Workgroup: OpenID Connect Working Group
Published: 3 November 2022
Author: G.F.F. Fletcher
Capital One

OpenID Connect Native SSO for Mobile Apps 1.0 - draft 05

Abstract

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

This document describes a mechanism that allows a mobile app to share the identity/authentication obtained by a different mobile app where both apps are written by the same vendor.

Table of Contents

1. Introduction

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 [RFC6749] protocol. It enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

As the industry moves to a more mobile oriented environment, vendors need a way to share identity across the multiple mobile apps they deploy. While the current OAuth2 best practice allows for SSO across any mobile app by sharing the session cookies in the system browser, this has risks such as a user clearing their system browser of cookies (possibly as requested by a customer care agent) or using private browsing on iOS and Android. On most mobile platforms, mobile apps signed by the same vendor certs can share information via the system "keychain" (Account Manager on Android).

This document specifies a new scope, extends the token endpoint and profiles the OAuth2 Token Exchange [RFC8693] spec allowing mobile apps to share identity (SSO) between apps produced and signed by the same vendor (i.e. signed with the same vendor certificate).

1.1. Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

In the .txt version of this specification, values are quoted to indicate that they are to be taken literally. When using these values in protocol messages, the quotes MUST NOT be used as part of the value. In the HTML version of this specification, values to be taken literally are indicated by the use of `this fixed-width font`.

1.2. Terminology

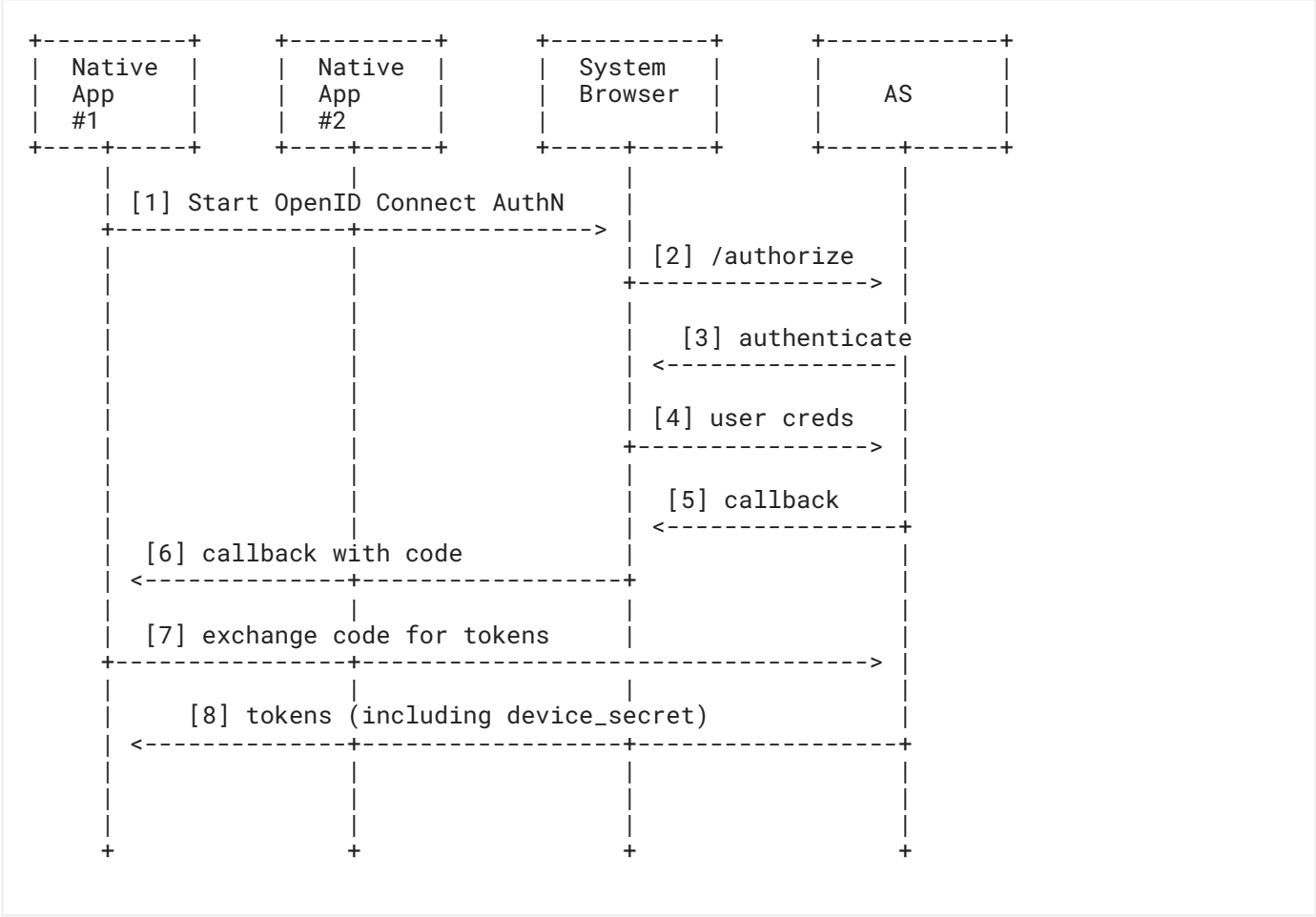
This specification uses the terms "Authorization Server", "Client", "Client Identifier", and "Redirection URI" defined by OAuth 2.0 [RFC6749], the term "User Agent" defined by RFC 7230 [RFC7230], the term "native app" defined by RFC 8252 [RFC8252] and the terms defined by OpenID Connect Core 1.0 [OpenID.Core].

This specification also defines the following terms:

Device Secret
: Opaque value to the client, issued by the Authorization Server, that uniquely identifies the device instance and provides credentials for the device.

Session ID
: Identifier for a user's authentication session.

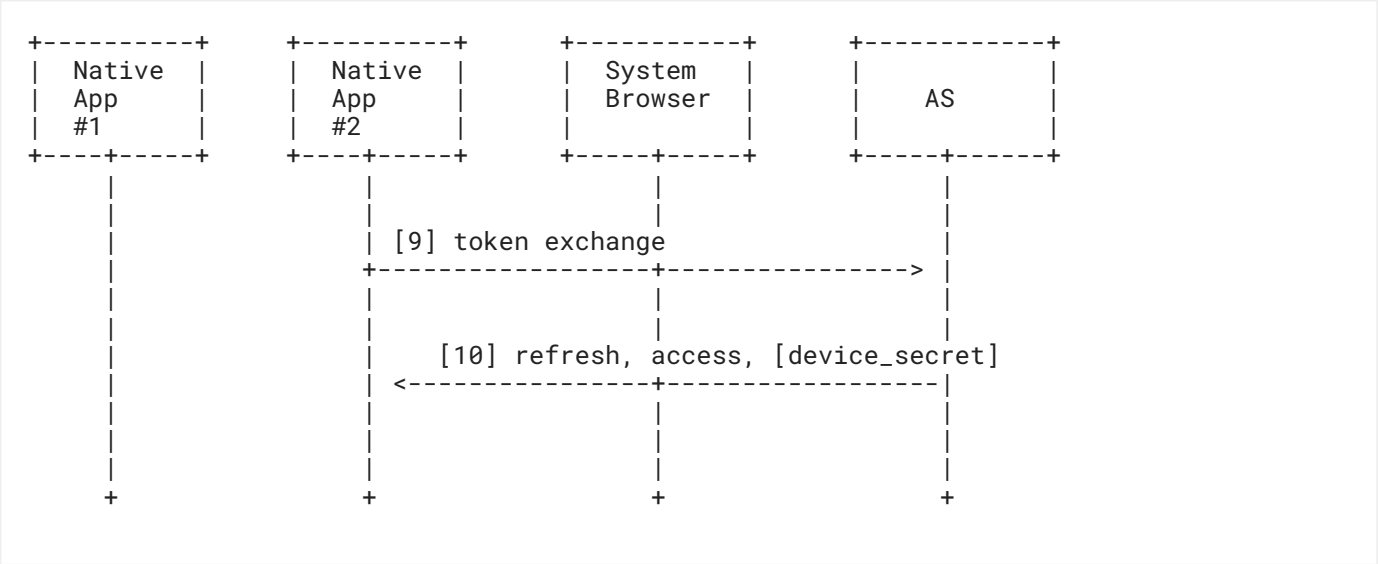
2. Abstract Flow



Steps [1] - [8] are the standard OpenID Connect authorization_code flow with the following extensions. In step 2, the `device_sso` scope is specified signifying that the client is requesting a `device_secret` to be returned when the code is exchanged for tokens.

After step 8, Native App #1 stores the device_secret and id_token in the protected device storage accessible only to Native App #2.

Native App #2 uses the stored data from the shared device storage to obtain tokens for the user thus enabling the app to access the user's resources (i.e. SSO)



Step [9] invokes the /token endpoint with the token exchange profile passing the id_token obtained from the shared device storage, the client_id and the device secret.

Step [10] returns the SSO generated refresh and access tokens for Native App #2.

3. Native App Authorization Extensions

The following sections describe the extensions required to the standard OpenID Connect Authentication flow which will enable a second mobile app to share the authentication of the first mobile app where both mobile applications are signed by the same vendor certificates.

3.1. Authorization Request

This specification defines a new scope value that is used to convey to the Authorization Server that when the code is exchanged for a token, a new `device_secret` will be returned in addition to the other tokens identified as part of the authorization request.

The new scope value is defined as `device_sso`. When this scope is present on the authorization request, when the code is exchanged for tokens, a new device_secret will be returned.

3.2. Device Secret

The device secret contains relevant data to the device and the current users authenticated with the device. The device secret is completely opaque to the client and as such the AS MUST adequately protect the value such as using a JWE if the AS is not maintaining state on the backend.

In the context of this extension the device secret may be shared between mobile apps that can obtain the value via the shared security mechanism (e.g. keychain on iOS). If a mobile app requests a device secret via the `device_sso` scope and a `device_secret` exists, then the client MUST provide the `device_secret` on the request to the /token endpoint to exchange code for tokens. The client SHOULD provide the `device_secret` to the /token endpoint during refresh token requests so that the AS may rotate the device_secret as necessary.

The exact construction of the `device_secret` is out of scope for this specification.

3.3. Token Request

During a normal user authentication via the system browser, after the mobile app receives the code and state response from the Authorization Server, this spec defines the following additional parameters to the /token endpoint for the authorization_code grant_type.

device_secret
: OPTIONAL. This token SHOULD be provided if the client requested the `device_sso` scope and the client already has a `device_secret` available. If no `device_secret` is specified and the refresh_token contains the `device_sso` scope, a new `device_secret` will be generated.

3.4. Token Response

When the authorization server receives the `device_secret` value it MUST process the authorization_code grant type per the OpenID Connect spec with the following additions applying to the id_token.

1. Add a `ds_hash` claim to the id_token to represent a function of the device_secret.

   ds_hash
   : REQUIRED. The ds_hash value provides a binding between the id_token and the issued device_secret. The exact binding between the ds_hash and device_secret is not specified by this profile. As this binding is managed solely by the Authorization Server, the AS can choose how to protect the relationship between the id_token and device_secret.

2. Add a session id to the id_token that represents the user's current authentication session.

   sid
   : REQUIRED. A string that uniquely identifies this user's authentication session. This value can be used in logout flows as well as the flow this spec is describing. For mobile apps where there is no explicit browser authentication this value SHOULD represent the underlying session associated with the refresh_token.

Note that the implementation of this spec and the specification of the ds_hash and sid value MUST NOT leak any data that would provide a security advantage to an attacker who gains access to the id_token.

When the authorization server receives the device_secret it must validate the token. If the token is invalid it must be discarded and the request processed as if no device_secret was specified.

If the authorization request included the device_sso scope then the authorization server MUST return a device_secret in the response. The device_secret is returned in the device_token claim of the returned JSON data.

If no devices_secret is specified, then the AS MUST generate the token. If a device_secret is specified and is valid, the AS MAY update the device_secret as necessary. Regardless a device_secret must be returned in the response.

4. Token Exchange Profile for Native SSO

This section profiles the OAuth2 Token Exchange [RFC8693] spec and describes the processing rules used to exchange a previous authentication for new refresh and access tokens requested by a mobile app created by the same vendor as the first mobile app and both apps signed by the same developer certificate.

4.1. OAuth2 Token Exchange Profile

The client authenticates using its registered token endpoint client authentication method. This could range from HTTP Basic Authentication, to OpenID Connect defined private_key_jwt. The AS must be able to obtain the client_id of the requesting mobile app (mobile app #2) from the client authentication method.

This profile defines the use of the following token exchange parameters.

grant_type
: REQUIRED. The value MUST be urn:ietf:params:oauth:grant-type:token-exchange

audience
: REQUIRED. This parameter defines the logical purview of the returned tokens. For the purposes of this profile, this value MUST be the issuer URI for the OpenID Provider that issued the id_token used in this profile.

subject_token
: REQUIRED. This parameter MUST contain the id_token obtained by the first native app. The id_token is used in the same manner as id_token_hint to identify the user to SSO into the invoking native app.

subject_token_type
: REQUIRED. This parameter MUST contain the value: urn:ietf:params:oauth:token-type:id_token

actor_token
: REQUIRED. This value defines the actor making the request which in this case is the device_secret issued to the device of the native application making the request. The device_secret MUST be presented per the definition of the urn:x-oath:params:oauth:token-type:device-secret token identifier described below.

actor_token_type
: REQUIRED. This value MUST be: urn:x-oath:params:oauth:token-type:device-secret

scope
: OPTIONAL. The scopes required by the requesting native application.

requested_token_type
: OPTIONAL. The desired token(s) to be returned. If no requested_token_type is defined, it is up to the AS to return the appropriate tokens for the requesting client. The full set of possible requested_token_type value is out of scope for this specification.

This profile also defines the following token type identifiers.

urn:x-oath:params:oauth:token-type:device-secret
: This token type identifier is used to describe the device_secret specified in the actor_token parameter.

4.2. Token Exchange Request

When a mobile app wants to request native SSO (i.e. obtain refresh and access tokens for an already signed in user) it makes a standard OAuth2 /token endpoint request following the profile for Token Exchange defined above.

The following is a non-normative example

```
POST /token HTTP/1.1
Host: as.example.com
Authorization: Basic ZGZhZGYyMzUyNDU0Og
...
grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Atoken-exchange
&audience=https%3A%3F%3Flogin.example.net&subject_token=<id_token>
&subject_token_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-type%3Aid-token
&actor_token=95twdf3w4y6wvftw35634t
&actor_token_type=urn%3Ax-oath%3Aparams%3Aoauth%3Atoken-type%3Adevice-secret
```

The client_id in this request is sent via the HTTP Basic Authentication method using the HTTP Authorization header.

4.3. Native SSO Processing Rules

When the authorization server receives a request at the token endpoint conforming to this profile it MUST perform the following checks before issuing any tokens.

1. Validate the device_secret to ensure the token is still valid. The format of this secret is defined by the Authorization server and is out of scope for this specification.
2. Verify the id_token for integrity protection by validating the signature of the id_token JWT.
3. Verify that the binding between the id_token and the device_secret (as defined in the extension to the /token response) is valid.
4. Verify that the session id in the id_token (sid claim) is still valid. If the session is no longer valid, the AS MUST return an error of invalid_grant. Note that in the case of a refresh_tokens issued with an offline_scope the 'sid' value SHOULD represent the offline "session" such that if the original refresh_token is revoked the 'ds_hash' value in the id_token is no longer valid.
5. Validate that the client requesting native SSO is authorized to do so. The AS SHOULD maintain a list of client_ids that can share user authentications. For example, the AS MAY take the 'aud' claim from the id_token and the client_id from the token request and ensures that both client_ids are allowed to share user authentications.
6. The AS SHOULD verify that the scopes requested by the client in the token request (either default scopes or explicitly specified in the optional scope parameter) do NOT require explicit user consent. If any requested scopes require explicit user consent the AS SHOULD fail the request and return an error of invalid_scope.

Based on the AS definition of the device_secret, the AS may perform additional checks to ensure the security of the request. Provided the above criteria is met, the AS will issue a normal Token Response object containing a refresh_token, access_token and id_token issued to the client_id of the mobile app making the request. The session associated with the new refresh_token SHOULD be the same as that used to verify the validity of the SSO exchange. If that session expires, all refresh_tokens associated with it MUST be invalidated.

4.4. Token Exchange Response

The Token Exchange response for this profile has the following characteristics:

access_token
: REQUIRED. This response field contains the access token issued to the mobile client identified by the client_id sent in the Authorization header.

issued_token_type
: REQUIRED. This value of this parameter MUST be: urn:ietf:params:oauth:token-type:access_token

token_type
: REQUIRED. The value of this parameter MUST be `bearer`.

expires_in
: RECOMMENDED. Identifies when the access_token expires.

scope
: OPTIONAL. Follows the token exchange spec definition.

refresh_token
: OPTIONAL. By default the AS should return a refresh_token that the mobile app can use to obtain additional access_tokens when the access_token expires.

id_token
: OPTIONAL. By default the AS should return an id_token that provides the mobile app with an identity assertion about the user.

device_secret
: OPTIONAL. The AS MAY return an updated device_secret in the response.

In the case of any errors, the AS MUST return a valid OAuth2 Error response as described in Section 2.2.2 of the Token Exchange spec.

The following is a non-normative example

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "issued_token_type": "urn:ietf:params:oauth:token-type:access_token",
  "token_type":"Bearer",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
  "id_token":"<id_token>",
  "device_secret":"casdfgarfgasdfg"
}
```

5. Security Considerations

5.1. Device Secret Protection

Sufficient care must be made to protect the `device_secret`. The device secret SHOULD be encrypted by the Authorization Service and periodically refreshed via the mechanisms described in this specification.

5.2. Cross-Device SSO

If it is possible to move correctly bound `device_secret` and `id_token` to a separate device, that device can obtain the user's authentication state on the new device. An implementation of this specification SHOULD use best efforts to bind the device_secret to the device instance.

5.3. id_token usage

Use of the id_token in this specification takes some liberties with id_token validation. For instance, the aud claim normally identifies the client receiving the id_token and not the authorization server that issued the id_token. Additionally, the id_token may have expired at the time of use designated by the specification.

6. Normative References

[IANA.OAuth.Parameters] IANA, "OAuth Parameters", <http://www.iana.org/assignments/oauth-parameters>.

[OpenID.Core] Sakimura, N., Bradley, J., Jones, M.B., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", 8 November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC8252] Denniss, W. and J. Bradley, "OAuth 2.0 for Native Apps", BCP 212, RFC 8252, DOI 10.17487/RFC8252, October 2017, <https://www.rfc-editor.org/info/rfc8252>.

[RFC8693] Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020, <https://www.rfc-editor.org/info/rfc8693>.

Appendix A. Acknowledgements

The OpenID Community would like to thank the following people for their contributions to this specification:

Filip Skokan

Joseph Heenan

Nat Sakimura

Naveen CM

Nov Matake

Appendix B. Document History

[[To be removed from the final specification]]

-00

- Initial Draft.
- Draft-03 - Applied updates from the community. Removed non-relevant IANA claims section.

- Draft-04 - Normative change to verify the id_token plus other edits based on feedback from Joseph Heenan. Also updated Security Considerations section and general cleanup.
- Draft-05 - Updated the draft version and tweaked some language based on updates from Naveen CM.

Author's Address

George F. Fletcher
Capital One
Email: gffletch@aol.com