NetApp BlueXP authentication and authorization

Basic concepts and terminology

You should be familiar with the basic authentication and authorization concepts before using the NetApp BlueXP REST APIs.

OAuth 2.0 authorization framework

OAuth 2.0 (OAuth2) is an authorization framework that allows client applications to obtain limited access to the protected resources at an HTTP service. The framework addresses many of the deficiencies of the traditional client/server authentication model by introducing an authorization layer as defined through four distinct roles. Within this framework, the client doesn't directly use the credentials of the resource owner at the server. Instead an authorization server issues an access token which is then presented to the server by the client.

OpenID Connect protocol

OpenID Connect (OIDC) is a protocol that extends and enhances the OAuth 2.0 framework. It allows third-party applications to confirm the identity of end users and obtain additional profile information.

Auth0 platform

Auth0 is an authentication and authorization platform based on the OAuth 2.0 standard and the OpenID Connect protocol. It orchestrates user authentication as well as authorization to protected resources. The Auth0 authorization server issues access tokens to the client applications. The NetApp BlueXP services use Auth0 to authenticate and authorize clients.

Auth0 user

An Auth0 user corresponds to a BlueXP customer account and can be used to authorize an application or client (including a NetApp BlueXP service) to access the account. The application's access is limited to the scope of the authorization granted (such as read or write). A BlueXP customer account obtains its Auth0 user as part of the onboarding process at the BlueXP website.

Access tokens

There are two types of tokens you can acquire and use to access the BlueXP REST APIs. Each token is valid for a fixed amount of time before it expires. Tokens are not revocable.

There are several ways the two access tokens can be used depending on the specific REST AI

Type	Description
User token	This is a token identifying a user who accesses the BlueXP through the REST API or web UI. A federated or non-federated authentication model can be used when requesting the token. See User access tokens for more information. The token is valid for six hours.
Service token	This token is used by software applications to perform a REST API call. Example clients include internal BlueXP service tasks and external clients. The token is valid for 24 hours. You can use client secret authentication or private key JWT authentication to create the service access token.

Grant type

A *grant type* within the OAuth 2.0 framework defines how an access token is acquired. The different grant types are adapted to several use cases based on the characteristics of the applications. Implicit with each grant type is the type of credentials provided and how identity is confirmed. See Grant types for more information.

Scope

A *scope* within the OAuth 2.0 framework provides a way to limit access to data and other resources. It can optionally be included when requesting an access token and is then applied at the resource server when the token is presented by the client.

JWT encoding

The JSON web token (JWT) is a standard way of representing claims between parties. It is encoded using base64url. See User access tokens for more information including an example of a token.

User access tokens

There are two types of access tokens that can be used to establish identity and authorization as part of performing a REST API call. The user access token is generally more common with customer automation applications.

Authentication models

There are two authentication models available when requesting a user access token. The one you select depends on your authentication environment. Each authentication model is supported through a different grant type. See Grant types for more information.

Federated authentication

Federated authentication allows multiple identity domains to be joined, providing shared access to resources across separate enterprise organizations. Each organization maintains its own identity management system which is recognized by the other participating organizations. Users who sign in using a federated user account must first generate a long-lived refresh token.

Non-federated authentication

Non-federated authentication is based on identity and resource access within a single domain. Users who sign in using a non-federated user account can obtain either a long-lived refresh token or a regular access token.

Contents of a user token

A user token is formatted as a standard JWT token and contains information about an individual user. It is normally includes the following fields:

Fields	**Description**
Header	JSON object encoded using base64url encoding. Contains information about the encryption algorithm used to encode the token and the public key ID for token validation.
Standard claims	JSON payload contains the user identification information, audience, the expiration time, scope of resource access given to the user.
Custom claims	Contains customized information of the user for example, the full name of the user, and so on.
Signature	Used to validate the issuer and the encoded message in the token.

Example of a user access token

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ik16VTNRa0kyUlRFeFJqZzVNRE5EUWpORl

JVVTNRVVpCTlVGQ05URkJRVFpDTWtFMVFqZzFSZyJ9.eyJodHRwOi8vY2xvdWQubmV0YXBwL mNvbS9md

WxsX25hbWUiOiJSYW51IEt1bmR1IiwiaHR0cDovL2Nsb3VkLm5ldGFwcC5jb20vZW1haWxfdm VyaWZpZ

WQiOnRydWUsImh0dHA6Ly9jbG91ZC5uZXRhcHAuY29tL2Nvbm5lY3Rpb25faWQiOiJjb25fZ1B qZmNzM

zVTUGZpemg0YiIsImh0dHA6Ly9jbG9xxxxxxoooooooooooooooooooooooooooooooooooooooo ooll

XRhcHAuY29tL2ludGVybmFsIjoiTmV0QXBwIiwiaXNzIjoiaHR0cHM6Ly9zdGFnallllllllllll

W5nLW5ldGFwcC1jbG91ZC1hY2NvdW50LmF1dGgwLmNvbS8iLCJzdWIiOiJhdXRoMHw2MDZi NjQ2N2QwZ

TA0MzAwNjk2ZjNjM2QiLCJhdWQiOlsiaHR0cHM6Ly9hcGkuY2xvdWQubmV0YXBwLxxxxxxxxx xxHBzO

i8vc3RhZ2luZy1uZXRhcHAtY2xvdWQtYWNjb3VudC5hdXRoMC5jb20vdXNlcmluZm8iXSwiaWF 0Ijox

NjM1MjQwMTM1LCJleHAiOjE2MzUyNjE3MzUsImF6cCI6IkZpaXZSWnF4TFdMamNsQ1V5Rm 1hYlo3MXJw

Q1IxNE5UIiwic2NvcGUiOiJvcGVuaWQgcHJvZmlsZSBlbWFpbCBjYzp1cGRhdGUtcGFzc3dvcmQi fQ.

mwavS7rJYUwMrnu0CL_8J3N-WJG2_UwdD3bplxAR-p6m8_2vZH8o4SjJAvaL3JwvQqcVHoh7YfyWd5TH

SKQAJawGyTQnqdOdp_2lVbEYQdncGRe9Ve22rWEvW

HHHCjr1xb8TKueYBCNgOkpl2LFlc3n3oOPqkf6I8iIMKi31-mC8XDqeiRbdW7nfvyypzRbfSAALHjszj

4LjFfyASO8GDenKZtg4egWzsrTYAQ6JSh4H-QBncFl0DzzJGA1hOKfJohPmzlG7CfKafzemHnXG9pkRz

AfLVrnwBXuapKUzyL--tU4jzpob5vji1bWAlkXo_p8SqJY4TzU99tiSp470bw

HEADER:ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT",
  "kid": "MzU3QkI2RTExRjg5MDNDQjNFRUU3QZXXXXXNTFBQTZCMkE1Qjg1Rg"
}
```

PAYLOAD:DATA

```
{
  "http://cloud.netapp.com/full_name": "<full_name>",
  "http://cloud.netapp.com/email_verified": true,
  "http://cloud.netapp.com/connection_id": "con_gPjfcs3cccczh4b",
  "http://cloud.netapp.com/is_federated": false,
  "http://cloud.netapp.com/internal": "NetApp",
  "iss": "https://netapp-cloud-account.auth0.com/",
  "sub": "auth0|606b6467d0e04300696f3c3d",
  "aud": [
    "https://api.cloud.netapp.com",
    "https://netapp-cloud-account.auth0.com/userinfo"
  ],
  "iat": 1635240035,
  "exp": 1635000735,
  "azp": "FiivRZqxLWLjclCUyFmabZ71rfffffR14NT",
  "scope": "openid profile email cc:update-password"
}
```

VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload)
)
```

Grant types

There are several OAuth 2.0 grant types available when requesting an access token to use with the NetApp BlueXP. The grant type is based on the application requirements and authentication environment, and determines how the token is requested.

> The refresh token grant is the most secure option when requesting a user access token. It ca
> environment.

Refresh token

This grant type generates a token used with either federated or non-federated authentication. You need to provide the OAuth2-compliant refresh token to the authentication endpoint to generate a long-lived access token. The generated token can be used to access resources across multiple domains. Here is an example of the JSON input used when requesting the token with a description of each parameter.

```
{
  "grant_type": "refresh_token",
  "refresh_token": "YOUR_REFRESH_TOKEN",
  "client_id": "Mu0V1ywgYteI6w1MbD15fKfVIUrNXGWC"
}
```

JSON

Copy

Parameter	Description
grant_type	Must be set to "refresh_token".
refresh_token	The refresh token issued by the OAuth2 server.
client_id	The client ID representing the BlueXP. This value is constant and the same for all customers.

The OAuth server verifies the Client ID and confirms that the request came from an authorized client. It also makes sure the refresh token is valid. After the request is validated, the server returns a response with the refresh token.

> A refresh token is **not** required when using a service account.

Password

This grant type generates a token used with non-federated authentication to access resources within a single domain. You need to provide the username and password login credentials to the authentication endpoint when requesting the access token. Here's an example of the JSON input used when requesting the token with a description of each parameter.

```
{
  "grant_type": "password",
  "username": "YOUR_EMAIL_ADDRESS",
  "password": "YOUR_PASSWORD",
  "audience": "https://api.cloud.netapp.com",
  "client_id": "QC3AgHk6qdbmC7Yyr82ApBwaaJLwRrNO"
}
```

JSON

Copy

Parameter	**Description**
grant_type	Must be set to "password".
username	The user’s email address.
password	The user’s password.
audience	The resource identifier.
client_id	The client ID issued when you registered the application.

Client credentials

In some cases, static clients or applications need to acquire an access token to perform operations on behalf of a user. Applications need to obtain an access token for their own service account without user intervention. In these situations you can use the client credentials grant type. Here is an example of the JSON input used when requesting the token with a description of each parameter.

```
{
   "audience": "https://api.cloud.netapp.com",
   "grant_type": "client_credentials",
   "client_id": "<CLIENT_ID>",
   "client_secret": "<CLIENT_SECRET>"


}
```

JSON

Copy

Parameter	**Description**
audience	The resource identifier.
grant_type	Must be set to "client_credentials".
client_id	The client ID issued when you registered the application.
client_secret	The client secret issued when you registered the application. This should be kept confidential.

The OAuth2 server verifies the application through the client ID with client secret and returns an access token in the response.

Use the NetApp BlueXP REST APIs

There is a general process to prepare and use the authentication and authorization information needed to issue a REST API call using the NetApp BlueXP.

Before you begin

There are several things you should consider before performing REST API call using the BlueXP.

NetApp BlueXP account

An account for the BlueXP is required to access the platform. The system prompts you to create an account when you attempt to sign in to the BlueXP web UI.

Sign up or log in to NetApp BlueXP.

Access token usage scenarios

Every REST API call for the BlueXP requires an access token or a combination of tokens. There might also be specific scope requirements for the tokens.

User token only

You only need to obtain and provide a user access token when issuing an API call.

Service token only

You only need to obtain and provide a service token when issuing an API call. This token is typically required by static clients or other internal applications when accessing the platform service APIs. It can also be used by external clients.

User and service tokens

Both a user token and service token are required.

REST API requirements and your authentication environment

You should review the API reference for the platform services you will use. The reference content includes requirements and other details about the REST calls. You also need to understand your authentication and authorization environment. Based on this you can determine the following requirements:

- Type of token or tokens needed
- Any additional scope requirements for the tokens
- Will federated or non-federated authentication used (user tokens only)
- BlueXP agent ID and client ID requirements

Perform an API call

The following steps describe how to obtain the required authentication information and perform a REST API call.

1. (Optional) Locate the BlueXP agent ID and client ID

Several BlueXP APIs require a BlueXP agent ID to route the request to the appropriate environment. You can also locate the related Client ID. See Get required identifiers for more information.

2. Obtain the access tokens

Depending on the REST API call, you might need a user token, a service token, or both. These tokens can be obtained from the Auth0 authorization service. The tokens need to be included with every API call. For more information, see:

- Create user token
- Create service token

3. Create the required request headers

You need to create several HTTP request headers and include them with each API call. See the applicable API reference to understand which headers should be used. The most common headers are described below. See Your first API call for an example.

Authorization

This header typically contains the user bearer access token.

x-agent-id

This header contains the agent ID for API calls that require it.

4. Issue a REST API call

You can use curl or the programming language of your choice to issue a REST API call. There are also workflow examples available for the ONTAP management API. See ONTAP management workflow processes for more information.