Severity: ERROR
Message: An IO error occurred while running profile '%s': '%s'.

ID: 2489

Severity: ERROR
Message: Unable to create '%s' backend (%s).

ID: 2490

Severity: ERROR
Message: Unable to create replication domain for base DN '%s' (%s).

ID: 2491

Severity: ERROR
Message: No backend has been created in the profile.

ID: 2492

Severity: ERROR
Message: '%s' setup profile failed to import data contained in '%s' file(s) into '%s' backend. Import LDIF tool options: '%s'. Exception message:'%s'. Tool output: %s.

ID: 2493

Severity: ERROR
Message: Unable to import data.

ID: 2494

Severity: ERROR
Message: Unable to import data (%s).

ID: 2495

Severity: ERROR
Message: Unable to import generated users (%s).

ID: 2496

Severity: ERROR
Message: Unable to import data because some entries have been rejected.

ID: 2497

Severity: ERROR
Message: Unable to import data because some entries have been skipped.

ID: 2498

Severity: ERROR
Message: '%s' setup profile failed to import data contained in '%s' file(s) into '%s' backend because of rejected entries (listed in '%s'). Import LDIF tool options: '%s'. Tool output: %s.

ID: 2499

Severity: ERROR
Message: '%s' setup profile failed to import data contained in '%s' file(s) into '%s' backend because of skipped entries (listed in '%s'). Import LDIF tool options: '%s'. Tool output: %s.

ID: 2500

Severity: ERROR
Message: Unable to update server configuration.

IDs: 2501-3000

ID: 2501

Severity: ERROR
Message: '%s' setup profile failed to update server configuration file, dsconfig tool options: '%s'. Exception message:'%s'. Tool output: %s.

ID: 2502

Severity: ERROR
Message: Unable to add index for attribute '%s' (%s).

ID: 2503

Severity: INFO
Message: Available profiles:.

ID: 2504

Severity: INFO
Message: Enter profile number(s) separated by comma (leave empty to skip this step):.

ID: 2505

Severity: INFO
Message: Enter profile number(s) separated by comma [%s]:.

ID: 2506

Severity: INFO
Message: No profiles configured or data to be imported.

ID: 2507

Severity: INFO
Message: Confirm the %s:.

ID: 2508

Severity: INFO
Message: Provide a value for '%s'.

ID: 2509

Severity: INFO
Message: Value set for '%s'.

ID: 2510

Severity: INFO
Message: Configure server for profile '%s'.

ID: 2511

Severity: INFO
Message: Configure server for profile '%s (%s)'.

ID: 2512

Severity: ERROR
Message: Please provide a value having the following syntax: %s.

ID: 2513

Severity: ERROR
Message: The value '%s' for parameter '%s' in profile '%s' is invalid. Please provide a value having the following syntax: %s.

ID: 2514

Severity: ERROR
Message: The value '%s' for parameter '%s' in profile '%s' is invalid because it does not match any allowed values (%s).

ID: 2515

Severity: ERROR
Message: The value for password parameter '%s' of profile '%s' is invalid (%s).

ID: 2516

Severity: ERROR
Message: Unable to encode password associated to parameter '%s' of profile '%s' (%s).

ID: 2517

Severity: ERROR
Message: The value for domain parameter '%s' of profile '%s' cannot be blank.

ID: 2518

Severity: ERROR
Message: The value '%s' for domain parameter '%s' of profile '%s' is invalid (%s).

ID: 2519

Severity: ERROR
Message: The value for DN parameter '%s' of profile '%s' cannot be empty.

ID: 2520

Severity: WARNING
Message: Version %s not found, best match is %s.

ID: 2521

Severity: INFO
Message: Profile '%s (%s)' does not have any parameters.

ID: 2522

Severity: INFO
Message: %s (%s) parameters.

ID: 2523

Severity: INFO
Message: Parameter:.

ID: 2524

Severity: INFO
Message: Syntax:.

ID: 2525

Severity: INFO
Message: %s or configuration expression.

ID: 2526

Severity: INFO
Message: String.

ID: 2527

Severity: INFO
Message: Number.

ID: 2528

Severity: INFO
Message: Decimal number.

ID: 2529

Severity: INFO
Message: Password.

ID: 2530

Severity: INFO
Message: Domain name.

ID: 2531

Severity: INFO
Message: File or directory path.

ID: 2532

Severity: INFO
Message: One of the following values:.

ID: 2533

Severity: INFO
Message: host:port.

ID: 2534

Severity: INFO
Message: DN.

ID: 2535

Severity: INFO
Message: Description:.

ID: 2536

Severity: INFO
Message: Default:.

ID: 2537

Severity: INFO
Message: Derived from the setup global options.

ID: 2538

Severity: ERROR
Message: Unable to bootstrap the Directory Server configuration because of the following error: %s.

ID: 2539

Severity: ERROR
Message: Unable to find resource '%s' of profile '%s' in profile version folders.

ID: 2540

Severity: INFO
Message: Which version of '%s' do you want to use (minimum %s)?.

ID: 2541

Severity: INFO
Message: 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.

ID: 2542

Severity: INFO
Message: {serverId}.

ID: 2543

Severity: INFO
Message: Unique server ID (used for identifying the server in the topology).

ID: 2544

Severity: INFO
Message: Server ID.

ID: 2545

Severity: ERROR
Message: An error occurred while configuring the server ID or the advertised listen address in global configuration: %s.

ID: 2546

Severity: ERROR
Message: Server ID '%s' is not valid. A Server ID is an ASCII alpha-numeric string possibly containing underscore and hyphen characters.

ID: 2547

Severity: INFO
Message: Do you want to set a value for parameter '%s'?.

ID: 2548

Severity: INFO
Message: No value specified for '%s'.

ID: 2549

Severity: INFO
Message: Do you want to add another value for parameter '%s'?.

ID: 2550

Severity: INFO
Message: Provide hostname for '%s'.

ID: 2551

Severity: INFO
Message: Provide port for '%s'.

ID: 2552

Severity: INFO
Message: This utility configures profiles in an offline OpenDJ server instance. %s.

ID: 2553

Severity: INFO
Message: Path of the server instance where profiles should be setup.

ID: 2554

Severity: INFO
Message: 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.

ID: 2555

Severity: INFO
Message: configure profiles in an offline OpenDJ server instance.

ID: 2556

Severity: ERROR
Message: You must select at least one profile by using the option '%s'. %s.

ID: 2557

Severity: ERROR
Message: Instance path '%s' is not a valid existing directory.

ID: 2558

Severity: ERROR
Message: Instance path '%s' does not contain an OpenDJ instance.

ID: 2559

Severity: ERROR
Message: An error occurred while initializing tool: %s.

ID: 2560

Severity: ERROR
Message: Profiles can be setup only when the server is offline, please stop the server and retry.

ID: 2561

Severity: ERROR
Message: An error occurred while reading file '%s': %s.

ID: 2562

Severity: ERROR
Message: Unable to setup the profile '%s' because it is already configured in the server.

ID: 2563

Severity: ERROR
Message: Unable to setup profiles '%s' because they are already configured in the server.

ID: 2564

Severity: INFO
Message: 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.

ID: 2565

Severity: INFO
Message: {deploymentId}.

ID: 2566

Severity: ERROR
Message: This deployment ID is using a key-pair generator which is not available in this Java environment. Retry the command with the Java environment that was used for generating this deployment ID.

ID: 2567

Severity: ERROR
Message: Unable to add certificates derived from deployment ID in keystore: %s.

ID: 2568

Severity: ERROR
Message: An error occurred while configuring replication: %s.

ID: 2569

Severity: INFO
Message: Deployment ID password.

ID: 2570

Severity: INFO
Message: {deploymentIdPassword}.

ID: 2571

Severity: INFO
Message: Provide the deployment ID password:.

ID: 2572

Severity: INFO
Message: Use deployment ID.

ID: 2573

Severity: ERROR
Message: An error occurred while attempting to update the administration connector port: %s.

ID: 2574

Severity: ERROR
Message: An error occurred while attempting to update the crypto manager in the Directory Server: %s.

ID: 2575

Severity: ERROR
Message: An error occurred while attempting to update the port on which to listen for LDAPS communication: %s.

ID: 2576

Severity: ERROR
Message: An error occurred while attempting to update the entry for the initial Directory Server root user: %s.

ID: 2577

Severity: ERROR
Message: Unable to bind to port %d. This port may already be in use, or you may not have permission to bind to it.

ID: 2578

Severity: ERROR
Message: Unable to bind to port %d. This port may already be in use, or you may not have permission to bind to it. On UNIX-based operating systems, non-root users may not be allowed to bind to ports 1 through 1024.

ID: 2579

Severity: ERROR
Message: Unable to write a required file. Make sure that the %s tool has sufficient permissions to create and write the file: %s.

ID: 2580

Severity: ERROR
Message: The root user password is missing, provide it with the --%s option.

ID: 2581

Severity: INFO
Message: install OpenDJ server.

ID: 2582

Severity: ERROR
Message: The provided password values do not match.

ID: 2583

Severity: INFO
Message: Use an existing certificate in a JCEKS keystore.

ID: 2584

Severity: INFO
Message: Use an existing certificate in a JKS keystore.

ID: 2585

Severity: INFO
Message: Use an existing certificate on a PKCS#11 token.

ID: 2586

Severity: INFO
Message: Use an existing certificate in a PKCS#12 keystore.

ID: 2587

Severity: INFO
Message: What would you like to do?.

ID: 2588

Severity: INFO
Message: Use nickname %s?.

ID: 2589

Severity: INFO
Message: Enable the server to run as a Windows Service?.

ID: 2590

Severity: INFO
Message: JCEKS keystore path:.

ID: 2591

Severity: INFO
Message: Java keystore (JKS) path:.

ID: 2592

Severity: INFO
Message: Keystore password:.

ID: 2593

Severity: INFO
Message: PKCS#12 keystore path:.

ID: 2594

Severity: ERROR
Message: Input tries limit reached (%d).

ID: 2595

Severity: ERROR
Message: Unable to resolve the server installation root directory.

ID: 2596

Severity: ERROR
Message: An error occurred while attempting to enable key manager provider entry: %s.

ID: 2597

Severity: INFO
Message: To see basic server status and configuration, you can launch %s.

ID: 2598

Severity: INFO
Message: Disabled.

ID: 2599

Severity: INFO
Message: Enabled.

ID: 2600

Severity: INFO
Message: Port on which the Administration Connector should listen for communication.

ID: 2601

Severity: INFO
Message: 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.

ID: 2602

Severity: INFO
Message: Start the server when the configuration is completed.

ID: 2603

Severity: INFO
Message: Enable StartTLS to allow secure communication with the server using the LDAP port.

ID: 2604

Severity: INFO
Message: Enable the server to run as a Windows Service.

ID: 2605

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

ID: 2606

Severity: INFO
Message: Port on which the Directory Server should listen for LDAP communication.

ID: 2607

Severity: INFO
Message: 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.

ID: 2608

Severity: INFO
Message: DN for the initial root user for the Directory Server.

ID: 2609

Severity: INFO
Message: Password for the initial root user for the Directory Server.

ID: 2610

Severity: INFO
Message: Skip the check to determine whether the specified ports are usable.

ID: 2611

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

ID: 2612

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

ID: 2613

Severity: INFO
Message: {path}.

ID: 2614

Severity: INFO
Message: Path of the file containing the keystore password. The specified path will be used as the configuration value in the new server.

ID: 2615

Severity: INFO
Message: {nickname}.

ID: 2616

Severity: INFO
Message: {rootUserDN}.

ID: 2617

Severity: INFO
Message: {rootUserPassword}.

ID: 2618

Severity: INFO
Message: {path}.

ID: 2619

Severity: INFO
Message: Path of the file containing the truststore password. The specified path will be used as the configuration value in the new server.

ID: 2620

Severity: ERROR
Message: The deployment ID password is missing, provide it with the --%s option.

ID: 2621

Severity: INFO
Message: Same as deployment ID password.

ID: 2622

Severity: INFO
Message: 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.

ID: 2623

Severity: INFO
Message: {bootstrapReplicationServer}.

ID: 2624

Severity: INFO
Message: 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).

ID: 2625

Severity: INFO
Message: Replication port.

ID: 2626

Severity: INFO
Message: Replication (secure).

ID: 2627

Severity: INFO
Message: What are the hostname and port number of the server to replicate with? (e.g ds.example.com on port 8989).

ID: 2628

Severity: INFO
Message: Replication server hostname.

ID: 2629

Severity: ERROR
Message: '%s' is not a valid hostname.

ID: 2630

Severity: INFO
Message: Replication port on %s.

ID: 2631

Severity: INFO
Message: Do you want to add another replication server.

ID: 2632

Severity: INFO
Message: Do you want to specify a server to replicate with? This step can be skipped if this is the first server in the topology.

ID: 2633

Severity: INFO
Message: Replicate with server.

ID: 2634

Severity: INFO
Message: Error Installing Backup Cloud Extension for Directory Server: %s.

ID: 2635

Severity: INFO
Message: The Directory Server shutdown process has been initiated by task %s.

ID: 2636

Severity: INFO
Message: The Directory Server shutdown process has been initiated by task %s: %s.

ID: 2637

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: 2638

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: 2639

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: 2640

Severity: ERROR
Message: An error occurred while attempting to load the contents of schema file %s into the server schema: %s.

ID: 2641

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: 2642

Severity: ERROR
Message: You do not have sufficient privileges to modify the server schema.

ID: 2643

Severity: ERROR
Message: You do not have sufficient privileges to initiate a Directory Server backup or backup purge.

ID: 2644

Severity: ERROR
Message: You do not have sufficient privileges to initiate a Directory Server restore.

ID: 2645

Severity: ERROR
Message: You do not have sufficient privileges to initiate an LDIF import.

ID: 2646

Severity: ERROR
Message: You do not have sufficient privileges to initiate an LDIF export.

ID: 2647

Severity: ERROR
Message: You do not have sufficient privileges to initiate a Directory Server restart.

ID: 2648

Severity: ERROR
Message: You do not have sufficient privileges to initiate a Directory Server shutdown.

ID: 2649

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: 2650

Severity: ERROR
Message: You do not have sufficient privileges to initiate an index rebuild.

ID: 2651

Severity: ERROR
Message: Invalid DN provided to the Initialize task: %s.

ID: 2652

Severity: ERROR
Message: Only users with the SERVER_LOCKDOWN privilege may place the server in lockdown mode.

ID: 2653

Severity: ERROR
Message: Only users with the SERVER_LOCKDOWN privilege connected from a loopback address may place the server in lockdown mode.

ID: 2654

Severity: ERROR
Message: Only users with the SERVER_LOCKDOWN privilege may cause the server to leave lockdown mode.

ID: 2655

Severity: ERROR
Message: Only users with the SERVER_LOCKDOWN privilege connected from a loopback address may cause the server to leave lockdown mode.

ID: 2656

Severity: ERROR
Message: You do not have sufficient privileges to terminate client connections.

ID: 2657

Severity: ERROR
Message: Unable to decode value %s as an integer connection ID.

ID: 2658

Severity: ERROR
Message: Attribute %s must be provided to specify the connection ID for the client to disconnect.

ID: 2659

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: 2660

Severity: INFO
Message: An administrator has terminated this client connection.

ID: 2661

Severity: ERROR
Message: There is no client connection with connection ID %s.

ID: 2662

Severity: INFO
Message: Add Schema File.

ID: 2663

Severity: INFO
Message: Backup.

ID: 2664

Severity: INFO
Message: Disconnect Client.

ID: 2665

Severity: INFO
Message: Lockdown.

ID: 2666

Severity: INFO
Message: Export.

ID: 2667

Severity: INFO
Message: Import.

ID: 2668

Severity: INFO
Message: Initialize Backend.

ID: 2669

Severity: INFO
Message: Initialize From Replica.

ID: 2670

Severity: INFO
Message: Leave Lockdown.

ID: 2671

Severity: INFO
Message: Rebuild Index.

ID: 2672

Severity: INFO
Message: Restore.

ID: 2673

Severity: INFO
Message: Set Generation ID.

ID: 2674

Severity: INFO
Message: Shutdown.

ID: 2675

Severity: INFO
Message: Unscheduled.

ID: 2676

Severity: INFO
Message: Disabled.

ID: 2677

Severity: INFO
Message: Waiting on start time.

ID: 2678

Severity: INFO
Message: Waiting on dependency.

ID: 2679

Severity: INFO
Message: Running.

ID: 2680

Severity: INFO
Message: Completed successfully.

ID: 2681

Severity: INFO
Message: Completed with errors.

ID: 2682

Severity: INFO
Message: Stopped by shutdown.

ID: 2683

Severity: INFO
Message: Stopped by error.

ID: 2684

Severity: INFO
Message: Stopped by administrator.

ID: 2685

Severity: INFO
Message: Canceled before starting.

ID: 2686

Severity: INFO
Message: Backend ID(s).

ID: 2687

Severity: INFO
Message: Backup Location.

ID: 2688

Severity: INFO
Message: LDIF File.

ID: 2689

Severity: INFO
Message: Backend ID.

ID: 2690

Severity: INFO
Message: Append To LDIF.

ID: 2691

Severity: INFO
Message: Compress LDIF.

ID: 2692

Severity: INFO
Message: Encrypt LDIF.

ID: 2693

Severity: INFO
Message: Sign Hash.

ID: 2694

Severity: INFO
Message: Include Attribute.

ID: 2695

Severity: INFO
Message: Exclude Attribute.

ID: 2696

Severity: INFO
Message: Include Filter.

ID: 2697

Severity: INFO
Message: Exclude Filter.

ID: 2698

Severity: INFO
Message: Include Branch.

ID: 2699

Severity: INFO
Message: Exclude Branch.

ID: 2700

Severity: INFO
Message: Wrap Column.

ID: 2701

Severity: INFO
Message: Backup Directory.

ID: 2702

Severity: INFO
Message: Backup ID.

ID: 2703

Severity: INFO
Message: LDIF File.

ID: 2704

Severity: INFO
Message: Backend ID.

ID: 2705

Severity: INFO
Message: Include Attribute.

ID: 2706

Severity: INFO
Message: Exclude Attribute.

ID: 2707

Severity: INFO
Message: Include Filter.

ID: 2708

Severity: INFO
Message: Exclude Filter.

ID: 2709

Severity: INFO
Message: Include Branch.

ID: 2710

Severity: INFO
Message: Exclude Branch.

ID: 2711

Severity: INFO
Message: Reject File.

ID: 2712

Severity: INFO
Message: Skip File.

ID: 2713

Severity: INFO
Message: Overwrite.

ID: 2714

Severity: INFO
Message: Skip Schema Validation.

ID: 2715

Severity: INFO
Message: Is Compressed.

ID: 2716

Severity: INFO
Message: Is Encrypted.

ID: 2717

Severity: INFO
Message: Clear Backend.

ID: 2718

Severity: INFO
Message: Process.

ID: 2719

Severity: INFO
Message: Cancel.

ID: 2720

Severity: INFO
Message: Disable.

ID: 2721

Severity: INFO
Message: Task was stopped by an administrator: %s.

ID: 2723

Severity: INFO
Message: Template File.

ID: 2724

Severity: INFO
Message: Random Seed.

ID: 2725

Severity: INFO
Message: Recurring.

ID: 2726

Severity: ERROR
Message: Index option cannot be specified when the rebuildAll or rebuildDegraded option is used.

ID: 2727

Severity: INFO
Message: Purge conflicts historical.

ID: 2728

Severity: ERROR
Message: Attribute %s has an invalid value. Reason: %s.

ID: 2729

Severity: INFO
Message: Reset change number index to begin with a given number and change.

ID: 2730

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: 2731

Severity: ERROR
Message: The change number index cannot be reset because this OpenDJ instance does not appear to be a replication server.

ID: 2732

Severity: ERROR
Message: Invalid change number (%d) specified, it must be greater than zero.

ID: 2733

Severity: ERROR
Message: Unable to reset the change number index: %s.

ID: 2734

Severity: ERROR
Message: The changes made by the add schema file task failed schema validation: %s.

ID: 2735

Severity: INFO
Message: Retrying.

ID: 2736

Severity: INFO
Message: Storage property.

ID: 2737

Severity: INFO
Message: Purge backup(s).

ID: 2738

Severity: INFO
Message: Backup location.

ID: 2739

Severity: INFO
Message: Backup ID(s).

ID: 2740

Severity: INFO
Message: Keep count.

ID: 2741

Severity: INFO
Message: Remove older than.

ID: 2742

Severity: INFO
Message: Force.

ID: 2743

Severity: INFO
Message: Backend name(s).

ID: 2744

Severity: ERROR
Message: Invalid DN provided to the Initialize Target task: %s.

ID: 2745

Severity: ERROR
Message: Invalid DN provided to the Purge Conflicts Historical task: %s.

ID: 2746

Severity: ERROR
Message: Invalid DN provided to the Reset Generation ID task: %s.

ID: 2747

Severity: INFO
Message: List available password storage schemes.

ID: 2748

Severity: INFO
Message: Clear-text password to encode or to compare against an encoded password.

ID: 2749

Severity: INFO
Message: Encoded password to compare against the clear-text password.

ID: 2750

Severity: INFO
Message: Scheme to use for the encoded password.

ID: 2751

Severity: ERROR
Message: An error occurred while parsing the command-line arguments: %s.

ID: 2752

Severity: ERROR
Message: No clear-text password was specified. Use --%s, --%s or --%s to specify the password to encode.

ID: 2753

Severity: ERROR
Message: No password storage scheme was specified. Use the --%s argument to specify the storage scheme.

ID: 2754

Severity: ERROR
Message: An unexpected error occurred while attempting to bootstrap the Directory Server client-side code: %s.

ID: 2755

Severity: ERROR
Message: An error occurred while trying to load the Directory Server configuration: %s.

ID: 2756

Severity: ERROR
Message: An error occurred while trying to load the Directory Server schema: %s.

ID: 2757

Severity: ERROR
Message: An error occurred while trying to initialize the core Directory Server configuration: %s.

ID: 2758

Severity: ERROR
Message: An error occurred while trying to initialize the Directory Server password storage schemes: %s.

ID: 2759

Severity: ERROR
Message: No password storage schemes have been configured for use in the Directory Server.

ID: 2760

Severity: ERROR
Message: Password storage scheme "%s" is not configured for use in the Directory Server.

ID: 2761

Severity: INFO
Message: The provided clear-text and encoded passwords match.

ID: 2762

Severity: INFO
Message: The provided clear-text and encoded passwords do not match.

ID: 2763

Severity: ERROR
Message: An error occurred while attempting to encode the clear-text password: %s.

ID: 2764

Severity: INFO
Message: Path to the LDIF file to write. All paths are relative to the server's installation directory, which can be remote.

ID: 2765

Severity: INFO
Message: Append an existing LDIF file rather than overwriting it.

ID: 2766

Severity: INFO
Message: Backend ID for the backend to export.

ID: 2767

Severity: INFO
Message: Base DN of a branch to exclude from the LDIF export.

ID: 2768

Severity: INFO
Message: Attribute to include in the LDIF export.

ID: 2769

Severity: INFO
Message: Attribute to exclude from the LDIF export.

ID: 2770

Severity: INFO
Message: Filter to identify entries to include in the LDIF export.

ID: 2771

Severity: INFO
Message: Filter to identify entries to exclude from the LDIF export.

ID: 2772

Severity: INFO
Message: Column at which to wrap long lines (0 for no wrapping).

ID: 2773

Severity: INFO
Message: Compress the LDIF data as it is exported.

ID: 2774

Severity: ERROR
Message: Unable to decode exclude filter string "%s" as a valid search filter: %s.

ID: 2775

Severity: ERROR
Message: Unable to decode include filter string "%s" as a valid search filter: %s.

ID: 2776

Severity: ERROR
Message: Unable to decode base DN string "%s" as a valid distinguished name: %s.

ID: 2777

Severity: ERROR
Message: Multiple Directory Server backends are configured with the requested backend ID "%s".

ID: 2778

Severity: ERROR
Message: None of the Directory Server backends are configured with the requested backend ID "%s".

ID: 2779

Severity: ERROR
Message: Unable to decode exclude branch string "%s" as a valid distinguished name: %s.

ID: 2780

Severity: ERROR
Message: Unable to decode wrap column value "%s" as an integer.

ID: 2781

Severity: ERROR
Message: An error occurred while attempting to process the LDIF export: %s.

ID: 2782

Severity: ERROR
Message: Unable to load class %s referenced in configuration entry %s for use as a Directory Server backend: %s.

ID: 2783

Severity: ERROR
Message: Unable to create an instance of class %s referenced in configuration entry %s as a Directory Server backend: %s.

ID: 2784

Severity: INFO
Message: Path to the LDIF file to import. All paths are relative to the server's installation directory, which can be remote.

ID: 2785

Severity: INFO
Message: Backend ID for the backend to import.

ID: 2786

Severity: INFO
Message: Base DN of a branch to exclude from the LDIF import.

ID: 2787

Severity: INFO
Message: Attribute to include in the LDIF import.

ID: 2788

Severity: INFO
Message: Attribute to exclude from the LDIF import.

ID: 2789

Severity: INFO
Message: Filter to identify entries to include in the LDIF import.

ID: 2790

Severity: INFO
Message: Filter to identify entries to exclude from the LDIF import.

ID: 2791

Severity: INFO
Message: Write rejected entries to the specified file.

ID: 2792

Severity: INFO
Message: Overwrite an existing rejects and/or skip file rather than appending to it.

ID: 2793

Severity: INFO
Message: LDIF file is compressed.

ID: 2794

Severity: ERROR
Message: Unable to decode exclude filter string "%s" as a valid search filter: %s.

ID: 2795

Severity: ERROR
Message: Unable to decode include filter string "%s" as a valid search filter: %s.

ID: 2796

Severity: ERROR
Message: Imported branches or backend IDs can not span across multiple Directory Server backends.

ID: 2797

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: 2798

Severity: ERROR
Message: Unable to decode exclude branch string "%s" as a valid distinguished name: %s.

ID: 2799

Severity: ERROR
Message: An error occurred while trying to open the rejects file %s for writing: %s.

ID: 2800

Severity: ERROR
Message: An error occurred while attempting to process the LDIF import: %s.

ID: 2801

Severity: INFO
Message: Base DN of a backend supporting indexing. Verification is performed on indexes within the scope of the given base DN.

ID: 2802

Severity: INFO
Message: 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.

ID: 2803

Severity: INFO
Message: 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.

ID: 2804

Severity: ERROR
Message: An error occurred while attempting to perform index verification: %s.

ID: 2805

Severity: ERROR
Message: Only one index at a time may be verified for cleanliness.

ID: 2806

Severity: ERROR
Message: The backend does not support indexing.

ID: 2807

Severity: ERROR
Message: The Directory Server backend with backend ID "%s" does not provide a mechanism for performing LDIF exports.

ID: 2808

Severity: ERROR
Message: The Directory Server backend with backend ID %s does not provide a mechanism for performing LDIF imports.

ID: 2809

Severity: INFO
Message: Base DN of a branch to include in the LDIF import.

ID: 2810

Severity: ERROR
Message: Unable to decode include branch string "%s" as a valid distinguished name: %s.

ID: 2811

Severity: ERROR
Message: Provided include base DN "%s" is not handled by the backend with backend ID %s.

ID: 2812

Severity: ERROR
Message: Multiple Directory Server backends are configured to support base DN "%s".

ID: 2813

Severity: ERROR
Message: None of the Directory Server backends are configured to support the requested base DN "%s".

ID: 2814

Severity: INFO
Message: Base DN of a branch to include in the LDIF export.

ID: 2815

Severity: ERROR
Message: Provided include base DN "%s" is not handled by the backend with backend ID %s.

ID: 2816

Severity: ERROR
Message: An error occurred while attempting to initialize the crypto manager: %s.

ID: 2817

Severity: ERROR
Message: An error occurred while attempting to initialize the subentry manager: %s.

ID: 2818

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: 2819

Severity: WARNING
Message: An error occurred while attempting to release the exclusive lock for backend %s: %s. This lock should automatically be cleared when the import process exits, so no further action should be required.

ID: 2820

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: 2821

Severity: WARNING
Message: An error occurred while attempting to release the shared lock for backend %s: %s. This lock should automatically be cleared when the export process exits, so no further action should be required.

ID: 2822

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: 2823

Severity: WARNING
Message: An error occurred while attempting to release the shared lock for backend %s: %s. This lock should automatically be cleared when the verification process exits, so no further action should be required.

ID: 2824

Severity: INFO
Message: Skip schema validation during the LDIF import.

ID: 2825

Severity: INFO
Message: Use the authentication password syntax rather than the user password syntax.

ID: 2826

Severity: ERROR
Message: Authentication password storage scheme "%s" is not configured for use in the Directory Server.

ID: 2827

Severity: ERROR
Message: The provided password is not a valid encoded authentication password value: %s.

ID: 2828

Severity: ERROR
Message: An error occurred while attempting to initialize the password policy components: %s.

ID: 2829

Severity: INFO
Message: Reason the server is being stopped or restarted.

ID: 2830

Severity: INFO
Message: 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.

ID: 2831

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: 2832

Severity: ERROR
Message: ERROR: An I/O error occurred while attempting to communicate with the Directory Server: %s.

ID: 2833

Severity: INFO
Message: Use quiet mode (no output).

ID: 2834

Severity: INFO
Message: Path to a MakeLDIF template to use to generate the import data.

ID: 2835

Severity: ERROR
Message: Neither the %s nor the %s argument was provided. One of these arguments must be given to specify the source for the LDIF data to be imported.

ID: 2836

Severity: ERROR
Message: Unable to parse the specified file %s as a MakeLDIF template file: %s.

ID: 2837

Severity: INFO
Message: 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.

ID: 2838

Severity: INFO
Message: Path to the file to watch for deletion.

ID: 2839

Severity: INFO
Message: Path to a file containing log output to monitor.

ID: 2840

Severity: INFO
Message: Maximum length of time in seconds to wait for the target file to be deleted before exiting.

ID: 2841

Severity: WARNING
Message: WARNING: Unable to open log file %s for reading: %s.

ID: 2842

Severity: INFO
Message: 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.

ID: 2843

Severity: INFO
Message: This utility can be used to export data from a Directory Server backend in LDIF form.

ID: 2844

Severity: INFO
Message: 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.

ID: 2845

Severity: INFO
Message: 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.

ID: 2846

Severity: INFO
Message: This utility ensures that index data is consistent within an indexed backend database. Stop the server before running this tool.

ID: 2847

Severity: INFO
Message: This utility can be used to wait for a file to be removed from the filesystem.

ID: 2848

Severity: ERROR
Message: The provided password is not a valid encoded user password value: %s.

ID: 2849

Severity: INFO
Message: Use the LDAP compare result as an exit code for the password comparison.

ID: 2850

Severity: INFO
Message: Exclude operational attributes from the LDIF export.

ID: 2851

Severity: INFO
Message: Server already stopped.

ID: 2852

Severity: INFO
Message: Stopping Server...

ID: 2853

Severity: ERROR
Message: Could not find the service name for the server.

ID: 2854

Severity: ERROR
Message: An unexpected error occurred starting the server as a windows service.

ID: 2855

Severity: ERROR
Message: An unexpected error occurred stopping the server windows service.

ID: 2856

Severity: INFO
Message: This utility can be used to configure the server as a Windows service.

ID: 2857

Severity: INFO
Message: Enables the server as a Windows service.

ID: 2858

Severity: INFO
Message: Disables the server as a Windows service and stops the server.

ID: 2859

Severity: INFO
Message: Provides information about the state of the server as a Windows service.

ID: 2860

Severity: ERROR
Message: You can only provide one of the following arguments: enableService, disableService, serviceState or cleanupService.

ID: 2861

Severity: ERROR
Message: You must provide at least one of the following arguments: enableService, disableService or serviceState or cleanupService.

ID: 2862

Severity: INFO
Message: %s.

ID: 2863

Severity: INFO
Message: Next Generation Directory Server. Installation path: %s.

ID: 2864

Severity: INFO
Message: The server was successfully enabled to run as a Windows service.

ID: 2865

Severity: INFO
Message: The server was already enabled to run as a Windows service.

ID: 2866

Severity: ERROR
Message: The server could not be enabled to run as a Windows service. The service name is already in use.

ID: 2867

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: 2868

Severity: INFO
Message: The server was successfully disabled as a Windows service.

ID: 2869

Severity: INFO
Message: The server was already disabled as a Windows service.

ID: 2870

Severity: WARNING
Message: The server has been marked for deletion as a Windows Service.

ID: 2871

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: 2872

Severity: INFO
Message: The server is enabled as a Windows service. The service name for the server is: %s.

ID: 2873

Severity: INFO
Message: The server is disabled as a Windows service.

ID: 2874

Severity: ERROR
Message: An unexpected error occurred trying to retrieve the state of the server as a Windows service.

ID: 2875

Severity: INFO
Message: Path to a file to which the command will write the output.

ID: 2876

Severity: WARNING
Message: WARNING: Unable to open output file %s for writing: %s.

ID: 2877

Severity: INFO
Message: Disables the server as a Windows service and removes the Windows registry information associated with the specified service.

ID: 2878

Severity: INFO
Message: Clean up of service %s was successful.

ID: 2879

Severity: ERROR
Message: Could not find the service with name %s.

ID: 2880

Severity: WARNING
Message: Service %s has been marked for deletion.

ID: 2881

Severity: ERROR
Message: An unexpected error occurred cleaning up the service %s.

ID: 2882

Severity: INFO
Message: This utility can be used to rebuild index data within an indexed backend database.

ID: 2883

Severity: INFO
Message: Base DN of a backend supporting indexing. Rebuild is performed on indexes within the scope of the given base DN.

ID: 2884

Severity: INFO
Message: 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.

ID: 2885

Severity: ERROR
Message: An error occurred while attempting to perform index rebuild: %s.

ID: 2886

Severity: ERROR
Message: The backend does not support rebuilding of indexes.

ID: 2887

Severity: ERROR
Message: At least one index must be specified for the rebuild process.

ID: 2888

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: 2889

Severity: WARNING
Message: An error occurred while attempting to release the shared lock for backend %s: %s. This lock should automatically be cleared when the rebuild process exits, so no further action should be required.

ID: 2890

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: 2891

Severity: ERROR
Message: The specified LDIF file %s cannot be read.

ID: 2892

Severity: INFO
Message: 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).

ID: 2893

Severity: INFO
Message: Write skipped entries to the specified file.

ID: 2894

Severity: ERROR
Message: An error occurred while trying to open the skip file %s for writing: %s.

ID: 2895

Severity: INFO
Message: 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).

ID: 2896

Severity: INFO
Message: Remove all entries for all base DNs in the backend before importing.

ID: 2897

Severity: ERROR
Message: Neither the %s nor the %s argument was provided. One of these arguments must be given to specify the backend for the LDIF data to be imported to.

ID: 2898

Severity: INFO
Message: %s task %s scheduled to start immediately.

ID: 2899

Severity: ERROR
Message: This tool may only be used on UNIX-based systems.

ID: 2900

Severity: INFO
Message: Create an RC script or systemd service that may be used to start, stop, and restart the Directory Server on UNIX-based systems.

ID: 2901

Severity: INFO
Message: The path to the RC script to create.

ID: 2902

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: 2903

Severity: ERROR
Message: An error occurred while attempting to generate the RC script: %s.

ID: 2904

Severity: INFO
Message: List the base DNs in a backend.

ID: 2905

Severity: INFO
Message: The backend ID of the backend.

ID: 2906

Severity: INFO
Message: The base DN within the backend.

ID: 2907

Severity: INFO
Message: The name of the index.

ID: 2908

Severity: INFO
Message: Do not try to decode backend data to their appropriate types.

ID: 2909

Severity: INFO
Message: 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.

ID: 2910

Severity: INFO
Message: Only show records with keys that should be ordered before the provided value using the comparator for the database container.

ID: 2911

Severity: INFO
Message: Only show records with keys that should be ordered after the provided value using the comparator for the database container.

ID: 2912

Severity: INFO
Message: Only show records whose data is no larger than the provided value.

ID: 2913

Severity: INFO
Message: Only show records whose data is no smaller than the provided value.

ID: 2914

Severity: INFO
Message: Backend ID.

ID: 2915

Severity: INFO
Message: Base DN.

ID: 2916

Severity: ERROR
Message: None of the Directory Server JE backends are configured with the requested backend ID %s.

ID: 2917

Severity: ERROR
Message: None of the entry containers are configured with the requested base DN %s in backend %s.

ID: 2918

Severity: ERROR
Message: Unable to decode base DN string "%s" as a valid distinguished name: %s.

ID: 2919

Severity: INFO
Message: Median.

ID: 2920

Severity: INFO
Message: Key Count.

ID: 2921

Severity: INFO
Message: Size.

ID: 2922

Severity: INFO
Message: Records.

ID: 2923

Severity: INFO
Message: Index Name.

ID: 2924

Severity: INFO
Message: Type.

ID: 2925

Severity: INFO
Message: Secure.

ID: 2926

Severity: INFO
Message: Entry Limit.

ID: 2927

Severity: INFO
Message: Mean.

ID: 2928

Severity: WARNING
Message: An error occurred while attempting to release the shared lock for backend %s: %s. This lock should automatically be cleared when the process exits, so no further action should be required.

ID: 2929

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: 2930

Severity: INFO
Message: Over.

ID: 2931

Severity: INFO
Message: 80th.

ID: 2932

Severity: INFO
Message: 95th.

ID: 2933

Severity: INFO
Message: 99th.

ID: 2934

Severity: ERROR
Message: A sub-command must be specified.

ID: 2935

Severity: INFO
Message: The name of the user account under which the server should run.

ID: 2936

Severity: INFO
Message: The path to the Java installation that should be used to run the server.

ID: 2937

Severity: INFO
Message: A set of arguments that should be passed to the JVM when running the server.

ID: 2938

Severity: ERROR
Message: The directory %s specified as the OPENDJ_JAVA_HOME path does not exist or is not a directory.

ID: 2939

Severity: ERROR
Message: The argument '%s' is incompatible with '%s'.

ID: 2940

Severity: INFO
Message: 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.

ID: 2941

Severity: INFO
Message: Print a summary of tasks.

ID: 2942

Severity: INFO
Message: ID of a particular task about which this tool will display information.

ID: 2943

Severity: INFO
Message: refresh.

ID: 2944

Severity: INFO
Message: cancel task.

ID: 2945

Severity: INFO
Message: view logs.

ID: 2946

Severity: INFO
Message: Enter the number of a task to cancel.

ID: 2947

Severity: ERROR
Message: Invalid menu item or task number '%s'.

ID: 2948

Severity: INFO
Message: ID.

ID: 2949

Severity: INFO
Message: Type.

ID: 2950

Severity: INFO
Message: Status.

ID: 2951

Severity: INFO
Message: Scheduled Start Time.

ID: 2952

Severity: INFO
Message: Actual Start Time.

ID: 2953

Severity: INFO
Message: Completion Time.

ID: 2954

Severity: INFO
Message: Dependencies.

ID: 2955

Severity: INFO
Message: Failed Dependency Action.

ID: 2956

Severity: INFO
Message: Log Message(s).

ID: 2957

Severity: INFO
Message: Last Log Message.

ID: 2958

Severity: INFO
Message: Email Upon Completion.

ID: 2959

Severity: INFO
Message: Email Upon Error.

ID: 2960

Severity: INFO
Message: Task %s canceled.

ID: 2961

Severity: ERROR
Message: Error retrieving task entry %s: %s.

ID: 2962

Severity: ERROR
Message: There are no tasks with ID %s.

ID: 2963

Severity: INFO
Message: Task Details.

ID: 2964

Severity: INFO
Message: %s Options.

ID: 2965

Severity: INFO
Message: No tasks exist.

ID: 2966

Severity: INFO
Message: None.

ID: 2967

Severity: INFO
Message: None Specified.

ID: 2968

Severity: INFO
Message: Immediate execution.

ID: 2969

Severity: INFO
Message: Error connecting to the directory server: '%s'. Verify that the connection options are correct and that the server is running.

ID: 2970

Severity: INFO
Message: ID of a particular task to cancel.

ID: 2971

Severity: INFO
Message: Show only tasks of this type.

ID: 2972

Severity: INFO
Message: {taskType}.

ID: 2973

Severity: INFO
Message: Show only tasks with this status.

ID: 2974

Severity: INFO
Message: {taskStatus}.

ID: 2975

Severity: ERROR
Message: Error canceling task '%s': %s.

ID: 2976

Severity: ERROR
Message: Error accessing logs for task '%s': %s.

ID: 2977

Severity: ERROR
Message: Task at index %d is not cancelable.

ID: 2978

Severity: INFO
Message: There are currently no cancelable tasks.

ID: 2979

Severity: ERROR
Message: There are no tasks defined with ID '%s'.

ID: 2980

Severity: ERROR
Message: Task '%s' has finished and cannot be canceled.

ID: 2981

Severity: ERROR
Message: State for task '%s' cannot be determined.

ID: 2982

Severity: INFO
Message: 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.

ID: 2983

Severity: ERROR
Message: The start date/time must in YYYYMMDDhhmmssZ format for UTC time or YYYYMMDDhhmmss for local time.

ID: 2984

Severity: INFO
Message: %s task %s scheduled to start %s.

ID: 2985

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: 2986

Severity: INFO
Message: Task Scheduling Options.

ID: 2987

Severity: INFO
Message: Task Backend Connection Options.

ID: 2988

Severity: INFO
Message: Email address of a recipient to be notified when the task completes. This option may be specified more than once.

ID: 2989

Severity: INFO
Message: Email address of a recipient to be notified if an error occurs when this task executes. This option may be specified more than once.

ID: 2990

Severity: INFO
Message: ID of a task upon which this task depends. A task will not start execution until all its dependencies have completed execution.

ID: 2991

Severity: INFO
Message: Action this task will take should one if its dependent tasks fail. The value must be one of %s. If not specified defaults to %s.

ID: 2992

Severity: ERROR
Message: The option %s is only applicable when scheduling this operation as a task.

ID: 2993

Severity: ERROR
Message: The value %s for option %s is not a valid email address.

ID: 2994

Severity: ERROR
Message: The failed dependency action value %s is invalid. The value must be one of %s.

ID: 2995

Severity: ERROR
Message: The failed dependency action option is to be used in conjunction with one or more dependencies.

ID: 2996

Severity: ERROR
Message: Error: task %s is not in a cancelable state.

ID: 2997

Severity: INFO
Message: {ldifFile}.

ID: 2998

Severity: INFO
Message: {seed}.

ID: 2999

Severity: INFO
Message: {host}.

ID: 3000

Severity: INFO
Message: {baseDN}.

IDs: 3001-3500

ID: 3001

Severity: INFO
Message: {path}.

ID: 3002

Severity: INFO
Message: {filter}.

ID: 3003

Severity: INFO
Message: {backendName}.

ID: 3004

Severity: INFO
Message: {startTime}.

ID: 3005

Severity: INFO
Message: {emailAddress}.

ID: 3006

Severity: INFO
Message: {taskID}.

ID: 3007

Severity: INFO
Message: {action}.

ID: 3008

Severity: INFO
Message: {serviceName}.

ID: 3009

Severity: INFO
Message: {userName}.

ID: 3010

Severity: INFO
Message: {args}.

ID: 3011

Severity: INFO
Message: {databaseName}.

ID: 3012

Severity: INFO
Message: {maxKeyValue}.

ID: 3013

Severity: INFO
Message: {minKeyValue}.

ID: 3014

Severity: INFO
Message: {maxDataSize}.

ID: 3015

Severity: INFO
Message: {minDataSize}.

ID: 3016

Severity: INFO
Message: {clearPW}.

ID: 3017

Severity: INFO
Message: {encodedPW}.

ID: 3018

Severity: INFO
Message: {scheme}.

ID: 3019

Severity: INFO
Message: {branchDN}.

ID: 3020

Severity: INFO
Message: {attribute}.

ID: 3021

Severity: INFO
Message: {wrapColumn}.

ID: 3022

Severity: INFO
Message: {templateFile}.

ID: 3023

Severity: INFO
Message: {rejectFile}.

ID: 3024

Severity: INFO
Message: {skipFile}.

ID: 3025

Severity: INFO
Message: {index}.

ID: 3026

Severity: INFO
Message: {stopReason}.

ID: 3027

Severity: INFO
Message: {stopTime}.

ID: 3028

Severity: INFO
Message: {seconds}.

ID: 3029

Severity: INFO
Message: %s task %s has been successfully completed.

ID: 3030

Severity: INFO
Message: %s task %s did not complete successfully.

ID: 3031

Severity: INFO
Message: {schedulePattern}.

ID: 3032

Severity: ERROR
Message: An error occurred while attempting to initialize server components to run the tool: %s.

ID: 3033

Severity: ERROR
Message: The %s argument is not supported for online imports.

ID: 3034

Severity: INFO
Message: Indicates the task is recurring and will be scheduled according to the value argument expressed in crontab(5) compatible time/date pattern.

ID: 3035

Severity: INFO
Message: Recurring %s task %s scheduled successfully.

ID: 3036

Severity: INFO
Message: Exporting to %s.

ID: 3037

Severity: INFO
Message: yes.

ID: 3038

Severity: INFO
Message: y.

ID: 3039

Severity: INFO
Message: no.

ID: 3040

Severity: INFO
Message: n.

ID: 3041

Severity: ERROR
Message: The specified start time '%s' has already passed.

ID: 3042

Severity: ERROR
Message: The specified stop time '%s' has already passed.

ID: 3043

Severity: INFO
Message: r.

ID: 3044

Severity: INFO
Message: c.

ID: 3045

Severity: INFO
Message: l.

ID: 3046

Severity: ERROR
Message: The timeout of '%d' seconds has been reached. You can use the argument '--%s' to increase this timeout.

ID: 3047

Severity: INFO
Message: Path to temporary directory for index scratch files during LDIF import.

ID: 3048

Severity: INFO
Message: {directory}.

ID: 3049

Severity: INFO
Message: 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.

ID: 3050

Severity: INFO
Message: {count}.

ID: 3051

Severity: ERROR
Message: The value %s for threadCount cannot be parsed: %s.

ID: 3052

Severity: INFO
Message: The password to encode or to compare against an encoded password is interactively asked to the user.

ID: 3053

Severity: INFO
Message: Please enter the password :.

ID: 3054

Severity: INFO
Message: Please renter the password:.

ID: 3055

Severity: ERROR
Message: Provided passwords don’t matched.

ID: 3056

Severity: ERROR
Message: Cannot read password from the input: %s.

ID: 3057

Severity: INFO
Message: Rebuild all indexes, including any DN2ID, DN2URI, VLV and extensible indexes. Cannot be used with the "-i" option or the "--rebuildDegraded" option.

ID: 3058

Severity: INFO
Message: {directory}.

ID: 3059

Severity: INFO
Message: Path to temporary directory for index scratch files during index rebuilding.

ID: 3060

Severity: INFO
Message: {period}.

ID: 3061

Severity: INFO
Message: 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.

ID: 3062

Severity: ERROR
Message: The Windows Service was successfully configured but there was an error starting it. Error code starting Windows Service: %d.

ID: 3063

Severity: INFO
Message: Do not display backend data, just statistics.

ID: 3064

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: 3065

Severity: INFO
Message: Rebuild all degraded indexes, including any DN2ID, DN2URI, VLV and extensible indexes. Cannot be used with the "-i" option or the "--rebuildAll" option.

ID: 3066

Severity: INFO
Message: 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.

ID: 3067

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: 3068

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: 3069

Severity: ERROR
Message: The OpenDJ binary version '%s' does not match the installed configuration version '%s'. Please run upgrade before continuing.

ID: 3070

Severity: INFO
Message: 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.

ID: 3071

Severity: INFO
Message: 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 '%s' option.

ID: 3072

Severity: INFO
Message: Upgrades only application data. OpenDJ configuration must have been upgraded before.

ID: 3073

Severity: INFO
Message: Upgrades OpenDJ configuration and application data so that it is compatible with the installed binaries. %n%nThis tool should be run immediately after upgrading the OpenDJ binaries and before restarting the server. %n%nNOTE: 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.

ID: 3074

Severity: ERROR
Message: The upgrade failed to complete for the following reason: %s.

ID: 3075

Severity: INFO
Message: Performing upgrade.

ID: 3076

Severity: INFO
Message: OpenDJ Upgrade Utility.

ID: 3077

Severity: ERROR
Message: OpenDJ cannot be upgraded because the server is currently running. Please stop the server and try again.

ID: 3078

Severity: ERROR
Message: OpenDJ has already been upgraded to version %s.

ID: 3079

Severity: ERROR
Message: An unexpected error occurred while attempting to display a notification: %s.

ID: 3080

Severity: ERROR
Message: An unexpected error occurred while attempting to display a confirmation : %s.

ID: 3081

Severity: INFO
Message: ...Change(s) done in %s (x%s).

ID: 3082

Severity: INFO
Message: ...No change applied in %s.

ID: 3083

Severity: ERROR
Message: An error occurred while performing an upgrade task: %s.

ID: 3084

Severity: INFO
Message: %s.%nDo you want to make this change?.

ID: 3085

Severity: INFO
Message: The upgrade is ready to proceed. Do you wish to continue?.

ID: 3086

Severity: INFO
Message: The upgrade has been canceled.

ID: 3087

Severity: ERROR
Message: No %s with OID %s exists in the schema.

ID: 3088

Severity: ERROR
Message: An error occurred when trying to upgrade the config/upgrade folder: %s.

ID: 3089

Severity: INFO
Message: Preparing to upgrade.

ID: 3090

Severity: INFO
Message: This tool cannot be used for upgrading versions of OpenDJ which are older than '%s'. Please upgrade to the latest 6.5 revision first before attempting further upgrades.

ID: 3092

Severity: INFO
Message: Do you accept the License Agreement?.

ID: 3093

Severity: INFO
Message: An error occurred while copying the file '%s' to '%s'.

ID: 3094

Severity: INFO
Message: An error occurred while deleting directory '%s'. Check that you have the rights to delete this directory and that there is no other application using it. Error was: %s.

ID: 3095

Severity: INFO
Message: An error occurred while deleting file '%s': %s. Check that you have the rights to delete this file and that there is no other application using it.

ID: 3096

Severity: ERROR
Message: The upgrade failed because %d errors were encountered. Please check log for further details.

ID: 3097

Severity: ERROR
Message: An error occurred while copying the schema file '%s': %s.

ID: 3098

Severity: ERROR
Message: An error occurred while adding one or more attributes to the schema file '%s': %s.

ID: 3099

Severity: INFO
Message: See '%s' for a detailed log of this operation.

ID: 3100

Severity: INFO
Message: Replacing schema file '%s'.

ID: 3101

Severity: INFO
Message: Archiving concatenated schema.

ID: 3102

Severity: INFO
Message: Adding '%s' configuration file.

ID: 3103

Severity: ERROR
Message: An error occurred while adding configuration file '%s': %s.

ID: 3104

Severity: INFO
Message: Rebuilding index(es) %s for base dn(s): %s.

ID: 3105

Severity: INFO
Message: Rebuild index task ends.

ID: 3106

Severity: INFO
Message: Performing post upgrade tasks.

ID: 3107

Severity: INFO
Message: Post upgrade tasks complete.

ID: 3108

Severity: ERROR
Message: An error occurred during post upgrade task. Process aborted. Please check log for further details.

ID: 3110

Severity: INFO
Message: You have to rebuild all indexes manually to get a fully functional server.

ID: 3111

Severity: ERROR
Message: Invalid log file %s.

ID: 3112

Severity: INFO
Message: The rebuild index tool arguments are %s.

ID: 3113

Severity: INFO
Message: Rebuilding all indexes.

ID: 3114

Severity: INFO
Message: End of the upgrade process.

ID: 3115

Severity: ERROR
Message: '%s' is missing or empty, it is probably corrupted.

ID: 3116

Severity: INFO
Message: No indexes to rebuild for backend '%s'.

ID: 3117

Severity: INFO
Message: The classes folder has been renamed to '%s' to avoid compatibility issues.

ID: 3118

Severity: INFO
Message: List the pluggable backends.

ID: 3119

Severity: INFO
Message: 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.

ID: 3120

Severity: INFO
Message: List the indexes associated with a pluggable backend. This subcommand may take a long time to complete depending on the size of the backend.

ID: 3121

Severity: INFO
Message: Dump records from an index, decoding keys and values. Depending on index size, this subcommand can generate lots of output.

ID: 3122

Severity: INFO
Message: 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.

ID: 3123

Severity: INFO
Message: Raw DB Name.

ID: 3124

Severity: ERROR
Message: An error occurred while listing the base DNs: %s.

ID: 3125

Severity: ERROR
Message: An error occurred while listing indexes: %s.

ID: 3126

Severity: ERROR
Message: An unexpected error occurred while attempting to initialize the backend '%s': %s.

ID: 3127

Severity: ERROR
Message: An unexpected error occurred while attempting to read and/or decode records from an index: %s.

ID: 3128

Severity: ERROR
Message: No index exists with the requested name '%s' in base DN '%s' and backend '%s'.

ID: 3129

Severity: ERROR
Message: Cannot specify a minimum key both as a string and as an hexadecimal string.

ID: 3130

Severity: ERROR
Message: Cannot specify a maximum key both as a string and as an hexadecimal string.

ID: 3131

Severity: ERROR
Message: An error occurred while processing arguments: %s.

ID: 3132

Severity: ERROR
Message: An error occurred while trying to execute %s: %s.

ID: 3133

Severity: INFO
Message: Total Keys.

ID: 3134

Severity: INFO
Message: Keys Size.

ID: 3135

Severity: INFO
Message: Values Size.

ID: 3136

Severity: INFO
Message: Total Size.

ID: 3137

Severity: INFO
Message: Uses SI Units for printing sizes.

ID: 3138

Severity: INFO
Message: Write hexadecimal data on a single line instead of pretty format.

ID: 3139

Severity: INFO
Message: %nTotal: %d%n.

ID: 3140

Severity: INFO
Message: %nIndex: %s%n.

ID: 3141

Severity: INFO
Message: Over index-entry-limit keys: %s%n.

ID: 3142

Severity: INFO
Message: %nTotal Records: %d%n.

ID: 3143

Severity: INFO
Message: Total / Average Key Size: %d bytes / %d bytes%n.

ID: 3144

Severity: INFO
Message: Total / Average Data Size: %d bytes / %d bytes%n.

ID: 3145

Severity: ERROR
Message: At key number %d, %s:.

ID: 3146

Severity: INFO
Message: Key (len %d):.

ID: 3147

Severity: INFO
Message: Value (len %d):.

ID: 3148

Severity: ERROR
Message: Data decoder for printing is not available, should use hex dump.

ID: 3149

Severity: ERROR
Message: No storage index exists with the requested name %s in backend %s.

ID: 3150

Severity: INFO
Message: This utility can be used to debug a backend.

ID: 3151

Severity: INFO
Message: {indexName}.

ID: 3152

Severity: INFO
Message: The raw database name.

ID: 3153

Severity: ERROR
Message: An error occurred while initializing server backends: %s.

ID: 3154

Severity: ERROR
Message: An error occurred while initializing plugins: %s.

ID: 3155

Severity: ERROR
Message: Subsystem %s should be initialized first.

ID: 3156

Severity: INFO
Message: OpenDJ configuration will be upgraded from version %s to %s.

ID: 3157

Severity: INFO
Message: OpenDJ data can't be upgraded because the configuration has not been upgraded yet. Upgrade the configuration first before upgrading the data.

ID: 3158

Severity: INFO
Message: OpenDJ data will be upgraded from version %s to %s.

ID: 3159

Severity: INFO
Message: OpenDJ data will be upgraded to version %s.

ID: 3160

Severity: INFO
Message: OpenDJ data was successfully upgraded to version %s.

ID: 3161

Severity: ERROR
Message: OpenDJ data has already been upgraded to version %s.

ID: 3162

Severity: INFO
Message: OpenDJ configuration was successfully upgraded to version %s.

ID: 3163

Severity: ERROR
Message: The OpenDJ binary version '%s' does not match the installed data version '%s'. Please run 'upgrade --dataOnly' before continuing.

ID: 3164

Severity: INFO
Message: Upgrade failed to delete the obsolete file '%s'. Reason: %s.

ID: 3165

Severity: INFO
Message: This utility can be used to debug changelog and changenumber files.

ID: 3166

Severity: INFO
Message: Dump the change number DB.

ID: 3167

Severity: INFO
Message: Dump the replica DB for a given domain and replica.

ID: 3168

Severity: INFO
Message: Dump a replica DB file.

ID: 3169

Severity: INFO
Message: The output directory for the dump files.

ID: 3170

Severity: INFO
Message: The lower bound of the range of change numbers to dump.

ID: 3171

Severity: INFO
Message: The upper bound of the range of change numbers to dump.

ID: 3172

Severity: INFO
Message: The lower bound of the range of changes to dump.

ID: 3173

Severity: INFO
Message: The upper bound of the range of changes to dump.

ID: 3174

Severity: INFO
Message: {directory}.

ID: 3175

Severity: INFO
Message: {change number}.

ID: 3176

Severity: INFO
Message: {csn}.

ID: 3177

Severity: INFO
Message: The base-dn of the changes contained in the provided replica DB file.

ID: 3178

Severity: INFO
Message: {base dn}.

ID: 3179

Severity: ERROR
Message: An error occurred when reading the replication server configuration entry '%s': %s.

ID: 3180

Severity: ERROR
Message: The replication domain '%s' is not found. Make sure the domain is replicated.

ID: 3181

Severity: ERROR
Message: Replica DB folder '%s' not found.

ID: 3182

Severity: ERROR
Message: The provided change number '%s' is not valid.

ID: 3183

Severity: ERROR
Message: The provided CSN '%s' is not valid.

ID: 3184

Severity: ERROR
Message: The output folder '%s' cannot be created.

ID: 3185

Severity: ERROR
Message: The provided output folder '%s' doesn’t exist.

ID: 3186

Severity: ERROR
Message: The provided DN '%s' is invalid.

ID: 3187

Severity: ERROR
Message: Cannot list the changelog files in '%s': %s.

ID: 3188

Severity: ERROR
Message: Cannot convert the filename '%s' to a valid change number.

ID: 3189

Severity: ERROR
Message: Cannot convert the filename '%s' to a valid CSN.

ID: 3190

Severity: ERROR
Message: Error while decoding the changelog file '%s' : %s.

ID: 3191

Severity: INFO
Message: Arguments for changelogstat: %s.

ID: 3192

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: 3193

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: 3194

Severity: INFO
Message: OpenDJ will be upgraded from version %s to %s.

ID: 3195

Severity: INFO
Message: OpenDJ was successfully upgraded to version %s.

ID: 3196

Severity: INFO
Message: The upgrade will not be performed because the configuration file config.ldif could not be parsed.

ID: 3197

Severity: INFO
Message: Removing file '%s'.

ID: 3198

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: 3199

Severity: INFO
Message: Removing configuration entries for the monitor providers.

ID: 3200

Severity: INFO
Message: Setting global server ID.

ID: 3201

Severity: INFO
Message: Setting new global server ID.

ID: 3202

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: 3203

Severity: INFO
Message: Setting global group ID and removing per domain values.

ID: 3204

Severity: WARNING
Message: Unable to find configuration for truststore backend "%s", using default values instead.

ID: 3205

Severity: WARNING
Message: Unable to find configuration for crypto manager "%s", replication SSL configuration will use default values.

ID: 3206

Severity: INFO
Message: Update replication SSL configuration from crypto manager and trust store backend to dedicated key and trust manager provider configuration elements.

ID: 3207

Severity: INFO
Message: Update replication SSL configuration.

ID: 3208

Severity: INFO
Message: Move SSL protocols and cipher suites configuration from crypto manager into each configuration element where they are used.

ID: 3209

Severity: INFO
Message: Removing global server ID from replication domains and replication server.

ID: 3210

Severity: INFO
Message: Adding Mail Servers.

ID: 3211

Severity: INFO
Message: Consolidating bootstrap replication servers from replication server and replication domains into Multimaster Synchronization provider.

ID: 3212

Severity: INFO
Message: Migrating truststore backend configuration to crypto manager.

ID: 3213

Severity: INFO
Message: Migrating truststore backend configuration.

ID: 3214

Severity: INFO
Message: Migrating security settings.

ID: 3215

Severity: ERROR
Message: Crypto manager configuration entry not found.

ID: 3216

Severity: NOTICE
Message: script to manage OpenDJ as a service on UNIX.

ID: 3217

Severity: NOTICE
Message: encode a password with a storage scheme.

ID: 3218

Severity: NOTICE
Message: export directory data in LDIF.

ID: 3219

Severity: NOTICE
Message: import directory data from LDIF.

ID: 3220

Severity: NOTICE
Message: manage server administration tasks.

ID: 3221

Severity: NOTICE
Message: rebuild index after configuration change.

ID: 3222

Severity: NOTICE
Message: start OpenDJ server.

ID: 3223

Severity: NOTICE
Message: display basic OpenDJ server information.

ID: 3224

Severity: NOTICE
Message: stop OpenDJ server.

ID: 3225

Severity: NOTICE
Message: upgrade OpenDJ configuration and application data.

ID: 3226

Severity: NOTICE
Message: check index for consistency or errors.

ID: 3227

Severity: NOTICE
Message: register OpenDJ server as a Windows Service.

ID: 3228

Severity: NOTICE
Message: gather OpenDJ backend debugging information.

ID: 3229

Severity: NOTICE
Message: debug changelog and changenumber files.

ID: 3230

Severity: NOTICE
Message: <xinclude:include href="description-upgrade.xml" />.

ID: 3231

Severity: ERROR
Message: The server has not been configured. Please run the 'setup' command first.

ID: 3232

Severity: NOTICE
Message: <xinclude:include href="variablelist-backendstat-index-status.xml" />.

ID: 3233

Severity: ERROR
Message: Rebuild index aborted: an error has occurred while rebuilding indexes for base DN '%s'.

ID: 3234

Severity: NOTICE
Message: <xinclude:include href="description-recurring-task-info.xml" />.

ID: 3235

Severity: ERROR
Message: An error occurred while reading configuration file: %s.

ID: 3236

Severity: ERROR
Message: An error occurred while copying data from '%s' to the server instance configuration directory. Error details: %s.

ID: 3238

Severity: INFO
Message: Reconfiguring PDB backends to JE backends.

ID: 3239

Severity: INFO
Message: Renaming PDB backend directory '%s' to '%s'.

ID: 3240

Severity: INFO
Message: You must reimport all your data into the JE backends in order to have a fully functional server.

ID: 3241

Severity: INFO
Message: Replacing low durability settings in JE backends.

ID: 3242

Severity: INFO
Message: Replacing high durability settings in JE backends.

ID: 3243

Severity: INFO
Message: Replacing medium durability settings in JE backends.

ID: 3244

Severity: INFO
Message: Renaming 'ds-cfg-json-schema' object class to 'ds-cfg-json-query-equality-matching-rule'.

ID: 3245

Severity: ERROR
Message: An error occurred while copying OpenDMK jar file '%s' to '%s': %s.

ID: 3246

Severity: INFO
Message: Migrating root DN '%s'.

ID: 3247

Severity: INFO
Message: Removing root DN users from configuration.

ID: 3248

Severity: ERROR
Message: Root DNs could not be migrated because the '%s' directory could not be created: %s.

ID: 3249

Severity: INFO
Message: OpenDJ 6.0.0 drops support for Root DNs and replaces them with standard user entries which are stored in LDIF backends. The root DN '%s' does not have an alias name so it must be renamed to '%s' so that it no longer resides inside the configuration. Do you want to proceed with the upgrade?.

ID: 3250

Severity: INFO
Message: OpenDJ 6.0.0 drops support for Root DNs and replaces them with standard user entries which are stored in LDIF backends. The root DN '%s' has multiple alias names, but only the alias name '%s' will be kept. Do you want to proceed with the upgrade?.

ID: 3251

Severity: INFO
Message: The upgrade will not be performed because some Root DNs need to be migrated.

ID: 3252

Severity: INFO
Message: Segregating mutable and immutable files.

ID: 3253

Severity: ERROR
Message: The directory '%s' could not be created: %s.

ID: 3254

Severity: ERROR
Message: The file '%s' could not be renamed to '%s': %s.

ID: 3255

Severity: INFO
Message: Update admin-backend.ldif file location.

ID: 3256

Severity: INFO
Message: Update tasks backend file location.

ID: 3257

Severity: INFO
Message: Replacing all pin related configuration attributes by a single pin configuration attribute.

ID: 3258

Severity: INFO
Message: The attribute has been removed because it was referring to a file that does not exist. You should replace it with the attribute %s before being able to enable the object corresponding to the configuration entry.

ID: 3259

Severity: INFO
Message: Removing send and receive window size configuration in replication.

ID: 3260

Severity: ERROR
Message: tool exit with error return code '%d'.

ID: 3261

Severity: INFO
Message: Migrating replication changelog files to 6.5.0 format.

ID: 3262

Severity: ERROR
Message: An error occurred reading the changelog files: %s.

ID: 3263

Severity: ERROR
Message: An error occurred while renaming the changelog files: %s.

ID: 3264

Severity: ERROR
Message: An error occurred while migrating replicas' offline states to the changelog files: %s.

ID: 3265

Severity: INFO
Message: Adding configuration entry '%s'.

ID: 3266

Severity: ERROR
Message: An error occurred while trying to modify %s : %s.

ID: 3267

Severity: INFO
Message: Replacing compute change number setting in replication server.

ID: 3268

Severity: INFO
Message: OpenDJ 6.5.0 changed the indexing algorithm for replication metadata. Its index must be rebuilt which may take a long time; without a working index every server start will take longer than normal. Do you want to rebuild the index automatically at the end of the upgrade?.

ID: 3269

Severity: INFO
Message: Replacing "reject unauthenticated requests" policy in global configuration.

ID: 3270

Severity: INFO
Message: Removing configuration for assured replication.

ID: 3271

Severity: INFO
Message: Removing generation-id data from configuration.

ID: 3272

Severity: INFO
Message: Removing synchronization state data from configuration.

ID: 3273

Severity: INFO
Message: Renaming the proxy backend configuration property 'service discovery mechanism' to 'shard'.

ID: 3274

Severity: INFO
Message: Adding objectClass to JSON, CSV, and External access logger configurations.

ID: 3275

Severity: ERROR
Message: Server is running. Please stop the server before running this tool.

ID: 3276

Severity: INFO
Message: Removing configuration for replication monitoring.

ID: 3277

Severity: ERROR
Message: An error occurred when trying to read the configuration version in %s : %s.

ID: 3278

Severity: ERROR
Message: An error occurred when trying to read the upgrade version in second line of file '%s' (%s).

ID: 3279

Severity: INFO
Message: Set the proxy backend configuration property 'hash-function' to MD5.

ID: 3280

Severity: INFO
Message: Use the old JE backend caches instead of the new shared cache.

ID: 3281

Severity: INFO
Message: Renaming ds-cfg-connection-pool-* attributes to ds-cfg-bind-connection-pool-*.

ID: 3282

Severity: INFO
Message: Set the database cache mode 'ds-cfg-db-cache-mode' to 'cache-ln'.

ID: 3283

Severity: INFO
Message: Add 'inheritFromDNParent' attribute type to the 'inheritedCollectiveAttributeSubentry' object class.

ID: 3284

Severity: NOTICE
Message: Could not find 'ads-certificate' entry in the truststore '%s'. This entry is required for the correct behavior of the directory server.

ID: 3285

Severity: WARNING
Message: Unable to retrieve the keys from the truststore.

ID: 3286

Severity: WARNING
Message: Unable to look up server key id '%s' in the admin backend.

ID: 3287

Severity: WARNING
Message: Unable to retrieve the hostname from the admin backend using the truststore as source of keys; 'advertised-listen-address' attribute in global configuration will use the local hostname as value. Cause: %s.

ID: 3288

Severity: NOTICE
Message: Unable to retrieve the local hostname, the 'advertised-listen-address' attribute in global configuration must be set manually.

ID: 3289

Severity: INFO
Message: Adding 'listen-address' and 'advertised-listen-address' attributes to the global configuration.

ID: 3290

Severity: INFO
Message: Removing 'listen-address' attributes that are redundant with default value provided by the global configuration.

ID: 3291

Severity: NOTICE
Message: An error occurred while removing some 'listen-address' attribute values from the configuration.

ID: 3292

Severity: INFO
Message: OpenDJ 7.0.0 changed the indexing algorithm for TelephoneNumber equality and substring matching rules. All TelephoneNumber syntax based attribute indexes must be rebuilt which may take a long time. Do you want to rebuild the indexes automatically at the end of the upgrade?.

ID: 3293

Severity: INFO
Message: Add ACI to make Root DSE fullVendorVersion operational attribute user visible.

ID: 3294

Severity: INFO
Message: Removing references to 'backup' backend.

ID: 3295

Severity: INFO
Message: Removing 'backup' backend.

ID: 3296

Severity: INFO
Message: Remove External change log domain configuration entries and migrate information to the domains and the replication server configuration entries.

ID: 3297

Severity: INFO
Message: Migrate source address attribute from replication server configuration to replication synchronization provider configuration.

ID: 3298

Severity: INFO
Message: Migrate common replication domain configuration attributes to replication synchronization provider configuration.

ID: 3299

Severity: NOTICE
Message: Configuration attribute %s has distinct values in replication domain configuration entries. Migration to replication synchronization provider configuration will keep only one value, '%s' and discarded '%s'.

ID: 3300

Severity: INFO
Message: Removing '%s' attribute(s) from backup tasks.

ID: 3301

Severity: ERROR
Message: '%s' attributes couldn’t be removed from backup tasks.

ID: 3302

Severity: INFO
Message: Removing restore tasks.

ID: 3303

Severity: ERROR
Message: The restore tasks couldn’t be removed.

ID: 3304

Severity: INFO
Message: DS 7.0.0 breaks restore tasks compatibility, all existing restore tasks will be removed.

ID: 3305

Severity: INFO
Message: Migrate isolation policy attribute to replication synchronization provider configuration.

ID: 3306

Severity: INFO
Message: Merge replication-purge-delay and conflicts-historical-purge-delay attribute into a single replication-purge-delay attribute in replication synchronizer provider configuration.

ID: 3307

Severity: ERROR
Message: An error occured when trying to read the configuration file %s: %s.

ID: 3308

Severity: WARNING
Message: Unable to find the task backend file location in the configuration file %s, entry '%s' or attribute '%s' is missing.

ID: 3309

Severity: INFO
Message: Removing profiler plugins.

ID: 3310

Severity: INFO
Message: Removing max-work-queue-capacity.

ID: 3311

Severity: INFO
Message: Migrating encrypted replication changelog files.

ID: 3312

Severity: ERROR
Message: An error occurred while migrating encrypted changelog files: %s.

ID: 3313

Severity: INFO
Message: Rename attribute 'ds-backup-directory-path' to 'ds-backup-location' in entries of objectClass '%s'.

ID: 3314

Severity: ERROR
Message: Attribute 'ds-backup-directory-path' could not be renamed to 'ds-backup-location' in entries of objectclass '%s'.

ID: 3315

Severity: INFO
Message: Renaming the 'use-mutual-tls' configuration property to 'use-sasl-external'.

ID: 3316

Severity: INFO
Message: Renaming the 'replication-server' configuration property to 'bootstrap-replication-server'.

ID: 3317

Severity: INFO
Message: The path to the systemd service file to create.

ID: 3318

Severity: ERROR
Message: An error occurred while attempting to generate the systemd service file: %s.

ID: 3319

Severity: INFO
Message: Gives an ID to the task.

ID: 3320

Severity: INFO
Message: {description}.

ID: 3321

Severity: INFO
Message: Gives a description to the task.

ID: 3322

Severity: INFO
Message: Description.

ID: 3323

Severity: INFO
Message: Allowing dsbackup purge tasks.

ID: 3324

Severity: INFO
Message: Removing empty configuration for groups.

ID: 3325

Severity: INFO
Message: Replacing non allowed comma in groupID definition with a dot.

ID: 3326

Severity: INFO
Message: Removing 'ds-cfg-load-balancing-algorithm' attribute from the proxy backend.

ID: 3327

Severity: INFO
Message: Renaming 'ds-cfg-bind-connection-pool-' attributes to 'ds-cfg-connection-pool-'.

ID: 3328

Severity: INFO
Message: Removing 'ds-cfg-request-connection-pool-size' attribute,'ds-cfg-connection-pool-size' will be used instead.

ID: 3329

Severity: INFO
Message: Add 'ldapSyntaxes' to attribute list which can be read in the schema access global access control policies.

ID: 3330

Severity: INFO
Message: Renaming replication changelog files from .log to .cdb suffix.

ID: 3331

Severity: ERROR
Message: Invalid sequence for task version: task %s tried to register for version %s but the previous task used a higher version: %s. This suggests there is a copy/paste error in the version number.

ID: 3332

Severity: INFO
Message: Renaming proxy 'ds-cfg-heartbeat-' **attributes to 'ds-cfg-keep-alive-**`.

ID: 3333

Severity: ERROR
Message: "Error parsing existing schema file '%s' - %s".

ID: 3334

Severity: INFO
Message: OpenDJ 7.2.0 fixed an indexing bug that may cause searches to return duplicate entries in certain circumstances. All indexes have to be rebuilt. This could take a long time to proceed. Do you want to launch this process automatically at the end of the upgrade?.

ID: 3335

Severity: INFO
Message: Removing look through limit from the configuration.

ID: 3336

Severity: INFO
Message: Renaming 'ds-cfg-cursor-entry-limit' to 'ds-cfg-max-candidate-set-size'.

ID: 3337

Severity: INFO
Message: Removing VLV indexes with base-object scope from the configuration.

ID: 3338

Severity: INFO
Message: Migrating JE environment properties to configuration attributes.

ID: 3339

Severity: INFO
Message: Removing replica degraded status threshold configuration.

ID: 3340

Severity: INFO
Message: Remove LDIF change record remnants from the configuration.

ID: 3341

Severity: INFO
Message: Changes in Unicode string normalization require that all indexes are rebuilt. Do you want to rebuild the indexes automatically at the end of the upgrade?.

ID: 3342

Severity: INFO
Message: Removing LDAP send reject notification configuration.

ID: 3343

Severity: INFO
Message: Removing debug logging configuration.

ID: 3344

Severity: INFO
Message: Replace multiple severity values by a single one in logging configurations.

ID: 3345

Severity: INFO
Message: Changing syntax for 'inheritAttribute' attribute type to IA5String.

ID: 3346

Severity: INFO
Message: OPENDJ-9550 on 7.3.0 caused static groups to lose their operational attributes on modify operations, corrupting data and the entryUUID index which needs to be rebuilt. Do you want to rebuild the indexes automatically at the end of the upgrade?.

ID: 3347

Severity: INFO
Message: The name of the group account under which the server should run.

ID: 3348

Severity: INFO
Message: {groupName}.

ID: 3349

Severity: ERROR
Message: You must provide either the --%s or the --%s argument.

ID: 3350

Severity: INFO
Message: Make the File-Based Error Loggers perform asynchronous logging for optimal performance after merging the debug and error logger.

ID: 3351

Severity: INFO
Message: Enabling change numbers in the replication changelog.

ID: 3352

Severity: INFO
Message: Removing 'ds-cfg-argon2-migration-memory' and adding 'ds-cfg-argon2-memory-pool-size' which specifies the size of the memory pool allocated for concurrent Argon2 password hashing computations.

ID: 3353

Severity: INFO
Message: Adding 'ds-cfg-exclude-values-of-attributes' with the list the attribute names which values will be excluded from the list of requested modifications.

ID: 3354

Severity: INFO
Message: Continue to allow invalid certificate lists, certificate pairs and postal addresses.

ID: 3355

Severity: INFO
Message: Renaming 'ds-cfg-big-index-matching-rule' attribute to 'ds-cfg-big-index-extensible-matching-rule`.

ID: 3356

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: 3357

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: 3358

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: 3359

Severity: ERROR
Message: Unable to parse LDIF entry starting at line %d because the line "%s" does not include an attribute name.

ID: 3360

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: 3361

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: 3362

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: 3363

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: 3364

Severity: WARNING
Message: Entry %s read from LDIF starting at line %d includes a duplicate attribute %s with value %s. The second occurrence of that attribute value has been skipped.

ID: 3365

Severity: ERROR
Message: Entry %s starting at line %d includes multiple values for single-valued attribute %s.

ID: 3366

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: 3367

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: 3368

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: 3369

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: 3370

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: 3371

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: 3372

Severity: ERROR
Message: The provided sender address %s is invalid: %s.

ID: 3373

Severity: ERROR
Message: The provided recipient address %s is invalid: %s.

ID: 3374

Severity: ERROR
Message: The specified e-mail message could not be sent using any of the configured mail servers.

ID: 3375

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: 3376

Severity: ERROR
Message: The provided string "%s" cannot be decoded as an LDAP URL because it does not contain a protocol scheme.

ID: 3377

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: 3378

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: 3379

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: 3380

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: 3381

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: 3382

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: 3383

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: 3384

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: 3385

Severity: ERROR
Message: The named character set is invalid because it does not contain a name.

ID: 3386

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: 3387

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: 3388

Severity: ERROR
Message: Cannot decode value "%s" as a named character set because there are no characters to include in the set.

ID: 3389

Severity: INFO
Message: %d seconds.

ID: 3390

Severity: INFO
Message: %d minutes, %d seconds.

ID: 3391

Severity: INFO
Message: %d hours, %d minutes, %d seconds.

ID: 3392

Severity: INFO
Message: %d days, %d hours, %d minutes, %d seconds.

ID: 3393

Severity: ERROR
Message: Unable to set permissions for file %s because it does not exist.

ID: 3394

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: 3395

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: 3396

Severity: WARNING
Message: Entry %s read from LDIF starting at line %d includes value "%s" for attribute %s that is invalid according to the associated syntax: %s.

ID: 3397

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: 3398

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: 3399

Severity: ERROR
Message: The embedded server with server root '%s' cannot be started because it is already running.

ID: 3400

Severity: ERROR
Message: Skipping entry %s because the DN is excluded by the exclude branch "%s".

ID: 3401

Severity: ERROR
Message: Skipping entry %s because the DN is excluded by the exclude filter "%s".

ID: 3402

Severity: ERROR
Message: Skipping entry %s because the DN is not included by any include branches.

ID: 3403

Severity: ERROR
Message: Skipping entry %s because the DN is not included by any include filters.

ID: 3404

Severity: ERROR
Message: Rejecting entry %s because it was rejected by a plugin.

ID: 3405

Severity: ERROR
Message: Rejecting entry %s because it was rejected by a plugin: %s.

ID: 3406

Severity: ERROR
Message: Unable to parse LDIF entry %s starting at line %d because it has an invalid binary option for attribute %s.

ID: 3407

Severity: ERROR
Message: Skipping entry %s because the following error was received when reading its attributes: %s.

ID: 3408

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: 3409

Severity: ERROR
Message: An error occurred while attempting to extract server archive '%s' before setup of embedded server with server root '%s': %s.

ID: 3410

Severity: ERROR
Message: An error occurred while attempting to rebuild index of embedded server with server root '%s': %s.

ID: 3411

Severity: ERROR
Message: An error occurred while attempting to start the embedded server with server root '%s' : %s.

ID: 3412

Severity: ERROR
Message: An error occurred while attempting to upgrade the embedded server with server root '%s' : %s.

ID: 3413

Severity: ERROR
Message: An error occurred while attempting to retrieve an internal connection to the server with the user DN '%s'.

ID: 3414

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: 3415

Severity: ERROR
Message: An error occurred while attempting to initialize the configuration framework or to read the configuration file '%s'.

ID: 3416

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: 3417

Severity: ERROR
Message: An error occurred while attempting to import LDIF file '%s' into embedded server with server root '%s': '%s'.

ID: 3418

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: 3419

Severity: ERROR
Message: An error occurred while attempting to retrieve the configuration version of the directory server: '%s'.

ID: 3420

Severity: ERROR
Message: An error occurred while attempting to retrieve the data version of the directory server: '%s'.

ID: 3421

Severity: ERROR
Message: An error occurred while initializing configuration of embedded server with server root '%s': %s.

ID: 3422

Severity: ERROR
Message: The directory to move %s does not exist.

ID: 3423

Severity: ERROR
Message: The directory to move %s exists but is a file.

ID: 3424

Severity: ERROR
Message: The target directory %s already exists.

ID: 3425

Severity: ERROR
Message: Configuration error: an LDAP port or an LDAPS port must be configured before finishing configuring the embedded server.

ID: 3426

Severity: WARNING
Message: Embedded Directory Servers are NOT SUPPORTED in production.

ID: 3427

Severity: ERROR
Message: The Directory server DS(%s) and Replication server RS(%s) are working on different dataset regarding the domain '%s' : the Directory server DS(%s) will reject all modification operations and must be re-initialized.

ID: 3428

Severity: ERROR
Message: You do not have sufficient privileges to perform the sort request.

ID: 3430

Severity: INFO
Message: Renaming 'ds-cfg-log-control-oids' attribute to 'ds-cfg-log-controls'.

ID: 3431

Severity: INFO
Message: Removing SNMP support.

ID: 3432

Severity: INFO
Message: 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. ID: 3433:: Severity: INFO
Message: 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. ID: 3434:: Severity: INFO
Message: 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. ID: 3435:: Severity: NOTICE
Message: A mandatory argument is missing. Choose one and only one argument from '--generate-recovery-id', '--generated-id' and '--user-generated-id'. ID: 3436:: Severity: INFO
Message: {generatedRecoveryId}. ID: 3437:: Severity: ERROR
Message: Disaster recovery must be started on a directory server or a combined directory/replication server. ID: 3438:: Severity:

ERROR
Message: An error occurred while computing the disaster recovery id for base DN '%s': %s. ID: 3439:: Severity: INFO
Message: Disaster recovery for base DN '%s' ended successfully on this server. ID: 3440:: Severity: INFO
Message: The disaster recovery process completed successfully for the first server. The next step is to restore the same data using the same procedure on all the remaining servers in the topology and run the following command on each server:. ID: 3441:: Severity: ERROR
Message: The disaster recovery procedure cannot be continued on this server because the domain '%s' does not contain the expected data. Make sure the server has been restored using exactly the same procedure used on the first server. ID: 3442:: Severity: ERROR
Message: An error occurred while rewriting the base entry of the domain '%s'. This server will not be able to connect to the servers which already have received the disaster recovery procedure. Make sure the filesystem is writeable and there is some free disk space, and try again. The error is: %s. ID: 3443:: Severity: ERROR
Message: An error occurred while clearing the changelog for the domain '%s'. This server will not be able to connect to the servers which have already completed the disaster recovery procedure. Make sure the filesystem at '%s' is readable with delete permissions and try again. The error is: %s. ID: 3444:: Severity: ERROR
Message: An error occurred while rewriting replication data for the domain '%s'. This server will not be able to connect to the servers which already have received the disaster recovery procedure. Make sure the filesystem at %s is writeable and there is some free disk space and try again. The error is: %s. ID: 3445:: Severity: ERROR
Message: An error occurred while clearing the change number index. The change number index on this server will expose erroneous data. Make sure the filesystem at '%s' is readable with delete permissions and try again. The error is: %s. ID: 3446:: Severity: ERROR
Message: The server is neither a replica nor a replication server for base DN '%s'. ID: 3447:: Severity: ERROR
Message: Disaster recovery for base DN '%s' failed on this server. ID: 3448:: Severity: WARNING
Message: The local replication server never received changes for base DN '%s'. Make sure you are running the disaster recovery on the right server. ID: 3449:: Severity: INFO
Message: Base DN of the domain to be recovered. ID: 3450:: Severity: ERROR
Message: The disaster recovery id '%s' is invalid. Verify the value of the disaster recovery id parameter matches the value printed when '--generate-recovery-id' was run on the first server. ID: 3451:: Severity: ERROR
Message: The provided base DN '%s' does not match the disaster recovery id '%s'. Please verify that the base DN and disaster recovery id correspond to the base DN and id from the first recovered server. ID: 3452:: Severity: INFO
Message: Starting disaster recovery on local server for base DN '%s'. ID: 3453:: Severity: ERROR
Message: The replication changelog file '%s' could not be deleted. Check file permissions and filesystem status. The error is: %s. ID: 3454:: Severity: ERROR
Message: The replication changelog files could not be retrieved. Check file permissions and filesystem status. The error is: %s. ID: 3455:: Severity: INFO
Message: Disaster recovery will erase replication metadata on this server. This server will then only be able to replicate changes with other recovered servers. You will have to run the recovery procedure on every other server of the topology. Servers which have not been recovered will not be able to connect to recovered servers anymore. ID: 3456:: Severity: INFO
Message: Continue the disaster recovery procedure. ID: 3457:: Severity: INFO
Message: dsrepl disaster-recovery --baseDn '%s' --generated-id %s. ID: 3458:: Severity: INFO
Message: Disaster recovery id: %s. ID: 3459:: Severity: INFO
Message: The disaster recovery process has been canceled, no operation was performed. This server may still not be operational. ID: 3460:: Severity: INFO
Message: Decodes one or more CSNs and displays them in a human readable JSON format. ID: 3461:: Severity: INFO
Message: csn [csn ...]. ID: 3462:: Severity: ERROR
Message: The CSN cannot be parsed as a valid CSN. ID: 3463:: Severity: NOTICE
Message: The current 7.4.0 server is configured to use an incompatible confidentiality setting which prevents automated upgrade from working. Follow the procedure detailed in the Upgrade Guide to upgrade this server. ID: 3464:: Severity: WARNING
Message: Peer replica '%s' is too late compared to changelog '%s' for domain '%s'. It asked for changes that are not present in

the changelog or have been purged. The peer replica will no longer receive replicated changes and must be re-initialized. Diagnostic information follows: %s. ID: 3465:: Severity: WARNING
Message: Peer replica asked for changes from replica '%1$s' starting from %2$s (CSN '%3$s'), but changelog DB only contains changes starting from %4$s (CSN '%5$s'). The last recorded purge information in 'domains.state' is: last update CSN: %6$s, last message CSN: %7$s. The replica DB description is: number of files: %8$d, replica offline: %9$b. The replica DB newest file content is: oldest CSN: %10$s, newest CSN: %11$s, number of updates: %12$d, number of ReplicaOfflineMsg: %13$d. ID: 3466:: Severity: WARNING
Message: Peer replica asked for changes from replica '%s' starting from %s (CSN '%s'), but the changelog DB has purged this change, and the last known generated change timestamp is %s (CSN '%s'). The last recorded purge information in 'domains.state' is: last update CSN: %s, last message CSN: %s. ID: 3467:: Severity: WARNING
Message: Peer replica asked for changes from replica '%1$s' starting from %2$s (CSN '%3$s'), but changelog DB only contains changes starting from %4$s (CSN '%5$s'). The administrator explicitly decided to ignore the recorded purge information in 'domains.state' by setting the 'org.forgerock.opendj.allowReplicaPastPurgeDelay' system property to 'true'. The replica DB description is: number of files: %6$d, replica offline: %7$b. The replica DB newest file content is: oldest CSN: %8$s, newest CSN: %9$s, number of updates: %10$d, number of ReplicaOfflineMsg: %11$d. ID: 3468:: Severity: ERROR
Message: The provided disaster recovery id had version %d, but only version 1 is supported. Verify the value of the disaster recovery id parameter matches the value printed when '--generate-recovery-id' was run on the first server. ID: 3469:: Severity: WARNING
Message: Replication domain '%s' status '%s' is not healthy. ID: 3470:: Severity: INFO
Message: Updating existing Prometheus endpoints to keep using the legacy format. ID: 3471:: Severity: WARNING
Message: Property '%s' in %s '%s' is DEPRECATED for removal since %s. Its usage is highly discouraged. ID: 3472:: Severity: WARNING
Message: Property '%s' in %s '%s' is LEGACY since %s. Its usage is highly discouraged. ID: 3473:: Severity: ERROR
Message: There is no data in domain '%s' to run disaster recovery on. ID: 3474:: Severity: INFO
Message: 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. ID: 3475:: Severity: INFO
Message: {userGeneratedRecoveryId}. ID: 3476:: Severity: ERROR
Message: Base Dn '%s' is not a replicated domain on this server. Disaster recovery must be run on a directory server or a combined directory/replication server. ID: 3477:: Severity: INFO
Message: Disaster recovery for domain '%s' has already been run with recovery ID '%s'. Verify all servers in the topology are being recovered with the same recovery ID. ID: 3478:: Severity: WARNING
Message: An error occurred while decoding a replicated request control: %s. ID: 3479:: Severity: WARNING
Message: An error occurred while decoding a internal modification control: %s. ID: 3480:: Severity: INFO
Message: Add a new HDAP authorization mechanism '%s'. ID: 3481:: Severity: INFO
Message: Replace the authentication mechanism of the '/hdap' endpoint '%s' with the new HDAP authorization mechanism '%s'. ID: 3482:: Severity: INFO
Message: Removing deprecated HTTP Basic DN and JWT authorization mechanisms. The HDAP authorization mechanism '%s' replaces them. ID: 3483:: Severity: ERROR
Message: The entry '%s' specified in the request does not exist in the Directory Server. ID: 3484:: Severity: ERROR
Message: The requested search operation included the persistent search control together with either the simple paged results control or the virtual list view control. These controls are mutually exclusive and cannot be used together. ID: 3485:: Severity: WARNING
Message: EXPERIMENTAL virtual threads support has been enabled. DO NOT USE in production. ID: 3486:: Severity: NOTICE
Message: Cannot continue the restore process, errors were encountered while reading the list of available backups: %s. ID: 3487:: Severity: NOTICE
Message: Cannot continue purging backups, errors were encountered while reading the list of available backups: %s. ID: 3488::

Severity: ERROR
Message: This server received a replication message to signal a disaster recovery. This means that this server is part of a mixed topology where the deprecated start-disaster-recovery command is used. The operation will continue, but it is not supported anymore and the disaster recovery procedure for mixed topologies must be used instead. ID: 3489:: Severity: INFO
Message: Hostname.

IDs: 3501-4000

ID: 3516

Severity: ERROR
Message: A record from replica '%s' in the replication changelog of domain '%s' cannot be decoded because its format (%x) is invalid.

ID: 3520

Severity: INFO
Message: replica '%s' in domain '%s'.

About this reference

This reference covers PingDS tools, which are bundled with the software. For the `dsconfig` command, also see the Configuration reference.

- addrate
- authrate
- backendstat
- base64
- changelogstat
- create-rc-script
- dsbackup
- dsconfig
- dskeymgr
- dsrepl
- encode-password
- export-ldif
- import-ldif
- ldapcompare
- ldapdelete
- ldapmodify
- ldappasswordmodify
- ldapsearch
- ldifdiff
- ldifmodify
- ldifsearch
- makeldif-template
- makeldif
- manage-account
- manage-tasks
- modrate
- rebuild-index
- searchrate

- setup-profile
- setup
- start-ds
- status
- stop-ds
- supportextract
- upgrade
- verify-index
- windows-service

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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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 or systemd service 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:

`-g | --groupName {groupName}`

The name of the group account under which the server should run.

`-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.

`-r | --rcScript {path}`

The path to the RC script to create.

`-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

`--providerArg {argument}`

Configuration argument for the PKCS#11 provider.

`--providerClass {class}`

Full class name of the PKCS#11 provider.

`--providerName {name}`

Name of the PKCS#11 provider.

`--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'.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

--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

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

--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)

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.

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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

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 decode-csn

```
dsrepl decode-csn csn [csn …]
```

Decodes one or more CSNs and displays them in a human readable JSON format.

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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

--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

`--providerArg {argument}`

Configuration argument for the PKCS#11 provider.

`--providerClass {class}`

Full class name of the PKCS#11 provider.

`--providerName {name}`

Name of the PKCS#11 provider.

--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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

--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.

The status of a server is one of the following.

- BAD - DATA MISMATCH: either the fractional replication configuration does not match the backend data, or the initial state of the replicated data does not match other servers and this server must be re-initialized;
- BAD - TOO LATE: the server has fallen further behind than the replication purge delay and must be re-initialized;
- GOOD: normal operation, nothing to do;
- SLOW: the server's replay delay is greater than five seconds;
- UNHEALTHY: read the server health errors in the server monitoring data for details.

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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

--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 write. All paths are relative to the server's installation directory, which can be remote.

`-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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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 import. All paths are relative to the server's installation directory, which can be remote.

`-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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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, 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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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, 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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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, 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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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, 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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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, 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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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.

Specifying attribute values

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:

```
description: Description for [org] under [suffix]
```

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:

```
givenName: <first>
sn: <last>
```

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:

```
mail: {givenName}.{sn}@[myDomain]
initials: {givenName:1}{sn:1}
```

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 (`> ] }`):

```
scimMail: \{"emails": \[\{"value": "{mail}", "type": "work", "primary": true}]}
xml: \<id>{uid}\</id>
```

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.

`--providerArg {argument}`

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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

--mvcc {mvcc}

Attribute name used in the assertion filter for multi-version concurrency control. For "read_modify" strategy only, ignored otherwise. Default: eTag

-S | --scriptFriendly

Use script-friendly mode. Default: false

--strategy {strategy}

The strategy for modifying entries, which must be one of "modify" or "read_modify". Default: MODIFY

-t | --numConcurrentRequests {numConcurrentRequests}

Number of concurrent requests per connection. Default: 1

--valueCount {valueCount}

Specifies the number of values an attribute should contain. 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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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. 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

Field	Allowed Values
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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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

--keyStorePasswordFilePath {path}

Path of the file containing the keystore password. The specified path will be used as the configuration value in the new server.

--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.

--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.

`--providerArg {argument}`

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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.

--providerArg {argument}

Configuration argument for the PKCS#11 provider.

--providerClass {class}

Full class name of the PKCS#11 provider.

--providerName {name}

Name of the PKCS#11 provider.

-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.

> **Note**
>
> On Linux systems, this tool uses the `top` command to capture details about individual threads, such as CPU use. Per thread CPU use (`-H`) is important in diagnosing performance issues, hangs, and high CPU problems. The output makes it possible to identify the offending threads in the `jstack` or `jcmd` output. The global `top -p` option only shows the overall CPU use. It is not sufficient to identify issues with particular threads.
> On some Linux distributions, the `top` command does not support the `-H` option. This makes it more difficult to diagnose performance issues, hangs, and high CPU problems.

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-java17/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/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.