



# LDAP configuration

## Astra Automation

NetApp  
February 12, 2024

# Table of Contents

- LDAP configuration ..... 1
  - Prepare for LDAP configuration ..... 1
  - Configure Astra to use an LDAP server ..... 3
  - Add LDAP entries to Astra ..... 10
  - Disable and reset LDAP ..... 16

# LDAP configuration

## Prepare for LDAP configuration

You can optionally integrate Astra Control Center with a Lightweight Directory Access Protocol (LDAP) server to perform authentication for selected Astra users. LDAP is an industry standard protocol for accessing distributed directory information and a popular choice for enterprise authentication.

### Related information

- [LDAP Technical Specification Road Map](#)
- [LDAP version 3](#)

## Overview of the implementation process

At a high level, there are several steps you need to perform to configure an LDAP server to provide authentication for Astra users.



While the steps presented below are in a sequence, in some cases you can perform them in a different order. For example, you can define the Astra users and groups before configuring the LDAP server.

1. Review [Requirements and limitations](#) to understand the options, requirements, and limitations.
2. Select an LDAP server and the desired configuration options (including security).
3. Perform the workflow [Configure Astra to use an LDAP server](#) to integrate Astra with the LDAP server.
4. Review the users and groups at the LDAP server to make sure they are defined properly.
5. Perform the appropriate workflow in [Add LDAP entries to Astra](#) to identify the users to be authenticated using LDAP.

## Requirements and limitations

You should review the Astra configuration essentials presented below, including limitations and configuration options, before configuring Astra to use LDAP for authentication.

### Only supported with Astra Control Center

The Astra Control platform provides two deployment models. LDAP authentication is only supported with Astra Control Center deployments.

### Configuration using REST API or web user interface

The current release of Astra Control Center supports configuration of LDAP authentication using both the Astra Control REST API as well as the Astra web user interface.

### LDAP server required

You must have an LDAP server to accept and process the Astra authentication requests. Microsoft's Active Directory is supported with the current Astra Control Center release.

### Secure connection to the LDAP server

When configuring the LDAP server in Astra, you can optionally define a secure connection. In this case a

certificate is needed for the LDAPS protocol.

### **Configure users or groups**

You need to select the users to be authenticated using LDAP. You can do this either by identifying the individual users or a group of users. The accounts must be defined at the LDAP server. They also need to be identified in Astra (type LDAP) which allows the authentication requests to be forwarded to LDAP.

### **Role constraint when binding a user or group**

With the current release of Astra Control Center, the only supported value for `roleConstraint` is `"*"`. This indicates the user is not restricted to a limited set of namespaces and can access all of them. See [Add LDAP entries to Astra](#) for more information.

### **LDAP credentials**

The credentials used by LDAP include the username (email address) and the associated password.

### **Unique email addresses**

All email addresses acting as usernames in an Astra Control Center deployment must be unique. You cannot add an LDAP user with an email address that is already defined to Astra. If a duplicate email exists, you need to first delete it from Astra. See [Remove users](#) at the Astra Control Center documentation site for more information.

### **Optionally define LDAP users and groups first**

You can add the LDAP users and groups to Astra Control Center even if they don't yet exist in LDAP or if the LDAP server is not configured. This allows you to preconfigure the users and groups before configuring the LDAP server.

### **A user defined in multiple LDAP groups**

If an LDAP user belongs to multiple LDAP groups and the groups have been assigned different roles in Astra, the user's effective role when authenticating will be the most privileged. For example, if a user is assigned the `viewer` role with `group1` but has the `member` role in `group2`, the user's role would be `member`. This is based on the hierarchy used by Astra (highest to lowest):

- Owner
- Admin
- Member
- Viewer

### **Periodic account synchronization**

Astra synchronizes its users and groups with the LDAP server approximately every 60 seconds. So if a user or group is added to or removed from LDAP, it can take up to one minute before it is available in Astra.

### **Disabling and resetting the LDAP configuration**

Before attempting to reset the LDAP configuration, you must first disable LDAP authentication. Also, to change the LDAP server (`connectionHost`), you need to perform both operations. See [Disable and reset LDAP](#) for more information.

### **REST API parameters**

The LDAP configuration workflows make REST API calls to accomplish the specific tasks. Each API call can include input parameters as shown in the provided samples. See [Online API reference](#) for information about how to locate the reference documentation.

# Configure Astra to use an LDAP server

You need to select an LDAP server and configure Astra to use the server as an authentication provider. The configuration task consists of the steps described below. Each step includes a single REST API call.

## 1. Add a CA certificate

Perform the following REST API call to add a CA certificate to Astra.



This step is optional and only required if you want Astra and the LDAP to communicate over a secure channel using LDAPS.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/certificates

## JSON input example

```
{
  "type": "application/astra-certificate",
  "version": "1.0",
  "certUse": "rootCA",
  "cert": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUMyVEN",
  "isSelfSigned": "true"
}
```

Note the following about the input parameters:

- `cert` is a JSON string containing a base64 encoded PKCS-11 formatted certificate (PEM encoded).
- `isSelfSigned` should be set to `true` if the certificate is self-signed. The default is `false`.

## Curl example

```
curl --location -i --request POST --data @JSONinput
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/certificates'
--header 'Content-Type: application/astra-certificate+json' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

## JSON response example

```
{
  "type": "application/astra-certificate",
  "version": "1.0",
  "id": "a5212e7e-402b-4cff-bba0-63f3c6505199",
  "certUse": "rootCA",
  "cert": "LS0tLS1CRUdJTtiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMyVEN",
  "cn": "adldap.example.com",
  "expiryTimestamp": "2023-07-08T20:22:07Z",
  "isSelfSigned": "true",
  "trustState": "trusted",
  "trustStateTransitions": [
    {
      "from": "untrusted",
      "to": [
        "trusted",
        "expired"
      ]
    },
    {
      "from": "trusted",
      "to": [
        "untrusted",
        "expired"
      ]
    },
    {
      "from": "expired",
      "to": [
        "untrusted",
        "trusted"
      ]
    }
  ],
  "trustStateDesired": "trusted",
  "trustStateDetails": [],
  "metadata": {
    "creationTimestamp": "2022-07-21T04:16:06Z",
    "modificationTimestamp": "2022-07-21T04:16:06Z",
    "createdBy": "8a02d2b8-a69d-4064-827f-36851b3e1e6e",
    "modifiedBy": "8a02d2b8-a69d-4064-827f-36851b3e1e6e",
    "labels": []
  }
}
```

## 2. Add the bind credentials

Perform the following REST API call to add the bind credentials.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/credentials

### JSON input example

```
{
  "name": "ldapBindCredential",
  "type": "application/astra-credential",
  "version": "1.1",
  "keyStore": {
    "bindDn": "dWlkPWFkbWluLG91PXM5c3RlbQ==",
    "password": "cGFzc3dvcmQ="
  }
}
```

Note the following about the input parameters:

- `bindDn` and `password` are the base64 encoded bind credentials of the LDAP admin user that is able to connect and search the LDAP directory. `bindDn` is the LDAP user's email address.

### Curl example

```
curl --location -i --request POST --data @JSONinput
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/credentials'
--header 'Content-Type: application/astra-credential+json' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

### JSON response example

```
{
  "type": "application/astra-credential",
  "version": "1.1",
  "id": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
  "name": "ldapBindCredential",
  "metadata": {
    "creationTimestamp": "2022-07-21T06:53:11Z",
    "modificationTimestamp": "2022-07-21T06:53:11Z",
    "createdBy": "527329f2-662c-41c0-ada9-2f428f14c137"
  }
}
```

Note the following the response parameters:

- The `id` of the credential is used in subsequent workflow steps.

### 3. Retrieve the UUID of the LDAP setting

Perform the following REST API call to retrieve the UUID of the `astra.account.ldap` setting that is included with Astra Control Center.



The curl example below uses a query parameter to filter the settings collection. You can instead remove the filter to get all the settings and then search for `astra.account.ldap`.

HTTP method	Path
GET	<code>/accounts/{account_id}/core/v1/settings</code>

#### Curl example

```
curl --location -i --request GET
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/settings?filter=name%20eq%20'astra.account.ldap'&include=name,id' --header 'Accept: */*'
--header 'Authorization: Bearer <API_TOKEN>'
```

#### JSON response example

```
{
  "items": [
    ["astra.account.ldap",
     "12072b56-e939-45ec-974d-2dd83b7815df"]
  ],
  "metadata": {}
}
```

### 4. Update the LDAP setting

Perform the following REST API call to update the LDAP setting and complete the configuration. Use the `id` value from the previous API call for the `<SETTING_ID>` value in the URL path below.



You can issue a GET request for the specific setting first to see the `configSchema`. This will provide more information about the required fields in the configuration.

HTTP method	Path
PUT	<code>/accounts/{account_id}/core/v1/settings/{setting_id}</code>



## JSON input example

```
{
  "type": "application/astra-setting",
  "version": "1.0",
  "desiredConfig": {
    "connectionHost": "myldap.example.com",
    "credentialId": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
    "groupBaseDN": "OU=groups,OU=astra,DC=example,DC=com",
    "isEnabled": "true",
    "port": 686,
    "secureMode": "LDAPS",
    "userBaseDN": "OU=users,OU=astra,DC=example,dc=com",
    "userSearchFilter": "((objectClass=User))",
    "vendor": "Active Directory"
  }
}
```

Note the following about the input parameters:

- `isEnabled` should be set to `true` or an error may occur.
- `credentialId` is the id of the bind credential created earlier.
- `secureMode` should be set to `LDAP` or `LDAPS` based on your configuration in the earlier step.
- Only 'Active Directory' is supported as a vendor.

## Curl example

```
curl --location -i --request PUT --data @JSONinput
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/settings/<SETTING_ID>' --header 'Content-Type: application/astra-setting+json' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

If the call is successful, the HTTP 204 response is returned.

## 5. Retrieve the LDAP setting

You can optionally perform the following REST API call to retrieve the LDAP settings and confirm the update.

HTTP method	Path
GET	/accounts/{account_id}/core/v1/settings/{setting_id}

## Curl example

```
curl --location -i --request GET
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/settings/<SETTING_ID>' --header 'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

## JSON response example

```
{
  "items": [
    {
      "type": "application/astra-setting",
      "version": "1.0",
      "metadata": {
        "creationTimestamp": "2022-06-17T21:16:31Z",
        "modificationTimestamp": "2022-07-21T07:12:20Z",
        "labels": [],
        "createdBy": "system",
        "modifiedBy": "00000000-0000-0000-0000-000000000000"
      },
      "id": "12072b56-e939-45ec-974d-2dd83b7815df",
      "name": "astra.account.ldap",
      "desiredConfig": {
        "connectionHost": "10.193.61.88",
        "credentialId": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
        "groupBaseDN": "ou=groups,ou=astra,dc=example,dc=com",
        "isEnabled": "true",
        "port": 686,
        "secureMode": "LDAPS",
        "userBaseDN": "ou=users,ou=astra,dc=example,dc=com",
        "userSearchFilter": "(objectClass=User)",
        "vendor": "Active Directory"
      },
      "currentConfig": {
        "connectionHost": "10.193.160.209",
        "credentialId": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
        "groupBaseDN": "ou=groups,ou=astra,dc=example,dc=com",
        "isEnabled": "true",
        "port": 686,
        "secureMode": "LDAPS",
        "userBaseDN": "ou=users,ou=astra,dc=example,dc=com",
        "userSearchFilter": "(objectClass=User)",
        "vendor": "Active Directory"
      },
      "configSchema": {
        "$schema": "http://json-schema.org/draft-07/schema#",

```

```
"title": "astra.account.ldap",
"type": "object",
"properties": {
  "connectionHost": {
    "type": "string",
    "description": "The hostname or IP address of your LDAP server."
  },
  "credentialId": {
    "type": "string",
    "description": "The credential ID for LDAP account."
  },
  "groupBaseDN": {
    "type": "string",
    "description": "The base DN of the tree used to start the group
search. The system searches the subtree from the specified location."
  },
  "groupSearchCustomFilter": {
    "type": "string",
    "description": "Type of search that controls the default group
search filter used."
  },
  "isEnabled": {
    "type": "string",
    "description": "This property determines if this setting is
enabled or not."
  },
  "port": {
    "type": "integer",
    "description": "The port on which the LDAP server is running."
  },
  "secureMode": {
    "type": "string",
    "description": "The secure mode LDAPS or LDAP."
  },
  "userBaseDN": {
    "type": "string",
    "description": "The base DN of the tree used to start the user
search. The system searches the subtree from the specified location."
  },
  "userSearchFilter": {
    "type": "string",
    "description": "The filter used to search for users according a
search criteria."
  },
  "vendor": {
    "type": "string",
```

```

        "description": "The LDAP provider you are using.",
        "enum": ["Active Directory"]
    }
},
"additionalProperties": false,
"required": [
    "connectionHost",
    "secureMode",
    "credentialId",
    "userBaseDN",
    "userSearchFilter",
    "groupBaseDN",
    "vendor",
    "isEnabled"
]
},
"state": "valid",
}
],
"metadata": {}
}

```

Locate the `state` field in the response which will have one of the values in the table below.

State	Description
pending	The configuration process is still active and not completed yet.
valid	Configuration has been completed successfully and <code>currentConfig</code> in the response matches <code>desiredConfig</code> .
error	The LDAP configuration process failed.

## Add LDAP entries to Astra

After LDAP is configured as an authentication provider for Astra Control Center, you can select the LDAP users that Astra will authenticate using the LDAP credentials. Each user must have a role in Astra before they can access Astra through the Astra Control REST API.

There are two ways you can configure Astra to assign roles. Choose the one that is appropriate for your environment.

- [Add and bind an individual user](#)
- [Add and bind a group](#)



The LDAP credentials are in the form of a username as an email address and the associated LDAP password.

## Add and bind an individual user

You can assign a role to each Astra user which is used after LDAP authentication. This is appropriate when there is a small number of users and each might have different administrative characteristics.

### 1. Add a user

Perform the following REST API call to add a user to Astra and indicate that LDAP is the authentication provider.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/users

### JSON input example

```
{
  "type" : "application/astra-user",
  "version" : "1.1",
  "authID" : "cn=JohnDoe,ou=users,ou=astra,dc=example,dc=com",
  "authProvider" : "ldap",
  "firstName" : "John",
  "lastName" : "Doe",
  "email" : "john.doe@example.com"
}
```

Note the following about the input parameters:

- The following parameters are required:
  - authProvider
  - authID
  - email
- authID is the distinguished name (DN) of the user in LDAP
- email must be unique for all users defined in Astra

If the email value is not unique, an error occurs and a 409 HTTP status code is returned in the response.

### Curl example

```
curl --location -i --request POST --data @JSONinput
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/users' --header
'Content-Type: application/astra-user+json' --header 'Accept: */*'
--header 'Authorization: Bearer <API_TOKEN>'
```

## JSON response example

```
{
  "metadata": {
    "creationTimestamp": "2022-07-21T17:44:18Z",
    "modificationTimestamp": "2022-07-21T17:44:18Z",
    "createdBy": "8a02d2b8-a69d-4064-827f-36851b3e1e6e",
    "labels": []
  },
  "type": "application/astra-user",
  "version": "1.2",
  "id": "a7b5e674-a1b1-48f6-9729-6a571426d49f",
  "authProvider": "ldap",
  "authID": "cn=JohnDoe,ou=users,ou=astra,dc=example,dc=com",
  "firstName": "John",
  "lastName": "Doe",
  "companyName": "",
  "email": "john.doe@example.com",
  "postalAddress": {
    "addressCountry": "",
    "addressLocality": "",
    "addressRegion": "",
    "streetAddress1": "",
    "streetAddress2": "",
    "postalCode": ""
  },
  "state": "active",
  "sendWelcomeEmail": "false",
  "isEnabled": "true",
  "isInviteAccepted": "true",
  "enableTimestamp": "2022-07-21T17:44:18Z",
  "lastActTimestamp": ""
}
```

## 2. Add a role binding for the user

Perform the following REST API call to bind the user to a specific role. You need to have the UUID of the user created in the previous step.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/roleBindings

## JSON input example

```
{
  "type": "application/astra-roleBinding",
  "version": "1.1",
  "accountID": "{account_id}",
  "userID": "a7b5e674-a1b1-48f6-9729-6a571426d49f",
  "role": "member",
  "roleConstraints": ["*"]
}
```

Note the following about the input parameters:

- The value used above for `roleConstraint` is the only option available for the current release of Astra. It indicates the user is not restricted to a limited set of namespaces and can access them all.

### Curl example

```
curl --location -i --request POST --data @JSONinput
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/roleBindings'
--header 'Content-Type: application/astra-roleBinding+json' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

### JSON response example

```
{
  "metadata": {
    "creationTimestamp": "2022-07-21T18:08:24Z",
    "modificationTimestamp": "2022-07-21T18:08:24Z",
    "createdBy": "8a02d2b8-a69d-4064-827f-36851b3e1e6e",
    "labels": []
  },
  "type": "application/astra-roleBinding",
  "principalType": "user",
  "version": "1.1",
  "id": "b02c7e4d-d483-40d1-aaff-e1f900312114",
  "userID": "a7b5e674-a1b1-48f6-9729-6a571426d49f",
  "groupID": "00000000-0000-0000-0000-000000000000",
  "accountID": "d0fdbfa7-be32-4a71-b59d-13d95b42329a",
  "role": "member",
  "roleConstraints": ["*"]
}
```

Note the following about the response parameters:

- The value `user` for the `principalType` field indicates the role binding was added for a user (not a

group).

## Add and bind a group

You can assign a role to an Astra group which is used after LDAP authentication. This is appropriate when there is a large number of users and each might have similar administrative characteristics.

### 1. Add a group

Perform the following REST API call to add a group to Astra and indicate that LDAP is the authentication provider.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/groups

### JSON input example

```
{
  "type": "application/astra-group",
  "version": "1.0",
  "name": "Engineering",
  "authProvider": "ldap",
  "authID": "CN=Engineering,OU=groups,OU=astra,DC=example,DC=com"
}
```

Note the following about the input parameters:

- The following parameters are required:
  - authProvider
  - authID

### Curl example

```
curl --location -i --request POST --data @JSONinput
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/groups' --header
'Content-Type: application/astra-group+json' --header 'Accept: */*'
--header 'Authorization: Bearer <API_TOKEN>'
```

### JSON response example



```

{
  "type": "application/astra-group",
  "version": "1.0",
  "id": "8b5b54da-ae53-497a-963d-1fc89990525b",
  "name": "Engineering",
  "authProvider": "ldap",
  "authID": "CN=Engineering,OU=groups,OU=astra,DC=example,DC=com",
  "metadata": {
    "creationTimestamp": "2022-07-21T18:42:52Z",
    "modificationTimestamp": "2022-07-21T18:42:52Z",
    "createdBy": "8a02d2b8-a69d-4064-827f-36851b3e1e6e",
    "labels": []
  }
}

```

## 2. Add a role binding for the group

Perform the following REST API call to bind the group to a specific role. You need to have the UUID of the group created in the previous step. Users that are members of the group will be able to sign in to Astra after LDAP performs the authentication.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/roleBindings

### JSON input example

```

{
  "type": "application/astra-roleBinding",
  "version": "1.1",
  "accountID": "{account_id}",
  "groupID": "8b5b54da-ae53-497a-963d-1fc89990525b",
  "role": "viewer",
  "roleConstraints": ["*"]
}

```

Note the following about the input parameters:

- The value used above for `roleConstraint` is the only option available for the current release of Astra. It indicates the user is not restricted to certain namespaces and can access them all.

### Curl example

```
curl --location -i --request POST --data @JSONinput
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/roleBindings'
--header 'Content-Type: application/astra-roleBinding+json' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

## JSON response example

```
{
  "metadata": {
    "creationTimestamp": "2022-07-21T18:59:43Z",
    "modificationTimestamp": "2022-07-21T18:59:43Z",
    "createdBy": "527329f2-662c-41c0-ada9-2f428f14c137",
    "labels": []
  },
  "type": "application/astra-roleBinding",
  "principalType": "group",
  "version": "1.1",
  "id": "2f91b06d-315e-41d8-ae18-7df7c08fbb77",
  "userID": "00000000-0000-0000-0000-000000000000",
  "groupID": "8b5b54da-ae53-497a-963d-1fc89990525b",
  "accountID": "d0fdbfa7-be32-4a71-b59d-13d95b42329a",
  "role": "viewer",
  "roleConstraints": ["*"]
}
```

Note the following about the response parameters:

- The value `group` for the `principalType` field indicates the role binding was added for a group (not a user).

## Disable and reset LDAP

There are two optional though related administrative tasks you can perform as needed for an Astra Control Center deployment. You can globally disable LDAP authentication and reset the LDAP configuration.

Both workflow tasks require the id for the `astra.account.ldap` Astra setting. Details for how to retrieve the setting id are included in **Configure the LDAP server**. See [Retrieve the UUID of the LDAP setting](#) for more information.

- [Disable LDAP authentication](#)
- [Reset the LDAP authentication configuration](#)

## Disable LDAP authentication

You can perform the following REST API call to globally disable LDAP authentication for a specific Astra deployment. The call updates the `astra.account.ldap` setting and the `isEnabled` value is set to `false`.

HTTP method	Path
PUT	/accounts/{account_id}/core/v1/settings/{setting_id}

### JSON input example

```
{
  "type": "application/astra-setting",
  "version": "1.0",
  "desiredConfig": {
    "connectionHost": "myldap.example.com",
    "credentialId": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
    "groupBaseDN": "OU=groups,OU=astra,DC=example,DC=com",
    "isEnabled": "false",
    "port": 686,
    "secureMode": "LDAPS",
    "userBaseDN": "OU=users,OU=astra,DC=example,dc=com",
    "userSearchFilter": "((objectClass=User))",
    "vendor": "Active Directory"
  }
}
```

```
curl --location -i --request PUT --data @JSONinput
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/settings/<SETTING_ID>' --header 'Content-Type: application/astra-setting+json' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

If the call is successful, the HTTP 204 response is returned. You can optionally retrieve the configuration settings again to confirm the change.

## Reset the LDAP authentication configuration

You can perform the following REST API call to disconnect Astra from the LDAP server and reset the LDAP configuration in Astra. The call updates the `astra.account.ldap` setting and the value of `connectionHost` is cleared.

The value of `isEnabled` must also be set to `false`. You can either set this value before making the reset call or as part of making the reset call. In the second case, `connectionHost` should be cleared and `isEnabled` set to `false` on the same reset call.



This is a disruptive operation and you should proceed with caution. It deletes all the imported LDAP users and groups. It also deletes all the related Astra users, groups, and roleBindings (LDAP type) that you created in Astra Control Center.

HTTP method	Path
PUT	/accounts/{account_id}/core/v1/settings/{setting_id}

### JSON input example

```
{
  "type": "application/astra-setting",
  "version": "1.0",
  "desiredConfig": {
    "connectionHost": "",
    "credentialId": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
    "groupBaseDN": "OU=groups,OU=astra,DC=example,DC=com",
    "isEnabled": "false",
    "port": 686,
    "secureMode": "LDAPS",
    "userBaseDN": "OU=users,OU=astra,DC=example,dc=com",
    "userSearchFilter": "((objectClass=User))",
    "vendor": "Active Directory"
  }
}
```

Note the following:

- To change the LDAP server, you must both disable and reset LDAP changing `connectHost` to a null value as shown in the example above.

```
curl --location -i --request PUT --data @JSONinput
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/settings/<SETTING_ID>' --header 'Content-Type: application/astra-setting+json' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

If the call is successful, the HTTP 204 response is returned. You can optionally retrieve the configuration again to confirm the change.

## Copyright information

Copyright © 2024 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

## Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.