NetApp

Infrastructure workflows

Astra Automation

NetApp
August 10, 2022

Table of Contents

Infrastructure workflows

Before you begin

You can use these workflows to create and maintain the infrastructure used with the Astra Control Center deployment model. In most case, the workflows can also be used with Astra Control Service.

> ℹ️ These workflows can be expanded and enhanced by NetApp at any time and so you should review them periodically.

General preparation

Before using any of the Astra workflows, make sure to review Prepare to use the workflows.

Workflow categories

The infrastructure workflows are organized in different categories to make it easier to locate the one you want.

Category	Description
Identity and access	These workflows allow you to manage identity and how Astra is accessed. The resources include users, credentials, and tokens.
LDAP configuration	You can optionally configure Astra Control Center to use LDAP to authenticate selected users.
Buckets	You can use these workflows to create and manage the S3 buckets used to store backups.
Storage	These workflows allow you to add and maintain storage backends and volumes.
Clusters	You can add managed Kubernetes clusters which allows you to protect and support the applications they contain.

Identity and access

List users

You can list the users that are defined for a specific Astra account.

1. List the users

Perform the following REST API call.

HTTP method	Path
GET	/account/{account_id}/core/v1/users

Additional input parameters

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
include	Query	No	Optionally select the values you want returned in the response.

Curl example: Return all data for all users

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/core/v1/users' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

Curl example: Return the first name, last name, and id for all users

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/core/v1/users?include=first
Name,lastName,id' --header 'Accept: */*' --header 'Authorization: Bearer
<API_TOKEN>'
```

JSON output example

```
{
  "items": [
    [
    "David",
    "Anderson",
    "844ec6234-11e0-49ea-8434-a992a6270ec1"
    ],
    [
      "Jane",
      "Cohen",
      "2a3e227c-fda7-4145-a86c-ed9aa0183a6c"
    ]
  ],
  "metadata": {}
}
```

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.

REST API configuration only

The current release of Astra Control Center only supports configuration of LDAP authentication using the Astra Control REST API. An important aspect of this limitation is the LDAP users are not displayed in the users tab of the Astra web interface. They are available through the REST API at the endpoint `../core/v1/users`.

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 it's users and groups with the LDAP server approximately every 60 second. 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 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	/account/{account_id}/core/v1/certificates

JSON input example

```
{
   "type": "application/astra-certificate",
   "version": "1.0",
   "certUse": "rootCA",
   "cert": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMyVEN",
   "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": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMyVEN",
  "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	/account/{account_id}/core/v1/credentials

JSON input example

```
{
  "name": "ldapBindCredential",
  "type": "application/astra-credential",
  "version": "1.1",
  "keyStore": {
    "bindDn": "dWlkPWFkbWluLG91PXN5c3RlbQ==",
    "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	/account/{account_id}/core/v1/settings

Curl example

```
curl --location -i --request GET
'https://astra.example.com/accounts/<ACCOUNT_ID>/core/v1/settings?filter=n
ame%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	/account/{account_id}/core/v1/settings/{setting_id}

JSON input example

```json
{
  "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	/account/{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

```json
{
  "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 `currentConfig` in the response matches `desiredConfig`.
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	/account/{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

```json
{
  "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	/account/{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	/account/{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	/account/{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

```json
{
  "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	/account/{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	/account/{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.

Buckets

List buckets

You can list the S3 buckets defined for a specific Astra account.

1. List the buckets

Perform the following REST API call.

HTTP method	Path
GET	/account/{account_id}/topology/v1/buckets

Curl example: Return all data for all buckets

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/topology/v1/buckets'
--header 'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

Storage

List storage backends

You can list the available storage backends.

1. List the backends

Perform the following REST API call.

HTTP method	Path
GET	/account/{account_id}/topology/v1/storageBackends

Curl example: Return all data for all storage backends

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/topology/v1/storageBackends
' --header 'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

JSON output example

```
{
  "items": [
    {
      "backendCredentialsName": "10.191.77.177",
      "backendName": "myinchunhcluster-1",
      "backendType": "ONTAP",
      "backendVersion": "9.8.0",
      "configVersion": "Not applicable",
      "health": "Not applicable",
      "id": "46467c16-1585-4b71-8e7f-f0bc5ff9da15",
      "location": "nalab2",
      "metadata": {
        "createdBy": "4c483a7e-207b-4f9a-87b7-799a4629d7c8",
        "creationTimestamp": "2021-07-30T14:26:19Z",
        "modificationTimestamp": "2021-07-30T14:26:19Z"
      },
      "ontap": {
        "backendManagementIP": "10.191.77.177",
        "managementIPs": [
          "10.191.77.177",
          "10.191.77.179"
        ]
      },
      "protectionPolicy": "Not applicable",
      "region": "Not applicable",
      "state": "Running",
      "stateUnready": [],
      "type": "application/astra-storageBackend",
      "version": "1.0",
      "zone": "Not applicable"
    }
  ]
}
```

Clusters

List managed clusters

You can list the Kubernetes clusters currently managed by Astra.

1. List the clusters

Perform the following REST API call.

HTTP method	Path
GET	/account/{account_id}/topology/v1/managedClusters

Curl example: Return all data for all clusters

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/topology/v1/managedClusters
' --header 'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```