Upgrade

This guide shows you how to upgrade Directory Services software.

About Upgrades

Read this first.

Directory Server

Upgrade a directory server.

Directory Proxy

Upgrade a directory proxy server.

Replication Server

Upgrade a standalone replication server.

Read the release notes before you upgrade DS software.

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

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

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, make sure you understand the key differences beforehand. With these in mind, plan the upgrade, how you will test the upgraded version, and how you will recover if the upgrade process does not go as expected:

Fully Compatible Replication

Some things never change. The replication protocol remains fully compatible with earlier versions back to OpenDJ 3.

This means you can still upgrade servers while the directory service is online, but the process has changed.

Key configuration differences

DS 6.5 and earlier	DS 7.0 and later
You configure replication after installation.	You configure replication during installation, before starting the server.
You configure which servers replicate.	You configure bootstrap replication servers. Replicas discover other servers through them.
You configure trust and TLS when configuring replication.	By default, you install servers with a shared deployment key 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 key.

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.

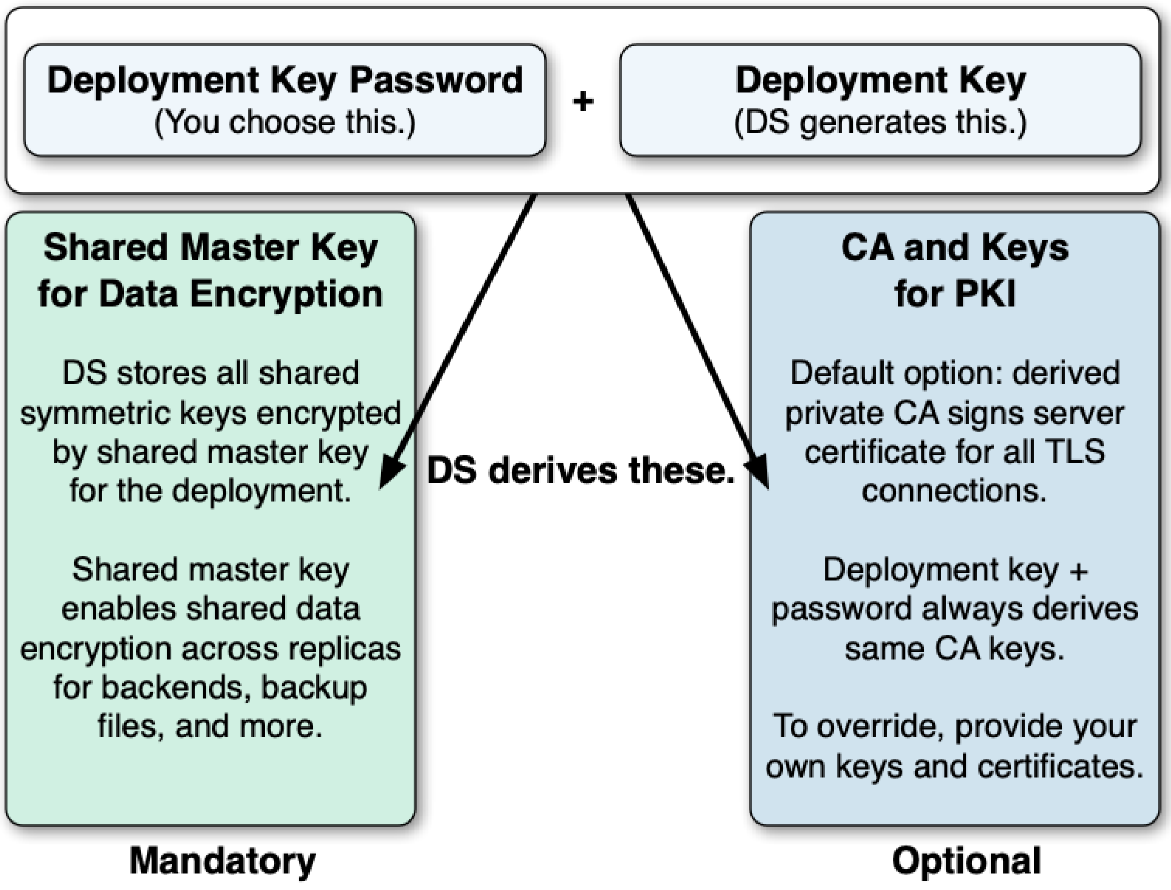
Deployment Key Required

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 key is a random string generated by DS software. A deployment key password is a secret string at least 8 characters long that you choose. The two are a pair. You must have a deployment key's password to use the key.

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 key and password. Even if you provide your own asymmetric keys for securing connections, you must use the deployment key and password to generate the shared master key.


Deployment Key Password
(You choose this.)
+
Deployment Key
(DS generates this.)
Shared Master Key for Data Encryption
DS stores all shared symmetric keys encrypted by shared master key for the deployment.
Shared master key enables shared data encryption across replicas for backends, backup files, and more.
DS derives these.
CA and Keys for PKI
Default option: derived private CA signs server certificate for all TLS connections.
Deployment key + password always derives same CA keys.
To override, provide your own keys and certificates.
Mandatory
Optional

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, see the rest of the DS 7 documentation.

IMPORTANT

To the extent possible, separate the upgrade process from the process of adopting new features. The DS **`upgrade`** command encourages this by maintaining compatibility where possible.

Once you have validated that the upgrade has completed successfully, take advantage of the new features available. Be sure to review the suggestions in After You Upgrade.

Supported Upgrades

From...	To...	Important Notes
Official ForgeRock release, version 3.0 or later	Official ForgeRock release, same edition of directory server or replication server	Supported.

From...	To...	Important Notes
Official ForgeRock release, OEM edition, version 3.0 or later	Official ForgeRock release, directory server or replication server	Supported. The OEM edition did not include Berkeley DB Java Edition, and did not support JE backends. Instead, the OEM edition uses PDB backends for local data. This release removes support for PDB backend databases. The upgrade process only converts PDB backend configuration entries to JE backend configuration entries. It renames the PDB backend database directories, appending a `.bak` suffix, but does not change the format of the databases. *The PDB backend database content is no longer accessible after upgrade.* Backup files for PDB backend databases are also no longer usable after upgrade. You must export data from any PDB backend databases to LDIF before upgrading, and then import the data into the new JE backend databases after upgrade. For instructions on exporting and importing LDIF, see Import and Export. After upgrading, configure backup tasks for the new JE backend databases as you had done previously for PDB backend databases.
Official ForgeRock release, version 2.6	Official ForgeRock release, directory server or replication server	Not supported. Workaround: First, upgrade all servers in the deployment to 6.5 before upgrading further. For details on upgrading to 6.5, see the DS 6.5 Installation Guide.

From...	To...	Important Notes
Official ForgeRock release, version 2.4 or 2.5	Official ForgeRock release, directory server or replication server	Not supported. Workaround: Upgrade all servers in the deployment to use at least 2.6.0 before upgrading further. For details on upgrading to that version, see Upgrading to OpenDJ 2.6.0.
Evaluation release, version 5.0 or later	Official ForgeRock release	Not supported. The evaluation version includes an additional server plugin and configuration. Official releases do not have an upgrade task to remove the plugin and its configuration.
Unofficial build, version 2.6.0 or later	Official ForgeRock release	Not supported.

Upgrade Strategies

When you upgrade to a new DS version, you choose between in-place upgrade, unpacking the new software over old, then running the `upgrade` command, or upgrade by adding new servers and retiring old ones.

DS software provides an **`upgrade`** command to simplify the process of upgrading a server in place.

IMPORTANT

For some scenarios, like upgrading Docker images in a Kubernetes deployment, in-place upgrade is the only kind that works.

Upgrade in Place

The most straightforward option when upgrading DS servers is to upgrade in place. One by one, you stop, upgrade, and restart each server individually, leaving the service running during upgrade:

Advantages	Disadvantages
No additional systems to manage.	During upgrade, the host system must meet the requirements for both the older version and the new release. For example, you might need to have more than one Java environment installed. The operating system must also be supported for both releases.
Simpler to understand.	Slower to roll back. Rollback involves restoring each server to its pre-upgrade state. Once a replica's databases have been upgraded, they cannot be rolled back.
Easier to maintain compatibility. To the extent possible, the **`upgrade`** command leaves the configuration as is.	You must manually enable new features after upgrade.

On Upgrading Replicas

IMPORTANT

The in-place upgrade process is designed to support a rolling (sequential) upgrade of replicated servers.

Do not upgrade all replicated servers at once in parallel, as this removes all replication changelog data simultaneously, breaking replication.

When upgrading in place, follow these steps for each replica:

1. Direct client application traffic away from the server to upgrade.
2. Upgrade the replica.
3. Direct client application traffic back to the upgraded server.

Add New Servers

Adding new servers and then retiring old ones is an alternative to upgrading in place. You replicate data between old and new systems, leaving the service running during upgrade:

Advantages	Disadvantages
Smoothly phase out old host systems. After successfully completing the upgrade, you gradually retire the old systems.	New host systems to manage.
Faster to roll back. Old servers remain in operation until upgrade completes successfully.	Harder to maintain compatibility. You must manually configure new servers to be fully compatible with existing servers, rather than relying on the **`upgrade`** command. This requires an in-depth understanding of both your existing configuration and the new configuration. Some new default settings may be incompatible with the old default settings, for example.
	Requires initializing the new replicas. Depending on the volume of data to synchronize, you can initialize at least the first new replica online. For deployments with medium to large data sets, initialize from exported LDIF, or from backup files created using an upgraded DS server. In either case, you must plan the operation.
	While the upgrade is in progress, replication monitoring is split between the older servers, which use `dsreplication status`, and the newer servers, which use `dsrepl status`. Run both commands to get a more complete picture of replication status.
	You must manually enable new features after upgrade.

Before You Upgrade

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

Supported Java

IMPORTANT

- Always use a JVM with the latest security fixes.
- Make sure you have a required Java environment installed on the system.

  If your default Java environment is not appropriate, use one of the following solutions:

  - Edit the `default.java-home` setting in the `opendj/config/java.properties` file.
  - Set `OPENDJ_JAVA_HOME` to the path to the correct Java environment.
  - Set `OPENDJ_JAVA_BIN` to the absolute path of the **`java`** command.
- When running the **`dskeymgr`** and **`setup`** commands, use the same Java environment everywhere in the deployment.

  Due to a change in Java APIs, the same DS deployment ID generates different CA key pairs with Java 11 and Java 17.

  Using different Java versions is a problem if you use deployment ID-based CA certificates. Replication breaks, for example, when you use the **`setup`** command for a new server with a more recent version of Java than was used to set up existing servers.

  For details on resolving the issue, refer to Incompatible Java versions.

DS software supports the following Java environments:

Supported Java Versions

Vendor	Versions

Vendor	Versions
OpenJDK, including OpenJDK-based distributions: • AdoptOpenJDK/Eclipse Temurin Java Development Kit (Adoptium) • Amazon Corretto • Azul Zulu • Red Hat OpenJDK ForgeRock tests most extensively with AdoptOpenJDK/Eclipse Temurin. ForgeRock recommends using the HotSpot JVM.	11[1], 17[2]
Oracle Java	11[1], 17[2]

[1] DS requires Java 11.0.6 or later. Earlier Java 11 updates lack required cryptography fixes. To use TLS 1.3 with PKCS#11, DS requires Java 11.0.8 or later. Use Java 11.0.12 or later for compatibility with third-party cryptographic tools.

[2] DS requires Java 17.0.3 or later. Earlier Java 17 updates lack required cryptography fixes.

Required Credentials

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

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

Back Up First

Before upgrading, perform a full file system backup of the current server so that you can revert on failure. Make sure you stop the directory server and *back up the file system directory where the current server is installed*.

Backup archives are *not guaranteed to be compatible* across major and minor server releases. *Restore backups only on directory servers of the same major or minor version.*

Disable Windows Service

If you are upgrading a server registered as a Windows service, disable the Windows service before upgrade:

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

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

When Adding New Servers

When upgrading by adding new servers, add the new directory servers or replication servers to the existing deployment, as described here.

IMPORTANT

- Set up replication before upgrade. Do not set up replication for the first time between servers of different versions.
- The new server you add must first connect to an existing replica that is a directory server, not a standalone replication server.
- Newer directory servers update LDAP schema definitions to add support for new features. The newer schema definitions are not all compatible with older servers.

1. Install and set up a new server, but do not start it, yet.

   Because replication is now configured at setup time, you may need to create the new server with some specific arguments. The following table indicates which arguments are needed for which kind of server:

New server is a…	Use this replication setup option
Combined DS/RS	`--replicationPort` *port*
Standalone DS	N/A
Standalone RS	`--replicationPort` *port*

   - Do not use the `setup --bootstrapReplicationServer` option. In a later step of this procedure, you will use the **`dsrepl add-local-server-to-pre-7-0-topology`** command. That command configures the bootstrap replication server settings for the new server based on the existing deployment.
   - Do not use the `setup --start` option. In a later step of this procedure, you will start the server.

For details about setup options, refer to Setup Hints, and many of the examples that use the **`setup`** command.

2. Configure the new server settings to be compatible with the settings of the existing servers.

   Examples of incompatible default settings include:

   - Password storage schemes not available in earlier versions.
   - String-based server IDs. Server IDs were limited to numbers between 1 and 65535.

     Remove leading `0` (zero) characters when setting a numeric server ID. DS servers classify a server ID with a leading `0` as a string, not a number.
   - String-based group IDs. Group IDs were also limited to numbers.
   - TLS protocols and cipher suites.

   For changes in the release, refer to Incompatible changes. If the existing servers run a release older than 6.5, refer to similar pages in the previous release notes.

3. Configure the new server as a replica of an existing server that is a directory server, and not a standalone replication server:

```
$ dsrepl \
 add-local-server-to-pre-7-0-topology \
 --hostname pre-7-ds.example.com \
 --port 4444 \
 --bindDn "cn=admin,cn=Administrators,cn=admin data" \
 --bindPassword password \
 --baseDn dc=example,dc=com \
 --trustAll \
 --no-prompt
```

   The existing server in this example is a directory server, as suggested by the *ds* in the hostname. The **`dsrepl add-local-server-to-pre-7-0-topology`** command does not support connecting to a standalone replication server.

   The command configures the new server, discovering the replication servers in the deployment, and setting the bootstrap replication servers.

   The command also generates one or more **`dsrepl initialize`** commands. Copy those commands, and add required credentials for use when initializing the new server.

   In the example command shown here:

- The `--bindDn` and `--bindPassword` options reflect either the UID and password of the existing servers' global replication administrator, or the DN and password of any user with sufficient access to act as global administrator on all servers.
- The insecure `--trustAll` option is used to simplify this procedure.

  To avoid using this option, add the remote server’s CA or signing certificate to the new server’s keystore, and use the appropriate keystore options.

4. Start the new server.
5. Initialize the new server with the **`dsrepl initialize`** command(s) from the previous step:

New server is a...	Initialize these base DNs
Combined DS/RS	`cn=admin data`, `cn=schema`, all directory data DNs
Standalone DS	`cn=admin data`, `cn=schema`, all directory data DNs
Standalone RS	`cn=admin data`

Directory Server

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

If you are adding a new server to an existing deployment, see When Adding New Servers instead.

IMPORTANT

Before upgrading, make sure you stop the server. Once you have unpacked the new server files, do not modify the server configuration until after you have completed the upgrade process.

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

1. Prepare for upgrade as described in Before You Upgrade.
2. Stop the server.
3. Proceed to upgrade the server:
   a. When upgrading a server installed from the cross-platform ZIP distribution:

- Unpack the new files over the old files as described in Unpack Files.
- Run the upgrade command to bring the server up to date with the new software delivery.

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

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

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

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

4. When the mutable data mounted at runtime differs from that of the instance where you first run the **`upgrade`** command, upgrade only mutable data by running the command again with the `--dataOnly` option at runtime.

   The `--dataOnly` option can be useful when running the server in a Docker container, for example.

   This improvement is available when upgrading from DS 6.0.0 or later releases.

5. Start the upgraded server.

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

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

   When you upgrade from OpenDJ 3.0, the upgrade process leaves the HTTP connection handler disabled.

   The newer REST to LDAP configuration is not necessarily compatible with the previous configuration. You must rewrite your configuration according to REST to LDAP Reference, and then configure the server to use the new configuration.

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

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

Directory Proxy

This page shows how to upgrade a directory proxy server in place. A directory proxy server has no local user data.

IMPORTANT

Before upgrading, make sure you stop the server. Once you have unpacked the new server files, do not modify the server configuration until after you have completed the upgrade process.

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

1. Prepare for upgrade as described in Before You Upgrade.
2. Stop the server.
3. Proceed to upgrade the server:
   a. When upgrading a server installed from the cross-platform ZIP distribution:
      - Unpack the new files over the old files as described in Unpack Files.
      - Run the upgrade command to bring the server up to date with the new software delivery.

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

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

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

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

4. When the mutable data mounted at runtime differs from that of the instance where you first run the **`upgrade`** command, upgrade only mutable data by running the command again with the `--dataOnly` option at runtime.

   The `--dataOnly` option can be useful when running the server in a Docker container, for example.

   This improvement is available when upgrading from DS 6.0.0 or later releases.
5. Start the upgraded server.

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

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

   When you upgrade from OpenDJ 3.0, the upgrade process leaves the HTTP connection handler disabled.

   The newer REST to LDAP configuration is not necessarily compatible with the previous configuration. You must rewrite your configuration according to REST to LDAP Reference, and then configure the server to use the new configuration.
6. If you disabled the Windows service to upgrade, enable it again:

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

Replication Server

This page shows how to upgrade a standalone replication server in place. A standalone replication server has no local user data. If the server holds user data, see Directory Server instead.

If you are adding a new server to an existing deployment, see When Adding New Servers instead.

IMPORTANT

Before upgrading, make sure you stop the server. Once you have unpacked the new server files, do not modify the server configuration until after you have completed the upgrade process.

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

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

2. Stop the server.
3. Proceed to upgrade the server:
   a. When upgrading a server installed from the cross-platform ZIP distribution:
      - Unpack the new files over the old files as described in Unpack Files.
      - Run the upgrade command to bring the server up to date with the new software delivery.

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

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

        When using the `--no-prompt` option, if the **`upgrade`** command cannot complete because it requires confirmation for a potentially long or critical task, then it exits with an error, and a message about how to finish making the changes. You can add the `--force` option to force a non-interactive upgrade to continue in this case, also performing long-running and critical tasks.
   b. When upgrading a server installed from native packages, use the system package management tools.
4. When the mutable data mounted at runtime differs from that of the instance where you first run the **`upgrade`** command, upgrade only mutable data by running the command again with the `--dataOnly` option at runtime.

   The `--dataOnly` option can be useful when running the server in a Docker container, for example.

   This improvement is available when upgrading from DS 6.0.0 or later releases.
5. Start the upgraded server.

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

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

   When you upgrade from OpenDJ 3.0, the upgrade process leaves the HTTP connection handler disabled.

The newer REST to LDAP configuration is not necessarily compatible with the previous configuration. You must rewrite your configuration according to REST to LDAP Reference, and then configure the server to use the new configuration.

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

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

REST to LDAP Gateway

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

DSML Gateway

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

After You Upgrade

The DS server upgrade process preserves the existing configuration as much as possible. This maintains compatibility, but there are additional steps you should take.

Overview

- Back up your directory data.

  Backup files are *not* compatible between versions.

- Update your scripts to account for incompatible changes.

  For details, refer to Incompatible changes.

- Plan your move away from deprecated features.

  For details, refer to Deprecated.

- Move to dedicated service accounts for your directory applications.

  You would not run all your UNIX applications as root, or all your Windows applications as Administrator. Stop using administrator accounts like `cn=Directory Manager` as service accounts.

Many DS setup profiles create service accounts for applications to use when authenticating to DS. For examples of AM service accounts, refer to the `base-entries.ldif` files in setup profiles under the `opendj/template/setup-profiles/AM` directory.

- Manually review and purge the DS server configurations for stale references to old servers.

  You can read the `opendj/config/config.ldif` file to find stale references, but always use the dsconfig command to make changes to the configuration.

- After you upgrade by adding new servers, but before you retire old servers, update bootstrap replication server settings to remove the old servers and add the new, DS 7 servers.

  For details, refer to Remove a Bootstrap Replication Server, and Add a Bootstrap Replication Server.

- Review what's new and changed in the intervening releases to identify useful changes.

  For this release, read the Release notes and plan to take advantage of new features and improvements.

- Apply the procedures that follow:
  - Use the New Security Model (in-place upgrades only)
  - Eliminate outdated password storage
  - Clean Up Admin Data
  - Add a Monitor User Account
  - Update LDAP Schema
  - Tune Settings
  - Use String-Based Server IDs

NOTE

Many example commands in this page use `cn=Directory Manager` as the name of the directory superuser account. This was the default before DS 7.

Here, `cn=Directory Manager` stands for the name of the directory superuser account in DS 6.5 and earlier.

Use the New Security Model

If you have upgraded DS servers in place, enable upgraded servers to use the new security model. While it is technically possible to continue with the old model, new features may require the new model.

NOTE

NOTE

If you started by adding DS 7 or later servers, as described in When Adding New Servers, then the new DS servers already have the keys. In that case, you can skip these steps.

DS release 7 changes the security model to let you configure replication at setup time, to make disaster recovery more straightforward, and to simplify symmetric key distribution:

- In prior releases, trust and symmetric key distribution in a replication topology depends on the replicated `cn=admin data` base DN. DS servers prior to release 7 reference each others' *instance keys*, and use them to protect symmetric keys in `cn=admin data` entries.
- DS servers now rely on a deployment key and password to derive a shared master key, and provide a default PKI to trust each others' certificates. DS servers protect symmetric keys using the shared master key to encrypt and decrypt them. For details, refer to Deployment Keys.

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 key for the deployment:

   ```
   ###
   # Generate a deployment key for the topology.
   # Do this once and SAVE THE DEPLOYMENT KEY:
   $ dskeymgr create-deployment-key --deploymentKeyPassword
   password
   <deployment-key>
   ```

   For more command options, refer to dskeymgr. The default validity for the deployment key is 10 years.
3. For each upgraded server, add at least the shared master key generated using the deployment key:

   ▼ ***Show details***

   ```
   ###
   # Use the same deployment key on each server:
   export DEPLOYMENT_KEY=<deployment-key>

   # Add a shared master key based on the deployment key:
   dskeymgr \
   ```

```
 export-master-key-pair \
 --alias master-key \
 --deploymentKey $DEPLOYMENT_KEY \
 --deploymentKeyPassword password \
 --keyStoreFile /path/to/opendj/config/keystore \
 --keyStorePassword:file
/path/to/opendj/config/keystore.pin

# Deployment key-based PKI?
# Add a deployment key CA certificate:
dskeymgr \
 export-ca-cert \
 --deploymentKey $DEPLOYMENT_KEY \
 --deploymentKeyPassword password \
 --keyStoreFile /path/to/opendj/config/keystore \
 --keyStorePassword:file
/path/to/opendj/config/keystore.pin

# Deployment key-based PKI?
# Add a deployment key-based TLS certificate:
dskeymgr \
 create-tls-key-pair \
 --deploymentKey $DEPLOYMENT_KEY \
 --deploymentKeyPassword 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.

▼ ***Show details***

The following example uses the private PKI keys based on the deployment key 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 key-based PKI?
dsconfig \
 create-trust-manager-provider \
 --set enabled:true \
 --set trust-store-file:config/keystore \
 --set trust-store-pin:\&{file:config/keystore.pin} \
 --set trust-store-type:PKCS12 \
 --type file-based \
 --provider-name PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

# Switch to the new keys to secure
# administrative and replication communications:
```

```
dsconfig \
 set-administration-connector-prop \
 --set ssl-cert-nickname:ssl-key-pair \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 set-synchronization-provider-prop \
 --provider-name "Multimaster Synchronization" \
 --set key-manager-provider:"Default Key Manager" \
 --set ssl-cert-nickname:ssl-key-pair \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

# Switch to the new keys for other secure communications:
dsconfig \
 set-connection-handler-prop \
 --handler-name HTTPS \
 --set ssl-cert-nickname:ssl-key-pair \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 set-connection-handler-prop \
 --handler-name LDAP \
 --set ssl-cert-nickname:ssl-key-pair \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 set-connection-handler-prop \
 --handler-name LDAPS \
 --set ssl-cert-nickname:ssl-key-pair \
 --set trust-manager-provider:PKCS12 \
 --hostname localhost \
 --port 4444 \
 --bindDn "cn=Directory Manager" \
 --trustAll \
 --bindPassword password \
 --no-prompt
```

6. For each upgraded server, restart the server, causing it to generate new secret keys, wrapped using the shared master key:

   ```
   stop-ds --restart
   ```

Eliminate outdated password storage

Reversible password storage schemes (3DES, AES, Blowfish, RC4) are deprecated since DS 7.0. Many password storage schemes are no longer enabled by default for new installations.

After upgrading to DS 7 and later, migrate active accounts away from the following deprecated and outdated password storage schemes:

- 3DES
- AES
- Base64
- Blowfish
- CRYPT
- Clear
- PBKDF2
- PKCS5S2
- SHA-1
- Salted SHA-1
- Salted SHA-256

- Salted SHA-384
- Salted SHA-512

For instructions on migrating accounts away from outdated storage schemes, refer to How do I change a password storage scheme and apply a new password policy to users in DS (All versions)?

Clean Up Admin Data

IMPORTANT

If, after cleanup, your deployment still stores secret keys under the replicated `cn=admin data` base DN, *do not disable `cn=admin data` or remove the `adminRoot` database*.

This applies, for example, to deployments that use (deprecated) reversible password storage schemes (3DES, AES, Blowfish, RC4). It also applies to deployments where servers were set up in production mode, and use keys with automatically generated, self-signed certificates to protect replication connections.

If you do choose to disable `cn=admin data` and remove the `adminRoot` database, you must first *manually* ensure that admin data is no longer used, and then remove references to it from your configuration.

1. Make sure you have upgraded all DS servers to version 7 or later.

   If you upgraded by adding new servers, and still have DS 6.5 or earlier servers, retire them before continuing.

   As explained in Overview at the top of this page, this means purging stale references to retired servers from the new servers' configurations, and updating bootstrap replication server settings to reference only the new, DS 7 servers.
2. If you upgraded in place, make sure you have followed the steps in Use the New Security Model.
3. Run the cleanup command.

   For example, run the cleanup command on each server with directory superuser credentials. If the credentials are the same on every server, it is sufficient to run the command once:

   1. After upgrade in place
   2. After adding new servers

```
$ dsrepl \
 cleanup-migrated-pre-7-0-topology \
 --bindDn "cn=Directory Manager" \
 --bindPassword password \
 --hostname localhost \
 --port 4444 \
 --trustAll \
 --no-prompt
```

```
$ dsrepl \
 cleanup-migrated-pre-7-0-topology \
 --bindDn uid=admin \
 --bindPassword password \
 --hostname localhost \
 --port 4444 \
 --trustAll \
 --no-prompt
```

The command is idempotent. You can run it multiple times if the initial run cannot fully complete the cleanup process.

4. Remove unused configuration settings:

▼ *Show details*

1. After upgrade in place

2. After adding new servers

```
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
```

```
dsconfig \
 delete-key-manager-provider \
 --provider-name "Crypto Manager Key Manager" \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --trustAll \
 --bindPassword password \
 --no-prompt

dsconfig \
 delete-key-manager-provider \
 --provider-name "Replication Key Manager" \
 --hostname localhost \
 --port 4444 \
 --bindDn uid=admin \
 --trustAll \
  --bindPassword password \
  --no-prompt

 dsconfig \
  delete-trust-manager-provider \
  --provider-name "Replication Trust Manager" \
  --hostname localhost \
  --port 4444 \
  --bindDn uid=admin \
  --trustAll \
  --bindPassword password \
  --no-prompt

 # Skip this command if the deployment has passwords
 stored
 # with reversible password storage schemes:
 dsconfig \
  delete-backend \
  --backend-name adminRoot \
  --hostname localhost \
  --port 4444 \
  --bindDn uid=admin \
  --trustAll \
  --bindPassword password \
  --no-prompt
```

5. Replace references to `Admin Data` in the server configuration.

   Find all references to admin data in your configuration:

   ```
   $ grep -i "admin data" /path/to/opendj/config/config.ldif
   ```

   How you replace or remove these references depends on your deployment.

6. Remove unused files:

   ```
   # Skip these commands if the deployment has passwords
   stored
   # with reversible password storage schemes:
   rm -rf /path/to/opendj/db/adminRoot
   rm -rf /path/to/opendj/db/ads-truststore
   ```

NOTE

After cleanup, the `dsrepl status` command can show `cn=admin data` status even if you have removed the backend.

Add a Monitor User Account

The **`dsrepl status`** command, and general server monitoring require an account with the `monitor-read` privilege. Since DS 6, you can create a monitor user account at setup time. However, the setup process does not *require* that you create such an account, and earlier versions do not offer the option.

If no such account exists, do one of the following:

- Add the `monitor-read` privilege to an existing, replicated user entry, as demonstrated in Monitor Privilege.
- Add a separate, replicated monitor user account, as demonstrated in How do I create a dedicated user for monitoring in DS?

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
   ```

```
Only in db/schema: 60-ds-evaluation-schema.ldif
Only in db/schema: 99-user.ldif
```

The following table summarizes the changes in detail:

Schema File	Notes	Action
`00-core.ldif`	Cosmetic changes due to schema replication: ◦ Each definition in `db/schema/00-core.ldif` has `X-SCHEMA-FILE '00-core.ldif'`. No definitions in `template/db/schema/00-core.ldif` have the `X-SCHEMA-FILE` extension. ◦ Some object classes in `db/schema/00-core.ldif` are explicitly defined as `STRUCTURAL`. Other minor differences: ◦ In 7, some attribute definitions have minimum upper bounds. ◦ The schema for collective attributes is extended.	**Replace** with template file
`03-pwpolicyextension.ldif`	The new version was rewritten to support fully featured replicated password policies.	**Replace** with template file
`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
$ ./bin/start-ds
```

Tune Settings

Major software releases include significant changes that can render existing tuning settings obsolete. When upgrading to a new major release of DS or Java software, revisit the system configuration, server configuration, and Java settings. Adjust the settings appropriately for your deployment as part of the upgrade process.

For information and suggestions on tuning, read the Release notes and Performance Tuning.

Use String-Based Server IDs

After upgrading from earlier releases, you can change server IDs to strings:

1. Make sure you have upgraded all DS servers to version 7 or later.
2. For each server, change the global server ID to the desired string.

   The following example shows a command that changes a server's global ID to a string:

```
$ dsconfig \
 set-global-configuration-prop \
 --hostname localhost \
 --port 4444 \
 --bindDN uid=admin \
 --bindPassword password \
 --set server-id:ds-us-west-1 \
 --usePkcs12TrustStore /path/to/opendj/config/keystore \
 --trustStorePassword:file
/path/to/opendj/config/keystore.pin \
 --no-prompt
```

3. Restart the server for the change to take effect.

Was this helpful?