



Authentication and authorization using OAuth 2.0

ONTAP 9

NetApp
January 23, 2026

Table of Contents

Authentication and authorization using OAuth 2.0	1
Overview of the ONTAP OAuth 2.0 implementation	1
Features and benefits	1
Implementation and configuration	2
Selected terminology	3
Additional resources	4
Concepts	4
OAuth 2.0 authorization servers and access tokens in ONTAP	4
Client authorization	7
OAuth 2.0 deployment scenarios with ONTAP	15
ONTAP client authentication using OAuth 2.0 Mutual TLS	17
Configure and deploy	19
Prepare to deploy OAuth 2.0 with ONTAP	19
Deploy OAuth 2.0 in ONTAP	21
Issue an ONTAP REST API call using OAuth 2.0	24

Authentication and authorization using OAuth 2.0

Overview of the ONTAP OAuth 2.0 implementation

Beginning with ONTAP 9.14, you have the option to control access to your ONTAP clusters using the Open Authorization (OAuth 2.0) framework. You can configure this feature using any of the ONTAP administrative interfaces, including the ONTAP CLI, System Manager, and REST API. However, the OAuth 2.0 authorization and access control decisions can only be applied when a client accesses ONTAP using the REST API.



OAuth 2.0 support was first introduced with ONTAP 9.14.0 and so its availability depends on the ONTAP release you are using. See the [ONTAP release notes](#) for more information.

Features and benefits

The major features and benefits of using OAuth 2.0 with ONTAP are described below.

Support for the OAuth 2.0 standard

OAuth 2.0 is the industry standard authorization framework. It is used to restrict and control access to protected resources using signed access tokens. There are several benefits to using OAuth 2.0:

- Many options for the authorization configuration
- Never reveal the client credentials including passwords
- Tokens can be set to expire based on your configuration
- Ideally suited for use with REST APIs

Tested with popular authorization servers

The ONTAP OAuth 2.0 implementation has been tested with several popular servers or services based on the ONTAP release as follows:

- ONTAP 9.16.1 (support for group UUID to name mapping and external roles):
 - Microsoft Entra ID
- ONTAP 9.14.1 (support for standard OAuth 2.0 features)
 - Auth0
 - Active Directory Federation Service (ADFS)
 - Keycloak

See [Authorization servers and access tokens](#) for more details about the features and capabilities available with each ONTAP release.

Support for multiple concurrent authorization servers

You can define up to eight authorization servers for a single ONTAP cluster. This gives you the flexibility to meet the needs of your diverse security environment.

Integration with the REST roles

The ONTAP authorization decisions are ultimately based on the REST roles assigned to users or groups. These roles are either carried in the access token as self-contained scopes or based on local ONTAP definitions along with Active Directory or LDAP groups.

Option to use sender-constrained access tokens

You can configure ONTAP and the authorization servers to use Mutual Transport Layer Security (mTLS) which strengthens client authentication. It guarantees the OAuth 2.0 access tokens are only used by the clients to which they were originally issued. This feature supports and aligns with several popular security recommendations, including those established by FAPI and MITRE.

Implementation and configuration

At a high level, there are several aspects of an OAuth 2.0 implementation and configuration you should consider when getting started.

OAuth 2.0 entities within ONTAP

The OAuth 2.0 authorization framework defines several entities that can be mapped to real or virtual elements within your data center or network. The OAuth 2.0 entities and their adaptation to ONTAP are presented in the table below.

OAuth 2.0 Entity	Description
Resource	The REST API endpoints that provide access to the ONTAP resources through internal ONTAP commands.
Resource owner	The ONTAP cluster user that created the protected resource or owns it by default.
Resource server	The host for the protected resources which is the ONTAP cluster.
Client	An application requesting access to a REST API endpoint on behalf of or with permission from the resource owner.
Authorization server	Typically a dedicated server responsible for issuing access tokens and enforcing administrative policy.

Core ONTAP configuration

You need to configure the ONTAP cluster to enable and use OAuth 2.0. This includes establishing a connection to the authorization server and defining the required ONTAP authorization configuration. You can perform this configuration using any of the administrative interfaces, including:

- ONTAP command line interface
- System Manager
- ONTAP REST API

Environment and supporting services

In addition to the ONTAP definitions, you also need to configure the authorization servers. If you're using group-to-role mapping, you need also to configure the Active Directory groups or LDAP equivalent.

Supported ONTAP clients

Beginning with ONTAP 9.14, a REST API client can access ONTAP using OAuth 2.0. Before issuing a REST API call, you need to obtain an access token from the authorization server. The client then passes this token to the ONTAP cluster as a *bearer token* using the HTTP authorization request header. Depending on the level of

security needed, you can also create and install a certificate at the client to use sender-constrained tokens based on mTLS.

Selected terminology

As you begin exploring an OAuth 2.0 deployment with ONTAP, it is helpful to become familiar with some of the terminology. See [Additional resources](#) for links to more information about OAuth 2.0.

Access token

A token issued by an authorization server and used by an OAuth 2.0 client application to make requests to access the protected resources.

JSON Web Token

The standard used to format the access tokens. JSON is used to represent the OAuth 2.0 claims in a compact format with the claims arranged in three main sections.

Sender-constrained access token

An optional feature based on the Mutual Transport Layer Security (mTLS) protocol. By using an additional confirmation claim in the token, this ensures the access token is only used by the client to which it was originally issued.

JSON Web Key Set

A JWKS is a collection of public keys used by ONTAP to verify the JWT tokens presented by the clients. The key sets are typically available at the authorization server through a dedicated URI.

Scope

Scopes provide a way to limit or control an application's access to protected resources such as the ONTAP REST API. They are represented as strings in the access token.

ONTAP REST role

REST roles were introduced with ONTAP 9.6 and are a core part of the ONTAP RBAC framework. These roles are different than the earlier traditional roles which are still supported by ONTAP. The OAuth 2.0 implementation in ONTAP only supports REST roles.

HTTP authorization header

A header included in the HTTP request to identify the client and associated permissions as part of making a REST API call. There are several flavors or implementations available depending on how authentication and authorization is performed. When presenting an OAuth 2.0 access token to ONTAP, the token is identified as a *bearer token*.

HTTP basic authentication

An early HTTP authentication technique still supported by ONTAP. The plaintext credentials (username and password) are concatenated with a colon and encoded in base64. The string is placed in the authorization request header and sent to the server.

FAPI

A working group at the OpenID Foundation providing protocols, data schemas, and security recommendations for the financial industry. The API was originally known as the Financial Grade API.

MITRE

A private not-for-profit company providing technical and security guidance to the United States Air Force and US government.

Additional resources

Several additional resources are provided below. You should review these sites to get more information about OAuth 2.0 and the related standards.

Protocols and standards

- [RFC 6749: The OAuth 2.0 Authorization Framework](#)
- [RFC 7519: JSON Web Tokens \(JWT\)](#)
- [RFC 7523: JSON Web Token \(JWT\) Profile for OAuth 2.0 Client Authentication and Authorization Grants](#)
- [RFC 7662: OAuth 2.0 Token Introspection](#)
- [RFC 7800: Proof-of-Possession Key for JWTs](#)
- [RFC 8705: OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens](#)

Organizations

- [OpenID Foundation](#)
- [FAPI Working Group](#)
- [MITRE](#)
- [IANA - JWT](#)

Products and services

- [Auth0](#)
- [Entra ID](#)
- [ADFS overview](#)
- [Keycloak](#)

Additional tools and utilities

- [JWT by Auth0](#)
- [OpenSSL](#)

NetApp documentation and resources

- [ONTAP automation documentation](#)

Concepts

OAuth 2.0 authorization servers and access tokens in ONTAP

Authorization servers perform several important functions as a central component within the OAuth 2.0 Authorization framework.

OAuth 2.0 authorization servers

Authorization servers are primarily responsible for creating and signing access tokens. These tokens contain identity and authorization information enabling a client application to selectively access protected resources. The servers are generally isolated from one another and can be implemented in several different ways, including as a standalone dedicated server or as part of a larger identity and access management product.



Different terminology can sometimes be used for an authorization server, especially when the OAuth 2.0 functionality is packaged within a larger identity and access management product or solution. For example, the term **identity provider (IdP)** is frequently used interchangeably with **authorization server**.

Administration

In addition to issuing access tokens, authorization servers also provide related administrative services, typically through a web user interface. For example, you can define and administer:

- Users and user authentication
- Scopes
- Administrative segregation through tenants and realms
- Policy enforcement
- Connection to various external services
- Support for other identity protocols (such as SAML)

ONTAP is compatible with authorization servers that are compliant with the OAuth 2.0 standard.

Defining to ONTAP

You need to define one or more authorization servers to ONTAP. ONTAP securely communicates with each server to verify tokens and perform other related tasks in support of the client applications.

The major aspects of ONTAP configuration are presented below. Also see [OAuth 2.0 deployment scenarios](#) for more information.

How and where the access tokens are validated

There are two options for validating access tokens.

- Local validation

ONTAP can validate access tokens locally based on information provided by the authorization server that issued the token. The information retrieved from the authorization server is cached by ONTAP and refreshed at regular intervals.

- Remote introspection

You can also use remote introspection to validate tokens at the authorization server. Introspection is a protocol allowing authorized parties to query an authorization server about an access token. It provides ONTAP a way to extract certain metadata from an access token and validate the token. ONTAP caches some of the data for performance reasons.

Network location

ONTAP may be behind a firewall. In this case, you need to identify a proxy as part of the configuration.

How the authorization servers are defined

You can define an authorization server to ONTAP using any of the administrative interfaces, including the CLI, System Manager, or REST API. For example, with the CLI you use the command `security oauth2 client create`.

Learn more about security oauth2 client create in the [ONTAP command reference](#).

Number of authorization servers

You can define up to eight authorization servers to a single ONTAP cluster. The same authorization server can be defined more than once to the same ONTAP cluster as long as the issuer or issuer/audience claims are unique. For example, with Keycloak this will always be the case when using different realms.

OAuth 2.0 features supported in ONTAP

Support for OAuth 2.0 was initially available with ONTAP 9.14.1 and continues to be enhanced with subsequent releases. The OAuth 2.0 features supported by ONTAP are described below.



Features introduced with a specific ONTAP release are carried forward to future releases.

ONTAP 9.16.1

ONTAP 9.16.1 expands the standard OAuth 2.0 features to include Entra ID specific extensions for native Entra ID groups. This involves the use of GUIDs in the access token instead of names. In addition, the release adds support for external role mapping to map the native identity provider roles to ONTAP roles using the "roles" field in the access token.

ONTAP 9.14.1

Beginning with ONTAP 9.14.1, authorization servers are supported through the following standard OAuth 2.0 features for applications using:

- OAuth 2.0 with the standard fields including "iss", "aud", and "exp" as described in [RFC6749: The OAuth 2.0 Authorization Framework](#) and [RFC 7519: JSON Web Token \(JWT\)](#). This also includes support for uniquely identifying users through fields in the access token such as "upn", "appid", "sub", "username" or "preferred_username".
- ADFS vendor-specific extensions for group names with the "group" field.
- Azure vendor-specific extensions for group UUIDs with the "group" field.
- ONTAP extensions for authorization support using self-contained and named roles within the OAuth 2.0 access token scope. This includes the "scope" and "scp" fields as well as group names within the scope.

Using OAuth 2.0 access tokens

The OAuth 2.0 access tokens issued by the authorization servers are verified by ONTAP and used to make role-based access decisions for the REST API client requests.

Acquiring an access token

You need to acquire an access token from an authorization server defined to the ONTAP cluster where you use the REST API. To acquire a token, you must contact the authorization server directly.



ONTAP does not issue access tokens or redirect requests from clients to the authorization servers.

How you request a token depends on several factors, including:

- Authorization server and its configuration options

- OAuth 2.0 grant type
- Client or software tool used to issue the request

Grant types

A *grant* is a well-defined process, including a set of network flows, used to request and receive an OAuth 2.0 access token. Several different grant types can be used depending on the client, environment, and security requirements. A list of the popular grant types is presented in the table below.

Grant type	Description
Client credentials	A popular grant type based on using only credentials (such as an ID and shared secret). The client is assumed to have a close trust relationship with the resource owner.
Password	The resource owner password credentials grant type can be used in cases where the resource owner has an established trust relation with the client. It can also be useful when migrating legacy HTTP clients to OAuth 2.0.
Authorization code	This is an ideal grant type for confidential clients and is based on a redirection-based flow. It can be used to obtain both an access token and refresh token.

JWT contents

An OAuth 2.0 access token is formatted as a JWT. The content is created by the authorization server based on your configuration. However, the tokens are opaque to the client applications. A client has no reason to inspect a token or to be aware of the contents.

Each JWT access token contains a set of claims. The claims describe characteristics of the issuer and the authorization based on administrative definitions at the authorization server. Some of the claims registered with the standard are described in the table below. All the strings are case sensitive.

Claim	Keyword	Description
Issuer	iss	Identifies the principal that issued the token. The claim processing is application specific.
Subject	sub	The subject or user of the token. The name is scoped to be globally or locally unique.
Audience	aud	The recipients the token is intended for. Implemented as an array of strings.
Expiration	exp	The time after which the token expires and must be rejected.

See [RFC 7519: JSON Web Tokens](#) for more information.

Client authorization

Overview and options for ONTAP client authorization

The ONTAP OAuth 2.0 implementation is designed to be flexible and robust, providing the features you need to secure your ONTAP environment. There are several mutually exclusive configuration options available. The authorization decisions are ultimately based on the ONTAP REST roles either contained in or derived from the OAuth 2.0

access tokens.



You can only use [ONTAP REST roles](#) when configuring authorization for OAuth 2.0. The earlier ONTAP traditional roles are not supported.

ONTAP applies the single most appropriate authorization option based on your configuration. See [How ONTAP determines access](#) for more about how ONTAP makes client access decisions.

OAuth 2.0 self-contained scopes

These scopes contain one or more custom REST roles, each encapsulated within a single string in the access token. They are independent of the ONTAP role definitions. You need to configure the scope strings at your authorization server. See [Self-contained OAuth 2.0 scopes](#) for more information.

Local ONTAP REST roles

A single named REST role, either builtin or custom, can be used. The scope syntax for a named role is `ontap-role-<URL-encoded-ONTAP-role-name>`. For example, if the ONTAP role is `admin` the scope string will be `ontap-role-admin`.

Users

The username in the access token defined with access to the application "http" can be used. A user is tested in the following order based on the defined authentication method: password, domain (Active Directory), nsswitch (LDAP).

Groups

The authorization servers can be configured to use ONTAP groups for authorization. If the local ONTAP definitions are examined but no access decision can be made, the Active Directory ("domain") or LDAP ("nsswitch") groups are used. Group information can be specified in one of two ways:

- OAuth 2.0 scope string

Supports confidential applications using the client credentials flow where there is no user with a group membership. The scope should be named `ontap-group-<URL-encoded-ONTAP-group-name>`. For example, if the group is "development" the scope string will be "ontap-group-development".

- In the "group" claim

This is intended for access tokens issued by ADFS using the resource owner (password grant) flow.

See [Working with OAuth 2.0 or SAML IdP groups in ONTAP](#) for more information.

Self-contained OAuth 2.0 scopes in ONTAP

Self-contained scopes are strings carried in the access token. Each is a complete custom role definition and includes everything ONTAP needs to make an access decision. The scope is separate and distinct from any of the REST roles defined within ONTAP itself.

Format of the scope string

At a base level, the scope is represented as a contiguous string and composed of six colon-separated values. The parameters used in the scope string are described below.

ONTAP literal

The scope must begin with the literal value `ontap` in lowercase. This identifies the scope as specific to ONTAP.

Cluster

This defines which ONTAP cluster the scope applies to. The values can include:

- Cluster UUID

Identifies a single cluster.

- Asterisk (*)

Indicates the scope applies to all clusters.

You can use the ONTAP CLI command `cluster identity show` to display the UUID of your cluster. If not specified, the scope applies to all clusters. Learn more about `cluster identity show` in the [ONTAP command reference](#).

Role

The name of the REST role contained in the self-contained scope. This value is not examined by ONTAP or matched to any existing REST roles defined to ONTAP. The name is used for logging.

Access level

This value indicates the access level applied to the client application when using the API endpoint in the scope. There are six possible values as described in the table below.

Access level	Description
none	Denies all access to the specified endpoint.
readonly	Allows only read access using GET.
read_create	Allows read access as well as the creation of new resource instances using POST.
read_modify	Allows read access as well as the ability to update existing resources using PATCH.
read_create_modify	Allows all access except delete. The allowed operations include GET (read), POST (create), and PATCH (update).
all	Allows full access.

SVM

The name of the SVM within the cluster the scope applies to. Use the `*` value (asterisk) to indicate all SVMs.



This feature is not fully supported with ONTAP 9.14.1. You can ignore the SVM parameter and use an asterisk as a placeholder. Review the [ONTAP release notes](#) to check for future SVM support.

REST API URI

The complete or partial path to a resource or set of related resources. The string must begin with `/api`. If you don't specify a value, the scope applies to all API endpoints at the ONTAP cluster.

Scope examples

A few examples of self-contained scopes are presented below.

`ontap:*:joes-role:read_create_modify:*:/api/cluster`

Provides the user assigned this role read, create, and modify access to the `/cluster` endpoint.

CLI administrative tool

To make the administration of the self-contained scopes easier and less error-prone, ONTAP provides the CLI command `security oauth2 scope` to generate scope strings based on your input parameters.

The command `security oauth2 scope` has two use cases based on your input:

- CLI parameters to scope string

You can use this version of the command to generate a scope string based on the input parameters.

- Scope string to CLI parameters

You can use this version of the command to generate the command parameters based on the input scope string.

Example

The following example generates a scope string with the output included after the command example below. The definition applies to all clusters.

```
security oauth2 scope cli-to-scope -role joes-role -access readonly -api  
/api/cluster
```

`ontap:*:joes-role:readonly:*:/api/cluster`

Learn more about `security oauth2 scope` in the [ONTAP command reference](#).

OAuth 2.0 external role mapping in ONTAP

An external role is defined at an identity provider configured for use by ONTAP. You can create and administer mapping relationships between these external roles and the ONTAP roles using the ONTAP CLI.



You can also configure the external role mapping feature using the ONTAP REST API. Learn more in the [ONTAP automation documentation](#).

External roles in an access token

Here's a fragment of a JSON access token containing two external roles.

```
...
"appidacr": "1",
"family_name": "User",
"name": "Test User 1",
"oid": "4c2215c7-6d52-40a7-ce71-096fa41379ba",
"roles": [
    "Global Administrator",
    "Application Administrator"
],
"ver": "1.0",
...
```

Configuration

You can use the ONTAP command line interface to administer the external role mapping feature.

Create

You can define a role mapping configuration with the `security login external-role-mapping create` command. You need to be at the ONTAP **admin** privilege level to issue this command as well as the related options.

Parameters

The parameters used to create a group mapping are described below.

Parameter	Description
<code>external-role</code>	The name of the role defined at the external identity provider.
<code>provider</code>	The name of the identity provider. This should be the identifier for the system.
<code>ontap-role</code>	Indicates the existing ONTAP role the external role is mapped to.

Example

```
security login external-role-mapping create -external-role "Global
Administrator" -provider entra -ontap-role admin
```

Learn more about `security login external-role-mapping create` in the [ONTAP command reference](#).

Additional CLI operations

The command supports several additional operations, including:

- Show
- Modify
- Delete

Related information

- [ONTAP command reference](#)

How ONTAP determines client access

To properly design and implement OAuth 2.0, you need to understand how your authorization configuration is used by ONTAP to make access decisions for the clients. The major steps used to determine access are presented below based on the ONTAP release.



There were no significant OAuth 2.0 updates with ONTAP 9.15.1. If you are using the 9.15.1 release, refer to the description for ONTAP 9.14.1.

Related information

- [OAuth 2.0 features supported in ONTAP](#)

ONTAP 9.16.1

ONTAP 9.16.1 expands the standard OAuth 2.0 support to include Microsoft Entra ID specific extensions for native Entra ID groups as well as external role mapping.

Determine client access for ONTAP 9.16.1

Step 1: Self-contained scopes

If the access token contains any self-contained scopes, ONTAP examines these scopes first. If there are no self-contained scopes, go to step 2.

With one or more self-contained scopes present, ONTAP applies each scope until an explicit **ALLOW** or **DENY** decision can be made. If an explicit decision is made, processing ends.

If ONTAP can't make an explicit access decision, continue to step 2.

Step 2: Check the local roles flag

ONTAP examines the boolean parameter `use-local-roles-if-present`. The value of this flag is set separately for each authorization server defined to ONTAP.

- If the value is `true` continue to step 3.
- If the value is `false` processing ends and access is denied.

Step 3: Named ONTAP REST role

If the access token contains a named REST role in the `scope` or `scp` field, or as a claim, ONTAP uses the role to make the access decision. This always results in an **ALLOW** or **DENY** decision and processing ends.

If there is no named REST role or the role is not found, continue to step 4.

Step 4: Users

Extract the username from the access token and attempt to match it to users that have access to the application "http". The users are examined based on the authentication method in the following order:

- password
- domain (Active Directory)
- nsswitch (LDAP)

If a matching user is found, ONTAP uses the role defined for the user to make an access decision. This always result in an **ALLOW** or **DENY** decision and processing ends.

If a user is not matched or if there's no username in the access token, continue to step 5.

Step 5: Groups

If one or more groups are included, the format is examined. If the groups are represented as UUIDs, an internal group mapping table is searched. If there's a group match and an associated role, ONTAP uses the role defined for the group to make an access decision. This always result in an **ALLOW** or **DENY** decision and processing ends. For more information see [Working with OAuth 2.0 or SAML IdP groups in ONTAP](#).

If groups are represented as names and configured with domain or nsswitch authorization, ONTAP attempts to match them to an Active Directory or LDAP group, respectively. If there's a group match, ONTAP uses the role defined for the group to make an access decision. This always result in an **ALLOW** or **DENY** decision and processing ends.

If there's no group match or if there's no group in the access token, access is denied and processing ends.

ONTAP 9.14.1

Initial OAuth 2.0 support is introduced with ONTAP 9.14.1 based on the standard OAuth 2.0 features.

Determine client access for ONTAP 9.14.1

Step 1: Self-contained scopes

If the access token contains any self-contained scopes, ONTAP examines these scopes first. If there are no self-contained scopes, go to step 2.

With one or more self-contained scopes present, ONTAP applies each scope until an explicit **ALLOW** or **DENY** decision can be made. If an explicit decision is made, processing ends.

If ONTAP can't make an explicit access decision, continue to step 2.

Step 2: Check the local roles flag

ONTAP examines the boolean parameter `use-local-roles-if-present`. The value of this flag is set separately for each authorization server defined to ONTAP.

- If the value is `true` continue to step 3.
- If the value is `false` processing ends and access is denied.

Step 3: Named ONTAP REST role

If the access token contains a named REST role in the `scope` or `scp` field, ONTAP uses the role to make the access decision. This always results in an **ALLOW** or **DENY** decision and processing ends.

If there is no named REST role or the role is not found, continue to step 4.

Step 4: Users

Extract the username from the access token and attempt to match it to users that have access to the application "http". The users are examined based on the authentication method in the following order:

- password
- domain (Active Directory)
- nsswitch (LDAP)

If a matching user is found, ONTAP uses the role defined for the user to make an access decision. This always result in an **ALLOW** or **DENY** decision and processing ends.

If a user is not matched or if there's no username in the access token, continue to step 5.

Step 5: Groups

If one or more groups are included and configured with domain or nsswitch authorization, ONTAP attempts to match them to an Active Directory or LDAP group, respectively.

If there's a group match, ONTAP uses the role defined for the group to make an access decision. This always result in an **ALLOW** or **DENY** decision and processing ends.

If there's no group match or if there's no group in the access token, access is denied and processing ends.

OAuth 2.0 deployment scenarios with ONTAP

There are several configuration options available when defining an authorization server to ONTAP. Based on these options, you can define an authorization server appropriate for your environment using one of several deployment scenarios.

Summary of the configuration parameters

There are several configuration parameters available when defining an authorization server to ONTAP. These parameters are generally supported in all the administrative interfaces.



The name used for an individual parameter or field can vary depending on the ONTAP administrative interface. To accommodate the differences in the administrative interfaces, a single generic name is used for each parameter in the table. The exact name used with a specific interface should be obvious based on the context.

Parameter	Description
Name	The name of the authorization server as it is known to ONTAP.
Application	The ONTAP internal application the definition applies to. This must be http .
Issuer URI	The FQDN with path identifying the site or organization that issues the tokens.
Provider JWKS URI	The FQDN with path and file name where ONTAP obtains the JSON Web Key Sets used to validate the access tokens.
JWKS refresh interval	The time interval determining how often ONTAP refreshes certificate information from the provider JWKS URI. The value is specified in ISO-8601 format.
Introspection endpoint	The FQDN with path that ONTAP uses to perform remote token validation through introspection.
Client ID	The name of the client as defined at the authorization server. When this value is included, you also need to provide the associated client secret based on the interface.
Outgoing proxy	This is to provide access to the authorization server when ONTAP is behind a firewall. The URI must be in curl format.
Use local roles if present	A boolean flag determining if the local ONTAP definitions are used, including a named REST role and local users.
Remote user claim	An alternative name that ONTAP uses to match local users. Use the <code>sub</code> field in the access token to match the local username.
Audience	This field defines the endpoints where the access token can be used.

Deployment scenarios

Several common deployment scenarios are presented below. They are organized based on whether token validation is performed locally by ONTAP or remotely by the authorization server. Each scenario includes a list of the required configuration options. See [Deploy OAuth 2.0 in ONTAP](#) for examples of the configuration commands.



After defining an authorization server, you can display its configuration through the ONTAP administrative interface. For example, use the command `security oauth2 client show` with the ONTAP CLI.

Local validation

The following deployment scenarios are based on ONTAP performing token validation locally.

Use self-contained scopes without a proxy

This is the simplest deployment using only OAuth 2.0 self-contained scopes. None of the local ONTAP identity definitions are used. You need to include the following parameters:

- Name
- Application (http)
- Provider JWKS URI
- Issuer URI

You also need to add the scopes at the authorization server.

Use self-contained scopes with a proxy

This deployment scenario uses the OAuth 2.0 self-contained scopes. None of the local ONTAP identity definitions are used. But the authorization server is behind a firewall and so you need to configure a proxy. You need to include the following parameters:

- Name
- Application (http)
- Provider JWKS URI
- Outgoing proxy
- Issuer URI
- Audience

You also need to add the scopes at the authorization server.

Use local user roles and default username mapping with a proxy

This deployment scenario uses local user roles with default name mapping. The remote user claim uses the default value of `sub` and so this field in the access token is used to match the local username. The username must be 40 characters or less. The authorization server is behind a firewall so you also need to configure a proxy. You need to include the following parameters:

- Name
- Application (http)
- Provider JWKS URI
- Use local roles if present (true)
- Outgoing proxy
- Issuer

You need to make sure the local user is defined to ONTAP.

Use local user roles and alternate username mapping with a proxy

This deployment scenario uses local user roles with an alternate username which is used to match a local ONTAP user. The authorization server is behind a firewall, so you need to configure a proxy. You need to include the following parameters:

- Name
- Application (http)
- Provider JWKS URI
- Use local roles if present (true)
- Remote user claim
- Outgoing proxy
- Issuer URI
- Audience

You need to make sure the local user is defined to ONTAP.

Remote introspection

The following deployment configurations are based on ONTAP performing token validation remotely through introspection.

Use self-contained scopes with no proxy

This is a simple deployment based on using the OAuth 2.0 self-contained scopes. None of the ONTAP identity definitions are used. You must include the following parameters:

- Name
- Application (http)
- Introspection endpoint
- Client ID
- Issuer URI

You need to define the scopes as well as the client and client secret at the authorization server.

Related information

- [security oauth2 client show](#)

ONTAP client authentication using OAuth 2.0 Mutual TLS

Depending on your security needs, you can optionally configure Mutual TLS (mTLS) to implement strong client authentication. When used with ONTAP as part of an OAuth 2.0 deployment, mTLS guarantees the access tokens are only used by the clients to which they were originally issued.

Mutual TLS with OAuth 2.0

Transport Layer Security (TLS) is used to establish a secure communication channel between two applications, typically a client browser and web server. Mutual TLS extends this by providing strong identification of the client through a client certificate. When used in an ONTAP cluster with OAuth 2.0, the base mTLS functionality

is extended by creating and using sender-constrained access tokens.

A sender-constrained access token can only be used by the client to which it was originally issued. To support this feature, a new confirmation claim (`cnf`) is inserted into the token. The field contains property `x5t#S256` which holds a digest of the client certificate used when requesting the access token. This value is verified by ONTAP as part of validating the token. Access tokens issued by authorization servers that are not sender-constrained do not include the additional confirmation claim.

You need to configure ONTAP to use mTLS separately for each authorization server. For example, the CLI command `security oauth2 client` includes the parameter `use-mutual-tls` to control mTLS processing based on three values as shown in the table below.



In each configuration, the outcome and action taken by ONTAP is dependent on the configuration parameter value as well as the contents of the access token and the client certificate. The parameters in the table are organized from the least to the most restrictive.

Parameter	Description
none	OAuth 2.0 mutual TLS authentication is completely disabled for the authorization server. ONTAP will not perform mTLS client certificate authentication even if the confirmation claim is present in the token or a client certificate is supplied with the TLS connection.
request	OAuth 2.0 mutual TLS authentication is enforced if a sender-constrained access token is presented by the client. That is, mTLS is enforced only if the confirmation claim (with property <code>x5t#S256</code>) is present in the access token. This is the default setting.
required	OAuth 2.0 mutual TLS authentication is enforced for all access tokens issued by the authorization server. Therefore, all access tokens must be sender-constrained. Authentication and the REST API request fail if the confirmation claim is not present in the access token or there is an invalid client certificate.

High-level implementation flow

The typical steps involved when using mTLS with OAuth 2.0 in an ONTAP environment are presented below. See [RFC 8705: OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens](#) for more details.

Step 1: Create and install a client certificate

Establishing client identity is based on proving knowledge of a client private key. The corresponding public key is placed in a signed X.509 certificate presented by the client. At a high level, the steps involved in creating the client certificate include:

1. Generate a public and private key pair
2. Create a certificate signing request
3. Send the CSR file to a well-known CA
4. CA verifies the request and issues the signed certificate

You can normally install the client certificate in your local operating system or use it directly with a common utility such as curl.

Step 2: Configure ONTAP to use mTLS

You need to configure ONTAP to use mTLS. This configuration is done separately for each authorization server. For example, with the CLI the command `security oauth2 client` is used with the optional parameter `use-mutual-tls`. See [Deploy OAuth 2.0 in ONTAP](#) for more information.

Step 3: Client requests an access token

The client needs to request an access token from the authorization server configured to ONTAP. The client application must use mTLS with the certificate created and installed in step 1.

Step 4: Authorization server generates the access token

The authorization server verifies the client request and generates an access token. As part of this, it creates a message digest of the client certificate which is included in the token as a confirmation claim (field `cnf`).

Step 5: Client application presents the access token to ONTAP

The client application makes a REST API call to the ONTAP cluster and includes the access token in the authorization request header as a **bearer token**. The client must use mTLS with the same certificate used to request the access token.

Step 6: ONTAP verifies client and token.

ONTAP receives the access token in an HTTP request as well as the client certificate used as part of mTLS processing. ONTAP first validates the signature in the access token. Based on the configuration, ONTAP generates a message digest of the client certificate and compares it to the confirmation claim `cnf` in the token. If the two values match, ONTAP has confirmed the client making the API request is the same client the access token was originally issued to.

Related information

- [security oauth2 client](#)

Configure and deploy

Prepare to deploy OAuth 2.0 with ONTAP

Before configuring OAuth 2.0 in an ONTAP environment, you should prepare for the deployment. A summary of the major tasks and decisions is included below. The arrangement of the sections is generally aligned with the order you should follow. But while it's applicable for most deployments, you should adapt it to your environment as needed. You should also consider creating a formal deployment plan.



Based on your environment, you can select the configuration for the authorization servers defined to ONTAP. This includes the parameter values you need to specific for each type of deployment. See [OAuth 2.0 deployment scenarios](#) for more information.

Protected resources and client applications

OAuth 2.0 is an authorization framework for controlling access to protected resources. Given this, an important first step with any deployment is to determine what the available resources are and which clients need access to them.

Identify client applications

You need to decide which clients will use OAuth 2.0 when issuing REST API calls and what API endpoints they need access to.

Review existing ONTAP REST roles and local users

You should review the existing ONTAP identity definitions, including the REST roles and local users. Depending on how you configure OAuth 2.0, these definitions can be used for making access decisions.

Global transition to OAuth 2.0

While you might implement OAuth 2.0 authorization gradually, you can also move all the REST API clients to OAuth 2.0 immediately by setting a global flag for each authorization server. This allows access decisions to be made based on your existing ONTAP configuration without the need for creating self-contained scopes.

Authorization servers

The authorization servers play an important role in your OAuth 2.0 deployment by issuing access tokens and enforcing administrative policy.

Select and install the authorization server

You need to select and install one or more authorization servers. It's important to become familiar with the configuration options and procedures of your identity providers, including how to define scopes. Note that some authorization servers, including Microsoft Entra ID, represent groups using UUIDs instead of names.

Determine if the authorization root CA certificate needs to be installed

ONTAP uses the authorization server's certificate to validate the signed access tokens presented by the clients. To do this, ONTAP needs the root CA certificate and any intermediate certificates. These might be pre-installed with ONTAP. If not, you need to install them.

Assess network location and configuration

If the authorization server is behind a firewall, ONTAP needs to be configured to use a proxy server.

Client authentication and authorization

There are several aspects of client authentication and authorization you need to consider.

Self-contained scopes or local ONTAP identity definitions

At a high level, you can either define self-contained scopes defined at the authorization server or rely on the existing local ONTAP identity definitions including roles and users.

Options with local ONTAP processing

If you use the ONTAP identity definitions, you must decide which to apply, including:

- Named REST role
- Match local users
- Active Directory or LDAP groups

Local validation or remote introspection

You need to decide if the access tokens will be validated locally by ONTAP or at the authorization server through introspection. There are also several related values to consider, such as the refresh interval.

Sender-constrained access tokens

For environments requiring a high level of security, you can use send-constrained access tokens based on mTLS. This requires a certificate for each client.

Groups as UUIDs and identity mapping

If you are using an authorization server that represents groups using UUIDs, you need to plan how to map

these to group names and possibly to associated roles.

Administrative interface

You can perform administration of OAuth 2.0 through any of the ONTAP interfaces, including:

- Command line interface
- System Manager
- REST API

How clients request access tokens

The client applications must request access tokens directly from the authorization server. You need to decide how this will be done, including the grant type.

Configure ONTAP

There are several ONTAP configuration tasks you need to perform.

Define REST roles and local users

Based on your authorization configuration, local ONTAP identify processing can be used. In this case, you need to review and define the REST roles and user definitions. And depending on your authorization server, this might also include administering groups based on UUID values.

Core configuration

There are three major steps needed to perform the core ONTAP configuration, including:

- Optionally install the root certificate (and any intermediate certificates) for the CA that signed the authorization server's certificate.
- Define the authorization server.
- Enable OAuth 2.0 processing for the cluster.

Deploy OAuth 2.0 in ONTAP

Deploying the core OAuth 2.0 functionality involves three primary steps.

Before you begin

You must prepare for the OAuth 2.0 deployment before configuring ONTAP. For example, you need to assess the authorization server, including how its certificate was signed and if it's behind a firewall. See [Prepare to deploy OAuth 2.0 with ONTAP](#) for more information.

Step 1: Install the authorization server root CA certificates

ONTAP includes a large number of pre-installed root CA certificates. So in many cases, the certificate for your authorization server will be immediately recognized by ONTAP without additional configuration. But depending on how the authorization server certificate was signed, you may need to install a root CA certificate and any intermediate certificates.

Follow the instructions provided below to install the certificate if it's needed. You should install all the required certificates at the cluster level.

Choose the correct procedure based on how you access ONTAP.

Example 1. Steps

System Manager

1. In System Manager, select **Cluster > Settings**.
2. Scroll down to the **Security** section.
3. Click → next to **Certificates**.
4. Under the **Trusted certificate authorities** tab click **Add**.
5. Click **Import** and select the certificate file.
6. Complete the configuration parameters for your environment.
7. Click **Add**.

CLI

1. Begin the installation:

```
security certificate install -type server-ca
```

2. Look for the following console message:

```
Please enter Certificate: Press <Enter> when done
```

3. Open the certificate file with a text editor.
4. Copy the entire certificate including the following lines:

```
-----BEGIN CERTIFICATE-----  
<certificate_value>  
-----END CERTIFICATE-----
```

5. Paste the certificate into the terminal after the command prompt.
6. Press **Enter** to complete the installation.
7. Confirm the certificate is installed using one of the following:

```
security certificate show-user-installed
```

```
security certificate show
```

Step 2: Configure the authorization server

You need to define at least one authorization server to ONTAP. You should choose the parameter values based on your configuration and deployment plan. Review [OAuth2 deployment scenarios](#) to determine the exact parameters needed for your configuration.



To modify an authorization server definition, you can delete the existing definition and create a new one.

The example provided below is based on the first simple deployment scenario at [Local validation](#). Self-

contained scopes are used without a proxy.

Choose the correct procedure based on how you access ONTAP. The CLI procedure uses symbolic variables that you need to replace before issuing the command.

Example 2. Steps

System Manager

1. In System Manager, select **Cluster > Settings**.
2. Scroll down to the **Security** section.
3. Click **+** next to **OAuth 2.0 authorization**.
4. Select **More options**.
5. Provide the required values for your deployment, such as:
 - Name
 - Application (http)
 - Provider JWKS URI
 - Issuer URI
6. Click **Add**.

CLI

1. Create the definition again:

```
security oauth2 client create -config-name <NAME> -provider-jwks-uri
<URI_JWKS> -application http -issuer <URI_ISSUER>
```

For example:

```
security oauth2 client create \
-config-name auth0 \
-provider-jwks-uri https://superzap.dev.netapp.com:8443/realms/my-
realm/protocol/openid-connect/certs \
-application http \
-issuer https://superzap.dev.netapp.com:8443/realms/my-realm
```

Learn more about `security oauth2 client create` in the [ONTAP command reference](#).

Step 3: Enable OAuth 2.0

The final step is to enable OAuth 2.0. This is a global setting for the ONTAP cluster.



Don't enable OAuth 2.0 processing until you confirm that ONTAP, the authorization servers, and any supporting services have all been properly configured.

Choose the correct procedure based on how you access ONTAP.

Example 3. Steps

System Manager

1. In System Manager, select **Cluster > Settings**.
2. Scroll down to the **Security section**.
3. Click → next to **OAuth 2.0 authorization**.
4. Enable **OAuth 2.0 authorization**.

CLI

1. Enable OAuth 2.0:

```
security oauth2 modify -enabled true
```

2. Confirm OAuth 2.0 is enabled:

```
security oauth2 show
Is OAuth 2.0 Enabled: true
```

Related information

- [security certificate install](#)
- [security certificate show](#)
- [security oauth2 modify](#)
- [security oauth2 show](#)

Issue an ONTAP REST API call using OAuth 2.0

The OAuth 2.0 implementation in ONTAP supports REST API client applications. You can issue a simple REST API call using curl to get started using OAuth 2.0. The example presented below retrieves the ONTAP cluster version.

Before you begin

You must configure and enable the OAuth 2.0 feature for your ONTAP cluster. This includes defining an authorization server.

Step 1: Acquire an access token

You need to acquire an access token to use with the REST API call. The token request is performed outside of ONTAP and the exact procedure depends on the authorization server and its configuration. You might request the token through a web browser, with a curl command, or using a programming language.

For illustration purposes, an example of how an access token can be requested from Keycloak using curl is presented below.

Keycloak example

```
curl --request POST \
--location
'https://superzap.dev.netapp.com:8443/realms/peterson/protocol/openid-
connect/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'client_id=dp-client-1' \
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'client_secret=5iTUF9QKLGxAoYa1iR33v1D5A2xq09V7'
```

You should copy and save the returned token.

Step 2: Issue the REST API call

After you have a valid access token, you can use a curl command with the access token to issue a REST API call.

Parameters and variables

The two variables in the curl example are described in the table below.

Variable	Description
\$FQDN_IP	The fully qualified domain name or IP address of the ONTAP management LIF.
\$ACCESS_TOKEN	The OAuth 2.0 access token issued by the authorization server.

You should first set these variables in the Bash shell environment before issuing the curl example. For example, in the Linux CLI type the following command to set and display the FQDN variable:

```
FQDN_IP=172.14.31.224
echo $FQDN_IP
172.14.31.224
```

After both variables are defined in your local Bash shell, you can copy the curl command and paste it into the CLI. Press **Enter** to substitute the variables and issue the command.

Curl example

```
curl --request GET \
--location "https://$FQDN_IP/api/cluster?fields=version" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

Copyright information

Copyright © 2026 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.