WG: Credential Issuance and Presentation

The new publications of the OpenIDConnect standards for credential issuance and presentation allows for standardized flows. This book is a place for documenting the design process of adopting these standards into the Nuts network.

- Notes on exploring Credential Issuance

Notes on exploring Credential Issuance

About this page

This page contains my (Steven) notes taken while exploring the new OIDC4CI and OIDC4P specifications.

The relevant specifications

- OpenID for Verifiable Credential Issuance. Main specificaions. Contains flows for issueing credentials, including OpenID Connect flows for obtaining `access_tokens`.
- OpenID for Verifiable Presentations, Specification used for requesting and presenting Verifiable Presentations, needed during the issuance flow.
- OpenID SIOPv2. Specification about communicating with a wallet.
- Status List 2021. Standard used for issuers to revoke or suspend credentials.
- Peer DID Method Specification. A DID format which adds transactions to the resolver results so key rotations can be supported. A simpler version of the keri format.
- DIF Presentation Exchange. Specification which describes a query format for requesting VPs and VCs from a wallet, used in the OpenIDConnecet for Verifiable Presentation flows.

Nuts node features:

What does the Nuts node need to support in order to be a decent trust layer?

1. Issuance of VCs to a holder
    1. Wallet initalizes the flow.
        1. LRZA credentials
        2. Request an `NutsAuthCredential` (Toestemmingsverzoek)
    2. Issuer initializes de flow:
        1. Issuance of a `NutsAuthCredential` (overdracht, bgz, netwerkzorg)
2. Issuer should be able to change state of the credential: suspending, revoking
3. APIClient should be able to obtain an `access_token` for (e.g. FHIR) API access
    1. AuthServer should be able to request additional VPs from client.
4. Searching for DIDs bases on relevant properties
    1. Relevance depends on the use-case
    2. Information should be published
    3. Trust relation between publisher and the searcher
    4. All claims should be able to be verified by VP request on actual interaction

5. Resolving of public key material bases on (potentially multiple) DID methods
    1. DID web method for Nuts? `did:nuts:web:nuts.example.com/123` where `nuts.example.com` represents the domain of the vendor. Con: no history of the DID document, trust based on DNS. Potentially add alternative methods, or define a simple web-method based nuts method bassed on did:peer.
6. Resolving of endpoint for services
    1. Most of the OpenIDConnect standards use the `.well_known/x` endpoints. Can this be used as an alternative for the Nuts services?

Issuer initiated flow with SIOP flow:

Example request and respond message for issuing a `AuthorisationCredential` in the Nuts network. We assume that the issuer already knows the subjects DID.

- Issuer DID: `did:nuts:123`
- Subject DID: `did:nuts:456`
- LRZA Issuer: `did:web:lrza`

Stepts are based on the following sequence diagram: Edit

sequence diagram

Image not found or type unknown

1. Credential offer request: Lookup the wallet meta data and take the `credential_offer_endpoint` This might be found in the services from the DID document, or found on a `.well_known` endpoint.

Create a `issuer_state` and store it with the current date and subjectDID.

Perform the following request `GET <credential_offer_endpoint>?credential_offer=offer` Where `offer` is the url encoded value of the following JSON object:

```
{
 "credential_issuer":"did:nuts:123",
 "credentials":[{
  "format": "ldp_vc",
  "@context": [
   "https://www.w3.org/2018/credentials/v1",
   "https://nuts.nl/credentials/v1"
  ],
  "types": [
   "VerifiableCredential",
   "UniversityDegreeCredential"
  ]
```

```
  }],
  "grants":{
    "authorization_code":{
      "issuer_state": "state123"
  }
}
```

2. Wallet: Store offer + datum in FetchCredentialQueue
3. Wallet: Take next item in the FetchCredentialQueue en create an AuthorizationRequest

Get issuer metadata. Lookup `credentials_supported`:

```
{
  "credential_issuer":"did:nuts:123",
  "credential_endpoint": "https://nuts.example.com/credentials",
  "credentials_supported":[{
    "format": "ldp_vc",
    "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://nuts.nl/credentials/v1"
    ],
    "types": [
        "VerifiableCredential",
        "NutsAuthorizationCredential"
    ],
    "cryptographic_binding_methods_supported": [
        "did"
    ],
    "cryptographic_suites_supported": [
        "JsonWebSignature2020"
    ]
  },
  {...}]
}
```

Create the `authorization_details` object

```
[
   {
     "type":"openid_credential",
     "format": "ldp_vc",
```

```
    "credential_definition": {
      "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://nuts.nl/credentials/v1"
      ],
      "types": [
        "VerifiableCredential",
        "NutsAuthorizationCredential"
      ]
    }
  },

]
```

4. Perform a

```
GET https://issuer.example.com/authorize?
  response_type=code
  &client_id=did:nuts:456
  &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
  &code_challenge_method=S256
  &authorization_details=...
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
  &wallet_issuer= https%3A%2F%2Fclient.example.org%2Fwallet%2Fdid%3Anuts%3A456
```

5. Issuer performs a wallet meta data lookup
6. Meta information returns
7. Auth request to wallet SIOP server: Build the `presentation_definition`:

```
{
  "id": "request-of-a-LRZA-credential",
  "input_descriptors": [{
    "id": "some-unique-id",
    "constraints": {
      "fields": [{
        "path": [
          "$.issuer"
        ],
        "filter": {
          "type": "string",
```

```
        "pattern": "did:web:lrza"
      }
    },
    {
      "path": [
        "$.credentialSubject.id"
      ],
      "filter": {
        "type": "string",
        "pattern": "did:nuts:456"
      }
    },
    {
      "path": [
        "$.type"
      ],
      "filter": {
        "type": "array",
        "uniqueItems": true,
        "contains": {
          "enum": ["VerifiableCredential", "LRZACredential"]
        },
        "minContains": 2
      }
    }
  ]
 }
],
"purpose": "validate the organization identity in order to issue an auth credential",
"format": {
  "ldp_vp": {
    "proof_type": ["JsonWebSignature2020"]
  }
}
}
```

```
GET https://client.example.org/wallets/did:nuts:456?
  response_type=vp_token
  &client_id=did:nuts:123
  &redirect_uri=https://issuer.example.org/cb
```

```
&presentation_definition=<definition>
&nonce=n-0S6_WzA2Mj
```

8. Token antwoord

```
Location: https://client.example.org/cb#
 presentation_submission=...
 &vp_token=...
```

vp_token:

```
{
   "@context": [
      "https://www.w3.org/2018/credentials/v1"
   ],
   "type": [
      "VerifiablePresentation"
   ],
   "verifiableCredential": [
      {
         "@context": [
            "https://www.w3.org/2018/credentials/v1",
            "https://nuts.nl/credentials/v1"
         ],
         "id": "did:web:lrza#abc",
         "type": [
            "VerifiableCredential",
            "LRZACredentials"
         ],
         "issuer": {
            "id": "did:web:lrza"
         },
         "issuanceDate": "2010-01-01T19:23:24Z",
         "credentialSubject": {
            "id":"did:nuts:456",
□□□□ "kvk":"776655",
            "name":"De regenboog",
            "city":"hengelo",
         },
         "proof": {
```

```
            "type": "JsonWebSignature2020",
            "created": "2021-03-19T15:30:15Z",
            "jws":
"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..PT8yCqVjj5ZHD0W36zsBQ47oc3El07WGPWaLU
uBTOT48IgKI5HDoiFUt9idChT_Zh5s8cF_2cSRWELuD8JQdBw",
            "proofPurpose": "assertionMethod",
            "verificationMethod": "did:web:lrza#keys-1"
         }
      }
   ],
   "id": "ebc6f1c2",
   "holder": "did:nuts:456",
   "proof": {
      "type": "JsonWebSignature2020",
      "created": "2021-03-19T15:30:15Z",
      "challenge": "n-0S6_WzA2Mj",
      "domain": "https://client.example.org/cb",
      "jws":
"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..GF5Z6TamgNE8QjE3RbiDOj3n_t25_1K7NVWMU
ASe_OEzQV63GaKdu235MCS3hIYvepcNdQ_ZOKpGNCf0vIAoDA",
      "proofPurpose": "authentication",
      "verificationMethod": "did:nuts:456#key-1"
   }
}
```


Thoughts: So, during requesting an access_token, instead of sending al information with the request, just tell the other side the scope (i.e. eOverdracht) and the AuthServer will request the required VPs. But what about the user session? Perhaps store the IRMA sessions in the wallet of the care provider and initialize the request with the irma session so the Nuts node knows which IRMA session to provide.