🤝 For selected customers

OAuth 2.0 Token Exchange

PingOne multi-tenant

Intended audience: Selected customers and Ping staff

Preface

We intend to support the OAuth 2.0 Token Exchange specification https://datatracker.ietf.org/doc/html/rfc8693. This RFC can service multiple use cases. This document describes the most common use case. Because this document describes a future capability, the final implementation is subject to change, and the disclaimer applies.

Changelog

Feb 8, 2026 Ivan Mok — made significant improvements to the sample use cases

Feb 3, 2026 Ivan Mok — added token exchange sample use cases

Nov 20, 2025 Ivan Mok — clarified Resources > Attributes > Expressions have access to `context.requestData.clientAssertionHeader[.property]` and `context.requestData.clientAssertion[.claim]` Additionally, clarified that the application using the Token Exchange grant type must be configured with a **Token Endpoint Authentication Method** other than **None**.

Nov 14, 2025 Ivan Mok — revised with more information

May 8, 2025 Ivan Mok

Table of contents

Disclaimer

The information in this document is confidential and proprietary to Ping Identity, Inc. (Ping) and may not be disclosed without the permission of Ping. This document is not subject to your license agreement or any other service or subscription agreement with Ping. Ping has no obligation to pursue any course of business outlined in this document or any related document, or to develop or release any functionality mentioned therein. This document, or any related document and Ping's strategy and possible future developments, products and or platforms directions and functionality are all subject to change and may be changed by Ping at any time for any reason without notice. The information on this document is not a commitment, promise or legal obligation to deliver any material, code or functionality. This document is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This document is for informational purposes and may not be incorporated into a contract. Ping assumes no responsibility for errors or omissions in this document, except if such damages were caused by Ping intentionally or grossly negligent.

All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.

Common PingOne OAuth/OIDC use case

A PingOne administrator registers an OIDC application and a custom resource in PingOne because the PingOne administrator, the developer of the application, and the developer of the custom resource agree to the following terms.

1. PingOne plays the Authorization Server and OpenID Provider roles and issues access tokens, optionally refresh tokens, and ID tokens (if applicable) to the application.
2. The application agrees to include the access token in its request as the `Authorization` HTTP request header to the custom resource.
3. The custom resource agrees to accept the access token as authorization and grants the application access to its data. The custom resource may send a request to PingOne's introspection endpoint to determine the validity of the said access token. Because PingOne's access token is a JWT, the custom resource can also validate the access token itself.

Example 1: Authorization Code flow

As a PingOne administrator, you added Xray as an application and Alpha as a custom resource in PingOne. You passed the application credentials to the developer of Xray and the custom resource credentials to the developer of Alpha.

Configuration

Application record	Custom resource record
Xray • Client ID: `a85f7a70` • Grant type: `Authorization Code`	Alpha • Client ID: `44278071` • Audience: `https://api.example.com/a`

NOTE: In this document, all UUIDs are shown with only the first eight digits for brevity and readability.

Sample requests and response

1. Xray sends to PingOne an authorization request with `scope=openid+a.crud`, `client_id=a85f7a70`, and other parameters.
2. PingOne authenticates the user, obtains the authorization from the user, and returns an authorization code to Xray.
3. Xray sends to PingOne a token request using the authorization code grant type.
4. PingOne returns to Xray an access token. In this access token:
    a. `iss` represents the issuer; the value is the PingOne issuer value of the environment.
    b. `client_id` represents the application that requested the access token; the value is the `client_id` of Xray, which is `a85f7a70`.
    c. `aud` represents the intended audience of the token; the value is the **Audience** value as defined in the custom resource record of Alpha, which is `https://api.example.com/a`.

5. Xray sends to Alpha an API request. For authorization, Xray includes the access token (from step 4) as the `Authorization` HTTP request header value in the API request.
6. Alpha evaluates the access token and understands that:
    a. The issuer of the access token is PingOne.
    b. The application that requested the token is Xray.
    c. The audience of the access token includes itself, Alpha.
7. Alpha sends to PingOne an introspection request to validate the access token.
    a. For client identification and authentication, Alpha uses the client ID and secret as defined in the custom resource record of Alpha (`client_id=44278071`).

    NOTE: Instead of sending an introspection request to PingOne, Alpha *could* validate the digital signature of the access token (because PingOne issues JWT-based access tokens). However, Alpha won't know whether the access token or the associated PingOne user session has been revoked.

8. Alpha returns an API response to Xray.

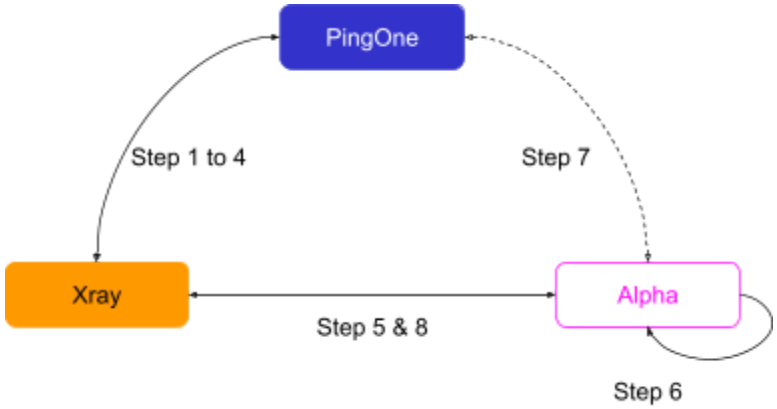

Figure1: Common PingOne OAuth/OIDC use case

Common Token Exchange use case

Referring to the aforementioned Common PingOne OAuth/OIDC use case, Alpha does not have all the data it needs to fulfill the API request it receives from Xray. Alpha needs some data from Beta, which is also a custom resource defined in PingOne.

If all parties support the OAuth 2.0 Token Exchange specification, Alpha can send a Token Exchange token request to PingOne to get an access token that it can use to send an API request to Beta to get the additional data.

In PingOne multi-tenant, only an application is allowed to send token requests to PingOne, regardless of the grant type the application wants to use. A custom resource is not allowed to send token requests to PingOne. As a result, for Alpha to send a Token Exchange token request to PingOne, two things must happen:

1. The PingOne administrator must create an application record for Alpha and pass the application credentials to the developer of Alpha. The **Token Endpoint Authentication Method** must be set to something other than **None**. In other words, client authentication is required at the token endpoint.
2. The developer of Alpha must update Alpha as follows.
    a. When sending introspection requests to PingOne, Alpha must identify itself using the custom resource credentials it previously received from the PingOne administrator.
    b. When sending Token Exchange token requests to PingOne, Alpha must identify itself using the new application credentials.

Essentially, Alpha is now both a custom resource and an application in PingOne the credentials it needs to use to identify itself changes depending on whether it wants to send an introspection request or it wants to send a Token Exchange token request.

Example 2: Token Exchange flow

Let's extend Example 1 from Common PingOne OAuth/OIDC use case to provide a Token Exchange sample.

Configuration

Application record	Custom resource record
Xray • Client ID: `a85f7a70` • Grant type: `Authorization Code`	Alpha • Client ID: `44278071` • Audience: `https://api.example.com/a`
Alpha Token Exchange App • Client ID: `e8f90620` • Grant type: `Token Exchange`	Beta • Client ID: `b0bc42b0` • Audience: `https://api.example.com/b`

NOTE: From Beta's point of view, Alpha is an application sending API requests to it, and Alpha's identifier is `e8f90620`, the Client ID as defined in the application record of Alpha (i.e. Alpha Token Exchange App).

Sample requests and response

After step 7 in Common PingOne OAuth/OIDC use case:

8. Alpha sends to PingOne a token request using the token exchange grant type.
    a. For client identification and authentication, Alpha uses the client ID and secret as defined in the application record of Alpha, i.e. “Alpha Token Exchange App” (`client_id=e8f90620`).
    b. Alpha also includes the following parameters in the token request: `subject_token`, optionally `actor_token`, and `scope`.
9. PingOne validates the subject token and actor token (if provided); for example:
    a. The subject token and the actor token (if provided) are valid JWTs.
    b. The issuer of the subject token and the actor tokens (if provided) matches the issuer of the current PingOne environment.
    c. The associated PingOne user session (if applicable) is valid.
10. PingOne mints an access token based on the attribute mappings as defined in the custom resource record of Beta.
11. PingOne returns to Alpha an access token. In this access token:
    a. `iss` represents the issuer; the value is the PingOne issuer value of the environment.
    b. `client_id` represents the application that requested the access token; the value is the `client_id` of Alpha Token Exchange App, which is `e8f90620`.
    c. `aud` represents the intended audience of the token; the value is the **Audience** value as defined in the custom resource record of Beta, which is `https://api.example.com/b`.
12. Alpha sends to Beta an API request to Beta. For authorization, Alpha includes the access token (from step 11) as the `Authorization` HTTP request header value in the API request.
13. Beta evaluates the access token and understands that:
    a. The issuer of the access token is PingOne.
    b. The application that requested the token is Alpha.
    c. The audience of the access token includes itself, Beta.
14. Beta sends to PingOne an introspection request to validate the access token.

    NOTE: Instead of sending an introspection request to PingOne, Beta *could* validate the digital signature of the access token (because PingOne issues JWT-based access tokens). However, Beta won’t know whether the access token or the associated PingOne user session has been revoked.

15. Beta returns to Alpha an API response. At this point, Alpha has everything it needs to fulfill the API request it receives from Xray (see step 5 in Common PingOne OAuth/OIDC use case).

16. Alpha returns an API response to Xray.

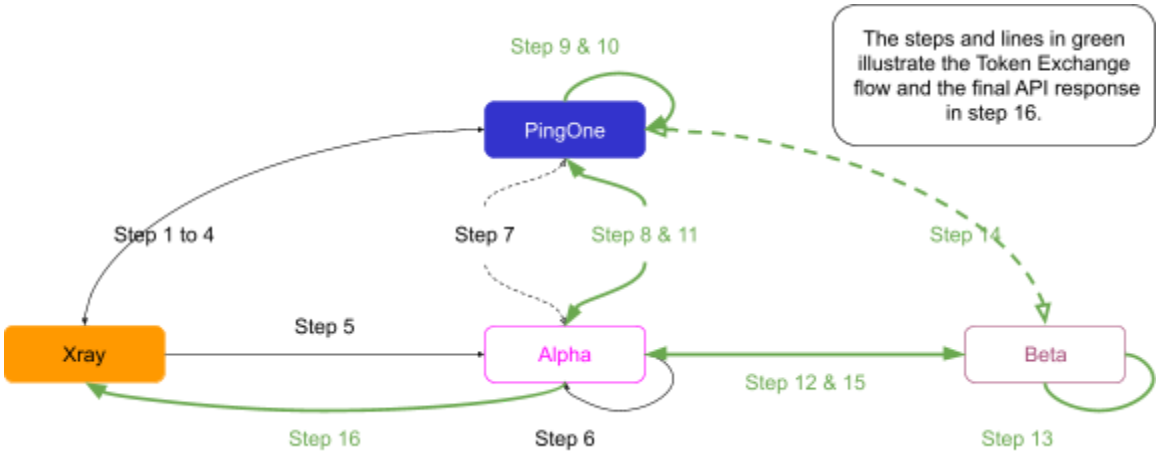

Figure 2: Common Token Exchange use case

Summary of this common Token Exchange use case

- From Xray's point of view, Alpha is a custom resource. Xray sends API requests to Alpha. Each request comes with an `Authorization` header where the value is an access token Xray obtains from PingOne using the authorization code grant type. In this access token:
  - The `iss` is the PingOne issuer value of the environment.
  - The `client_id` is the Client ID of Xray, which is `a85f7a70`.
  - The `aud` is the **Audience** value as defined in the custom resource record of Alpha, which is `https://api.example.com/a`.
- Alpha sends Token Exchange token requests to PingOne to obtain access tokens to access data at Beta. When sending Token Exchange requests to PingOne, Alpha must identify itself using the Client ID as defined in the application record of Alpha (i.e. "Alpha Token Exchange App"), which is `e8f90620`. In the access token as a result of the Token Exchange flow:
  - The `iss` is the PingOne issuer value of the environment.
  - The `client_id` is the Client ID of Alpha Token Exchange App, which is `e8f90620`.
  - The `aud` is the **Audience** value as defined in the custom resource record of Beta, which is `https://api.example.com/b`.
- From Beta's point of view, Alpha is an application sending API requests to it. Alpha's identifier is `e8f90620`, the Client ID as defined in the application record of Alpha (i.e. Alpha Token Exchange App).

Detailed sample use cases

The following sections provide detailed information of several use cases.

Impersonation example

As a PingOne administrator, you added Xray as an application and Alpha as a custom resource in PingOne. You passed the application credentials to the developer of Xray and the custom resource credentials to the developer of Alpha. Xray is expected to use the authorization code grant type to obtain access tokens from PingOne to send API requests to Alpha.

You also added an application named Alpha Token Exchange App and another custom resource named Beta. You passed the application credentials to the developer of Alpha and the custom resource credentials to Beta. Alpha is expected to use the token exchange grant type to obtain access tokens from PingOne to send API requests to Beta. In this arrangement, Alpha is expected to impersonate the user.

Configuration

Application record	Custom resource record
Xray • Client ID: `a85f7a70` • Grant type: `Authorization Code` • Scopes: `openid` and `a.crud`	Alpha • Client ID: `44278071` • Audience: `https://api.example.com/a` • Scopes: `a.crud` • Attribute: `sub` ← **User ID**
Alpha Token Exchange App • Client ID: `e8f90620` • Grant type: `Token Exchange` • Scopes: `openid` and `b.read`	Beta • Client ID: `b0bc42b0` • Audience: `https://api.example.com/b` • Scopes: `b.read` • Attributes: `sub` ← `#root.context.requestData.subjectToken.sub`

The first row of configuration allows Xray to use the authorization code grant type to obtain access tokens from PingOne for the purposes of sending API requests to Alpha. Each of these access tokens contains a subject (`sub`), where the source is the direct interaction between the user and Xray.

The second row of configuration allows Alpha to use the token exchange grant type to obtain access tokens from PingOne for the purposes of sending API requests to Beta. Each of these access tokens contains a subject (`sub`).

Runtime requests and responses

1. Xray sends to PingOne an authorization request with `scope=openid+a.crud`, `client_id=a85f7a70`, and other parameters.
2. PingOne authenticates the user, obtains the authorization from the user, and returns an authorization code to Xray.
3. Xray sends to PingOne a token request using the authorization code grant type.
4. PingOne returns to Xray an access token. The decode payload reads:

```
{
  "client_id": "a85f7a70-c9ae-46cc-99cb-ff78a4ce486e", # the client ID of Xray
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "ea50083a-a3d2-4c4a-978b-1175d83c59fd",
  "iat": 1770155043,
  "exp": 1770158643,
  "aud": [
    "https://api.example.com/a" # the audience is Alpha
  ],
  "scope": "a.crud", # the scope of this access token
  "sub": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d", # the id of the user using Xray
  "sid": "2fc3e42c-f074-4efe-9bc7-ad91627be19b",
  "auth_time": 1770155030,
  "acr": "1Single_Factor",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c"
}
```

5. Xray sends to Alpha an API request. For authorization, Xray includes the access token (from step 4) as the `Authorization` HTTP request header value in the API request.
6. Alpha evaluates the access token and understands that:
    a. The issuer of the access token is PingOne.
    b. The application that requested the token is Xray.
    c. The audience of the access token includes itself, Alpha.
7. Alpha sends to PingOne an introspection request to validate the access token.
    a. For client identification and authentication, Alpha uses the client ID and secret as defined in the custom resource record of Alpha (`client_id=44278071`).
8. Alpha sends to PingOne a token request using the token exchange grant type.
    a. For client identification and authentication, Alpha uses the client ID and secret as defined in the application record of Alpha, i.e. “Alpha Token Exchange App” (`client_id=e8f90620`).
    b. Alpha also includes the following parameters in the token request:
        i. `scope=b.read`
        ii. `subject_token=<access token from step 5>`
        iii. `subject_token_type=urn:ietf:params:oauth:token-type:access_token`
        iv. `requested_token_type=urn:ietf:params:oauth:token-type:access_token`
9. PingOne validates the subject token; for example:
    a. The subject token is a valid JWT.
    b. The issuer of the subject token matches the issuer of the current PingOne environment.
    c. The associated PingOne user session is valid.

10. PingOne mints an access token based on the attribute mappings as defined in the custom resource record of Beta.
11. PingOne returns to Alpha a token response, which contains an access token. The token response reads:

```
{
    "access_token": "eyJ…", # the access token as a result of a Token Exchange flow
    "token_type": "Bearer",
    "expires_in": 3600,
    "scope": "b.read",
    "issued_token_type": "urn:ietf:params:oauth:token-type:access_token"
}
```

    The decoded access token reads:

```
{
  "client_id": "e8f90620-43e7-4d56-af96-fb0efb77076f", # the client_id of Alpha Token Exchange App
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "b6c4af53-4396-452d-a3da-88008846cf76",
  "iat": 1770155325,
  "exp": 1770158925,
  "aud": [
    "https://api.example.com/b" #  the audience is Beta
  ],
  "scope": "b.read", # the scope of this access token
  "sub": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d", # the id of the user using Xray
  "sid": "2fc3e42c-f074-4efe-9bc7-ad91627be19b",
  "auth_time": 1770155030,
  "acr": "1Single_Factor",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.userId": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d"
}
```

12. Alpha sends to Beta an API request to Beta. For authorization, Alpha includes the access token (from step 11) as the `Authorization` HTTP request header value in the API request.
13. Beta evaluates the access token and understands that:
    a. The issuer of the access token is PingOne.
    b. The application that requested the token is Alpha.
    c. The audience of the access token includes itself, Beta.
14. Beta sends to PingOne an introspection request to validate the access token.
15. Beta returns to Alpha an API response. At this point, Alpha has everything it needs to fulfill the API request it receives from Xray (in step 5).
16. Alpha returns to Xray an API response.

The request-and-response sequence matches Figure 2.

The access token Xray obtains through the use of the authorization code grant type

```
{
  "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
  "alg": "RS256"
}.{
  "client_id": "a85f7a70-c9ae-46cc-99cb-ff78a4ce486e", # the client ID of Xray
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "ea50083a-a3d2-4c4a-978b-1175d83c59fd",
  "iat": 1770155043,
  "exp": 1770158643,
  "aud": [
    "https://api.example.com/a" # the audience is Alpha
  ],
  "scope": "a.crud", # the scope of this access token
  "sub": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d", # the id of the user using Xray
  "sid": "2fc3e42c-f074-4efe-9bc7-ad91627be19b",
  "auth_time": 1770155030,
  "acr": "1Single_Factor",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c"
}.[Signature]
```

The token response Alpha receives as a result of the Token Exchange token request

```
{
    "access_token": "eyJ…", # the access token as a result of a Token Exchange flow
    "token_type": "Bearer",
    "expires_in": 3600,
    "scope": "b.read",
    "issued_token_type": "urn:ietf:params:oauth:token-type:access_token"
}
```

The access_token decoded

```
{
  "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
  "alg": "RS256"
}.{
  "client_id": "e8f90620-43e7-4d56-af96-fb0efb77076f", # the client_id of Alpha Token Exchange App
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "b6c4af53-4396-452d-a3da-88008846cf76",
  "iat": 1770155325,
  "exp": 1770158925,
  "aud": [
    "https://api.example.com/b" #  the audience is Beta
  ],
  "scope": "b.read", # the scope of this access token
  "sub": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d", # the id of the user using Xray
  "sid": "2fc3e42c-f074-4efe-9bc7-ad91627be19b",
  "auth_time": 1770155030,
  "acr": "1Single_Factor",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.userId": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d"
}.[Signature]
```

Delegation example

As a PingOne administrator, you added Yankee as an application and Gamma as a custom resource in PingOne. You passed the application credentials to the developer of Yankee and the custom resource credentials to the developer of Gamma. Yankee is expected to use the authorization code grant type to obtain access tokens from PingOne to send API requests to Gamma. The access token contains the user identity (the `sub` claim) and the identity that is allowed to act on behalf of the user (the `may_act` claim).

You also added an application named Gamma Token Exchange App and another custom resource named Delta. You passed the application credentials to the developer of Gamma and the custom resource credentials to Delta. Gamma is expected to use the token exchange grant type to obtain access tokens from PingOne to send API requests to Delta. In this arrangement, PingOne is expected to include the appropriate user delegation information in the access token (the `act` claim) so that Delta understands someone else is acting on behalf of the user when it receives API requests from Gamma.

Configurations

Application record	**Custom resource record**

Yankee
• Client ID: `f6c78a5b`
• Grant type: `Authorization Code`
• Scopes: `openid` and `g.crud`
	Gamma
	• Client ID: `75d8cea3`
	• Audience: `https://api.example.com/g`
	• Scopes: `g.crud`
	• Attribute:
	  ○ `sub` ← **Username**
	  ○ `may_act` ← `{"sub": "45f60a71"}`

Gamma Token Exchange App
• Client ID: `45f60a71`
• Grant type: `Client Credentials` and `Token Exchange`
• Scopes: `openid` and `d.read`
	Delta
	• Client ID: `81ca41a2`
	• Audience: `https://api.example.com/d`
	• Scopes: `d.read`
	• Attributes:
	  ○ `sub` ← (explained later)
	  ○ `act` ← (explained later)

The first row of configuration allows Yankee to use the authorization code grant type to obtain access tokens from PingOne for the purposes of sending API requests to Gamma. Each of these access tokens contains a subject (`sub`) and the identity that may act on behalf of the subject (`may_act`).

The second row of configuration allows Gamma to use the token exchange grant type to obtain access tokens from PingOne for the purposes of sending API requests to Delta. Each of these access tokens contains a subject (`sub`) and the identity acting on behalf of the subject (`act`).

Attribute mappings for custom resource Delta

Attribute	Mapping	Required
`sub`	`#root.context.requestData.subjectToken.sub` This expression populates the `sub` claim value in the issued token with the `sub` claim value from the subject token.	☐
`act`	`(#root.context.requestData.grantType == "client_credentials")?"noActor":((#root.context.requestData.subjectToken.may_act.sub == #root.context.requestData.actorToken.client_id)?#root.context.requestData.subjectToken.may_act:null)` This expression checks whether the token request is using the client credentials grant type, or not. If it's the client credentials grant type, it populates `act` with a hard-coded value of `noActor`. This value provides a signal to Delta that no delegation is involved. If it's not the client credential grant type, it assumes it's the token exchange grant type and checks whether the `may_act` claim value in the subject token matches the `client_id` in the actor token. If it's a match, it populates `act` with the `may_act` claim value from the subject token; otherwise, it sets `act` as *null*. Because the `act` claim is marked as a required attribute, the Token Exchange token request fails.	☑

Runtime requests

1. Yankee sends to PingOne an authorization request with `scope=openid+g.crud`, `client_id=f6c78a5b`, and other parameters.
2. PingOne authenticates the user, obtains the authorization from the user, and returns an authorization code to Yankee.
3. Yankee sends to PingOne a token request using the authorization code grant type.
4. PingOne returns to Xray an access token. The decode payload reads:


```
{
  "client_id": "f6c78a5b-9d39-4cd7-b94e-81dad33c8773",
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "54ffa426-1410-4383-8ec5-344a7b1b948e",
  "iat": 1770574186,
  "exp": 1770577786,
  "aud": [
    "https://api.example.com/g"
  ],
  "scope": "g.crud",
  "sub": "user@example.net",
  "sid": "86635114-c633-4c13-b1eb-4a8a3f0e7dcd",
  "auth_time": 1770573761,
  "acr": "1Single_Factor",
  "may_act": {
    "sub": "45f60a71-df8c-42d6-9410-f64f0454874d" # the client ID of Gamma Token Exchange App
  },
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.userId": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d"
}
```


5. Yankee sends to Gamma an API request. For authorization, Yankee includes the access token (from step 4) as the `Authorization` HTTP request header value in the API request.
6. Gamma evaluates the access token and understands that:
    a. The issuer of the access token is PingOne.
    b. The application that requested the token is Yankee.
    c. The audience of the access token includes itself, Gamma.
7. Gamma sends to PingOne an introspection request to validate the access token.
    a. For client identification and authentication, Gamma uses the client ID and secret as defined in the custom resource record of Gamma (`client_id=75d8cea3`).
8. Gamma sends to PingOne a token request using the client credentials grant type.
    a. For client identification and authentication, Gamma uses the client ID and secret as defined in the application record of Gamma, i.e. “Gamma Token Exchange App” (`client_id=45f60a71`).
9. PingOne returns to Gamma an access token. The decoded payload reads:


```
{
  "client_id": "45f60a71-df8c-42d6-9410-f64f0454874d", # the client ID of Gamma Token Exchange App
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "47d54766-7b9a-4485-96d9-6c09d410b943",
  "iat": 1770574217,
  "exp": 1770577817,
  "aud": [
```



```
    "https://api.example.com/d"
  ],
  "scope": "d.read",
  "act": "noActor",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.rid": "47d54766-7b9a-4485-96d9-6c09d410b943"
}
```

10. Gamma sends to PingOne a token request using the token exchange grant type.
    a. For client identification and authentication, Gamma uses the client ID and secret as defined in the application record of Gamma, i.e. “Gamma Token Exchange App” (`client_id=45f60a71`).
    b. Gamma also includes the following parameters in the token request:
        i. `scope=d.read`
        ii. `subject_token=<access token from step 5>`
        iii. `subject_token_type=urn:ietf:params:oauth:token-type:access_token`
        iv. `actor_token=<access token from step 9>`
        v. `actor_token_type=urn:ietf:params:oauth:token-type:access_token`
        vi. `requested_token_type=urn:ietf:params:oauth:token-type:access_token`
11. PingOne validates the subject token; for example:
    a. The subject token and the actor token are valid JWTs.
    b. The issuer of the subject token and the actor token matches the issuer of the current PingOne environment.
    c. The associated PingOne user session is valid.
12. PingOne mints an access token based on the attribute mappings as defined in the custom resource record of Delta.
13. PingOne returns to Gamma a token response, which contains an access token. The token response reads:

```
{
    "access_token": "eyJ…",
    "token_type": "Bearer",
    "expires_in": 3600,
    "scope": "d.read",
    "issued_token_type": "urn:ietf:params:oauth:token-type:access_token"
}
```

The decoded access token reads:

```
{
  "client_id": "45f60a71-df8c-42d6-9410-f64f0454874d", # the client ID of Gamma Token Exchange App
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "337a5736-bea4-44dd-9692-dc5f696ef710",
  "iat": 1770574249,
  "exp": 1770577849,
  "aud": [
    "https://api.example.com/d"
  ],
  "scope": "d.read",
  "sub": "user@example.net",
```

```
  "sid": "86635114-c633-4c13-b1eb-4a8a3f0e7dcd",
  "auth_time": 1770573761,
  "acr": "1Single_Factor",
  "act": {
    "sub": "45f60a71-df8c-42d6-9410-f64f0454874d" # the client ID of Gamma Token Exchange App
  },
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.userId": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d"
}
```

14. Gamma sends to Delta an API request to Delta. For authorization, Gamma includes the access token (from step 13) as the `Authorization` HTTP request header value in the API request.
15. Delta evaluates the access token and understands that:
    a. The issuer of the access token is PingOne.
    b. The application that requested the token is Gamma.
    c. The audience of the access token includes itself, Delta.
16. Delta sends to PingOne an introspection request to validate the access token.
17. Delta returns to Gamma an API response. At this point, Gamma has everything it needs to fulfill the API request it receives from Yankee (in step 5).
18. Gamma returns to Yankee an API response.

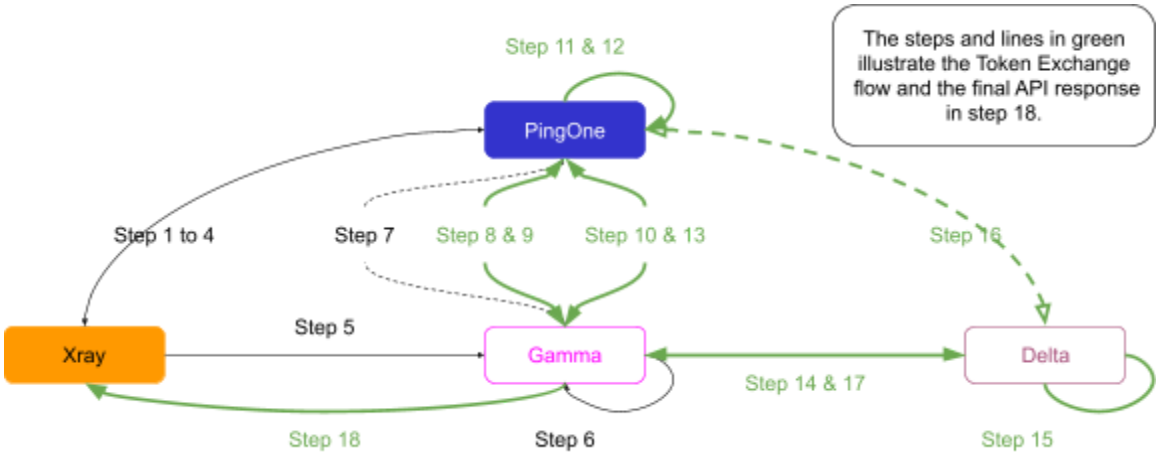

Figure 3: Delegation token exchange flow

The access token Yankee obtains through the use of the authorization code grant type

```
{
  "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
  "alg": "RS256"
}.{
  "client_id": "f6c78a5b-9d39-4cd7-b94e-81dad33c8773",
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "54ffa426-1410-4383-8ec5-344a7b1b948e",
  "iat": 1770574186,
  "exp": 1770577786,
  "aud": [
    "https://api.example.com/g"
  ],
  "scope": "g.crud",
  "sub": "user@example.net",
  "sid": "86635114-c633-4c13-b1eb-4a8a3f0e7dcd",
  "auth_time": 1770573761,
  "acr": "1Single_Factor",
  "may_act": {
    "sub": "45f60a71-df8c-42d6-9410-f64f0454874d" # the client ID of Gamma Token Exchange App
  },
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.userId": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d"
}.[Signature]
```

The access token Gamma obtains through the use of the client credentials grant type

```
{
  "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
  "alg": "RS256"
}.{
  "client_id": "45f60a71-df8c-42d6-9410-f64f0454874d", # the client ID of Gamma Token Exchange App
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "47d54766-7b9a-4485-96d9-6c09d410b943",
  "iat": 1770574217,
  "exp": 1770577817,
  "aud": [
    "https://api.example.com/d"
  ],
  "scope": "d.read",
  "act": "noActor",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.rid": "47d54766-7b9a-4485-96d9-6c09d410b943"
}.[Signature]
```

The token response Gamma receives as a result of the Token Exchange token request


```
{
    "access_token": "eyJ…",
    "token_type": "Bearer",
    "expires_in": 3600,
    "scope": "d.read",
    "issued_token_type": "urn:ietf:params:oauth:token-type:access_token"
}
```


The access_token decoded


```
{
  "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
  "alg": "RS256"
}.{
  "client_id": "45f60a71-df8c-42d6-9410-f64f0454874d", # the client ID of Gamma Token Exchange App
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "337a5736-bea4-44dd-9692-dc5f696ef710",
  "iat": 1770574249,
  "exp": 1770577849,
  "aud": [
    "https://api.example.com/d"
  ],
  "scope": "d.read",
  "sub": "user@example.net",
  "sid": "86635114-c633-4c13-b1eb-4a8a3f0e7dcd",
  "auth_time": 1770573761,
  "acr": "1Single_Factor",
  "act": {
    "sub": "45f60a71-df8c-42d6-9410-f64f0454874d" # the client ID of Gamma Token Exchange App
  },
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.userId": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d"
}.[Signature]
```

Machine-to-machine example

As a PingOne administrator, you added Zulu as an application and Epsilon as a custom resource in PingOne. You passed the application credentials to the developer of Zulu and the custom resource credentials to the developer of Epsilon. Zulu is expected to use the client credentials grant type to obtain access tokens from PingOne to send API requests to Epsilon.

You also added an application named Epsilon Token Exchange App and another custom resource named Zeta. You passed the application credentials to the developer of Epsilon and the custom resource credentials to Delta. Epsilon is expected to use the token exchange grant type to obtain access tokens from PingOne to send API requests to Zeta.

Configuration

Application record	**Custom resource record**

Zulu
• Client ID: `4076de38`
• Grant type: `Client Credentials`
• Scopes: `openid` and `e.crud`
	Epsilon
	• Client ID: `bc82af8d`
	• Audience: `https://api.example.com/e`
	• Scopes: `e.crud`
	• Attribute:
	  ○ `sub` ← **User ID**
	  ○ `e.attr` ← `'Eee'`

Epsilon Token Exchange App
• Client ID: `b03ae60a`
• Grant type: `Token Exchange`
• Scopes: `openid` and `z.read`
	Zeta
	• Client ID: `bf53b521`
	• Audience: `https://api.example.com/b`
	• Scopes: `b.read`
	• Attributes:
	  ○ `sub` ← **User ID**
	  ○ `z.attr` ← `'Zee'`

The first row of configuration allows Zulu to use the client credentials grant type to obtain access tokens from PingOne for the purposes of sending API requests to Epsilon. When minting these access tokens, PingOne populates the `e.attr` claim value with `Eee` and does not include the `sub` claim because **User ID** is *null* when no user is involved.

The second row of configuration allows Epsilon to use the token exchange grant type to obtain access tokens from PingOne for the purposes of sending API requests to Zeta. Each of these access tokens contains a subject (`sub`). When minting these access tokens, PingOne populates the `z.attr` claim value with `Zee` and does not include the `sub` claim because **User ID** is *null* when no user is involved.

NOTE: `e,attr` and `z.attr` are sample attributes with static values. They are not essential to demonstrate the machine-to-machine token exchange flow.

Runtime requests

1. Zulu sends to PingOne a token request using the client credentials grant type.
    a. For client identification and authentication, Zulu uses the client ID and secret as defined in the application record of Zulu (`client_id=4076de38`).
    b. Zulu includes `scope=e.crud` in its token request.
2. PingOne returns to Zulu an access token based on the attribute mappings as defined in the custom resource record of Epsilon. The decoded payload reads:

```
{
  "client_id": "4076de38-d226-49c8-8b47-5f8df21ef3a2", # the client ID of Zulu
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "98dccaf3-44d9-47a3-ba56-9f2db0888fca",
  "iat": 1770161254,
  "exp": 1770164854,
  "aud": [
    "https://api.example.com/e" # the audience is Epsilon
  ],
  "scope": "e.crud",
  "e.attr": "Eee",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.rid": "98dccaf3-44d9-47a3-ba56-9f2db0888fca"
}
```

3. Zulu sends to Epsilon an API request. For authorization, Zulu includes the access token (from step 2) as the `Authorization` HTTP request header value in the API request.
4. Epsilon evaluates the access token and understands that:
    a. The issuer of the access token is PingOne.
    b. The application that requested the token is Zulu.
    c. The audience of the access token includes itself, Epsilon.
5. Epsilon sends to PingOne an introspection request to validate the access token.
    a. For client identification and authentication, Epsilon uses the client ID and secret as defined in the custom resource record of Epsilon (`client_id=bc82af8d`).
6. Epsilon sends to PingOne a token request using the token exchange grant type.
    a. For client identification and authentication, Alpha uses the client ID and secret as defined in the application record of Epsilon, i.e. “Epsilon Token Exchange App” (`client_id=b03ae60a`).
    b. Alpha also includes the following parameters in the token request:
        i. `scope=z.read`
        ii. `subject_token=<access token from step 3>`
        iii. `subject_token_type=urn:ietf:params:oauth:token-type:access_token`
        iv. `requested_token_type=urn:ietf:params:oauth:token-type:access_token`
7. PingOne validates the subject token; for example:
    a. The subject token is a valid JWT.
    b. The issuer of the subject token matches the issuer of the current PingOne environment.
8. PingOne mints an access token based on the attribute mappings as defined in the custom resource record of Zeta.

9. PingOne returns to Epsilon a token response, which contains an access token. The token response reads:

```
{
    "access_token": "eyJ…",
    "token_type": "Bearer",
    "expires_in": 3600,
    "scope": "z.read",
    "issued_token_type": "urn:ietf:params:oauth:token-type:access_token"
}
```

   The decoded access token reads:

```
{
  "client_id": "b03ae60a-e4f9-4e9e-ae3d-52592e61d939", # the client ID of Epsilon Token Exchange App
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "03965dbd-6db1-40a9-92ae-7267faf7808a",
  "iat": 1770161259,
  "exp": 1770164859,
  "aud": [
    "https://api.example.com/z" # the audience is Zeta
  ],
  "scope": "z.read",
  "z.attr": "Zee",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.rid": "03965dbd-6db1-40a9-92ae-7267faf7808a"
}
```

10. Epsilon sends to Zeta an API request to Beta. For authorization, Epsilon includes the access token (from step 9) as the `Authorization` HTTP request header value in the API request.
11. Zeta evaluates the access token and understands that:
    a. The issuer of the access token is PingOne.
    b. The application that requested the token is Epsilon Token Exchange App.
    c. The audience of the access token includes itself, Zeta.
12. Zeta sends to PingOne an introspection request to validate the access token.
13. Zeta returns to Epsilon an API response. At this point, Epsilon has everything it needs to fulfill the API request it receives from Zulu (in step 3).
14. Epsilon returns to Zulu an API response.

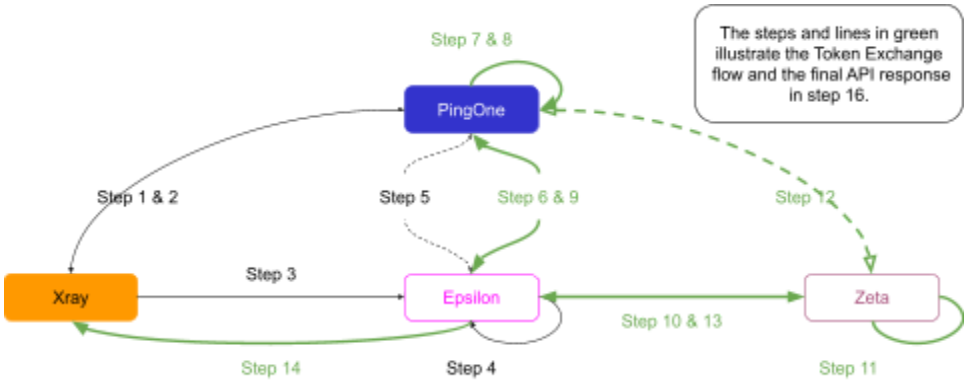

Figure 4: Machine-to-machine token exchange flow

The access token Zulu obtains through the use of the client credentials grant type

```
{
  "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
  "alg": "RS256"
}.{
  "client_id": "4076de38-d226-49c8-8b47-5f8df21ef3a2", # the client ID of Zulu
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "98dccaf3-44d9-47a3-ba56-9f2db0888fca",
  "iat": 1770161254,
  "exp": 1770164854,
  "aud": [
    "https://api.example.com/e" # the audience is Epsilon
  ],
  "scope": "e.crud",
  "e.attr": "Eee",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.rid": "98dccaf3-44d9-47a3-ba56-9f2db0888fca"
}.[Signature]
```

The token response Epsilon receives as a result of the Token Exchange token request

```
{
    "access_token": "eyJ…",
    "token_type": "Bearer",
    "expires_in": 3600,
    "scope": "z.read",
    "issued_token_type": "urn:ietf:params:oauth:token-type:access_token"
}
```

The access_token decoded

```
{
  "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
  "alg": "RS256"
}.{
  "client_id": "b03ae60a-e4f9-4e9e-ae3d-52592e61d939", # the client ID of Epsilon Token Exchange App
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "03965dbd-6db1-40a9-92ae-7267faf7808a",
  "iat": 1770161259,
  "exp": 1770164859,
  "aud": [
    "https://api.example.com/z" # the audience is Zeta
  ],
  "scope": "z.read",
  "z.attr": "Zee",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.rid": "03965dbd-6db1-40a9-92ae-7267faf7808a"
}.[Signature]
```

#root.context.appConfig and #root.context.requestData

This decoded access_token as a result of a Token Exchange token request illustrates the data accessible to the PingOne administrators when they build their expressions in a custom resource record.

The sample did not contain `actor_token`, `actor_token_type`, `actorTokenUser`, and `actorTokenHeader` because the application didn't include the `actor_token` and `actor_token_type` parameters in its Token Exchange token request. If it did, the response would include them in the `requestData` claim value as well.


```
{
  "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
  "alg": "RS256"
}.{
  "client_id": "e8f90620-43e7-4d56-af96-fb0efb77076f",
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "c15028b0-02aa-445b-aeb4-e07e7d8d9d3f",
  "iat": 1770161939,
  "exp": 1770165539,
  "aud": [
    "https://api.example.com/t"
  ],
  "scope": "t.read",
  "sub": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d",
  "sid": "e4842c1c-d1e0-416d-8fae-ee565a14704d",
  "auth_time": 1770161827,
  "acr": "1Single_Factor",
  "requestData": {
    "scope": "t.read",  // requested scope
    "grantType": "urn:ietf:params:oauth:grant-type:token-exchange",  // grant type; expression can branch on
it
    "subjectToken": {
      "client_id": "a85f7a70-c9ae-46cc-99cb-ff78a4ce486e",
      "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
      "jti": "01f7fcfb-69d0-43cb-a691-983e2218b274",
      "iat": 1770161919,
      "exp": 1770165519,
      "aud": [
        "https://api.example.com/a"
      ],
      "scope": "a.crud",
      "sub": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d",
      "sid": "e4842c1c-d1e0-416d-8fae-ee565a14704d",
      "auth_time": 1770161827,
      "acr": "1Single_Factor",
      "env": "6991589d-87eb-47f4-9131-284cebe106b3",
      "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c"
    },
    "subjectTokenType": "urn:ietf:params:oauth:token-type:access_token",
    "subjectTokenUser": {  // the user as identified by the subject_token
      "id": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d",
      "_links": {
        "self": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d"
        },
        "environment": {
```



```
          "href": "http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3"
        },
        "population": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/populations/6b9cb9a7-54f9-4a4c-b
71f-73e46834dfa5"
        },
        "devices": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d/devices"
        },
        "roleAssignments": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d/roleAssignments"
        },
        "password": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d/password"
        },
        "password.reset": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d/password"
        },
        "password.set": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d/password"
        },
        "password.check": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d/password"
        },
        "password.recover": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d/password"
        },
        "linkedAccounts": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d/linkedAccounts"
        },
        "account.sendVerificationCode": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d"
        },
        "memberOfGroups": {
          "href":
"http://10.76.127.228:4140/environments/6991589d-87eb-47f4-9131-284cebe106b3/users/8ca2b15a-e3bd-43a5-bee1-1e
533bae759d/memberOfGroups"
        }
      },
      "environment": {
        "id": "6991589d-87eb-47f4-9131-284cebe106b3"
      },
      "account": {
```



```
        "canAuthenticate": true,
        "status": "OK"
      },
      "createdAt": "2026-02-03T16:27:50.469Z",
      "enabled": true,
      "identityProvider": {
        "type": "PING_ONE"
      },
      "lastSignOn": {
        "at": "2026-02-03T23:37:07.238Z",
        "remoteIp": "209.133.96.106"
      },
      "lifecycle": {
        "status": "ACCOUNT_OK"
      },
      "mfaEnabled": false,
      "population": {
        "id": "6b9cb9a7-54f9-4a4c-b71f-73e46834dfa5"
      },
      "updatedAt": "2026-02-03T23:37:07.266Z",
      "username": "user@example.net",
      "verifyStatus": "NOT_INITIATED"
    },
    "subjectTokenHeader": {
      "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
      "alg": "RS256"
    },
    "requestedTokenType": "urn:ietf:params:oauth:token-type:access_token"
  },
  "appConfig": {
    "clientId": "e8f90620-43e7-4d56-af96-fb0efb77076f",
    "envId": "6991589d-87eb-47f4-9131-284cebe106b3",
    "orgId": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
    "tokenEndpointAuthMethod": "CLIENT_SECRET_POST"
  },
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c"
}.[Signature]
```


Additional verification

Looking back to the **Impersonation** example:

Application record	Custom resource record
Xray • Client ID: `a85f7a70` • Grant type: `Authorization Code`	Alpha • Client ID: `44278071` • Audience: `https://api.example.com/a`
Alpha Token Exchange App • Client ID: `e8f90620` • Grant type: `Token Exchange`	Beta • Client ID: `b0bc42b0` • Audience: `https://api.example.com/b`

Xray obtains from PingOne an access token and includes it as the `Authorization` HTTP request header value in its API request to Alpha.

```
{
  "aud": [
    "https://api.example.com/a"  // the audience of the intended custom resource, Alpha
  ],
  ...
}
```

Alpha extracts it out of the header and includes it as the `subject_token` value in its Token Exchange token request to PingOne. For client identification and authentication, Alpha uses the client ID and secret as defined in the application record of Alpha, i.e. “Alpha Token Exchange App” (`client_id=e8f90620`)

Generally speaking, when processing a Token Exchange token request, PingOne validates that the subject token and the actor token are valid JWTs, the issuer matches the issuer of the current PingOne environment, and the associated user session is valid. As needed, administrators can include additional checks to verify whether the application that sends the token request (the `client_id` from the token request) and the intended audience of the subject and actor tokens of the access token type (the `aud` claim in the token) refer to the same resource server.

For example, the following expression verifies that (1) the `aud` claim array contains the **Audience** as defined in the custom resource record of Alpha (`https://api.example.com/a`) ***and*** (2) the `client_id` of the application sending the Token Exchange token request matches the **Client ID** as defined in the application record of Alpha Token Exchange App (`e8f90620-43e7-4d56-af96-fb0efb77076f`).

```
#core.ifelse(#data.containsAll(#root.context.requestData.subjectToken.aud,
{"https://api.example.com/a"}) && #root.context.appConfig.clientId ==
"e8f90620-43e7-4d56-af96-fb0efb77076f", #root.context.requestData.subjectToken.sub, null)
```

If both conditions are met, PingOne populates the `sub` claim value with the `sub` claim value from the subject token; otherwise, it sets the `sub` claim to *null*. To make this expression effective, the `sub` claim must also be marked as a required attribute, in which case *null* will trigger a failure.

https://auth

https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as/authorize?client_id=a85f7a70-c9ae-46cc-99cb-ff78a4ce486e&redirect_uri=https://decoder.pingidentity.cloud/hybrid&response_type=token&scope=a.crud

https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as/authorize?client_id=f6c78a5b-9d39-4cd7-b94e-81dad33c8773&redirect_uri=https://decoder.pingidentity.cloud/hybrid&response_type=token&scope=g.crud — user@example.net

https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as/authorize?client_id=f6c78a5b-9d39-4cd7-b94e-81dad33c8773&redirect_uri=https://decoder.pingidentity.cloud/hybrid&response_type=id_token&scope=openid — admin@example.net

Tab 3

Release planning

We intend to deliver Token Exchange Phase 1 by the end of 2026 Q1. We will continue to improve it in multiple phases thereafter.

Phase 1 coverage

Token Exchange is a new grant type. No application is allowed to use the Token Exchange grant type until an administrator enables it explicitly.

For the `subject_token` or the `actor_token`, a PingOne environment accepts an access token or an ID token it previously issued. In other words, it must be a token from the same environment. A token from another PingOne environment (regardless of whether it is under the same organization, or not) or an external authorization server is not supported. The token must be valid; for example, the token has not expired, the digital signature is valid, etc.

For the `requested_token_type`, PingOne supports `urn:ietf:params:oauth:token-type:access_token`. The `access_token` in the token response is an access token intended for one or more custom resources. PingOne does not include a refresh token in its token response as a result of using the Token Exchange grant type.

Similar to what an application can do in its authorization request, an application using the Token Exchange grant type can use the ***scope*** parameter to indicate the desired custom resource(s). Also similar to how the ***scope*** parameter is processed, the scope value(s) in the ***scope*** parameter must be added to the application using the Token Exchange grant type.

When the `requested_token_type` is `urn:ietf:params:oauth:token-type:access_token`, its fulfillment is controlled by the attribute mapping configuration of the applicable custom resource(s). Administrators can use expressions to implement fulfillment logic. For example, to fulfill an attribute differently based on the grant type used, an administrator can check the `context.requestData.grantType` value and fulfill the attribute differently. Expression has access to various data from the token request, such as `context.requestData.subjectToken[.claim]` to access individual claims from the subject token or `context.requestData.scope` to access the ***scope*** parameter value. `context.appConfig.clientId`, `context.appConfig.tokenEndpointAuthMethod`, `context.appConfig.envId`, and `context.appConfig.orgId` are also made available.

Note that Resources > Attributes > Expressions already have access to `context.requestData.clientAssertionHeader[.property]` and `context.requestData.clientAssertion[.claim]`, which can help drive or populate claims in the resulting tokens.

Future phases

After phase 1, we intend to make further improvements in multiple phases thereafter. Here are some of the items:

- Possibly Q2 to Q3: Expand to support https://datatracker.ietf.org/doc/html/draft-ietf-oauth-identity-assertion-authz-grant (XAA)
    - Expand `requested_token_type` to support `urn:ietf:params:oauth:token-type:id-jag`
    - Potentially expand `subject_token_type` to support `urn:ietf:params:oauth:token-type:refresh_token` (as defined in rfc8693#section-3 and mentioned in Identity Assertion JWT Authorization Grant draft #section-4.4.3)
- Possibly Q3+: Support third-party authorization servers (JWT-based access token per https://datatracker.ietf.org/doc/html/rfc9068)

No commitments can be made at this point.

Tab 4

Old	New
Xray	XL Delivery
ALpha (RS)	Address RS (resource server)
a.crud	a.crud
ALpha TOken EXchange APp	Address Token Exchange App
Beta	Buzzer RS
b.read	b.read

Impersonation example

As a PingOne administrator, you added XL Delivery as an application and Alpha as a custom resource in PingOne. You passed the application credentials to the developer of XL Delivery and the custom resource credentials to the developer of Alpha. XL Delivery is expected to use the authorization code grant type to obtain access tokens from PingOne to send API requests to Alpha.

You also added an application named Address Token Exchange App and another custom resource named Beta. You passed the application credentials to the developer of Alpha and the custom resource credentials to Beta. Alpha is expected to use the token exchange grant type to obtain access tokens from PingOne to send API requests to Beta. In this arrangement, Alpha is expected to impersonate the user.

Configuration

Application record	Custom resource record
XL Delivery • Client ID: `a85f7a70` • Grant type: `Authorization Code` • Scopes: `openid` and `a.crud`	Alpha • Client ID: `44278071` • Audience: `https://api.example.com/a` • Scopes: `a.crud` • Attribute: `sub` ← **User ID**
Address Token Exchange App • Client ID: `e8f90620` • Grant type: `Token Exchange` • Scopes: `openid` and `b.read`	Beta • Client ID: `b0bc42b0` • Audience: `https://api.example.com/b` • Scopes: `b.read` • Attributes: `sub` ← `#root.context.requestData.subjectToken.sub`

The first row of configuration allows XL Delivery to use the authorization code grant type to obtain access tokens from PingOne for the purposes of sending API requests to Alpha. Each of these access tokens contains a subject (`sub`), where the source is the direct interaction between the user and XL Delivery.

The second row of configuration allows Alpha to use the token exchange grant type to obtain access tokens from PingOne for the purposes of sending API requests to Beta. Each of these access tokens contains a subject (`sub`).

Runtime requests and responses

1. XL Delivery sends to PingOne an authorization request with `scope=openid+a.crud`, `client_id=a85f7a70`, and other parameters.
2. PingOne authenticates the user, obtains the authorization from the user, and returns an authorization code to XL Delivery.
3. XL Delivery sends to PingOne a token request using the authorization code grant type.
4. PingOne returns to XL Delivery an access token. The decode payload reads:

```
{
  "client_id": "a85f7a70-c9ae-46cc-99cb-ff78a4ce486e", # the client ID of XL Delivery
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "ea50083a-a3d2-4c4a-978b-1175d83c59fd",
  "iat": 1770155043,
  "exp": 1770158643,
  "aud": [
    "https://api.example.com/a" # the audience is Alpha
  ],
  "scope": "a.crud", # the scope of this access token
  "sub": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d", # the id of the user using XL Delivery
  "sid": "2fc3e42c-f074-4efe-9bc7-ad91627be19b",
  "auth_time": 1770155030,
  "acr": "1Single_Factor",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c"
}
```

5. XL Delivery sends to Alpha an API request. For authorization, XL Delivery includes the access token (from step 4) as the `Authorization` HTTP request header value in the API request.
6. Alpha evaluates the access token and understands that:
    a. The issuer of the access token is PingOne.
    b. The application that requested the token is XL Delivery.
    c. The audience of the access token includes itself, Alpha.
7. Alpha sends to PingOne an introspection request to validate the access token.
    a. For client identification and authentication, Alpha uses the client ID and secret as defined in the custom resource record of Alpha (`client_id=44278071`).
8. Alpha sends to PingOne a token request using the token exchange grant type.
    a. For client identification and authentication, Alpha uses the client ID and secret as defined in the application record of Alpha, i.e. “Address Token Exchange App” (`client_id=e8f90620`).
    b. Alpha also includes the following parameters in the token request:
        i. `scope=b.read`
        ii. `subject_token=<access token from step 5>`
        iii. `subject_token_type=urn:ietf:params:oauth:token-type:access_token`
        iv. `requested_token_type=urn:ietf:params:oauth:token-type:access_token`
9. PingOne validates the subject token; for example:
    a. The subject token is a valid JWT.
    b. The issuer of the subject token matches the issuer of the current PingOne environment.
    c. The associated PingOne user session is valid.

10. PingOne mints an access token based on the attribute mappings as defined in the custom resource record of Beta.
11. PingOne returns to Alpha a token response, which contains an access token. The token response reads:

```
{
    "access_token": "eyJ…", # the access token as a result of a Token Exchange flow
    "token_type": "Bearer",
    "expires_in": 3600,
    "scope": "b.read",
    "issued_token_type": "urn:ietf:params:oauth:token-type:access_token"
}
```

    The decoded access token reads:

```
{
  "client_id": "e8f90620-43e7-4d56-af96-fb0efb77076f", # the client_id of Address Token Exchange App
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "b6c4af53-4396-452d-a3da-88008846cf76",
  "iat": 1770155325,
  "exp": 1770158925,
  "aud": [
    "https://api.example.com/b" #  the audience is Beta
  ],
  "scope": "b.read", # the scope of this access token
  "sub": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d", # the id of the user using XL Delivery
  "sid": "2fc3e42c-f074-4efe-9bc7-ad91627be19b",
  "auth_time": 1770155030,
  "acr": "1Single_Factor",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.userId": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d"
}
```

12. Alpha sends to Beta an API request to Beta. For authorization, Alpha includes the access token (from step 11) as the `Authorization` HTTP request header value in the API request.
13. Beta evaluates the access token and understands that:
    a. The issuer of the access token is PingOne.
    b. The application that requested the token is Alpha.
    c. The audience of the access token includes itself, Beta.
14. Beta sends to PingOne an introspection request to validate the access token.
15. Beta returns to Alpha an API response. At this point, Alpha has everything it needs to fulfill the API request it receives from XL Delivery (in step 5).
16. Alpha returns to XL Delivery an API response.

The request-and-response sequence matches Figure 2.

The access token XL Delivery obtains through the use of the authorization code grant type

```
{
  "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
  "alg": "RS256"
}.{
  "client_id": "a85f7a70-c9ae-46cc-99cb-ff78a4ce486e", # the client ID of XL Delivery
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "ea50083a-a3d2-4c4a-978b-1175d83c59fd",
  "iat": 1770155043,
  "exp": 1770158643,
  "aud": [
    "https://api.example.com/a" # the audience is Alpha
  ],
  "scope": "a.crud", # the scope of this access token
  "sub": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d", # the id of the user using XL Delivery
  "sid": "2fc3e42c-f074-4efe-9bc7-ad91627be19b",
  "auth_time": 1770155030,
  "acr": "1Single_Factor",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c"
}.[Signature]
```

The token response Alpha receives as a result of the Token Exchange token request

```
{
    "access_token": "eyJ…", # the access token as a result of a Token Exchange flow
    "token_type": "Bearer",
    "expires_in": 3600,
    "scope": "b.read",
    "issued_token_type": "urn:ietf:params:oauth:token-type:access_token"
}
```

The access_token decoded

```
{
  "kid": "376110a0-da12-11f0-8a1e-014df6b27a55",
  "alg": "RS256"
}.{
  "client_id": "e8f90620-43e7-4d56-af96-fb0efb77076f", # the client_id of Address Token Exchange App
  "iss": "https://auth.pingone.com/6991589d-87eb-47f4-9131-284cebe106b3/as",
  "jti": "b6c4af53-4396-452d-a3da-88008846cf76",
  "iat": 1770155325,
  "exp": 1770158925,
  "aud": [
    "https://api.example.com/b" #  the audience is Beta
  ],
  "scope": "b.read", # the scope of this access token
  "sub": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d", # the id of the user using XL Delivery
  "sid": "2fc3e42c-f074-4efe-9bc7-ad91627be19b",
  "auth_time": 1770155030,
  "acr": "1Single_Factor",
  "env": "6991589d-87eb-47f4-9131-284cebe106b3",
  "org": "d4229c38-0f5e-4bf7-9292-9d3b0df7294c",
  "p1.userId": "8ca2b15a-e3bd-43a5-bee1-1e533bae759d"
}.[Signature]
```