Table of Contents ⊞

8.5. Integration with LDAP Enterprise Edition

This section describes Neo4j support for integrating with LDAP systems.

This section describes the following:

- Introduction
- Configure the LDAP auth provider
  - Configuration for Active Directory
  - Configuration for openLDAP
- Use 'ldapsearch' to verify the configuration
- The auth cache
- Available methods of encryption
  - Use LDAP with encryption via StartTLS
  - Use LDAP with encrypted LDAPS
- Use a self-signed certificate in a test environment

§ 8.5.1. Introduction

Neo4j supports the LDAP protocol which allows for integration with Active Directory, OpenLDAP or other LDAP-compatible authentication services. We will show example configurations where management of federated users is deferred to the LDAP service, using that service's facilities for administration. This means that we completely turn off native Neo4j user and role administration and map LDAP groups to the Neo4j native roles (../native-user-role-management/native-roles/), and to custom roles.

8.5.2. Configure the LDAP auth provider

All settings need to be defined at server startup time in the default configuration file *neo4j.conf*. First configure Neo4j to use LDAP as authentication and authorization provider.

```
# Turn on security:
dbms.security.auth_enabled=true

# Choose LDAP connector as security provider for both authentication and authorization:
dbms.security.auth_provider=ldap
```

8.5.2.1. Configuration for Active Directory

See below for an example configuration for Active Directory:

```
# Configure LDAP to point to the AD server:
dbms.security.ldap.host=ldap://myactivedirectory.example.com

# Provide details on user structure within the LDAP system:
dbms.security.ldap.authentication.user_dn_template=cn={0},cn=Users,dc=example,dc=com
dbms.security.ldap.authorization.user_search_base=cn=Users,dc=example,dc=com
dbms.security.ldap.authorization.user_search_filter=(&(objectClass=*)(cn={0}))
dbms.security.ldap.authorization.group_membership_attributes=memberOf

# Configure the actual mapping between groups in the LDAP system and roles in Neo4j:
dbms.security.ldap.authorization.group_to_role_mapping=\
  "cn=Neo4j Read Only,cn=Users,dc=neo4j,dc=com"      = reader       ;\
  "cn=Neo4j Read-Write,cn=Users,dc=neo4j,dc=com"     = publisher    ;\
  "cn=Neo4j Schema Manager,cn=Users,dc=neo4j,dc=com" = architect    ;\
  "cn=Neo4j Administrator,cn=Users,dc=neo4j,dc=com"  = admin        ;\
  "cn=Neo4j Procedures,cn=Users,dc=neo4j,dc=com"     = allowed_role

# In case defined users are not allowed to search for themselves, we can specify credentials for a user with read access to all users and groups.
# Note that this account only needs read-only access to the relevant parts of the LDAP directory and does not need to have access rights to Neo4j
or any other systems.
# dbms.security.ldap.authorization.use_system_account=true
# dbms.security.ldap.authorization.system_username=cn=search-account,cn=Users,dc=example,dc=com
# dbms.security.ldap.authorization.system_password=secret
```

Below is an alternative configuration for Active Directory that allows for logging in with `sAMAccountName`:

```
# Configure LDAP to point to the AD server:
dbms.security.ldap.host=ldap://myactivedirectory.example.com

# Provide details on user structure within the LDAP system:
dbms.security.ldap.authorization.user_search_base=cn=Users,dc=example,dc=com
dbms.security.ldap.authorization.user_search_filter=(&(objectClass=*)(samaccountname={0}))
dbms.security.ldap.authorization.group_membership_attributes=memberOf

# Configure the actual mapping between groups in the LDAP system and roles in Neo4j:
dbms.security.ldap.authorization.group_to_role_mapping=\
  "cn=Neo4j Read Only,cn=Users,dc=neo4j,dc=com"      = reader       ;\
  "cn=Neo4j Read-Write,cn=Users,dc=neo4j,dc=com"     = publisher    ;\
  "cn=Neo4j Schema Manager,cn=Users,dc=neo4j,dc=com" = architect    ;\
  "cn=Neo4j Administrator,cn=Users,dc=neo4j,dc=com"  = admin        ;\
  "cn=Neo4j Procedures,cn=Users,dc=neo4j,dc=com"     = allowed_role

# In case defined users are not allowed to search for themselves, we can specify credentials for a user with read access to all users and groups.
# Note that this account only needs read-only access to the relevant parts of the LDAP directory and does not need to have access rights to Neo4j
or any other systems.
dbms.security.ldap.authorization.use_system_account=true
dbms.security.ldap.authorization.system_username=cn=search-account,cn=Users,dc=example,dc=com
dbms.security.ldap.authorization.system_password=secret

# Perform authentication with sAMAccountName instead of DN.
# Using this setting requires dbms.security.ldap.authorization.system_username and dbms.security.ldap.authorization.system_password to be used,
since there is no way to log in through LDAP directly with the sAMAccountName.
# Instead, the login name will be resolved to a DN that will be used to log in with.
dbms.security.ldap.authentication.use_samaccountname=true
```

Below is an alternative configuration for Active Directory that allows for authenticating users from different organizational units by using the Active Directory attribute `sAMAccountName`:

```
# Configure LDAP to point to the AD server:
dbms.security.ldap.host=ldap://myactivedirectory.example.com

dbms.security.ldap.authentication.user_dn_template={0}@example.com
dbms.security.ldap.authorization.user_search_base=dc=example,dc=com
dbms.security.ldap.authorization.user_search_filter=(&(objectClass=user)(sAMAccountName={0}))
dbms.security.ldap.authorization.group_membership_attributes=memberOf

# Configure the actual mapping between groups in the LDAP system and roles in Neo4j:
dbms.security.ldap.authorization.group_to_role_mapping=\
"cn=Neo4j Read Only,cn=Users,dc=example,dc=com" = reader ;\
"cn=Neo4j Read-Write,cn=Users,dc=example,dc=com" = publisher ;\
"cn=Neo4j Schema Manager,cn=Users,dc=example,dc=com" = architect ;\
"cn=Neo4j Administrator,cn=Users,dc=example,dc=com" = admin ;\
"cn=Neo4j Procedures,cn=Users,dc=example,dc=com" = allowed_role
```

Specifying the `{0}@example.com` pattern in the `user_dn_template` enables the authentication to start at the root domain. The whole tree is checked to find the user, regardless of where it is located within the tree.

Note that the setting `dbms.security.ldap.authentication.use_samaccountname` is not configured in this example.

8.5.2.2. Configuration for openLDAP

See below for an example configuration for openLDAP:

```
# Configure LDAP to point to the OpenLDAP server:
dbms.security.ldap.host=myopenldap.example.com

# Provide details on user structure within the LDAP system:
dbms.security.ldap.authentication.user_dn_template=cn={0},ou=users,dc=example,dc=com
dbms.security.ldap.authorization.user_search_base=ou=users,dc=example,dc=com
dbms.security.ldap.authorization.user_search_filter=(&(objectClass=*)(uid={0}))
dbms.security.ldap.authorization.group_membership_attributes=gidnumber

# Configure the actual mapping between groups in the OpenLDAP system and roles in Neo4j:
dbms.security.ldap.authorization.group_to_role_mapping=\
  101 = reader       ;\
  102 = publisher    ;\
  103 = architect    ;\
  104 = admin        ;\
  105 = allowed_role

# In case defined users are not allowed to search for themselves, we can specify credentials for a user with read access to all users and groups.
# Note that this account only needs read-only access to the relevant parts of the LDAP directory and does not need to have access rights to Neo4j
or any other systems.
# dbms.security.ldap.authorization.use_system_account=true
# dbms.security.ldap.authorization.system_username=cn=search-account,ou=users,dc=example,dc=com
# dbms.security.ldap.authorization.system_password=search-account-password
```

We would like to draw attention to some details in the configuration examples. A comprehensive overview of LDAP configuration options is available in Section A.1, “Configuration settings” (../../reference/configuration-settings/).

Parameter name	Default value	Description
dbms.security.ldap.authentication.user_dn_template (../../reference/configuration-settings/#config_dbms.security.ldap.authentication.user_dn_template)	`uid={0},ou=users,dc=example,dc=com`	Converts usernames into LDAP-specific fully qualified names required for logging in.
dbms.security.ldap.authorization.user_search_base (../../reference/configuration-settings/#config_dbms.security.ldap.authorization.user_search_base)	`ou=users,dc=example,dc=com`	Sets the base object or named context to search for user objects.
dbms.security.ldap.authorization.user_search_filter (../../reference/configuration-settings/#config_dbms.security.ldap.authorization.user_search_filter)	`(&(objectClass=*)(uid={0}))`	Sets up an LDAP search filter to search for a user principal.
dbms.security.ldap.authorization.group_membership_attributes (../../reference/configuration-settings/#config_dbms.security.ldap.authorization.group_membership_attributes)	`[memberOf]`	Lists attribute names on a user object that contains groups to be used for mapping to roles.

Parameter name	Default value	Description
dbms.security.ldap.authorization.group_to_role_mapping (../../reference/configuration-settings/#config_dbms.security.ldap.authorization.group_to_role_mapping)		Lists an authorization mapping from groups to the pre-defined built-in roles `admin`, `architect`, `publisher` and `reader`, or to any other custom-defined roles.

8.5.3. Use 'ldapsearch' to verify the configuration

We can use the LDAP command-line tool `ldapsearch` to verify that the configuration is correct, and that the LDAP server is actually responding. We do this by issuing a search command that includes LDAP configuration setting values.

These example searches verify both the authentication (using the `simple` mechanism) and authorization of user 'john'. See the `ldapsearch` documentation for more advanced usage and how to use SASL authentication mechanisms.

With `dbms.security.ldap.authorization.use_system_account=false` (default):

```
#ldapsearch -v -H ldap://<dbms.security.ldap.host> -x -D <dbms.security.ldap.authentication.user_dn_template : replace {0}> -W -b
<dbms.security.ldap.authorization.user_search_base> "<dbms.security.ldap.authorization.user_search_filter : replace {0}>"
<dbms.security.ldap.authorization.group_membership_attributes>
ldapsearch -v -H ldap://myactivedirectory.example.com:389 -x -D cn=john,cn=Users,dc=example,dc=com -W -b cn=Users,dc=example,dc=com "(&
(objectClass=*)(cn=john))" memberOf
```

With `dbms.security.ldap.authorization.use_system_account=true`:

```
#ldapsearch -v -H ldap://<dbms.security.ldap.host> -x -D <dbms.security.ldap.authorization.system_username> -w
<dbms.security.ldap.authorization.system_password> -b <dbms.security.ldap.authorization.user_search_base> "
<dbms.security.ldap.authorization.user_search_filter>" <dbms.security.ldap.authorization.group_membership_attributes>
ldapsearch -v -H ldap://myactivedirectory.example.com:389 -x -D cn=search-account,cn=Users,dc=example,dc=com -w secret -b
cn=Users,dc=example,dc=com "(&(objectClass=*)(cn=john))" memberOf
```

Then verify that we get a successful response, and that the value of the returned membership attribute is a group that is mapped to a role in `dbms.security.ldap.authorization.group_to_role_mapping`.

```
# extended LDIF
#
# LDAPv3
# base <cn=Users,dc=example,dc=com> with scope subtree
# filter: (cn=john)
# requesting: memberOf
#

# john, Users, example.com
dn: CN=john,CN=Users,DC=example,DC=com
memberOf: CN=Neo4j Read Only,CN=Users,DC=example,DC=com

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

8.5.4. The auth cache

The *auth cache* is the mechanism by which Neo4j caches the result of authentication via the LDAP server in order to aid performance. It is configured with the `dbms.security.ldap.authentication.cache_enabled (../../reference/configuration-settings/#config_dbms.security.ldap.authentication.cache_enabled)` and `dbms.security.auth_cache_ttl (../../reference/configuration-settings/#config_dbms.security.auth_cache_ttl)` parameters.

```
# Turn on authentication caching to ensure performance
dbms.security.ldap.authentication.cache_enabled=true
dbms.security.auth_cache_ttl=10m
```

Parameter name	Default value	Description
dbms.security.ldap.authentication.cache_enabled (../../reference/configuration-settings/#config_dbms.security.ldap.authentication.cache_enabled)	`true`	Determines whether or not to cache the result of authentication via the LDAP server. Whether authentication caching should be enabled or not must be considered in view of your company's security guidelines. It should be noted that when using the REST API, disabling authentication caching will result in re-authentication and possibly re-authorization of users on every request, which may severely impact performance on production systems, and put heavy load on the LDAP server.

Parameter name	Default value	Description
dbms.security.auth_cache_ttl (../../reference/configuration-settings/#config_dbms.security.auth_cache_ttl)	`600 seconds`	is the time to live (TTL) for cached authentication and authorization info. Setting the TTL to 0 will disable all auth caching. A short TTL will require more frequent re-authentication and re-authorization, which can impact performance. A very long TTL will also mean that changes to the users settings on an LDAP server may not be reflected in the Neo4j authorization behaviour in a timely manner. Valid units are `ms`, `s`, `m`; default unit is `s`.

An administrator can clear the auth cache to force the re-querying of authentication and authorization information from the federated auth provider system.

Example 8.17. Clear the auth cache

Use Neo4j Browser or Neo4j Cypher Shell to execute this statement.

```
CALL dbms.security.clearAuthCache()
```

8.5.5. Available methods of encryption

All the following ways of specifying the `dbms.security.ldap.host (../../reference/configuration-settings/#config_dbms.security.ldap.host)` parameter are valid. Doing so will configure using LDAP without encryption. Not specifying the protocol or port will result in `ldap` being used over the default port `389`.

```
dbms.security.ldap.host=myactivedirectory.example.com
dbms.security.ldap.host=myactivedirectory.example.com:389
dbms.security.ldap.host=ldap://myactivedirectory.example.com
dbms.security.ldap.host=ldap://myactivedirectory.example.com:389
```

8.5.5.1. Use LDAP with encryption via StartTLS

To configure Active Directory with encryption via StartTLS, set the following parameters:

```
dbms.security.ldap.use_starttls=true
dbms.security.ldap.host=ldap://myactivedirectory.example.com
```


8.5.5.2. Use LDAP with encrypted LDAPS

To configure Active Directory with encrypted LDAPS, set `dbms.security.ldap.host (../../reference/configuration-settings/#config_dbms.security.ldap.host)` to one of the following. Not specifying the port will result in `ldaps` being used over the default port `636`.

```
dbms.security.ldap.host=ldaps://myactivedirectory.example.com
dbms.security.ldap.host=ldaps://myactivedirectory.example.com:636
```

This method of securing Active Directory is being deprecated and is therefore not recommended. Instead, use Active Directory with encryption via StartTLS.

8.5.6. Use a self-signed certificate in a test environment

Production environments should always use an SSL certificate issued by a Certificate Authority for secure access to the LDAP server. However, there are scenarios, for example in test environments, where you may wish to use a self-signed certificate on the LDAP server. In these scenarios you will have to tell Neo4j about the local certificate. This is done by entering the details of the certificate using `dbms.jvm.additional` in *neo4j.conf*.

Example 8.18. Specify details for self-signed certificate on LDAP server

This example shows how to specify details for a self-signed certificate on an LDAP server. The path to the certificate file `MyCert.jks` is an absolute path on the Neo4j server.

```
dbms.jvm.additional=-Djavax.net.ssl.keyStore=/path/to/MyCert.jks
dbms.jvm.additional=-Djavax.net.ssl.keyStorePassword=secret
dbms.jvm.additional=-Djavax.net.ssl.trustStore=/path/to/MyCert.jks
dbms.jvm.additional=-Djavax.net.ssl.trustStorePassword=secret
```