Self-Issued OP

~DIF and OIDF, or Decentralized identity and OIDC~

Kristina Yasuda, Microsoft Identity
Oliver Terbu, Consensus Mesh
1. Three work-streams
   i.  Self-Issued OpenID Provider (SIOP V2)
   ii. Presentation of W3C verifiable credentials using OIDC
   iii. Issuance of aggregated/client-bound claims from Claims Providers

2. Use-cases
1-Ł. Self-Issued OpenID Provider

Specific model where users control their own
OpenID Providers - extension of Chapter 7

Issues raised

- Different Trust Model between Self-Issued
  OP and RP from that of the rest of OIDC?
  - Ad Hoc Registration is proposed
  - Need to communicate info about SIOP’s
    provider? iss=self-issued.me
  - consent, etc.
  - will deep-dive on two
1-i. SIOP V2 Issues progress (1/2)

- Which SIOP/wallet under user’s possession to invoke? following options:
  
  - 1. SIOP Chooser ([https://bitbucket.org/openid/connect/issues/1212/siop-chooser](https://bitbucket.org/openid/connect/issues/1212/siop-chooser))
    - a combination of 1/ a list of wallets (maintained by the trust framework); 2/ universal links to open wallet from the browser; and 3/ share sheet to choose between several wallets under the user’s control.
    - a current best solution that will work with different kind of wallets - native apps, PWAs, browser wallets.
  
  - 2. Each wallet pre-registering custom URL schema with RP
    - NASCAR problem remains

Not the ideal solution, but the most viable without OS vendor’s collaboration.
1-i. SIOP V2 Issues progress (1/2)

- Need for a user to prove control over the Self-Issued OP
  - in addition to jwk thumbprint, allow DIDDs to be used as holder identifier by checking if ID Token is signed by the keys in the DIDDoc controlled by the user - benefit of a key rotation

3.2. Self-Issued OpenID Provider Response
sub
REQUIRED. Subject identifier value, represented by a URI. When sub type is jkt, the value is the base64url encoded representation of the thumbprint of the key in the sub_jwk Claim. When sub type is did, the value is a decentralized identifier.
1-ii. Presentation of W3C verifiable credentials using OIDC

- Support request and presentation of Verifiable Credentials in ID Tokens and Userinfo responses
- Usable with all OpenID Connect Flows (SIOP, code, CIBA, …)
- Leverage OpenID Connect as simple to use protocol for wallet integrations
- Leverage W3C verifiable credentials to existing OpenID Connect deployments
Current Spec work

- Request
  - via “claims” parameter
  - Simply claims or credential type or credential type + claims (selective disclosure)
- Working on a draft that allows for both options to gather implementation feedback with a goal of making a decision on which option to adopt
  - A) Embedding entire VP/VC in any format
    - https://github.com/Sakurann/vp-token-spec
    - ease of adoption in existing implementations
  - B) VP Token as separate artifact returned alongside ID Token from the authorization endpoint
    - https://github.com/awoie/vp-token-spec
    - ‘clean’ technical solution

→ So that VPs are returned using same syntax in both options, will also define generic container to convey VPs - something like an array with objects containing a format identifier and the actual payload (+ potentially some additional metadata).

Will be contributed to the WG & call for adoption in coming week
A. vp_jwt Claim

```
{
  "id_token": {
    "acr": null,
    "vp_jwt": {
      "credential_types": [
        "https://www.w3.org/2018/credentials#Credential"
      ],
      "iss": "https://self-issued.me",
      "iat": 1615910538,
      "exp": 1615911138,
      "sub": "did:ion:EiC6Y9_aDaCsITlY06HId4seJjj-9...s3NBI1n19",
      "auth_time": 1615910535,
      "nonce": "960848874",
      "vp_jwt": [
        "ewogICAglmlyzY6Imh0dHBzOi8vYm9vay5pdHNvdXJ3ZWJub...\n      I\n    ],
    "sub_jwk": {
      "crv": "P-384",
      "kty": "EC",
      "kid": "c7298a61a6904426a580b1df31e42d0",
      "x": "jf3a6dqucl24Pj0JMUn8RwucG9T103hpU_S_79sHqi7VZBD9e2VKXPts9UjyBm",
      "y": "38VLVE3kNIEMeJkLFe4W04DqdTKK6QuZm771C\n      MN2x9bENzoGF2EYFiB5s0sq0"
    }
  }
}
```
A. vp_ldp Claim

```
{
    "id_token": {
        "vp_ldp": {
            "credential": {
                "claims": {
                    "given_name": "John",
                    "family_name": "Doe",
                    "birthdate": "1990-01-01"
                }
            },
            "type": ["VerifiablePresentation"]
        },
        "verifiableCredential": {
            "@context": [
                "https://www.w3.org/2018/credentials/v1",
                "https://www.w3.org/2018/credentials/examples/v1"
            ],
            "id": "https://example.com/credentials/1872",
            "type": ["VerifiableCredential",
                      "IDCardCredential"
            ],
            "issuer": {
                "id": "did:example:issuer"
            },
            "issuanceDate": "2010-01-01T19:23:24Z",
            "credentialSubject": {
                "@context": ["https://www.w3.org/2018/credentials/v1"],
                "type": ["Passport"
                          "VoterRegistration"
                ],
                "given_name": "John",
                "family_name": "Doe",
                "gender": "Male",
                "date_of_birth": "1990-01-01"
            }
        }
    }
}
```
B. Separate artifact

- ‘VP Token’

`claims` parameter in the request

```json
{
  "id_token":{
    "acr":null
  },
  "vp_token":{
    "format": "json-ld",
    "credential_types": [
      {
        "type": "https://www.w3.org/2018/vch/credentials/v1",
        "claims": {
          "given_name": null,
          "family_name": null,
          "birthdate": null
        }
      }
    ]
  }
}
```

ID Token contains a `vp_hash`

```
{
  "iat": 1615910538,
  "exp": 1615911138,
  "sub": "urn:uuid:68f874e2-377c-437f-a447-b304967ca351",
  "auth_time": 1615910535,
  "vp_hash": "77QmuUPtdFz3WtF2AnpK9RQ",
  "nonce": "960848874",
  "sub_ikw": {
    ...
  }
}
```

‘VP Token’ contains an entire VP

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1"
  ],
  "type": [
    "VerifiablePresentation"
  ],
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://www.w3.org/2018/credentials/examples/v1"
      ],
      ...
    }
  ]
}
```
1-iii. Issuance of aggregated/client-bound claims from Claims

Specify the methods for an application to:
- perform discovery for a Claims Provider
- register a client to a Claims Provider
- obtain claims from the Claims Provider
- return aggregated claims from Claims Providers to requesting clients
OpenID Connect has 3 claims models

1. Simple Claims
2. Aggregated Claims
3. Distributed Claims

- C acts as an OP to D which is an RP in this context
- A&B acts as an OP to D which is an RP in this context
- E acts as a resource to D

Note: Credit to Nat Sakimura
Weakness of the Connect Core defined aggregated claims

- How to get a token from CP is hand-wavy.
- No specified method to down scope the userinfo of the CP.
- No way to provide a binding information between CP:sub and IdP:sub.

OIDC Claims aggregation draft (WG adopted, issues filled in)

https://bitbucket.org/openid/connect/src/master/openid-connect-claims-aggregation/openid-connect-claims-aggregation-1_0.md

(Discussions to converge with Credential Provider draft - to be contributed)

https://github.com/mattrglobal/oidc-client-bound-assertions-spec

Note: Credit to Nat Sakimura
OIDC flows

Authorization Code Flow

Relying Party (RP)

1. Authn & Authz Req
2. User Authn & Authz
3. Token Req

OpenID Provider (OP)

Authorization Endpoint

Token Endpoint

User Info Endpoint

Claims Aggregation

RP

1. Authn & Authz Req
2. User Authn & Authz

Token Endpoint (OP)

ID Token

Access Token

Authz Code

Claims

Endpoint (OP)

UserInfo Endpoint (OP)

Claims

Endpoint (belongs to Claims Provider)

Note: Credit to @TakahikoKawasaki
1. Give me claims \{a,b\}

2. Is it ok to give \{a,b\} to D?

3. I grant.

4. Give me a.
   Token = Ta

5. Signed claims

6. Give me b.
   Token = Tb

7. Signed claims

8. Here are \{a,b\} with the user identification claims c.

Identity Register

CP (A)

CP (B)

User

IdP (wallet etc.)

D Client

Signed Claims (Token)

Note: Credit to Nat Sakimura
2. **Use-cases**

User’s having OPs that they control; users being able to receive and present verifiable credentials

-> “What problem it solves that current technology does not solve”

- Privacy preservation - no issuer call home at presentation.
  - mDL (mobile Driving License defined as ISO/IEC 18013-5)
- Addressing issuers-ceased-to-exist use case.
  - University issues student cards for alumni, which alumni can use regardless of university existence. (also cost saving because university potentially does not have to maintain alumni records in the registry) -> Keio University
- Claims Aggregation & User-consent
  - NHS verifying doctors’ eligibility using digital claims from several sources and saving patient treating time
- Also remote onboarding, getting app access and self-service recovery
- Other use-cases?
- Weekly SIOP Special Topic Calls
  - Alternating Pacific and Atlantic time-zone calls

- OIDC AB/Connect WG calls
  - Weekly Pacific time-zone calls and
  - Bi-weekly Atlantic time-zone calls

+ Bitbucket issues, PRs 😊