



Concepts

ONTAP 9

NetApp
January 23, 2026

Table of Contents

- Concepts 1
 - OAuth 2.0 authorization servers and access tokens in ONTAP 1
 - OAuth 2.0 authorization servers 1
 - OAuth 2.0 features supported in ONTAP 2
 - Using OAuth 2.0 access tokens 3
 - Client authorization 4
 - Overview and options for ONTAP client authorization 4
 - Self-contained OAuth 2.0 scopes in ONTAP 5
 - OAuth 2.0 external role mapping in ONTAP 7
 - How ONTAP determines client access 8
 - OAuth 2.0 deployment scenarios with ONTAP 11
 - Summary of the configuration parameters 11
 - Deployment scenarios 11
 - ONTAP client authentication using OAuth 2.0 Mutual TLS 13
 - Mutual TLS with OAuth 2.0 13
 - High-level implementation flow 14

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. |

| Claim | Keyword | Description |
|------------|---------|--|
| 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. |

| Access level | Description |
|--------------------|--|
| 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 identify 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 |
|---------------|--|
| external-role | The name of the role defined at the external identity provider. |
| provider | The name of the identity provider. This should be the identifier for the system. |
| ontap-role | 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 supported 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](#)

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.