Red Hat Directory Server 12

User management and authentication

Managing users, groups, roles, and authentication-related settings

Last Updated: 2024-10-10

Red Hat Directory Server 12 User management and authentication

Managing users, groups, roles, and authentication-related settings

Legal Notice

Abstract

Learn how to manage user access permissions, resource limits, user groups, and user roles. You can configure a password and account lockout policy, deny access for a group of users, and limit system resources depending on their bind distinguished name (bind DN).

Table of Contents

PROVIDING FEEDBACK ON RED HAT DIRECTORY SERVER

We appreciate your input on our documentation and products. Please let us know how we could make it better. To do so:

- For submitting feedback on the Red Hat Directory Server documentation through Jira (account required):
  1. Go to the Red Hat Issue Tracker.
  2. Enter a descriptive title in the **Summary** field.
  3. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
  4. Click **Create** at the bottom of the dialogue.
- For submitting feedback on the Red Hat Directory Server product through Jira (account required):
  1. Go to the Red Hat Issue Tracker.
  2. On the **Create Issue** page, click **Next**.
  3. Fill in the **Summary** field.
  4. Select the component in the **Component** field.
  5. Fill in the **Description** field including:
     a. The version number of the selected component.
     b. Steps to reproduce the problem or your suggestion for improvement.
  6. Click **Create**.

CHAPTER 1. USING GROUPS IN DIRECTORY SERVER

You can add users to groups in Directory Server. Groups are one of the mechanisms to group directory entries, that simplifies management of the user accounts.

When you use a group, Directory Server stores the distinguished name (DN) of the users who are members of this group in a membership attribute of the group entry. This special attribute is defined by the object class you choose when creating a group entry. For details about the group types, see Group types in Directory Server.

Groups are faster then roles. However, for a group to have benefits of a role, you need to enable the MemberOf plug-in. By default, the MemberOf plug-in automatically adds the **memberOf** attribute to a user entry if this user is a member of the group. As a result, the information about the membership is stored in both the group and user entries. For details about the MemberOf plug-in, see Listing group membership in user entries.

1.1. GROUP TYPES IN DIRECTORY SERVER

In Directory Server, you can add members to a **static** or **dynamic** group. For more details about definition of each group type, see About groups in Directory Server. A group object class defines a membership attribute, and to add a member to the group, you need to add a value to this membership attribute of the group entry.

The following table lists group object classes and corresponding membership attributes.

Group type	Object class	Membership attribute
Static	**groupOfNames**	**member**
	groupOfUniqueNames	**uniqueMember**
Dynamic	**groupOfURLs**	**memberURL**
	groupOfCertificates	**memberCertificate**

Object classes that you can use when you create a group:

- **groupOfNames** is a simple group. You can add any entry to this group. The **members** attribute determines the group membership. The **members** attribute values are distinguished names (DN) of user entries that are members of the group.
- **groupOfUniqueNames** lists user DNs as members, however the DNs must be unique. This group prevents self-referential group memberships. The **uniqueMember** attribute determines the group membership.
- **groupOfURLs** uses a list of LDAP URLs to filter and create its membership list. Any dynamic group requires this object class and you can use it in conjunction with **groupOfNames** and **groupOfUniqueNames**. The **memberURL** attribute determines the group membership.
- **groupOfCertificates** uses an LDAP filter to search for certificate names to identify group members. Use the **groupOfCertificates** object class for group-based access control, because you can give special access permissions to this group. The **memberCertificateDescription** attribute determines the group membership.

IMPORTANT

If you use an object class of a static group together with one of the dynamic object classes, the group becomes dynamic.

The MemberOf plug-in **does not** support dynamic groups. Therefore, the plug-in does not add the **memberOf** attribute to the user entry if the user entry matches the filter of a dynamic group.

1.2. CREATING A STATIC GROUP

You can create a static group by using the command line or the web console.

1.2.1. Creating a static group using the command line

Use the **dsidm** utility to create a static group with the **groupOfNames** object class. Use the **ldapmodify** utility to create a static group with the **groupOfUniqueNames** object class.

The following example creates two static groups in the **ou=groups,dc=example,dc=com** entry.

Prerequisites

- The **ou=groups,dc=example,dc=com** parent entry exists.

Procedure

- To create **cn=simple_group** group with the **groupOfNames** object class, run:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b
"dc=example,dc=com" group create --cn "simple_group"
Successfully created simple_group
```

  Note that the **dsidm group create** command creates groups only in the **ou=group** sub-entry. If you want to create a group in another entry, use **ldapmodify** utility.

- To create **cn=unique_members_group** group with the **groupOfUniqueNames** object class, run:

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: cn=unique_members_group,ou=groups,dc=example,dc=com
changetype: add
objectClass: top
objectClass: groupOfUniqueNames
cn: unique_members_group
description: A static group with unique members

adding new entry "cn=unique_members_group,ou=groups,dc=example,dc=com"
```

Verification

- Use **dsidm group list** command to list groups with the the **groupOfNames** object class:

```
# dsidm --basedn "dc=example,dc=com" instance_name group list
simple_group
```

- Use **dsidm uniquegroup list** command to list groups with the unique members:

```
# dsidm --basedn "dc=example,dc=com" instance_name uniquegroup list
unique_members_group
```

Next steps

- Adding members to a static group .

1.2.2. Creating a static group in the LDAP Browser

You can use the web console to create a static group. The following example creates a **static_group** in the **ou=groups,dc=example,dc=com** parent entry.

Prerequisites

- The **ou=groups,dc=example,dc=com** parent entry exists.
- You have permissions to log in to the instance in the web console. For more information about logging in to the web console, see Logging in to the Directory Server by using the web console .

Procedure

1. Navigate to **LDAP Browser** menu.
2. Using the **Tree** or **Table** view, expand the parent entry **ou=groups,dc=example,dc=com** under which you want to create the group.
3. Click the Options menu (▮) and select **New** to open the wizard window.
4. Select the **Create a group** and click **Next**.
5. Select the **Basic Group** for the groupe type and click **Next**.
6. Add the group name, group description, and select the membership attribute for the group:
   - **member** for the group with the **groupOfNames** object class.
   - **uniquemember** for the group with the **groupOfUniqueNames** object class.
7. Click **Next**.
8. Optional: Add members to the group and click **Next**.
9. Verify the group information, click **Create**, and **Finish**.

Verification

- Expand the newly created group entry in the suffix tree.

1.3. ADDING MEMBERS TO STATIC GROUPS

You can add a member to a group by using the command line of the web console.

1.3.1. Adding members to a static group using the command line

To add a member to a static group use the **ldapmodify** utility.

Prerequisites

- The group entry exists.
- The users entry exist.

Procedure

- To add a member to a static group with the **groupOfNames** object class, add the user distinguished name (DN) as the value to the **member** attribute of the group entry:

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: cn=simple_group,ou=groups,dc=example,dc=com
changetype: modify
add: member
member: uid=jsmith,ou=people,dc=example,dc=com

modifying entry "cn=simple_group,ou=groups,dc=example,dc=com"
```

  The command adds the **uid=jsmith** user to the **cn=simple_group** group.

- To add a member to a static group with the **groupOfUniqueNames** object class, add the user distinguished name (DN) as the value to the **uniqueMember** attribute of the group entry:

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: cn=unique_members_group,ou=groups,dc=example,dc=com
changetype: modify
add: uniqueMember
uniqueMember: uid=ajonson,ou=people,dc=example,dc=com

modifying entry "cn=unique_members_group,ou=groups,dc=example,dc=com"
```

  The command adds the **uid=ajonson** user to the **cn=unique_members_group** group.

Verification

- List the members of the group:

```
# ldapsearch -xLL -D "cn=Directory Manager" -W -b dc=example,dc=com "
(cn=simple_group)"

dn: cn=simple_group,ou=Groups,dc=example,dc=com
objectClass: top
objectClass: groupOfNames
objectClass: nsMemberOf
cn: simple_group
member: uid=jsmith,ou=people,dc=example,dc=com
member: uid=mtomson,ou=people,dc=example,dc=com
```

1.3.2. Adding members to a static group in LDAP Browser

You can add a member to a static group in the web console by using LDAP Browser.

Prerequisites

- The group entry exists.
- The user entry exists.
- You are logged in to the instance in the web console. For more details about logging in to the web console, see Logging in to the Directory Server by using the web console .

Procedure

1. Navigate to **LDAP Browser** menu.
2. Using the **Tree** or **Table** view, expand the group entry to which you want to add the member. For example. you want to add a member to **cn=unique_members_group,ou=groups,dc=example,dc=com**.
3. Click the Options menu (▮) and select **Edit** to open the wizard window. The window displays the current members list.
4. Select **Find New Members** tab.
5. Type the part of the **uid** or **cn** attribute value of the member in the search bar and press **Enter**. The **Available Members** field displays the user distinguished names (DN) that you can add to the group.
6. Select the member DN and move it to the **Chosen Members** field by click on the arrow (>).
7. Click **Add Member** button.

Verification

- Expand the **cn=unique_members_group,ou=groups,dc=example,dc=com** group entry and find the added user in the entry details.

1.4. CREATING A DYNAMIC GROUP USING THE COMMAND LINE

Directory Server supports creating dynamic groups by using only the command line. Use the **ldapmodify** utility to create a dynamic group with the **groupOfURLs** and **groupOfCertificates** object classes.

The following example creates two dynamic groups in the **ou=groups,dc=example,dc=com** entry.

Prerequisites

- The **ou=groups,dc=example,dc=com** parent entry exists.

Procedure

- To create **cn=example_dynamic_group** group with the **groupOfURLs** object class, run:

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: cn=example_dynamic_group,ou=groups,dc=example,dc=com
```

```
changetype: add
objectClass: top
objectClass: groupOfURLs
cn: example_dynamic_group
description: Example dynamic group for user entries
memberURL: ldap:///dc=example,dc=com??sub?(&(objectclass=person)(cn=*sen))

adding new entry "cn=example_dynamic_group,ou=groups,dc=example,dc=com"
```

The command creates a dynamic group that filters members with the **person** object class and the **sen** substring in the right part of the common name (**cn**) value.

- To create **cn=example_certificates_group** group with the **groupOfCertificates** object class, run:

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x
dn: cn=example_certificates_group,ou=groups,dc=example,dc=com
changetype: add
objectClass: top
objectClass: groupOfCertificates
cn: example_certificates_group
description: Example dynamic group for certificate entries
memberCertificateDescription: {ou=people, l=USA, dc=example, dc=com}

adding new entry "cn=example_certificates_group,ou=groups,dc=example,dc=com"
```

The command creates a dynamic group that filters members whose certificate subject DNs contain **ou=people,l=USA,dc=example,dc=com**.

Verification

- Search for the newly created group with the **groupOfURLs** object class:

```
# ldapsearch -xLLL -D "cn=Directory Manager" -W -H ldap://server.example.com -b
"dc=example,dc=com" "objectClass=groupOfURLs" 1.1

dn: cn=example_dynamic_group,ou=groups,dc=example,dc=com
```

- Search for the newly created group with the **groupOfCertificates** object class:

```
# ldapsearch -xLLL -D "cn=Directory Manager" -W -H ldap://server.example.com -b
"dc=example,dc=com" "objectClass=groupOfCertificates" 1.1

dn: cn=example_certificates_group,ou=groups,dc=example,dc=com
```

Additional resources

- Adding an LDAP entry using the command line

1.5. LISTING GROUP MEMBERSHIP IN USER ENTRIES

A group defines entries that belong to this group by using the membership attribute. It is easy to look at the group and find its members. For example, a static group with the **groupOfNames** object class stores distinguished names (DNs) of its members as values of the **member** attribute. However, you cannot

quickly find out what groups a single user belongs to. With groups, a user entry does not contain anything that indicates the user memberships, unlike with roles.

To solve this problem, you can use the MemberOf plug-in. The MemberOf plug-in analyzes the membership attribute in a group entry and automatically writes the **memberOf** attribute in the user entry that points to the group. By default, the plug-in checks the **member** attribute in the groups, however, you can use several attributes to support different group types.

When you add or delete a member of a group, the plug-in updates the **memberOf** attributes in the user entries. With the MemberOf plug-in, you can do a simple search against a specific user entry to find all groups that the user is a member of. The MemberOf Plug-in shows direct and indirect memberships for all groups.

IMPORTANT

The MemberOf plug-in manages membership attributes only for **static** groups.

Additional resources

- Group types in Directory Server

1.5.1. Considerations when using the MemberOf plug-in

When using the MemberOf plug-in, consider the following:

- The MemberOf plug-in in a replication topology
  In a replication topology, you can manage the MemberOf plug-in in two ways:
  - Enable the MemberOf plug-in on all supplier and consumer servers in the topology. In this case, you must exclude the **memberOf** attribute of user entries from replication in all replication agreements.
  - Enable the MemberOf plug-in only on all supplier servers in the topology. To do this:
    - You must disable replication of the **memberOf** attribute to all write-enabled suppliers in the replication agreement.
    - You must enable replication of the **memberOf** attribute to all consumer replicas in their replication agreement.
    - You must disable the MemberOf plug-in on consumer replicas.
- The MemberOf plug-in with distributed databases
  As described in Creating and maintaining databases, you can store sub-trees of your directory in separate databases. By default, the MemberOf plug-in only updates user entries that are stored within the same database as the group. To update users across all databases, you must set the **memberOfAllBackends** parameter to **on**. For more details about setting the **memberOfAllBackends** parameter, see Configuring the MemberOf plug-in on each server using the web console.

1.5.2. Required object classes for the MemberOf plug-in

By default, the MemberOf plug-in adds the **nsMemberOf** object class to user entries to provide the **memberOf** attribute. The **nsMemberOf** object class is sufficient for the plug-in to work correctly.

Alternatively, you can create user entries that contain the **inetUser**,**inetAdmin**, **inetOrgPerson** object class. These object classes support the **memberOf** attribute.

To configure nested groups, the group must use the **extensibleObject** object class.

NOTE

If directory entries do not contain an object class that supports required attributes operations fail with the following error:

```
LDAP: error code 65 - Object Class Violation
```

1.5.3. The MemberOf plug-in syntax

When configuring the MemberOf plug-in, you set the main two attributes:

- **memberOfGroupAttr**. Defines which membership attribute to poll from the group entry. The **memberOfGroupAttr** attribute is multi-valued. Therefore, the plug-in can manage multiple types of groups. By default, the plug-in polls the **member** attribute.
- **memberOfAttr**. Defines which membership attribute to create and manage in the member's user entry. By default, the plug-in adds the **memberOf** attribute to the user entry.

In addition, the plug-in syntax provides the plug-in path, function to identify the MemberOf plug-in, the plug-in state, and other configuration parameters.

The following example shows the default MemberOf plug-in entry configuration:

```
dn: cn=MemberOf Plugin,cn=plugins,cn=config
cn: MemberOf Plugin
memberofallbackends: off
memberofattr: memberOf
memberofentryscope: dc=example,dc=com
memberofgroupattr: member
memberofskipnested: off
nsslapd-plugin-depends-on-type: database
nsslapd-pluginDescription: memberof plugin
nsslapd-pluginEnabled: off
nsslapd-pluginId: memberof
nsslapd-pluginInitfunc: memberof_postop_init
nsslapd-pluginPath: libmemberof-plugin
nsslapd-pluginType: betxnpostoperation
nsslapd-pluginVendor: 389 Project
nsslapd-pluginVersion: 2.4.5
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
```

For details about the parameters in the example and other parameters you can set, see MemberOf plug-in section in the "Configuration and schema reference" documentation.

1.5.4. Enabling the MemberOf plug-in

You can enable the MemberOf plug-in by using the command line or the web console.

1.5.4.1. Enabling the MemberOf plug-in using the command line

Use the **dsconf** utility to enable the MemberOf plug-in.

Procedure

1. Enable the plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof enable
```

2. Restart the instance:

```
# dsctl instance_name restart
```

Verification

- View the plug-in configuration details:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof show

dn: cn=MemberOf Plugin,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
...
```

Additional resources

- Configuring the MemberOf plug-in on each server

1.5.4.2. Enabling the MemberOf plug-in using the web console

You can use the web console to enable the MemberOf plug-in.

Prerequisites

- You are logged in to the instance in the web console. For more details about logging in to the web console, see Logging in to the Directory Server by using the web console .

Procedure

1. Navigate to the **Plugins** menu.
2. Select the **MemberOf** plug-in in the list of plug-ins.
3. Change the status to **ON** to enable the plug-in.
4. Restart the instance. For instructions for restarting the instance, see Starting and stopping a Directory Server instance by using the web console.

Additional resources

- Configuring the MemberOf plug-in on each server using the web console

1.5.5. Configuring the MemberOf plug-in on each server

If you do not want to replicate the configuration of the MemberOf plug-in, configure the plug-in manually on each server by using the command line or the web console.

1.5.5.1. Configuring the MemberOf plug-in on each server using the command line

By default, the MemberOf plug-in reads the **member** membership attribute from the group entries and adds the **memberOf** attribute to the user entries. However, you can configure the plug-in to read other membership attribute from the group, add another attribute to the user entry, skip nested groups, work on all databases and other settings.

For example, you want the MemberOf plug-in to do the following:

- Read **uniqueMember** attribute from group entries to identify membership.
- Skip nested groups.
- Search for user entries in all databases.

Prerequisites

- You enabled the MemberOf plug-in. For details, see Enabling the MemberOf plug-in.

Procedure

1. Optionally: Display the MemberOf plug-in configuration to see which membership attribute the plug-in currently reads from groups entries:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof show

...
memberofgroupattr: member
...
```

   The plug-in currently reads the **member** attribute from the group entry to retrieve members.

2. Set the **uniqueMember** attribute as the value to the **memberOfGroupAttr** parameter in the plug-in configuration:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --groupattr uniqueMember
```

   The **memberOfGroupAttr** parameter is multi-valued and you can set several values by passing them all to the **--groupattr** parameter. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --groupattr member uniqueMember
```

3. In an environment that uses distributed databases, configure the plug-in to search user entries in all databases instead of only the local database:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --allbackends on
```

The command sets the **memberOfAllBackends** parameter.

4. Configure the plug-in to skip nested groups:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --skipnested on
```

The command sets the **memberOfSkipNested** parameter.

5. Optional: By default, the plug-in adds **nsMemberOf** object class to user entries if the user entries do not have the object class that allows the **memberOf** attribute. To configure the plug-in to add the **inetUser** object class to the user entries instead of **nsMemberOf**, run:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --autoaddoc inetUser
```

The command sets the **memberOfAutoAddOC** parameter.

6. Restart the instance:

```
# dsctl instance_name restart
```

Verification

- View the MemberOf plug-in configuration:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof show
dn: cn=MemberOf Plugin,cn=plugins,cn=config
cn: MemberOf Plugin
memberofallbackends: on
memberofattr: memberOf
memberofautoaddoc: inetuser
memberofentryscope: dc=example,dc=com
memberofgroupattr: uniqueMember
memberofskipnested: on
...
nsslapd-pluginEnabled: on
...
```

Additional resources

- Updating the **memberOf** attribute values in user entries using the **fixup** task
- Sharing the MemberOf plug-in configuration between servers
- Setting the scope of the MemberOf plug-in
- Required object classes for the MemberOf plug-in

1.5.5.2. Configuring the MemberOf plug-in on each server using the web console

By default, the MemberOf plug-in reads the **member** membership attribute from the group entries and adds the **memberOf** attribute to the user entries. However, you can configure the plug-in to read other membership attribute from the group, skip nested groups, work on all databases and other settings by

using the web console.

For example, you want the MemberOf plug-in to do the following:

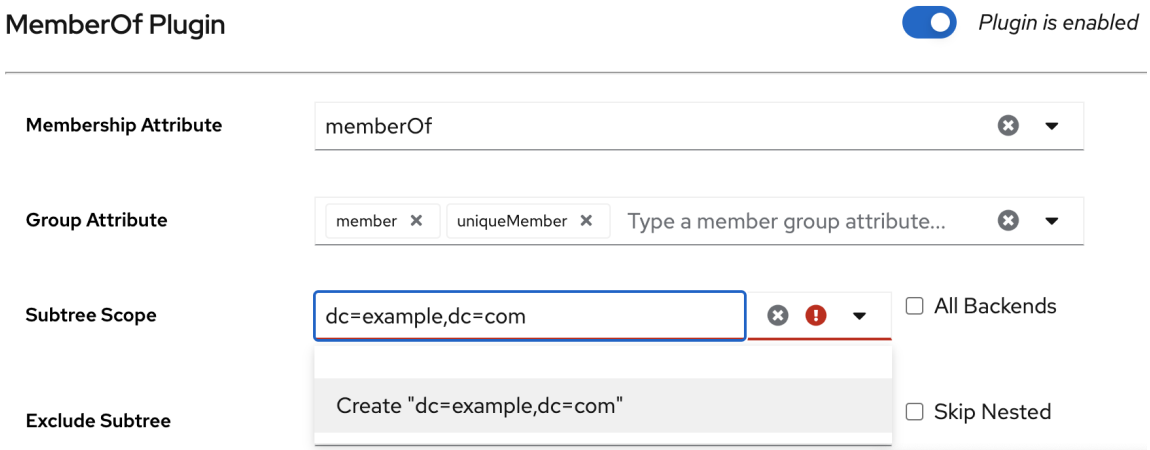
- Read **member** and **uniqueMember** attributes from group entries to identify membership.
- Set the scope of the plug-in to **dc=example,dc=com**.
- Skip nested groups.
- Search for user entries in all databases.

Prerequisites

- You are logged in to the instance in the web console. For more details about logging in to the web console, see Logging in to the Directory Server by using the web console .
- You enabled the MemberOf plug-in. For details, see Enabling the MemberOf plug-in.

Procedure

1. Navigate to **LDAP Browser** menu.
2. Select the **MemberOf** plug-in from the plug-ins list.
3. Add the **uniqueMember** attribute to the **Group Attribute** field.
4. Set the scope of the plug-in to **dc=example,dc=com**:
   a. Enter **dc=example,dc=com** to the **Subtree Scope** field.
   b. Click **Create "dc=example,dc=com"** in the drop-down list.



5. Optional: Set a subtree to exclude. For example, you do not want the plug-in to work on the **ou=private,dc=example,dc=com** subtree:
   a. Enter **ou=private,dc=example,dc=com** to the **Exclude Subtree** field.
   b. Click **Create "ou=private,dc=example,dc=com"** in the drop-down list.
6. Check **All Backends** to configure the plug-in to search user entries in all databases instead of only the local database.
7. Check **Skip Nested** to configure the plug-in to skip nested groups.

8. Click **Save Config**.

Additional resources

- Sharing the MemberOf plug-in configuration between servers
- Setting the scope of the MemberOf plug-in using the command line
- MemberOf plug-in
- Updating the **memberOf** attribute values in user entries using the **fixup** task

1.5.6. Sharing the MemberOf plug-in configuration between servers

By default, each server stores its own configuration of the MemberOf plug-in. With the shared configuration of the plug-in, you can use the same settings without configuring the plug-in manually on each server. Directory Server stores the shared configuration outside of the **cn=config** suffix and replicates it.

For example, you want to store the plug-in shared configuration in the **cn=shared_MemberOf_config,dc=example,dc=com** entry.

IMPORTANT

After enabling the shared configuration, the plug-in ignores all parameters set in the **cn=MemberOf Plugin,cn=plugins,cn=config** plug-in entry and only uses settings from the shared configuration entry.

Prerequisites

- You enabled the MemberOf plug-in on all servers in the replication topology. For details, see Enabling the MemberOf plug-in.

Procedure

1. Enable the shared configuration entry on a server:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof
config-entry add "cn=shared_MemberOf_config,dc=example,dc=com" --attr memberOf
--groupattr member

Successfully created the cn=shared_MemberOf_config,dc=example,dc=com
MemberOf attribute nsslapd-pluginConfigArea (config-entry) was set in the main plugin
config
```

   The command sets **nsslapd-pluginConfigArea** attribute value to **cn=shared_MemberOf_config,dc=example,dc=com**.

2. Restart the instance:

```
# dsctl instance_name restart
```

3. Enable the shared configuration on other servers in the replication topology that should use the shared configuration:

a. Set the distinguished name (DN) of the configuration entry that stores the shared configuration:

```
# dsconf -D "cn=Directory Manager" ldap://server2.example.com plugin memberof set --config-entry cn=shared_MemberOf_config,dc=example,dc=com
```

b. Restart the instance:

```
# dsctl instance_name restart
```

Verification

1. Check that the MemberOf plug-in uses the shared configuration:

```
# dsconf -D "cn=Directory Manager" ldap://server1.example.com plugin memberof show

dn: cn=MemberOf Plugin,cn=plugins,cn=config
cn: MemberOf Plugin
...
nsslapd-pluginConfigArea: cn=shared_MemberOf_config,dc=example,dc=com
...
```

2. Optional: Check the shared configuration settings:

```
# dsconf -D "cn=Directory Manager" ldap://server1.example.com plugin memberof config-entry show "cn=shared_MemberOf_config,dc=example,dc=com"

dn: cn=shared_MemberOf_config,dc=example,dc=com
cn: shared_MemberOf_config
memberofattr: memberOf
memberofgroupattr: member
objectClass: top
objectClass: extensibleObject
```

Additional resources

- nsslapd-pluginConfigArea

1.5.7. Setting the scope of the MemberOf plug-in

If you configured several backends or multiple-nested suffixes, you can use the **memberOfEntryScope** and **memberOfEntryScopeExcludeSubtree** parameters to set what suffixes the **MemberOf** plug-in works on.

If you add a user to a group, the MemberOf plug-in only adds the **memberOf** attribute to the group if both the user and the group are in the plug-in's scope.

For example, the following procedure configures the MemberOf plug-in to work on all entries in **dc=example,dc=com**, but to exclude entries in **ou=private,dc=example,dc=com**.

Prerequisites

- You enabled the MemberOf plug-in on all servers in the replication topology. For details, see Enabling the MemberOf plug-in.

Procedure

1. Set the scope value for the MemberOf plug-in to **dc=example,dc=com**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --scope "dc=example,dc=com"
```

2. Exclude entries in **ou=private,dc=example,dc=com**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof set --exclude "ou=private,dc=example,com"
```

   If you moved a user entry out of the scope by using the **--scope** DN parameter:

   - The MemberOf plug-in updates the membership attribute, such as **member**, in the group entry to remove the user DN value.
   - The MemberOf plug-in updates the **memberOf** attribute in the user entry to remove the group DN value.

NOTE

The value set in the **--exclude** parameter has a higher priority than values set in **--scope**. If the scopes set in both parameters overlap, the MemberOf plug-in only works on the non-overlapping directory entries.

For details about setting the scope for the MemberOf plug-in, see Configuring the MemberOf plug-in on each server using the web console.

1.5.8. Updating the memberOf attribute values in user entries using the fixup task

The MemberOf plug-in automatically manages **memberOf** attributes in group member entries based on the configuration in the group entry. However, you need to run the **fixup** task in the following situations to avoid inconsistency between the **memberOf** configuration that the server plug-in manages and the actual memberships defined in user entries:

- You added group members to a group before you enabled the MemberOf plug-in.
- You manually edited the **memberOf** attribute in a user entry.
- You imported or replicated new user entries to the server that already have the **memberOf** attribute.

Note that you can run the **fixup** tasks only locally. In a replication environment, Directory Server updates the **memberOf** attribute for entries on other servers after Directory Server replicates the updated entries.

Prerequisites

- You enabled the MemberOf plug-in on all servers in the replication topology. For details, see Enabling the MemberOf plug-in.

Procedure

- For example, to update the **memberOf** values in **dc=example,dc=com** entry and subentries, run:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof fixup
"dc=example,dc=com"
Attempting to add task entry...
Successfully added task entry
```

By default, the **fixup** task updates **memberOf** values in all entries that contain the **inetUser**, **inetAdmin**, or **nsMemberOf** object class.

If you want the **fixup** task to also work on entries that contain other object classes, use **-f** filter option:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin memberof fixup
-f "(|(objectclass=inetuser)(objectclass=inetadmin)(objectclass=nsmemberof)
(objectclass=nsmemberof)(objectclass=inetOrgPerson))" "dc=example,dc=com"
```

This **fixup** task updates **memberOf** values in all entries that contain the **inetUser**, **inetAdmin**, **nsMemberOf**, or **inetOrgPerson** object class.

Additional resources

- LDAP search filters

CHAPTER 2. USING ROLES IN DIRECTORY SERVER

You can group Directory Server entries by using roles. Roles behave as both a static and a dynamic group. Roles are easier to use than groups because they are more flexible in their implementation. For example, an application can get the list of roles to which an entry belongs by querying the entry itself rather than selecting a group and browsing the members list of several groups.

You can manage roles by using the command line or the web console.

2.1. ROLES IN DIRECTORY SERVER

A role behaves as both a static and a dynamic group, similarly to a hybrid group:

- With a group, Directory Server adds entries to the group entry as members.
- With a role, Directory Server adds the role attribute to the entry and then uses this attribute to automatically identify members in the role entry.

Role members are entries that possess the role. You can specify members of the role explicitly or dynamically depending on the role type. Directory Server supports the following types of roles:

- *Managed roles*
  Managed roles have an explicit list of members. You can use managed roles to perform the same tasks that you perform with static groups.
- *Filtered roles*
  You can filter the role members by using filtered roles, similarly to filtering with dynamic groups. Directory Server assigns entries to a filtered role depending on whether the entry possesses a specific attribute defined in the role.
- *Nested roles*
  Nested roles can contain managed and filtered roles.

When you create a role, determine if users can add or remove themselves from the role. For more details, see Section 2.2, “Using roles securely in Directory Server” .

NOTE

Evaluating roles is more resource-intensive for the Directory Server than evaluating groups because the server does the work for the client application. With roles, the client application can check role membership by searching for the **nsRole** attribute. The **nsRole** attribute is a computed attribute that identifies which roles an entry belongs to. Directory Server does not store the **nsRole** attribute. From the client application point of view, the method for checking membership is uniform and is performed on the server side.

Find considerations for using roles in [Deciding between groups and roles] in the *Planning and designing a directory service*. documentation.

Additional resources

- Managing groups in Directory Server

2.2. USING ROLES SECURELY IN DIRECTORY SERVER

When creating a new role, consider if users can easily add or remove themselves from a role. For example, you can allow users of the **Mountain Biking** interest group role to add or remove themselves easily. However, you must not allow users who are assigned the **Marketing** role to add or remove themselves from the role.

One potential security risk is inactivating user accounts by inactivating roles. Inactive roles have special access control instructions (ACIs) defined for their suffix. If an administrator allows users to add and remove themselves from roles freely, these users can remove themselves from an inactive role to unlock their accounts.

For example, a user is assigned a managed role. When Directory Server locks this managed role by using account inactivation, the user can not bind to the server because Directory Server computes the **nsAccountLock** attribute as **true** for that user. However, if the user was already bound to Directory Server and now is locked through the managed role, the user can remove the **nsRoleDN** attribute from his entry and unlock himself if no restricting ACIs are specified.

To prevent users from removing the **nsRoleDN** attribute, use the following ACIs depending on the type of role:

- Managed roles. For entries that are members of a managed role, use the following ACI:

```
aci: (targetattr="nsRoleDN")
(targattrfilters= add=nsRoleDN:(!(nsRoleDN=cn=AdministratorRole,dc=example,dc=com)),
del=nsRoleDN:(!(nsRoleDN=cn=nsManagedDisabledRole,dc=example,dc=com)))
(version3.0;acl "allow mod of nsRoleDN by self but not to critical values"; allow(write)
userdn=ldap:///self;)
```

- Filtered roles. Protect attributes that are part of the filter (**nsRoleFilter**). Do not allow a user to add, delete, or modify the attribute that the filtered role uses. If Directory Server computes the value of the filter attribute, then you must protect all attributes that can modify this filter attribute value.

- Nested roles. A nested role can contain filtered and managed roles. Thus, you must restrict modify operations in ACIs for each attribute of the roles that the nested role contains.

Additional resources

- Manually inactivating users and roles

2.3. MANAGING ROLES IN DIRECTORY SERVER BY USING THE COMMAND LINE

You can view, create, and delete roles by using the command line.

2.3.1. Creating a managed role in Directory Server

Managed roles are roles that have an explicit enumerated list of members. You can use the **ldapmodify** utility to create a managed role. The following example creates a managed role for a marketing team.

Prerequisites

- The **ou=people,dc=example,dc=com** parent entry exists in Directory Server.
- The **cn=Bob Jones,ou=people,dc=example,dc=com** user entry exists in Directory Server.

Procedure

1. Create the **cn=Marketing** managed role entry by using the **ldapmodify** command with the **-a** option:

```
# ldapmodify -a -D "cn=Directory Manager" -W -H ldap://server.example.com -x << EOF
dn: cn=Marketing,ou=people,dc=example,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsSimpleRoleDefinition
objectclass: nsManagedRoleDefinition
cn: Marketing
description: managed role for the marketing team
EOF
```

   The managed role entry must contain the following object classes:

   - **LDAPsubentry**
   - **nsRoleDefinition**
   - **nsSimpleRoleDefinition**
   - **nsManagedRoleDefinition**

2. Assign the **cn=Marketing,ou=people,dc=example,dc=com** managed role to the **cn=Bob Jones,ou=people,dc=example,dc=com** user entry by adding the **nsRoleDN** attribute to this user entry:

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x << EOF
dn: cn=Bob Jones,ou=people,dc=example,dc=com
changetype: modify
add: nsRoleDN
nsRoleDN: cn=Marketing,ou=people,dc=example,dc=com
EOF

modifying entry "cn=Bob Jones,ou=people,dc=example,dc=com"
```

3. Optional: Configure the *equality* index for the **nsRoleDN** attribute in the **userRoot** database to avoid unindexed searches:

```
#  dsconf -D "cn=Directory Manager" ldap://server.example.com backend index add --
index-type eq --attr nsroleDN --reindex userRoot
```

Verification

- List user entries that now belong to the **cn=Marketing,ou=people,dc=example,dc=com** managed role:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -b
"dc=example,dc=com" "(nsRole=cn=Marketing,ou=people,dc=example,dc=com)" dn
```

```
dn: cn=Bob Jones,ou=people,dc=example,dc=com

dn: cn=Tom Devis,ou=people,dc=example,dc=com
```

Additional resources

- Creating a role in the LDAP Browser
- Providing input to the ldapadd, ldapmodify, and ldapdelete utilities
- Maintaining the indexes of a specific database using the command line
- **ldapmodify(1)** man page

2.3.2. Creating a filtered role in Directory Server

Directory Server assigns entries to a filtered role if the entries have a specific attribute defined in the role. The role definition specifies the **nsRoleFilter** LDAP filter. Entries that match the filter are members of the role.

You can use **ldapmodify** utility to create a filtered role. The following example creates a filtered role for sales department managers.

Prerequisites

- The **ou=people,dc=example,dc=com** parent entry exists in Directory Server.

Procedure

1. Create the **cn=SalesManagerFilter** filtered role entry by using the **ldapmodify** command with the **-a** option:

```
# ldapmodify -a -D "cn=Directory Manager" -W -H ldap://server.example.com -x << EOF
dn: cn=SalesManagerFilter,ou=people,dc=example,dc=com
changetype: add
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsFilteredRoleDefinition
cn: SalesManagerFilter
nsRoleFilter: o=sales managers
Description: filtered role for sales managers
EOF
```

   The **cn=SalesManagerFilter** filtered role entry has the **o=sales managers** filter for the role. All user entries that have the **o** attribute with the value of **sales managers** are members of the filtered role.

   Example of the user entry that is now a member of the filtered role:

```
dn: cn=Pat Smith,ou=people,dc=example,dc=com
objectclass: person
cn: Pat
```

```
sn: Smith
userPassword: password
o: sales managers
```

The filtered role entry must have the following object classes:

- **LDAPsubentry**
- **nsRoleDefinition**
- **nsComplexRoleDefinition**
- **nsFilteredRoleDefinition**

2. Optional: Configure the *equality* index for the attribute that you use in the **nsRoleFilter** role filter to avoid unindexed searches. In the given example, the role uses **o=sales managers** as the filter. Therefore, index the **o** attribute to improve the search performance:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com backend index add --index-type eq --attr o --reindex userRoot
```

Verification

- List user entries that now belong to the **cn=SalesManagerFilter,ou=people,dc=example,dc=com** filtered role:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -b "dc=example,dc=com" "(nsRole=cn=SalesManagerFilter,ou=people,dc=example,dc=com)" dn

dn: cn=Jess Mor,ou=people,dc=example,dc=com

dn: cn=Pat Smith,ou=people,dc=example,dc=com
```

Additional resources

- Creating a role in the LDAP Browser
- Providing input to the ldapadd, ldapmodify, and ldapdelete utilities
- **ldapmodify(1)** man page

2.3.3. Creating a nested role in Directory Server

Nested roles can contain managed and filtered roles. A nested role entry requires the **nsRoleDN** attribute to identify the roles to nest.

You can use **ldapmodify** utility to create a nested role. The following example creates a nested role that contains the managed and the filtered roles you created in Creating a managed role in Directory Server and Creating a filtered role in Directory Server .

Prerequisites

- The **ou=people,dc=example,dc=com** parent entry exists in Directory Server.

Procedure

1. Create the **cn=MarketingSales** nested role entry that contains the **cn=SalesManagerFilter** filtered role and the **cn=Marketing** managed role by using the **ldapmodify** command with the **-a** option:

```
# ldapmodify -a -D "cn=Directory Manager" -W -H ldap://server.example.com -x << EOF
dn: cn=MarketingSales,ou=people,dc=example,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: nsRoleDefinition
objectclass: nsComplexRoleDefinition
objectclass: nsNestedRoleDefinition
cn: MarketingSales
nsRoleDN: cn=SalesManagerFilter,ou=people,dc=example,dc=com
nsRoleDN: cn=Marketing,ou=people,dc=example,dc=com
EOF
```

   Optionally, the role can have the **description** attribute.

   The nested role entry must have the following object classes:

   - **LDAPsubentry**
   - **nsRoleDefinition**
   - **nsComplexRoleDefinition**
   - **nsNestedRoleDefinition**

Verification

- List user entries that now belong to the **cn=MarketingSales** nested role:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -x -b
"dc=example,dc=com" "(nsRole=cn=MarketingSales,ou=people,dc=example,dc=com)"
dn

dn: cn=Bob Jones,ou=people,dc=example,dc=com
dn: cn=Pat Smith,ou=people,dc=example,dc=com
dn: cn=Jess Mor,ou=people,dc=example,dc=com
dn: cn=Tom Devis,ou=people,dc=example,dc=com
```

Additional resources

- Creating a role in the LDAP Browser
- Providing input to the ldapadd, ldapmodify, and ldapdelete utilities
- **ldapmodify(1)** man page

2.3.4. Viewing roles for an entry

To view roles for an entry, use the **ldapsearch** command with explicitly specified **nsRole** virtual attribute.

Prerequisites

- Roles entry exists.
- You assigned roles to the **uid=user_name** user entry.

Procedure

- Search for the **uid=*user_name*** entry with specified **nsRole** virtual attribute:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -b
"dc=example,dc=com" -s sub -x "(uid=user_name)" nsRole

dn: uid=user_name,ou=people,dc=example,dc=com
...
nsRole: cn=Role for Managers,dc=example,dc=com
nsRole: cn=Role for Accounting,dc=example,dc=com
```

The command retrieves all roles which the **uid=*user_name*** user is a member of.

2.3.5. Deleting roles in Directory Server

To delete a role in Directory Server, you can use **ldapmodify** command.

The following is an example of deleting the **cn=Marketing** managed role from Directory Server.

Procedure

- To delete the **cn=Marketing** managed role entry, enter:

```
# ldapmodify -D "cn=Directory Manager" -W -H ldap://server.example.com -x << EOF
dn: cn=Marketing,ou=People,dc=example,dc=com
changetype: delete
EOF

deleting entry "cn=Marketing,ou=People,dc=example,dc=com"
```

NOTE

When you delete a role, Directory Server deletes only the role entry and does not delete the **nsRoleDN** attribute for each role member. To delete the **nsRoleDN** attribute for each role member, enable the Referential Integrity plug-in and configure this plug-in to manage the **nsRoleDN** attribute.

For more information about the Referential Integrity plug-in, see Using Referential Integrity to maintain relationships between entries.

Additional resources

- Deleting a role in the LDAP browser
- **ldapmodify(1)** man page

2.4. MANAGING ROLES IN DIRECTORY SERVER BY USING THE WEB CONSOLE

You can view, create, and delete roles by using **LDAP browser** in the web console.

2.4.1. Creating a role in the LDAP Browser

You can create a role for a Red Hat Directory Server entry by using the **LDAP Browser** wizard in the web console.

Prerequisites

- Access to the web console.
- A parent entry exists in Directory Server.

Procedure

1. Log in to the web console and click **Red Hat Directory Server**.
2. After the web console loads the **Red Hat Directory Server** interface, open the **LDAP Browser**.
3. Select an LDAP entry and open the **Options** menu.
4. From the drop-down menu select **New** and click **Create a new role**.
5. Follow the steps in the wizard and click the **Next** button after you complete each step.
6. To create the role, review the role settings in the **Create Role** step and click the **Create** button. You can click the **Back** button to modify the role settings or click the **Cancel** button to cancel the role creation.
7. To close the wizard window, click the **Finish** button.

Verification

- Expand the LDAP entry and verify the new role appears among the entry parameters.

2.4.2. Deleting a role in the LDAP browser

You can delete the role from the Red Hat Directory Server entry by using the **LDAP Browser** in the web console.

Prerequisites

- Access to the web console.
- A parent entry exists in Directory Server.

Procedure

1. Log in to the web console and click **Red Hat Directory Server**.
2. After the web console loads the **Red Hat Directory Server** interface, click **LDAP browser**.

3. Expand the LDAP entry select the role which you want to delete.
4. Open the **Options** menu and select **Delete**.
5. Verify the data about the role you want to delete and click the **Next** button until you reach the **Deletion** step.
6. Toggle the switch to the **Yes, I'm sure** position and click the **Delete** button.
7. To close the wizard window, click the **Finish** button.

Verification

- Expand the LDAP entry and verify the role is no longer a part of the entry parameters.

2.4.3. Modifying a role in the LDAP browser

You can modify the role parameters for a Red Hat Directory Server entry by using the **LDAP Browser** in the web console.

Prerequisites

- Access to the web console.
- A parent entry exists in the Red Hat Directory Server.

Procedure

1. Log in to the web console and click **Red Hat Directory Server**.
2. After the web console loads the **Red Hat Directory Server** interface, click **LDAP Browser**.
3. Expand the LDAP entry and select the role you are modifying.
4. Click the **Options** menu and select **Edit** to modify the parameters of the role or **Rename** to rename the role.
5. In the wizard window modify the necessary parameters and click **Next** after each step until you observe the **LDIF Statements** step.
6. Check the updated parameters and click **Modify Entry** or **Change Entry Name**.
7. To close the wizard window, click the **Finish** button.

Verification

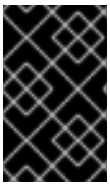
- Expand the LDAP entry and verify the updated parameters are listed for the role.

CHAPTER 3. CHANGING THE DIRECTORY MANAGER PASSWORD

The Directory Manager is the privileged database administrator, comparable to the **root** user in a Linux operating system. The Directory Manager entry and the corresponding password are set during the instance installation. As an administrator, you can change the Directory Manager password to use a different one.

3.1. CHANGING THE DIRECTORY MANAGER PASSWORD USING THE COMMAND LINE

You can set a new password for the Directory Manager using the **dsconf** command line utility or manually by setting the **nsslapd-rootpw** parameter.



IMPORTANT

Set the password using an encrypted connection only. Using an unencrypted connection can expose the password to the network. If your server does not support encrypted connections, use the web console to update the Directory Manager password.

Procedure

- Set the Directory Manager password using one of the following options:
  - To encrypt the password automatically:

    ```
    # dsconf -D "cn=Directory Manager" ldaps://server.example.com config replace nsslapd-rootpw=password
    ```

    Directory Server automatically encrypts the plain text value that you set in the **nsslapd-rootpw** parameter.

WARNING

Do not use curly braces **{}** in the password. Directory Server stores the password in the **{password-storage-scheme}hashed_password** format. The server interprets characters in curly braces as the password storage scheme. If the string is an invalid storage scheme or if the password is not correctly hashed, the Directory Manager cannot connect to the server.

  - To encrypt the password manually:

    1. Generate a new password hash. For example:

    ```
    # pwdhash -D /etc/dirsrv/slapd-instance_name password
    {PBKDF2_SHA256}AAAgAMwPYIhEkQozTagoX6RGG5E7d6/6oOJ8TVty...
    ```

The password is encrypted using the password storage scheme set in the **nsslapd-rootpwstoragescheme** attribute of the Directory Server instance configuration.

2. Using a STARTTLS connection, set the **nsslapd-rootpw** attribute to the value displayed in the previous step:

```
# dsconf -D "cn=Directory Manager" ldaps://server.example.com config replace nsslapd-rootpw="{PBKDF2_SHA256}AAAgAMwPYlhEkQozTagoX6RGG5E7d6/6oOJ8TVty..."
```

Additional resources

- Changing the Directory Manager password using the web console

3.2. CHANGING THE DIRECTORY MANAGER PASSWORD USING THE WEB CONSOLE

You can set a new password for the Directory Manager using the web console.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Open the **Server → Server Settings → Directory Manager** menu.
2. Enter the new password into the **Directory Manager Password** and **Confirm Password** fields.
3. Optional: Set a different password storage scheme.
4. Click **Save**.

CHAPTER 4. RESETTING THE DIRECTORY MANAGER PASSWORD

The Directory Manager is the privileged database administrator, comparable to the **root** user in a Linux operating system. The Directory Manager password is set during the instance installation. If you lose the password, you can reset it to regain privileged access to the directory.

4.1. RESETTING THE DIRECTORY MANAGER PASSWORD USING THE COMMAND LINE

If you have the root access to the Directory Server instance, you can reset the password of the Directory Manager.

Procedure

1. Generate a new password hash. For example:

```
# pwdhash -D /etc/dirsrv/slapd-instance_name new_password
{PBKDF2_SHA256}AAAgABU0bKhyjY53NcxY33ueoPjOUWtl4iyYN5uW...
```

   Because you specified the path to the Directory Server instance configuration, the **pwdhash** generator automatically uses the password storage scheme set in the **nsslapd-rootpwstoragescheme** attribute to encrypt the new password.

2. Stop the Directory Server instance:

```
# dsctl instance_name stop
```

3. Edit the **/etc/dirsrv/slapd-*instance_name*/dse.ldif** file and set the **nsslapd-rootpw** attribute to the value displayed in the first step:

```
nsslapd-rootpw: {PBKDF2_SHA256}AAAgABU0bKhyjY53NcxY33ueoPjOUWtl4iyYN5uW...
```

4. Start the Directory Server instance:

```
# dsctl instance_name start
```

CHAPTER 5. CONFIGURING PASSWORD POLICIES

A password policy minimizes the risks associated with using passwords by enforcing a certain level of security. For example, you can define a password policy to ensure that:

- Users must change their passwords according to a schedule
- Users must provide non-trivial passwords
- The password syntax must meet certain complexity requirements

5.1. HOW PASSWORD POLICIES WORK

Directory Server supports fine-grained password policies, which work in an inverted pyramid, from general to specific. A global password policy is superseded by a subtree-level password policy, which is superseded by a user-level password policy.

You can define:

- Global password policy, applied to the entire directory
- Local password policy
  - Subtree-level policy, applied to a particular subtree
  - User-level policy, applied to a particular user

Password policies are not additive: only one password policy applies to an entry. For example, when you configure a particular attribute in the global or subtree-level password policy, but not in the user-level password policy, this attribute does not apply to the user. In this case, when the user attempts to log in, only the user-level policy is active.

WARNING

When using a password administrator account or the Directory Manager (root DN) to set a password, you bypass the password policies. Do not use these accounts for regular user password management. Use them only to perform password administration tasks that require bypassing the password policies, such as adding a prehashed password, or purposefully overriding current password constraints for setting temporary passwords after a reset.

The complete password policy that applies to a user account consists of the following elements:

- **The type or level of password policy checks.**This information indicates whether the server should check for and enforce a global password policy or local password policies.
- **Password add and modify information.**The password information includes password syntax and password history details.
- **Bind information.** The bind information includes the number of grace logins permitted, password aging attributes, and tracking bind failures.

NOTE

After establishing a password policy, you can protect user passwords from potential threats by configuring an account lockout policy. Account lockout protects against attempts to break into the directory by repeatedly guessing a user's password.

Additional resources

- Configuring a password-based account lockout policy
- Configuring time-based account lockout policies

5.2. CONFIGURING THE GLOBAL PASSWORD POLICY USING THE COMMAND LINE

By default, global password policy settings are disabled. You can configure the global password policy using the **dsconf** command line utility.

Procedure

1. Display the current settings:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy get
Global Password Policy: cn=config
----------------------------------
passwordstoragescheme: PBKDF2_SHA256
passwordChange: on
passwordMustChange: off
passwordHistory: off
passwordInHistory: 6
...
```

2. Adjust the password policy settings. For a full list of available settings, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --help
```

For example, to enable password syntax checking and set the minimum length of passwords to **12** characters, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --
pwdchecksyntax on --pwdmintokenlen 12
```

3. Enable the the account lockout feature for the password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --
pwdlockout on
```

5.3. CONFIGURING THE GLOBAL PASSWORD POLICY USING THE WEB CONSOLE

By default, global password policy settings are disabled. You can configure the global password policy using the web console.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Open the **Database → Password Policies → Global Policy** menu.
2. Set the global password policy settings. You can set parameters in the following categories:
   - General settings, such as the password storage scheme
   - Password expiration settings, such as the time when a password expires
   - Account lockout settings, such as after how many failed login attempts an account should be locked
   - Password syntax settings, such as the minimum password length
     To display a tool tip and the corresponding attribute name in the **cn=config** entry for a parameter, hover the mouse cursor over the setting.
3. Click **Save**.

5.4. LOCAL PASSWORD POLICY ENTRIES

When you use the **dsconf localpwp addsubtree** or **dsconf localpwp adduser** commands, Directory Server automatically creates an entry to store the local password policy attributes.

For a subtree, the following entries are added:

Table 5.1. Local password policy entries for a subtree

Entry name	Description	Contents
nsPwPolicyContainer	A container entry at the subtree level	Various password policy-related entries for the subtree and all its children
nsPwPolicyEntry	The actual password policy specification entry	All the password policy attributes that are specific to the subtree
nsPwTemplateEntry	The CoS Template Entry	The **pwdpolicysubentry** value pointing to the **nsPwPolicyEntry** entry
<CoS definition entry DN>	The CoS definition entry at the subtree level	CoS definition entry

Example 5.1. The nsPwPolicyContainer entry for a subtree ou=people,dc=example,dc=com

```
dn: cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectClass: top
objectClass: nsContainer
```

```
cn: nsPwPolicyContainer
```

Example 5.2. The nsPwPolicyEntry entry for a subtree ou=people,dc=example,dc=com

```
dn: cn="cn=nsPwPolicyEntry,ou=people,dc=example,dc=com",
 cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: ldapsubentry
objectclass: passwordpolicy
```

Example 5.3. The nsPwTemplateEntry entry for a subtree ou=people,dc=example,dc=com

```
dn: cn="cn=nsPwTemplateEntry,ou=people,dc=example,dc=com",
 cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: costemplate
objectclass: ldapsubentry
cosPriority: 1
pwdpolicysubentry: cn="cn=nsPwPolicyEntry,ou=people,dc=example,dc=com",
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
```

Example 5.4. The CoS specification entry for a subtree ou=people,dc=example,dc=com

```
dn: cn=newpwdpolicy_cos,ou=people,dc=example,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=cn=nsPwTemplateEntry\,ou=people\,dc=example,dc=com,
 cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
cosAttribute: pwdpolicysubentry default operational
```

For a user, the following entries are added:

Table 5.2. Local password policy entries for a user

Entry name	Description	Contents
nsPwPolicyContainer	A container entry at the parent level	Various password policy-related entries for the user and all its children
nsPwPolicyEntry	The actual password policy specification entry	All the password policy attributes that are specific to the user

Example 5.5. The nsPwPolicyContainer entry for a user uid=user_name,ou=people,dc=example,dc=com

```
dn: cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectClass: top
objectClass: nsContainer
cn: nsPwPolicyContainer
```

Example 5.6. The nsPwPolicyEntry entry for a user uid=user_name,ou=people,dc=example,dc=com

```
dn: cn="cn=nsPwPolicyEntry,uid=user_name,ou=people,dc=example,dc=com",
 cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: ldapsubentry
objectclass: passwordpolicy
```

5.5. CONFIGURING A LOCAL PASSWORD POLICY USING THE COMMAND LINE

In contrast to a global password policy, which defines settings for the entire directory, a local password policy is a policy for a specific user or subtree. Currently, you can only set up a local password policy using the command line.

Prerequisites

- User or subtree entries that you want to create the policy for already exist in the directory.

Procedure

1. Verify if a local password policy already exists for the subtree or user entry. For example:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp get
"ou=People,dc=example,dc=com"
Enter password for cn=Directory Manager on ldap://server.example.com:
Error: No password policy was found for this entry
```

   If no local policy exists, create one:

   - To create a **subtree** password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp
addsubtree "ou=People,dc=example,dc=com"
```

   - To create a **user** password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp adduser
"uid=user_name,ou=People,dc=example,dc=com"
```

2. Set local policy attributes. For a full list of available settings, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp set --help
```

For example, to enable password expiration and set the maximum password age to 14 days (**1209600** seconds):

- On a **subtree** password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp set --
pwdexpire on --pwdmaxage 1209600 "ou=People,dc=example,dc=com"
```

- On a **user** password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp set --
pwdexpire on --pwdmaxage 1209600
"uid=user_name,ou=People,dc=example,dc=com"
```

5.6. DISABLING A LOCAL PASSWORD POLICY USING THE COMMAND LINE

When you create a new local policy, the **nsslapd-pwpolicy-local** parameter in the **cn=config** entry is automatically set to **on**. If the local password policy should not be enabled, you can disable it manually using the command line.

Procedure

- Disable all local policies or remove a particular local policy:
  - To disable all local password policies:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --
pwdlocal off
```

  - To remove a single existing **subtree** password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp remove
"ou=People,dc=example,dc=com"
```

  - To remove a single existing **user** password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp remove
"uid=user_name,ou=People,dc=example,dc=com"
```

5.7. TRACKING PASSWORD CHANGE TIME

Password change operation is a typical modify operation on an entry, and Directory Server records the update time to the **lastModified** operational attribute. However, to make it easier to update passwords in Active Directory synchronization or to connect with other LDAP clients, you can record the time of the last password change separately.

Configure the **passwordTrackUpdateTime** parameter within a global or local password policy to record the time of the last password change to the **pwdUpdateTime** operation attribute of the user entry.

Prerequisites

- You have **root** permissions.

Procedure

1. Set the **passwordTrackUpdateTime** parameter to **on**:
   - For the global policy, run:

     ```
     # dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwdtrack on
     ```

   - For the subtree or user-level policy, run:

     ```
     # dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp set "ou=people,dc=example,dc=com" --pwdtrack on
     ```

Verification

- Display the current settings of the policy:
  - For the global policy, run:

    ```
    # dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy get

    Global Password Policy: cn=config
    ---------------------------------
    nsslapd-pwpolicy-local: on
    passwordstoragescheme: PBKDF2-SHA512
    ...
    passwordtrackupdatetime: on
    ...
    ```

  - For the subtree or user-level policy, run:

    ```
    # dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp get "ou=people,dc=example,dc=com"

    Local Subtree Policy Policy for "ou=people,dc=example,dc=com":
    cn=cn\3DnsPwPolicyEntry_subtree\2Cou\3Dpeople\2Cdc\3Dexample\2Cdc\3Dcom,cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
    ---------------------------------
    ...
    passwordtrackupdatetime: on
    ```

Additional resources

- Configuring password policies
- passwordTrackUpdateTime

CHAPTER 6. CONFIGURING TEMPORARY PASSWORD RULES

Directory Server password policies support setting temporary passwords on user accounts. If you assign a temporary password to a user, Directory Server rejects any other operation than a password change until the user changes its password.

The following are the features of temporary passwords:

- Only the **cn=Directory Manager** account can assign temporary passwords.
- Directory Server allows authentication attempts only for a fixed number of times to avoid that an attacker probes the password.
- Directory Server allows authentication attempts after a specified delay to configure that the temporary passwords are not usable directly after you set them.
- Directory Server allows authentication attempts only for a specified time so that the temporary password expires if a user does not use or reset it.
- If the authentication was successful, Directory Server requires that the user resets the password before the server performs any other operation.

By default, temporary password rules are disabled. You can configure them in global or local password policies.

6.1. ENABLING TEMPORARY PASSWORD RULES IN THE GLOBAL PASSWORD POLICY

To enable the temporary password feature for the whole Directory Server instance:

1. Enable that users must change their password if an administrator resets it.
2. Configure the feature in the global password policy.

If an administrator updates the **userPassword** attribute of a user and sets the **passwordMustChange** attribute to **on**, Directory Server applies the temporary password rules.

Procedure

1. Configure that a user must change its password after an administrator resets it:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwdmustchange on
```

2. Configure the temporary password rules settings in a global password policy:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwptprmaxuse 5 --pwptprdelayexpireat 3600 --pwptprdelayvalidfrom 60
```

In this example:

- The **--pwptprmaxuse** option sets the maximum number of attempts a user can use the temporary password to **5**.

- The **--pwptprdelayexpireat** option sets the time before the temporary password expires to **3600** seconds (1 hour).
- The **--pwptprdelayvalidfrom** option configures that the time set in **--pwptprdelayexpireat** starts **60** seconds after an administrator reset the password of a user.

Verification

- Display the attributes that store the temporary password rules:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy get | grep -i TPR
passwordTPRMaxUse: 5
passwordTPRDelayExpireAt: 3600
passwordTPRDelayValidFrom: 60
```

6.2. ENABLING TEMPORARY PASSWORD RULES IN A LOCAL PASSWORD POLICY

To enable the temporary password feature for a specific user or sub-tree, enable that users must change their password if an administrator resets it, and configure the feature in a local password policy.

If an administrator updates the **userPassword** attribute of a user and sets the **passwordMustChange** attribute to **on**, Directory Server applies the temporary password rules if the user:

- Has the local password policy enabled
- Is stored in a sub-tree that has the local password policy enabled

Procedure

1. Configure that a user must change its password after an administrator resets it:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwdmustchange on
```

2. Configure the temporary password rules settings:
   - For an existing sub-tree:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp addsubtree --pwptprmaxuse 5 --pwptprdelayexpireat 3600 --pwptprdelayvalidfrom 60 ou=People,dc=example,dc=com
```

   - For an existing user:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp adduser --pwptprmaxuse 5 --pwptprdelayexpireat 3600 --pwptprdelayvalidfrom 60 uid=example,ou=People,dc=example,dc=com
```

   In these examples:

   - The **--pwptprmaxuse** option sets the maximum number of attempts a user can use the temporary password to **5**.

- The **--pwptprdelayexpireat** option sets the time before the temporary password expires to **3600** seconds (1 hour).
- The **--pwptprdelayvalidfrom** option configures that the time set in **--pwptprdelayexpireat** starts **60** seconds after an administrator reset the password of a user.

Verification

- Display the local password policy of the distinguished name (DN):

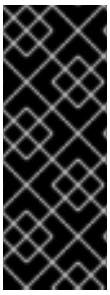
```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp get <DN> |
grep -i TPR
passwordTPRMaxUse: 5
passwordTPRDelayExpireAt: 3600
passwordTPRDelayValidFrom: 60
```

CHAPTER 7. ASSIGNING PASSWORD ADMINISTRATOR PERMISSIONS

The Directory Manager can assign the *password administrator* role to a user or a group of users. Because password administrators need access control instructions (ACIs) with the appropriate permissions, Red Hat recommends that you configure a group to allow a single ACI set to manage all password administrators.

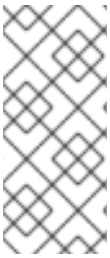
Using the password administrator role is beneficial in the following scenarios:

- setting up an atribute that forces the user to change their password at the time of the next login
- changing a user's password to a different storage scheme defined in the password policy



IMPORTANT

A password administrator can perform any user password operations. When using a password administrator account or the Directory Manager (root DN) to set a password, password policies are bypassed and not verified. Do not use these accounts for regular user password management. Red Hat recommends performing ordinary password updates under an existing role in the database with permissions to update only the **userPassword** attribute.



NOTE

You can add a new **passwordAdminSkipInfoUpdate: on**/**off** setting under the **cn=config** entry to provide a fine grained control over password updates performed by password administrators. When you enable this setting, passwords updates do not update certain attributes, for example, **passwordHistory**, **passwordExpirationTime**, **passwordRetryCount**, **pwdReset**, and **passwordExpWarned**.

7.1. ASSIGNING PASSWORD ADMINISTRATOR PERMISSIONS IN A GLOBAL POLICY

In a global policy, you can assign the password administrator role to a user or a group of users. Red Hat recommends that you configure a group to allow a single access control instruction (ACI) set to manage all password administrators.

Prerequisites

- You have created a group named **password_admins** that includes all of the users to whom you want to assign the password administrator role.

Procedure

1. Create the ACI that defines the permissions for a password administrator role:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x << EOF
dn: ou=people,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="userPassword || nsAccountLock || userCertificate || nsSshPublicKey")
```

```
(targetfilter="(objectClass=nsAccount)")(version 3.0; acl "Enable user password reset"; allow
(write, read)(groupdn="ldap:///cn=password_admins,ou=groups,dc=example,dc=com");)
EOF
```

2. Assign the password administrator role to the group:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --
pwdadmin "cn=password_admins,ou=groups,dc=example,dc=com"
```

7.2. ASSIGNING PASSWORD ADMINISTRATOR PERMISSIONS IN A LOCAL POLICY

In a local policy, you can assign the password administrator role to a user or a group of users. Red Hat recommends that you configure a group to allow a single access control instruction (ACI) set to manage all password administrators.

Prerequisites

- You have created a group named **password_admins** that includes all of the users to whom you want to assign the password administrator role.

Procedure

1. Create the ACI that defines the permissions for a password administrator role:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x << EOF
dn: ou=people,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="userPassword || nsAccountLock || userCertificate || nsSshPublicKey")
(targetfilter="(objectClass=nsAccount)")(version 3.0; acl "Enable user password reset"; allow
(write, read)(groupdn="ldap:///cn=password_admins,ou=groups,dc=example,dc=com");)
EOF
```

2. Assign the password administrator role to the group:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com localpwp set
ou=people,dc=example,dc=com --pwdadmin
"cn=password_admins,ou=groups,dc=example,dc=com"
```

7.3. ADDITIONAL RESOURCES

- Backup up Directory Server (change it)

CHAPTER 8. DISABLING ANONYMOUS BINDS

If a user attempts to connect to Directory Server without supplying any credentials, this operation is called **anonymous bind**. Anonymous binds simplify searches and read operations, such as finding a phone number in the directory by not requiring users to authenticate first. However, anonymous binds can also be a security risk, because users without an account are able to access the data.

WARNING

By default, anonymous binds are enabled in Directory Server for search and read operations. This allows unauthorized access to user entries as well as configuration entries, such as the root directory server entry (DSE).

8.1. DISABLING ANONYMOUS BINDS USING THE COMMAND LINE

To increase the security, you can disable anonymous binds.

Procedure

- Set the **nsslapd-allow-anonymous-access** configuration parameter to **off**:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com config replace
nsslapd-allow-anonymous-access=off
```

Verification

- Run a search without specifying a user account:

```
# ldapsearch -H ldap://server.example.com -b "dc=example,dc=com" -x
ldap_bind: Inappropriate authentication (48)
 additional info: Anonymous access is not allowed
```

8.2. DISABLING ANONYMOUS BINDS USING THE WEB CONSOLE

To increase the security, you can disable anonymous binds.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Navigate to **Server → Server Settings → Advanced Settings**.
2. Set the **Allow Anonymous Access** parameter to **off**.
3. Click **Save**.

Verification

- Run a search without specifying a user account:

```
# ldapsearch -H ldap://server.example.com -b "dc=example,dc=com" -x
ldap_bind: Inappropriate authentication (48)
        additional info: Anonymous access is not allowed
```

CHAPTER 9. MANUALLY INACTIVATING USERS AND ROLES

manually-inactivating-users-and-roles

In Directory Server, you can temporarily inactivate a single user account or a set of accounts. Once an account is inactivated, a user cannot bind to the directory. The authentication operation fails.

9.1. INACTIVATION AND ACTIVATION OF USERS AND ROLES USING THE COMMAND LINE

You can manually inactivate users and roles using the command line or the operational attribute.

Roles behave as both a static and a dynamic group. With a group, entries are added to a group entry as members. With a role, the role attribute is added to an entry and then that attribute is used to identify members in the role entry automatically.

Users and roles are inactivated executing the same procedures. However, when a role is inactivated, the members of the role are inactivated, not the role entry itself.

To inactivate users and roles, execute the following commands in the command line:

- For inactivation of a user account:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b
"dc=example,dc=com" account lock "uid=user_name,ou=People,dc=example,dc=com"
```

- For inactivation of a role:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b
"dc=example,dc=com" role lock "cn=Marketing,ou=People,dc=example,dc=com"
```

To activate users and roles, execute the following commands in the command line:

- For activation of a user account:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b
"dc=example,dc=com" account unlock
"uid=user_name,ou=People,dc=example,dc=com"
```

- For activation of a role:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b
"dc=example,dc=com" role unlock "cn=Marketing,ou=People,dc=example,dc=com"
```

Optionally, instead of using the commands, you can add the operational attribute **nsAccountLock** to the entry. When an entry contains the **nsAccountLock** attribute with a value of **true**, the server rejects the bind.

Additional resources

- Using roles in Directory Server
- Managing directory attributes and values

- Configuring directory databases

9.2. COMMANDS FOR DISPLAYING THE STATUS OF AN ACCOUNT OR A ROLE

You can display the status of an account or a role in Directory Server using the corresponding commands in the command line.

Commands for displaying the status

- Display the status of an account:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b
"dc=example,dc=com" account entry-status
"uid=user_name,ou=People,dc=example,dc=com"
Entry DN: uid=user_name,ou=People,dc=example,dc=com
Entry Creation Date: 20210813085535Z (2021-08-13 08:55:35)
Entry Modification Date: 20210813085535Z (2021-08-13 08:55:35)
Entry State: activated
```

Optional: The **-V** option displays additional details.

Example 9.1. Detailed output for an active account

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b
"dc=example,dc=com" account entry-status
"uid=user_name,ou=People,dc=example,dc=com" -V
Entry DN: uid=user_name,ou=People,dc=example,dc=com
Entry Creation Date: 20210824160645Z (2021-08-24 16:06:45)
Entry Modification Date: 20210824160645Z (2021-08-24 16:06:45)
Entry Last Login Date: 20210824160645Z (2021-08-24 16:06:45)
Entry Time Until Inactive: 2 seconds (2021-08-24 16:07:45)
Entry State: activated
```

Example 9.2. Detailed output for an inactive account

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b
"dc=example,dc=com" account entry-status
"uid=user_name,ou=People,dc=example,dc=com" -V
Entry DN: uid=user_name,ou=People,dc=example,dc=com
Entry Creation Date: 20210824160645Z (2021-08-24 16:06:45)
Entry Modification Date: 20210824160645Z (2021-08-24 16:06:45)
Entry Last Login Date: 20210824160645Z (2021-08-24 16:06:45)
Entry Time Since Inactive: 3 seconds (2021-08-24 16:07:45)
Entry State: inactivity limit exceeded
```

- Display the status of a role:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b
"dc=example,dc=com" role entry-status
```

```
"cn=Marketing,ou=People,dc=example,dc=com"
Entry DN: cn=Marketing,ou=people,dc=example,dc=com
Entry State: activated
```

- Display the status of a sub-tree:

```
# dsidm -D "cn=Directory Manager" ldap://server.example.com -b
"dc=example,dc=com" account subtree-status "ou=People,dc=example,dc=com" -f
"(uid=*)" -V -o "2021-08-25T14:30:30"
```

  To filter the results of the search in a sub-tree, you can use:

  - The **-f** option to set the search filter
  - The **-s** option to set the search scope
  - The **-i** option to return only inactive accounts
  - The **-o** option to return only accounts which will be inactive before the specified date **YYYY-MM-DDTHH:MM:SS**

CHAPTER 10. SYNCHRONIZING ACCOUNT LOCKOUT ATTRIBUTES ACROSS ALL SERVERS IN A REPLICATION ENVIRONMENT

Directory Server stores account lockout attributes locally. In an environment with multiple servers, configure replication for these attributes to prevent attackers from attempting to log in to one server until the account lockout count is reached and then continue on other servers.

10.1. HOW DIRECTORY SERVER HANDLES PASSWORD AND ACCOUNT LOCKOUT POLICIES IN A REPLICATION ENVIRONMENT

Directory Server enforces password and account lockout policies as follows:

- Password policies are enforced on the data supplier
- Account lockout policies are enforced on all servers in a replication topology

Directory Server replicates the following password policy attributes:

- **passwordMinAge**
- **passwordMaxAge**
- **passwordExp**
- **passwordWarning**

However, by default, Directory Server does not replicate the general account lockout attributes:

- **passwordRetryCount**
- **retryCountResetTime**
- **accountUnlockTime**

To prevent attackers from attempting to log in to one server until the account lockout count is reached and then continue on other servers, replicate these account lockout attributes.

Additional resources

- Configuring Directory Server to replicate account lockout attributes

10.2. CONFIGURING DIRECTORY SERVER TO REPLICATE ACCOUNT LOCKOUT ATTRIBUTES

If you use an account lockout policy or password policy that updates the **passwordRetryCount**, **retryCountResetTime**, or **accountUnlockTime** attributes, configure Directory Server to replicate these attributes so that their values are the same across all servers.

Perform this procedure on all suppliers in the replication topology.

Prerequisites

- You configured an account lockout policy or a password policy that updates one or more of the mentioned attributes.
- You use Directory Server in a replication environment.

Procedure

1. Enable replication of password policy attributes:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com pwpolicy set --pwdisglobal="on"
```

2. If you use fractional replication, display the list of attributes that are excluded from replication:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt get --suffix "dc=example,dc=com" example-agreement | grep "nsDS5ReplicatedAttributeList"
```

   Using the default settings, no output is shown, and Directory Server replicates the account lockout attributes. However, if the command returns a list of excluded attributes, such as in the following example, verify the attribute list:

```
nsDS5ReplicatedAttributeList: (objectclass=*) $ EXCLUDE accountUnlockTime passwordRetryCount retryCountResetTime example1 example2
```

   In this example, the **accountUnlockTime**, **passwordRetryCount**, and **retryCountResetTime** lockout policy attributes are excluded from replication, along with two other attributes.

3. If the output of the previous command lists any of the account lockout attributes, update the fractional replication settings to only include attributes other than the lockout policy attributes:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com repl-agmt set --suffix "dc=example,dc=com" --frac-list "example1 example2" example-agreement
```

Verification

1. Attempt to perform a search as a user using an invalid password:

```
# ldapsearch -H ldap://server.example.com -D "uid=example,ou=People,dc=example,dc=com" -w "invalid-password" -b "dc=example,dc=com" -x
ldap_bind: Invalid credentials (49)
```

2. Display the **passwordRetryCount** attribute of the user:

```
# ldapsearch -H ldap://server.example.com -D "cn=Directory Manager" -W -b "uid=example,ou=People,dc=example,dc=com" -x passwordRetryCount
...
dn: uid=example,ou=People,dc=example,dc=com
passwordRetryCount: 1
```

3. Run the previous command on a different server in the replication topology. If the value of the **passwordRetryCount** attribute is the same, Directory Server replicated the attribute.

Additional resources

Additional resources

- Configuring a password-based account lockout policy

CHAPTER 11. USING REFERENTIAL INTEGRITY TO MAINTAIN RELATIONSHIPS BETWEEN ENTRIES

Referential Integrity is a database mechanism that ensures that Directory Server maintains relationship between related entries. You can use this feature to ensure that an update to one entry in the directory is correctly reflected in other entries that reference the updated entry.

For example, if you remove a user from the directory and the Referential Integrity plug-in is enabled, the server also removes the user from any group in which the user is a member. If the plug-in is not enabled, the user remains a member of the group until an administrator manually removes it.

Referential Integrity is an important feature if you integrate Directory Server with other products that rely on Directory Server for user and group management.

11.1. HOW THE REFERENTIAL INTEGRITY PLUG-IN WORKS

When you enable the Referential Integrity plug-in, it performs integrity updates on the **member**, **uniqueMember**, **owner**, and **seeAlso** attributes, by default, immediately after an operation.

For example, if an administrator deletes, updates, renames, or moves a group or user within the directory, Directory Server logs the operation in the Referential Integrity log file. Directory Server then uses the distinguished name (DN) from this log file and searches entries matching the attribute specified in the plug-in's configuration, and then updates the matching entries. For example, after deleting the **cn=demo,dc=example,dc=com** entry the plug-in searches for entries with the **member** attribute set to **cn=demo,dc=example,dc=com** and removes these **member** attributes. Afterwards, the plug-in does the same for the **uniqueMember**, **owner**, and **seeAlso** attributes.

By default, Directory Server does searches and updates in the same transaction as the original operation. Because search and update operations can take a lot of time, it is possible to delay them after the completion of the original operation. You can use the **--update-delay** option of the **dsconf plugin referential-integrity set** command to separate the original operations from integrity updates.

To avoid poor performance of modify and delete operations, index the attributes you specify in the Referential Integrity plug-in configuration.

Additional resources

- Managing indexes

11.2. CONFIGURING THE REFERENTIAL INTEGRITY PLUG-IN USING THE COMMAND LINE

You can use the command line to configure the Referential Integrity plug-in.

Perform this procedure on every supplier in a replication topology.

Procedure

1. Enable the Referential Integrity plug-in:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity enable
```

2. Set the subtree in which the plug-in searches for delete or rename operations of user entries:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --entry-scope "ou=People,dc=example,dc=com"
```

3. Optional: Exclude a subtree under the entry scope:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --exclude-entry-scope "ou=Special Users,ou=People,dc=example,dc=com"
```

This command configures the plug-in to ignore delete or rename operations performed in the **ou=Special Users,ou=People,dc=example,dc=com** subtree.

4. Configure the subtree in which the plug-in updates group entries:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --container-scope "ou=Groups,dc=example,dc=com"
```

5. By default, the plug-in performs integrity updates on the **member**, **uniqueMember**, **owner**, and **seeAlso** attributes. To specify other attributes, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --membership-attr attribute_1 attribute_2
```

Note that this command overrides the list of attributes in the plug-in's configuration. If you want to add an attribute, pass the current list of attributes and the additional one to the **--membership-attr** option.

6. Optional: By default, Directory Server performs referential integrity checks immediately. If you want to set a delay, enter:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity set --update-delay=5
```

This command delays the referential integrity checks by **5** seconds. Note that, if you enabled the Referential Integrity on multiple suppliers, setting a delay can cause replication loops and directory inconsistencies. To avoid such problems, enable the plug-in only on one supplier in the topology.

7. Restart the instance:

```
# dsctl instance_name restart
```

Verification

1. Display the Referential Integrity plug-in configuration:

```
# dsconf -D "cn=Directory Manager" ldap://server.example.com plugin referential-integrity show
...
nsslapd-plugincontainerscope: ou=Groups,dc=example,dc=com
nsslapd-pluginentryscope: ou=People,dc=example,dc=com
...
```

```
referint-membership-attr: member
referint-membership-attr: uniquemember
referint-membership-attr: owner
referint-membership-attr: seeAlso
referint-update-delay: 0
...
```

2. List the members of a group by displaying the **member** attributes of the groups:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -b
"cn=demoGroup,ou=Groups,dc=example,dc=com" member
...
member: uid=demoUser,ou=People,dc=example,dc=com
```

3. Delete the **uid=demoUser,ou=People,dc=example,dc=com** user:

```
# dsidm -D "cn=Directory manager" ldap://server.example.com -b
"dc=example,dc=com" user delete "uid=demoUser,ou=People,dc=example,dc=com"
```

4. Display the members of the group again:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -b
"cn=demoGroup,ou=People,dc=example,dc=com" member
```

If **uid=demoUser,ou=People,dc=example,dc=com** is no longer listed as a member of the group, the Referential Integrity plug-in works.

11.3. CONFIGURING THE REFERENTIAL INTEGRITY PLUG-IN USING THE WEB CONSOLE

You can use the Directory Server web console to configure the Referential Integrity plug-in.

Perform this procedure on every supplier in a replication topology.

Prerequisites

- You are logged in to the instance in the web console.

Procedure

1. Navigate to **Plugins → Referential Integrity**.
2. Enable the plug-in.
3. Click **Actions → Restart Instance**.
4. Navigate again to **Plugins → Referential Integrity**.
5. By default, the plug-in performs integrity updates on the **member**, **uniqueMember**, **owner**, and **seeAlso** attributes. To specify other attributes, update the list in the **Membership Attribute** field.
6. Set the **Entry Scope** field to the DN of the subtree in which the plug-in should search for delete or rename operations of user entries.

7. Optional: To exclude a subtree under the entry scope, enter the DN of the subtree in the **Exclude Entry Scope** field.

8. Set the **Container Scope** field to the DN of the subtree in which the plug-in should update group entries.

9. Optional: Update the path to the Referential Integrity log file. Directory Server uses this file to track changes in the directory. Note that the **dirsrv** user must have write permissions to this location.

10. Optional: By default, Directory Server performs referential integrity checks immediately. If you want to set a delay, set it in the **Update Delay** field.
    Note that, if you enabled the Referential Integrity on multiple suppliers, setting a delay can cause replication loops and directory inconsistencies. To avoid such problems, enable the plug-in only on one supplier in the topology.

11. Click **Save Config**.

Verification

1. List the members of a group by displaying the **member** attributes of the groups:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -b
"cn=demoGroup,ou=Groups,dc=example,dc=com" member
...
member: uid=demoUser,ou=People,dc=example,dc=com
```

2. Delete the **uid=demoUser,ou=People,dc=example,dc=com** user:

```
# dsidm -D "cn=Directory manager" ldap://server.example.com -b
"dc=example,dc=com" user delete "uid=demoUser,ou=People,dc=example,dc=com"
```

3. Display the members of the group again:

```
# ldapsearch -D "cn=Directory Manager" -W -H ldap://server.example.com -b
"cn=demoGroup,ou=People,dc=example,dc=com" member
```

If **uid=demoUser,ou=People,dc=example,dc=com** is no longer listed as a member of the group, the Referential Integrity plug-in works.