Diagrams of All The OpenID Connect Flows

Takahiko Kawasaki Follow
Oct 30, 2017 · 8 min read

Introduction

OpenID Connect has been developed by extending **OAuth 2.0**.

OAuth 2.0 is a specification as to how to issue **access tokens**. It is defined in **RFC 6749 (The OAuth 2.0 Authorization Framework)**. (c.f. *"The Simplest Guide To OAuth 2.0"*)

OpenID Connect is a specification as to how to issue **ID tokens**. The main part is defined in **OpenID Connect Core 1.0**.

RFC 6749 includes the definition of a Web API called "**authorization endpoint**". The API requires `response_type` as a mandatory request parameter. OpenID Connect has defined flows to issue ID tokens by extending the specification of the `response_type` request parameter.

In RFC 6749, the value of `response_type` is either `code` or `token`. OpenID Connect has added a new value, `id_token`, and allowed **any combination** of `code`, `token` and `id_token`. A special value, `none`, has been added, too. As a result, now `response_type` can take any one of the following values.

1. `code`
2. `token`
3. `id_token`
4. `id_token token`
5. `code id_token`
6. `code token`

7. `code id_token token`
8. `none`

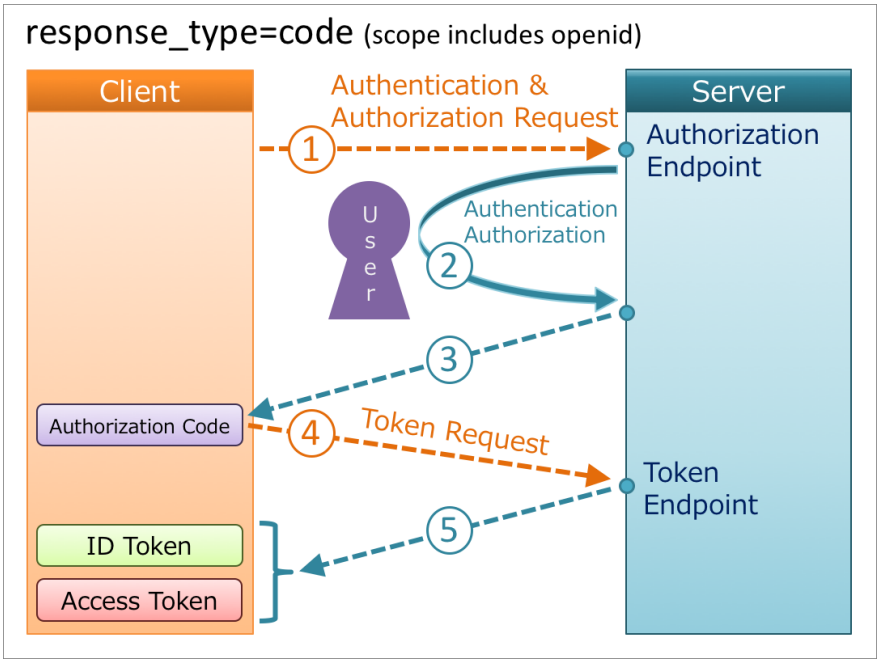
Note that a request for an ID token has to include `openid` in the `scope` request parameter. Especially, if `openid` is not included in `scope`, the case of `response_type=code` is regarded as the original authorization code flow defined in RFC 6749 and an ID token won't be issued. The same is true of the case of `response_type=code token`, too.

1. response_type=code

When the value of `response_type` is `code`, but if `openid` is not included in the `scope` request parameter, the request is just an authorization code flow which is defined in RFC 6749. On the other hand, if `openid` is included in the `scope` request parameter, an ID token is issued from the token endpoint in addition to an access token.

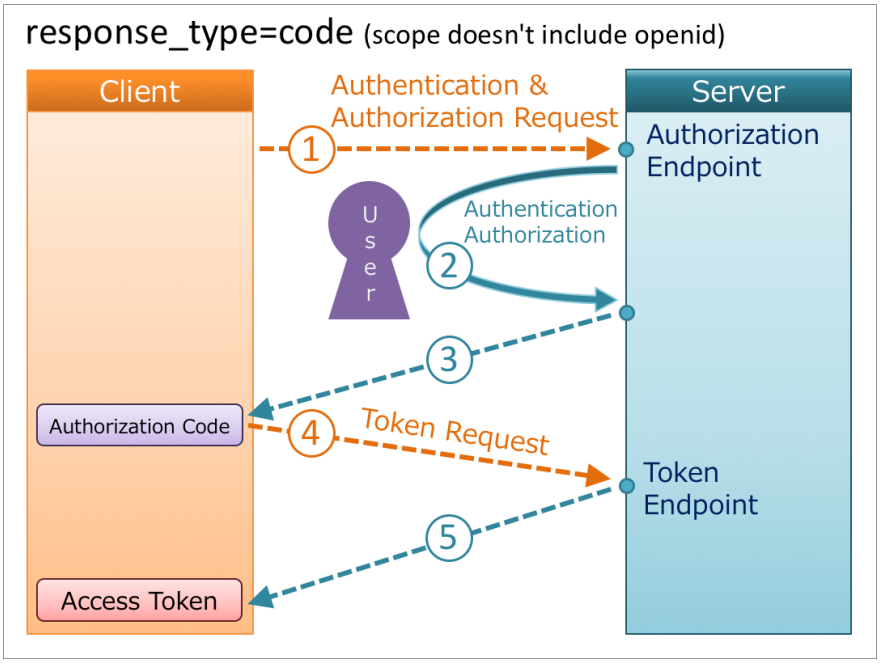
if `openid` is included

Endpoint	Authorization Code	Access Token	ID Token
Authorization	Issued	X	X
Token	X	Issued	Issued



if `openid` is not included (authorization code flow defined in RFC 6749)

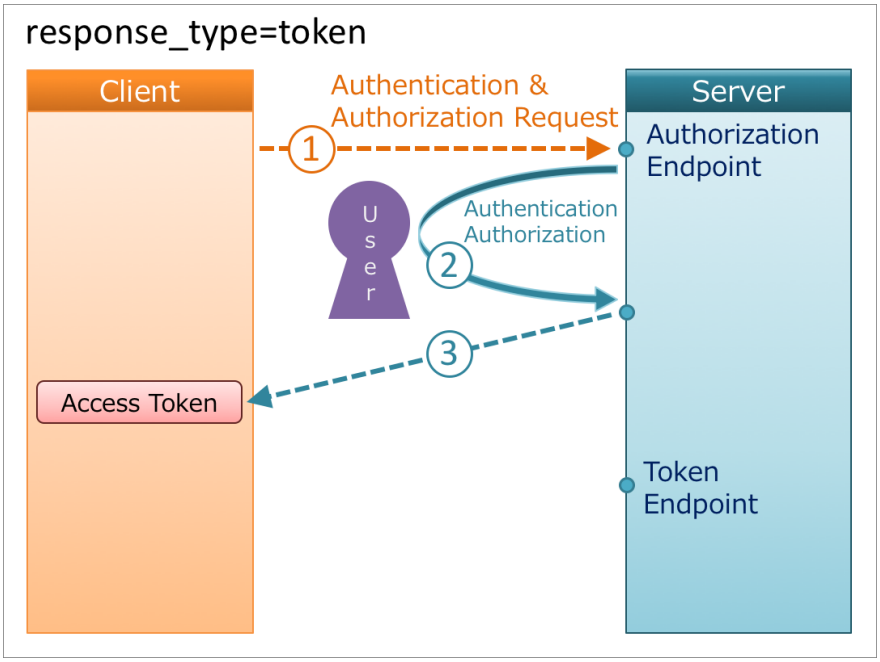
Endpoint	Authorization Code	Access Token	ID Token
Authorization	Issued	X	X
Token	X	Issued	X



2. response_type=token

When the value of `response_type` is `token`, the request is an implicit flow defined in RFC 6749. Even if `openid` is included in the `scope` request parameter, an ID token is not issued. This flow uses the authorization endpoint but does not use the token endpoint.

Endpoint	Authorization Code	Access Token	ID Token
Authorization	X	Issued	X



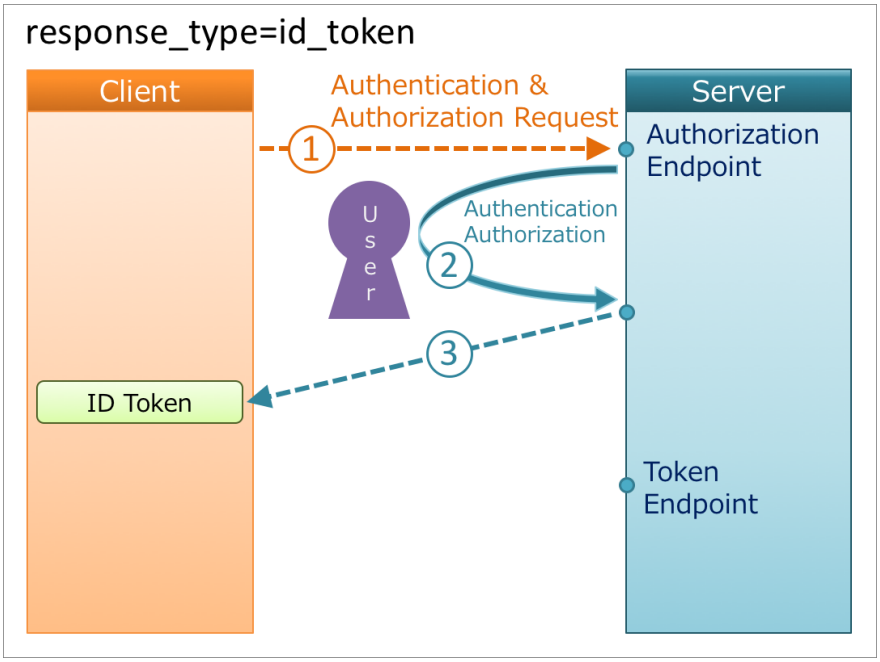
At the end of "*3. Authentication*", OpenID Connect Core 1.0 explicitly states that OpenID Connect does not use `token` as follows:

> *NOTE: While OAuth 2.0 also defines the* **`token`** *Response Type value for the Implicit Flow,* ***OpenID Connect does not use this Response Type****, since no ID Token would be returned.*

3. response_type=id_token

When the value of `response_type` is `id_token`, an ID token is issued from the authorization endpoint. This flow does not use the token endpoint.

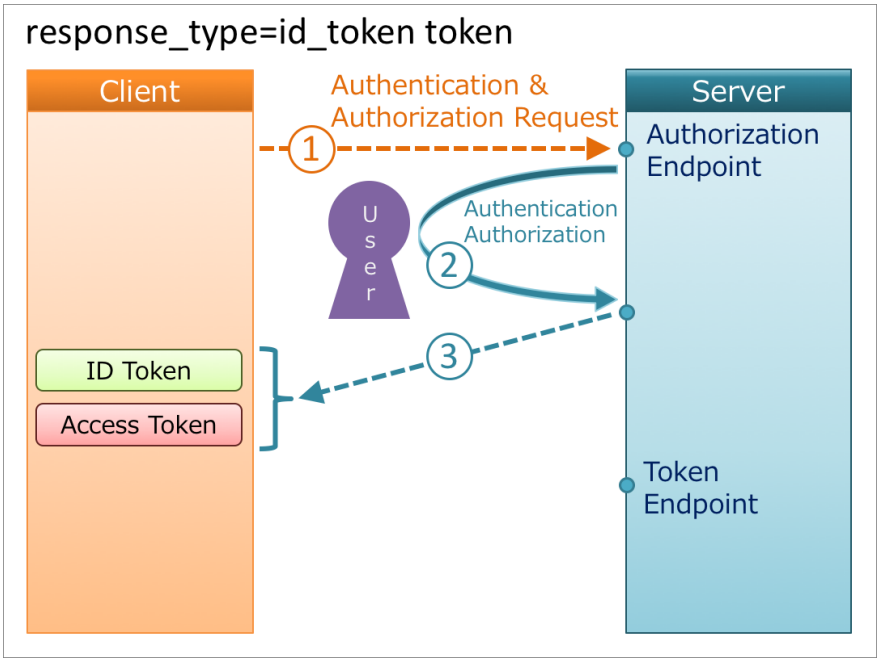
Endpoint	Authorization Code	Access Token	ID Token
Authorization	X	X	Issued



4. response_type=id_token token

When the value of `response_type` is `id_token token`, an ID token and an access token are issued from the authorization endpoint. This flow does not use the token endpoint.

Endpoint	Authorization Code	Access Token	ID Token
Authorization	X	Issued	Issued



When an access token is issued together with an ID token from the authorization endpoint, the hash value of the access token calculated in a certain way has to be embedded in the ID token. So, be careful when you implement this flow. *"3.2.2.10 ID Token"* in OpenID Connect Core 1.0 says as follows:

> ***at_hash***

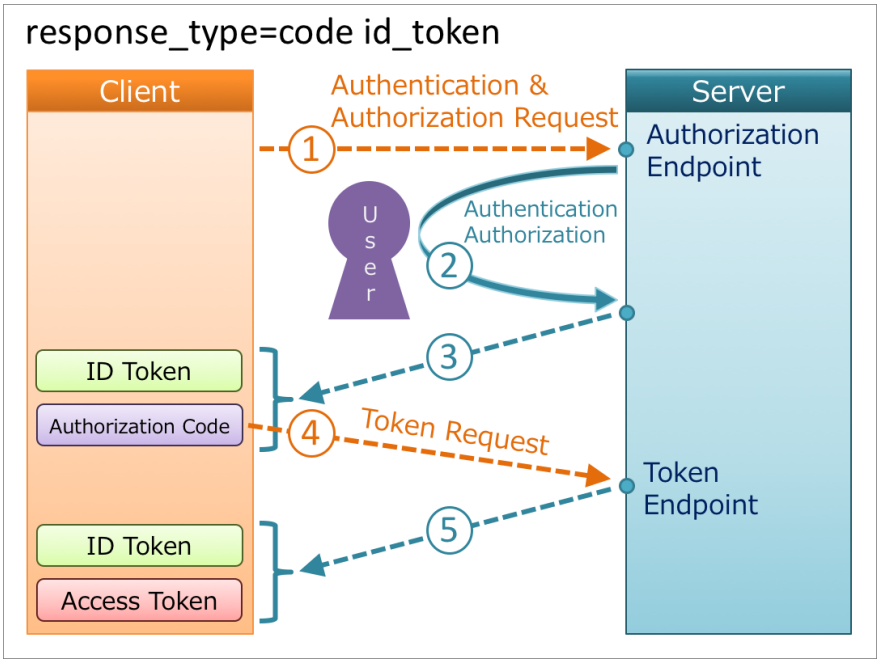
> *Access Token hash value. Its value is the base64url encoding of the left-most half of the hash of the octets of the ASCII representation of the* `access_token` *value, where the hash algorithm used is the hash algorithm used in the* `alg` *Header Parameter of the ID Token's JOSE Header. For instance, if the* `alg` *is* `RS256`*, hash the* `access_token` *value with SHA-256, then take the left-most 128 bits and base64url encode them. The* `at_hash` *value is a case sensitive string.*

> ***If the ID Token is issued from the Authorization Endpoint with an*** `access_token` ***value, which is the case for the*** `response_type` ***value*** `id_token token`***, this is REQUIRED****; it MAY NOT be used when no Access Token is issued, which is the case for the* `response_type` *value* `id_token`*.*

5. response_type=code id_token

When the value of `response_type` is `code id_token`, an authorization code and an ID token are issued from the authorization endpoint, and an access token and an ID token are issued from the token endpoint.

Endpoint	Authorization Code	Access Token	ID Token
Authorization	Issued	X	Issued
Token	X	Issued	Issued



Both the authorization endpoint and the token endpoint issue an ID token, but the contents of the ID tokens are not always the same. Regarding this, *"3.3.3.6 ID Token"* in OpenID Connect Core 1.0 says as follows:

> *If an ID Token is returned from both the Authorization Endpoint and from the Token Endpoint, which is the case for the* `response_type` *values* `code id_token` *and* `code id_token token`*, the* `iss` *and* `sub` *Claim Values MUST be identical in both ID Tokens. All Claims about the Authentication event present in either SHOULD be present in both. If either ID Token contains Claims about the End-User, any that are present in both SHOULD have the same values in both. Note that* ***the OP MAY choose to return fewer Claims about the End-User from the Authorization Endpoint, for instance, for privacy reasons****. The* `at_hash` *and* `c_hash` *Claims MAY be omitted from the ID Token returned from the Token Endpoint even*

> *when these Claims are present in the ID Token returned from the Authorization Endpoint, because the ID Token and Access Token values returned from the Token Endpoint are already cryptographically bound together by the TLS encryption performed by the Token Endpoint.*

When an authorization code is issued together with an ID token from the authorization endpoint, the hash value of the authorization code calculated in a certain way has to be embedded in the ID token. So, be careful when you implement this flow. *"3.3.2.11. ID Token"* in OpenID Connect Core 1.0 says as follows:

> ***c_hash***

> *Code hash value. Its value is the base64url encoding of the left-most half of the hash of the octets of the ASCII representation of the* `code` *value, where the hash algorithm used is the hash algorithm used in the* `alg` *Header Parameter of the ID Token's JOSE Header. For instance, if the* `alg` *is* `HS512`*, hash the* `code` *value with SHA-512, then take the left-most 256 bits and base64url encode them. The* `c_hash` *value is a case sensitive string.*

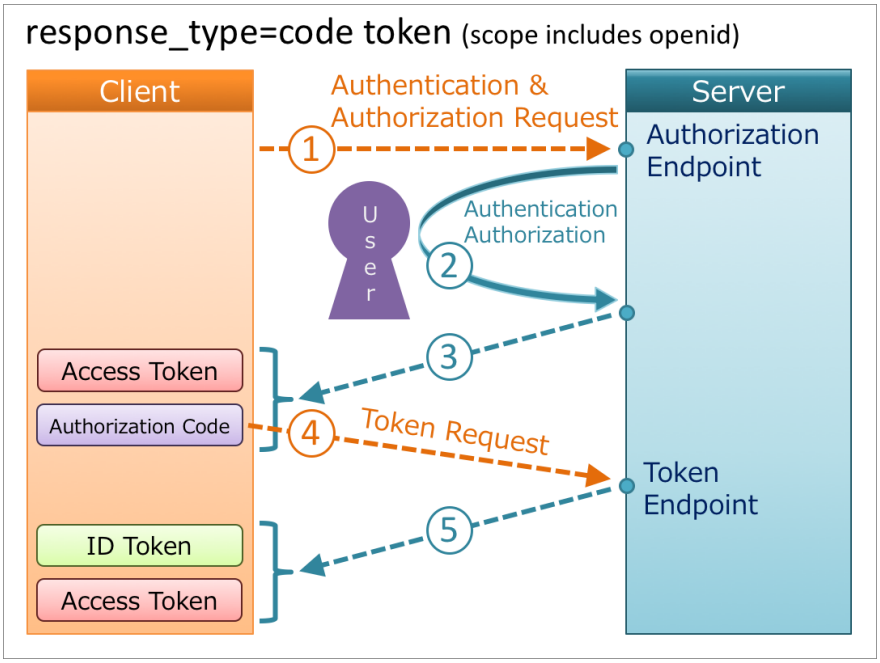
> ***If the ID Token is issued from the Authorization Endpoint with a*** `code`***, which is the case for the*** `response_type` ***values*** `code id_token` ***and*** `code id_token token`***, this is REQUIRED****; otherwise, its inclusion is OPTIONAL.*

6. response_type=code token

When the value of `response_type` is `code token`, an authorization code and an access token are issued from the authorization endpoint, and an access token is issued from the token endpoint. In addition, if `openid` is included in the `scope` request parameter, an ID token is issued from the token endpoint, too.

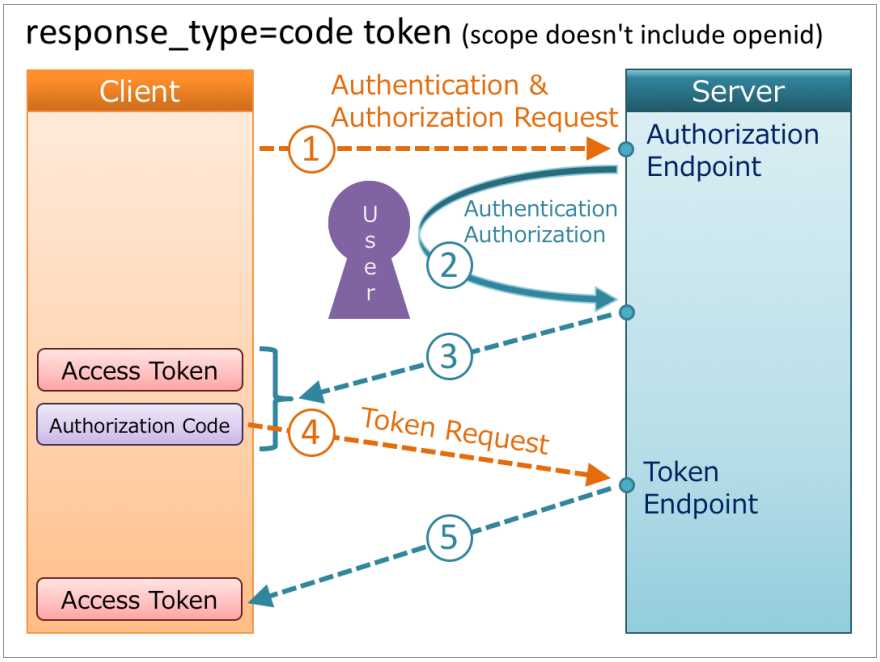
if `openid` is included

Endpoint	Authorization Code	Access Token	ID Token
Authorization	Issued	Issued	X
Token	X	Issued	Issued



if `openid` is not included

Endpoint	Authorization Code	Access Token	ID Token
Authorization	Issued	Issued	X
Token	X	Issued	X



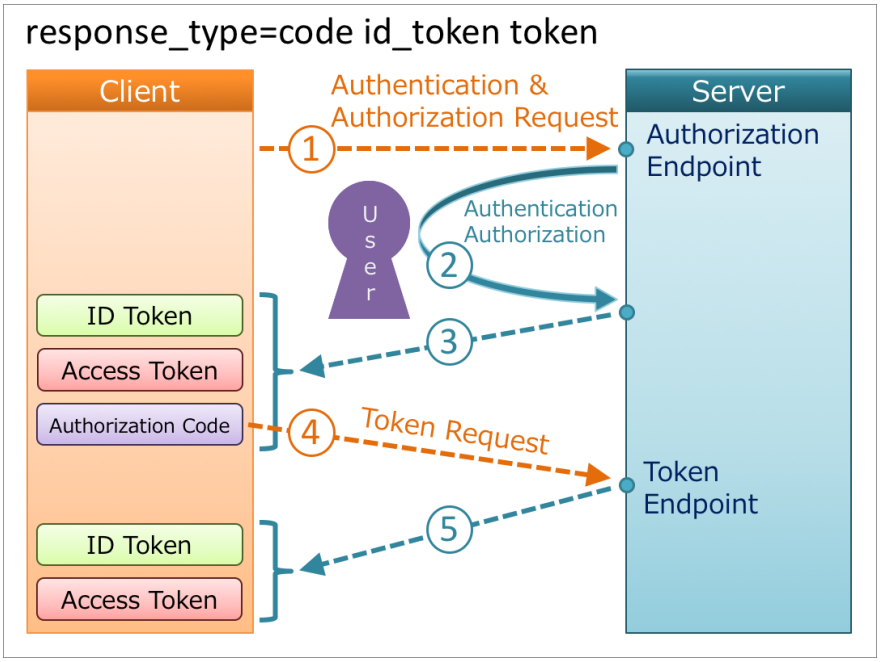
Both the authorization endpoint and the token endpoint issue an access token, but the contents of the access tokens are not always the same. Regarding this, *"3.3.3.8. Access Token"* in OpenID Connect Core 1.0 says as follows:

> *If an Access Token is returned from both the Authorization Endpoint and from the Token Endpoint, which is the case for the* `response_type` *values* `code token` *and* `code id_token token`*, their values MAY be the same or they MAY be different. Note that* ***different Access Tokens might be returned be due to the different security characteristics of the two endpoints and the lifetimes and the access to resources granted by them might also be different****.*

7. response_type=code id_token token

When the value of `response_type` is `code id_token token`, an authorization code, an access token and an ID token are issued from the authorization endpoint, and an access token and an ID token are issued from the token endpoint.

Endpoint	Authorization Code	Access Token	ID Token
Authorization	Issued	Issued	Issued
Token	X	Issued	Issued



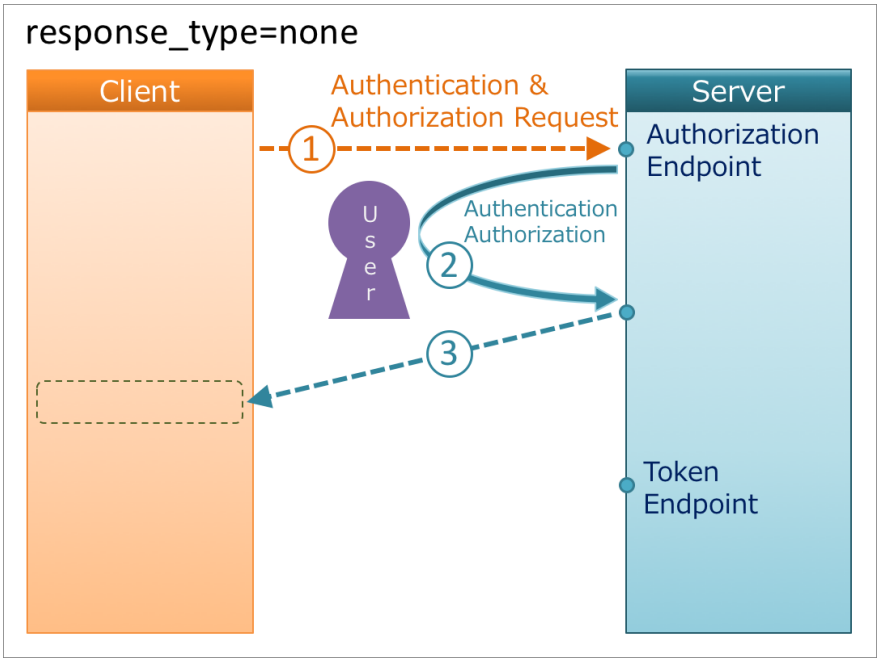
Both the authorization endpoint and the token endpoint issue an access token, but the contents of the access tokens are not always the same, likewise *"6. response_type=code token"*. As for the specification, please refer to *"3.3.3.8. Access Token"* in OpenID Connect Core 1.0.

When an ID token is issued from the authorization endpoint, the hash value of the access token has to be embedded in the ID token if an access token is also issued, and the hash value of the authorization code has to be embedded in the ID token if an authorization code is also issued, likewise *"4. response_type=id_token token"* and *"5. response_type=code id_token"*. As for the specification, please refer to *"3.3.2.11. ID Token"* in OpenID Connect Core 1.0.

8. response_type=none

When the value of `response_type` is `none`, nothing is issued from the authorization endpoint. This flow does not use the token endpoint.

Endpoint	Authorization Code	Access Token	ID Token
Authorization	X	X	X



The definition of `none` is in *"4. None Response Type"* in OAuth 2.0 Multiple Response Type Encoding Practices.

9. Support

Software claiming OpenID Connect support does not always support all the flows described above. Even among the implementations that have OpenID Certification, about the half don't cover Hybrid OP Profile (= don't support hybrid flows).

Financial API Working Group of OpenID Foundation is discussing and defining **Financial API** (**FAPI**). When a client application complying with the FAPI specification makes a request for an access token for write operations, the value of `response_type` of the request must be either `code id_token` or `code id_token token`. *"5.2.2. Authorization Server"* in Financial Services - Financial API - Part 2: Read and Write API Security Profile says as follows:

> *shall require the* `response_type` *values* `code id_token` *or* `code id_token token` *;*

That is, if you are planning to comply with Financial API, you have to use an authorization server which supports hybrid flows. Be careful when you select an authorization server because there exist

implementations which support `response_type=code` only but claim they support OpenID Connect.

Finally

If you feel the flows are too complicated to implement, please consider ***Authlete***. Read ***"New Architecture of OAuth 2.0 and OpenID Connect implementation"***, and you will love the architecture of Authlete 😉