Workgroup: connect
Internet-Draft: openid-connect-4-verifiable-presentations-00
Published: 3 May 2021
Intended Status: Standards Track
Authors: O. Terbu *ConsenSys Mesh*
T. Lodderstedt *yes.com*
K. Yasuda *Microsoft*
A. Lemmon *Convergence.tech*
T. Looker *Mattr*

OpenID Connect for W3C Verifiable Credential Objects

Abstract

This specification defines an extension of OpenID Connect to allow presentation of claims in the form of W3C Verifiable Credentials as part of the protocol flow in addition to claims provided in the `id_token` and/or via Userinfo responses.

Table of Contents

1. Introduction
2. Use Cases
   2.1. Verifier accesses Wallet via OpenID Connect
   2.2. Existing OpenID Connect RP integrates SSI wallets
   2.3. Existing OpenID Connect OP as custodian of End-User Credentials
   2.4. Federated OpenID Connect OP adds device-local mode
3. Terminology
4. Overview
5. Container Format
6. JWT parameters extention
7. New Tokens extention
8. Requesting Verifiable Presentations
   8.1. Embedded Verifiable Presentations
      8.1.1. VP Token
9. Examples
   9.1. Self-Issued OpenID Provider with Verifiable Presentation in ID Token
      9.1.1. Authentication request
      9.1.2. Authentication Response
   9.2. Self-Issued OpenID Provider with Verifiable Presentation in ID Token (selective disclosure)
      9.2.1. `claims` parameter
      9.2.2. Authentication Response
   9.3. Authorization Code Flow with Verifiable Presentation in ID Token
      9.3.1. Authentication Request
      9.3.2. Authentication Response
      9.3.3. Token Request
   9.4. Authorization Code Flow with Verifiable Presentation returned from the UserInfo endpoint
      9.4.1. Authentication Request
      9.4.2. Authentication Response
      9.4.3. Token Request

9.4.4. Token Response

9.4.5. UserInfo Response

9.5. Authorization Code Flow with Verifiable Presentation returned from the UserInfo endpoint (LDP)

9.5.1. Claims parameter

9.5.2. Token Response

9.5.3. UserInfo Response

9.6. SIOP with vp_token

9.6.1. Authentication request

9.6.2. Authentication Response (including vp_token)

9.7. Authorization Code Flow with vp_token

9.7.1. Authentication Request

9.7.2. Authentication Response

9.7.3. Token Request

9.7.4. Token Response (including vp_token)

Appendix A. IANA Considerations

Appendix B. Acknowledgements

Appendix C. Notices

Appendix D. Document History

Appendix E. Related Issues

Authors' Addresses

1. Introduction

This specification extends OpenID Connect with support for presentation of claims via W3C Verifiable Credentials. This allows existing OpenID Connect RPs to extends their reach towards claims sources asserting claims in this format. It also allows new applications built using Verifiable Credentials to utilize OpenID Connect as integration and interoperability layer towards credential holders.

2. Use Cases

2.1. Verifier accesses Wallet via OpenID Connect

A Verifier uses OpenID Connect to obtain verifiable presentations. This is a simple and mature way to obtain identity data. From a technical perspective, this also makes integration with OAuth-protected APIs easier as OpenID Connect is based on OAuth.

2.2. Existing OpenID Connect RP integrates SSI wallets

An application currently utilizing OpenID Connect for accessing various federated identity providers can use the same protocol to also integrate with emerging SSI-based wallets. Thats an conveient transition path leveraging existing expertise and protecting investments made.

2.3. Existing OpenID Connect OP as custodian of End-User Credentials

An existing OpenID Connect may extends its service by maintaining credentials issued by other claims sources on behalf of its customers. Customers can mix claims of the OP and from their credentials to fulfil authentication requests.

2.4. Federated OpenID Connect OP adds device-local mode

An extisting OpenID Connect OP with a native user experience (PWA or native app) issues Verifiable Credentials and stores it on the user's device linked to a private key residing on this device under the user's control. For every authentication request, the native user experience first checks whether this request can be fulfilled using the locally stored credentials. If so, it generates a presentations signed with the user's keys in order to prevent replay of the credential.

This approach dramatically reduces latency and reduces load on the OP's servers. Moreover, the user can identity, authenticate, and authorize even in situations with unstable or without internet connectivity.

3. Terminology

Credential

A set of one or more claims made by an issuer. (see https://www.w3.org/TR/vc-data-model/#terminology)

Verifiable Credential (VC)

A verifiable credential is a tamper-evident credential that has authorship that can be cryptographically verified. Verifiable credentials can be used to build verifiable presentations, which can also be cryptographically verified. The claims in a credential can be about different subjects. (see https://www.w3.org/TR/vc-data-model/#terminology)

Presentation

Data derived from one or more verifiable credentials, issued by one or more issuers, that is shared with a specific verifier. (see https://www.w3.org/TR/vc-data-model/#terminology)

Verified Presentation (VP)

A verifiable presentation is a tamper-evident presentation encoded in such a way that authorship of the data can be trusted after a process of cryptographic verification. Certain types of verifiable presentations might contain data that is synthesized from, but do not contain, the original verifiable credentials (for example, zero-knowledge proofs). (see https://www.w3.org/TR/vc-data-model/#terminology)

W3C Verifiable Credential Objects

Both verifiable credentials and verifiable presentations

4. Overview

This specification defines mechanisms to allow RPs to request and OPs to provide Verifiable Presentations via OpenID Connect.

Verifiable Presentations are used to present claims along with cryptographic proofs of the link between presenter and subject of the verifiable credentials it contains. A verifiable presentation can contain a subset of claims asserted in a certain credential (selective disclosure) and it can assemble claims from different credentials.

There are two credential formats to VCs and VPs: JSON or JSON-LD. There are also two proof formats to VCs and VPs: JWT and Linked Data Proofs. Each of those formats has different properties and capabilites and each of them comes with different proof types. Proof formats are agnostic to the credential format chosen. However, the JSON credential format is commonly used with JSON Web Signatures (https://www.w3.org/TR/vc-data-model/#json-web-token). JSON-LD is commonly used with different kinds of Linked Data Proofs and JSON Web Signatures (https://www.w3.org/TR/vc-data-model/#json-ld). Applications can use all beforementioned assertion and proof formats with this specification.

This specification introduces the following representations to exchange verifiable credentials objectes between OpenID OPs and RPs.

- The JWT claim `verifiable_presentations` used as generic container to embed verifiable presentation objects into ID tokens or userinfo responses.
- The new token types "VP Token" used as generic container for verifiable presentation objects in authentication and token responses in addition to ID Tokens.

All representations share the same container format.

5. Container Format

A verifiable presentation container is an array of objects, each of them containing the following fields:

`format`: REQUIRED A JSON string denoting the proof format the presentation was returned in. This specification introduces the values `jwt_vp` and `ldp_vp` to denote credentials in JSON-LD and JWT format, respectively, as defined in https://identity.foundation/presentation-exchange/.

`presentation` : REQUIRED. A W3C Verifiable Presentation with a cryptographically verifiable proof in the defined proof format.

Note that OP would first encode VPs using the rules defined in the Verifiable Credential specification either in JWT format or JSON-LD format, before encoded VPs as container objects.

Here is an example:

```
[
   {
      "format":"vp_jwt",
      "presentation":
      "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImRpZDpleGFtcGxlOmFiZmUxM2Y3MTIxMjA0
      MzFjMjc2ZTEyZWNhYiNrZXlzLTEifQ.eyJzdWIiOiJkaWQ6ZXhhbXBsZTplYmZlYjFmNzEyZWJjNmYxY
      zI3NmUxMmVjMjEiLCJqdGkiOiJodHRwOi8vZXhhbXBsZS5lZHUvY3JlZGVudGlhbHMvMzczMiIsImlzc
      yI6Imh0dHBzOi8vZXhhbXBsZS5jb20va2V5cy9mb28uandrIiwibmJmIjoxNTQxNDkzNzI0LCJpYXQiO
      jE1NDE0OTM3MjQsImV4cCI6MTU3MzAyOTcyMywibm9uY2UiOiI2NjAhNjM0NUZTZXIiLCJ2YyI6eyJAY
      29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkZW50aWFscy92MSIsImh0dHBzOi8vd
      3d3LnczLm9yZy8yMDE4L2NyZWRlbnRpYWxzL2V4YW1wbGVzL3YxIl0sInR5cGUiOlsiVmVyaWZpYWJsZ
      UNyZWRlbnRpYWwiLCJVbml2ZXJzaXR5RGVncmVlQ3JlZGVudGlhbCJdLCJjcmVkZW50aWFsU3ViamVjd
      CI6eyJkZWdyZWUiOnsidHlwZSI6IkJhY2hlbG9yRGVncmVlIiwibmFtZSI6IjxzcGFuIGxhbmc9J2ZyL
      UNBJz5CYWNjYWxhdXLDqWF0IGVuIG11c2lxdWVzIG51bcOpcmlxdWVzPC9zcGFuPiJ9fX19.KLJo5GAy
      BND3LDTn9H7FQokEsUEi8jKwXhGvoN3JtRa51xrNDgXDb0cq1UTYB-rK4Ft9YVmR1NI_ZOF8oGc_7wAp
      8PHbF2HaWodQIoOBxxT-4WNqAxft7ET6lkH-4S6Ux3rSGAmczMohEEf8eCeN-jC8WekdPl6zKZQj0YPB
      1rx6X0-xlFBs7cl6Wt8rfBP_tZ9YgVWrQmUWypSioc0MUyiphmyEbLZagTyPlUyflGlEdqrZAv6eSe6R
      txJy6M1-lD7a5HTzanYTWBPAUHDZGyGKXdJw-W_x0IWChBzI8t3kpG253fg6V3tPgHeKXE94fz_QpYfg
      --7kLsyBAfQGbg"
   },
   {
      "format":"vp_ldp",
      "presentation":{
         "@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"
               ],
               "id":"https://example.com/credentials/1872",
               "type":[
                  "VerifiableCredential",
                  "IDCardCredential"
               ],
               "issuer":{
                  "id":"did:example:issuer"
               },
               "issuanceDate":"2010-01-01T19:23:24Z",
               "credentialSubject":{
                  "given_name":"Fredrik",
                  "family_name":"Str&#246;mberg",
                  "birthdate":"1949-01-22"
               },
               "proof":{
                  "type":"Ed25519Signature2018",
                  "created":"2021-03-19T15:30:15Z",

"jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..PT8yCqVjj5ZHD0W36zsBQ4
7oc3El07WGPWaLUuBTOT48IgKI5HDoiFUt9idChT_Zh5s8cF_2cSRWELuD8JQdBw",
                  "proofPurpose":"assertionMethod",
                  "verificationMethod":"did:example:issuer#keys-1"
               }
            }
         ],
         "id":"ebc6f1c2",
         "holder":"did:example:holder",
         "proof":{
            "type":"Ed25519Signature2018",
```

```
            "created":"2021-03-19T15:30:15Z",
            "challenge":"()&)()0__sdf",

"jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..GF5Z6TamgNE8QjE3RbiDOj
3n_t25_1K7NVWMUASe_OEzQV63GaKdu235MCS3hIYvepcNdQ_ZOKpGNCf0vIAoDA",
            "proofPurpose":"authentication",
            "verificationMethod":"did:example:holder#key-1"
         }
      }
   }
]
```

6. JWT parameters extention

Verifiable credential objects can be exchanged between OP and RP enveloped in JWT claims in ID tokens or userinfo responses.

This specification introduces the following JWT claim for that purpose:

- `verifiable_presentations`: A claim whose value is a verifiable presentations container object as defined above.

This claim can be added to ID Tokens, Userinfo responses as well as Access Tokens and Introspection response. It MAY also be included as aggregated or distributed claims (see Section 5.6.2 of the OpenID Connect specification [OpenID]).

Note that above claim has to be distinguished from `vp` or `vc` claims as defined in JWT proof format. `vp` or `vc` claims contain those parts of the standard verifiable credentials and verifiable presentations where no explicit encoding rules for JWT exist. They are used as part of a verifiable credential or presentation in JWT format. They are not meant to include complete verifiable credentials or verifiable presentations objects which is the purpose of the claims defined in this specification.

7. New Tokens extention

This specifications introduces the following new token:

- VP Token: a token containing a verifiable presentations container as defined above. Such a token is provided to the RP in addition to an `id_token` in the `vp_token` parameter.

`vp_token` is provided in the same response as the `id_token`. Depending on the response type, this can be either the authentication response or the token response. Authentication event information is conveyed via the id token while it's up to the RP to determine what (additional) claims are allocated to `id_token` and `vp_token`, respectively, via the `claims` parameter.

If the `vp_token` is returned in the frontchannel, a hash of the respective token MUST be included in `id_token`.

`vp_hash` OPTIONAL. Hash value of `vp_token` that represents the W3C VP. Its value is the base64url encoding of the left-most half of the hash of the octets of the ASCII representation of the `vp_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 vp_token value with SHA-256, then take the left-most 128 bits and base64url encode them. The `vp_hash` value is a case sensitive string.

8. Requesting Verifiable Presentations

This section illustrates how the `claims` parameter can be used for requesting verified presentations. It serves as a starting point to drive discussion about this aspect. There are other candidate approaches for this purpose (most notably DIF Presentation Exchange. They will be evaluated as this draft evolves.

8.1. Embedded Verifiable Presentations

A Verifiable Presentation embedded in an ID Token (or userinfo response) is requested by adding an element `verifiable_presentations` to the `id_token` (or `userinfo`) top level element of the `claims` parameter. This element must contain the following element:

`credential_types` Object array containing definitions of credential types the RP wants to obtain along with an (optional) definition of the claims from the respective credential type the RP is requesting. Each of those object has the following fields:

- `type` REQUIRED String denoting a credential type
- `claims` OPTIONAL An object determining the claims the RP wants to obtain using the same notation as used underneath `id_token`.
- `format` OPTION String designating the VP format. Predefined values are `vp_ldp` and `vp_jwt`.

Here is a non-normative example:

```
{
   "id_token":{
      "acr":null,
      "verifiable_presentations":{
         "credential_types":[
            {

"type":"https://www.w3.org/2018/credentials/examples/v1/IDCardCredential",
               "claims":{
                  "given_name":null,
                  "family_name":null,
                  "birthdate":null
               }
            }
         ]
      }
   }
}
```

8.1.1. VP Token

A VP Token is requested by adding a new top level element `vp_token` to the `claims` parameter. This element contains the sub elements as defined above.

9. Examples

This section illustrates examples when W3C Verifiable Credentials objects are requested using `claims` parameter and returned inside ID Tokens.

9.1. Self-Issued OpenID Provider with Verifiable Presentation in ID Token

Below are the examples when W3C Verifiable Credentials are requested and returned inside ID Token as part of Self-Issued OP response. ID Token contains a `verifiable_presentations` claim with the Verifiable Presentation data. It can also contain `verifiable_credentials` element with the Verifiable Credential data.

9.1.1. Authentication request

The following is a non-normative example of how an RP would use the `claims` parameter to request the `verifiable_presentations` claim in the `id_token`:

```
  HTTP/1.1 302 Found
  Location: openid://?
    response_type=id_token
    &client_id=https%3A%2F%2Fclient.example.org%2Fcb
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &scope=openid

&claims=claims=%7B%22id_token%22%3A%7B%22vc%22%3A%7B%22types%22%3A%5B%22https%3A%2F%
     2Fdid.itsourweb.org%3A3000%2Fsmart-credential%2FOntario-Health-Insurance-Plan
     %22%5D%7D%7D%7D
    &state=af0ifjsldkj
    &nonce=960848874
    &registration_uri=https%3A%2F%2F
      client.example.org%2Frf.txt%22%7D
```

9.1.1.1. `claims` parameter

Below is a non-normative example of how the `claims` parameter can be used for requesting verified presentations signed as a JWT.

```
{
   "id_token":{
      "acr":null,
      "verifiable_presentations":{
         "credential_types":[
            {
               "type":"https://did.itsourweb.org:3000/smartcredential/Ontario-Health-
Insurance-Plan"
            }
         ]
      }
   }
}
```

9.1.2. Authentication Response

Below is a non-normative example of ID Token that includes `verifiable_presentations` claim.

```
{
  "kid": "did:ion:EiC6Y9_aDaCsITlY06HId4seJjJ...b1df31ec42d0",
  "typ": "JWT",
  "alg": "ES256K"
}.{
   "iss":"https://self-issued.me",
   "aud":"https://book.itsourweb.org:3000/client_api/authresp/uhn",
   "iat":1615910538,
   "exp":1615911138,
   "sub":"did:ion:EiC6Y9_aDaCsITlY06HId4seJjJ-9...mS3NBIn19",
   "auth_time":1615910535,
   "nonce":"960848874",
   "verifiable_presentations":[
      {
         "format":"vp_jwt",
         "presentation":"ewogICAgImlzcyI6Imh0dHBzOi8vYm9vay5pdHNvdXJ3ZWIub...IH0="
      }
   ],
   "sub_jwk":{
      "crv":"P-384",
      "kty":"EC",
      "kid": "c7298a61a6904426a580b1df31ec42d0",
      "x":"jf3a6dquclZ4PJ0JMU8RuucG9T1O3hpU_S_79sHQi7VZBD9e2VKXPts9lUjaytBm",
      "y":"38VlVE3kNiMEjklFe4Wo4DqdTKkFbK6QrmZf77lCMN2x9bENZoGF2EYFiBsOsnq0"
   }
}
```

Below is a non-normative example of a decoded Verifiable Presentation object that was included in `verifiable_presentations`. Note that `vp` is used to contain only "those parts of the standard verifiable presentation where no explicit encoding rules for JWT exist" [VC-DATA-MODEL]

```
  {
    "iss":"did:ion:EiC6Y9_aDaCsITlY06HId4seJjJ...b1df31ec42d0",
    "aud":"https://book.itsourweb.org:3000/ohip",
    "iat":1615910538,
    "exp":1615911138,
    "nbf":1615910538,
    "nonce":"acIlfiR6AKqGHg",
    "vp":{
        "@context":[
          "https://www.w3.org/2018/credentials/v1",
          "https://ohip.ontario.ca/v1"
        ],
        "type":[
          "VerifiablePresentation"
        ],
        "verifiableCredential":[

"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6InVybjp1dWlkOjU0ZDk2NjE2LTE1MWUt...OLryT1g
"
        ]
    }
  }
```

9.2. Self-Issued OpenID Provider with Verifiable Presentation in ID Token (selective disclosure)

9.2.1. `claims` parameter

Below is a non-normative example of how the `claims` parameter can be used for requesting verified presentations signed as Linked Data Proofs.

```
{
   "id_token":{
      "verifiable_presentations":[
         {
            "credential_types":[
               "https://www.w3.org/2018/credentials/examples/v1/IDCardCredential"
            ],
            "claims":{
               "given_name":null,
               "family_name":null,
               "birthdate":null
            }
         }
      ]
   }
}
```

9.2.2. Authentication Response

Below is a non-normative example of ID Token that includes `verifiable_presentations` claim.

```
{
   "iss":"https://self-issued.me",
   "aud":"https://book.itsourweb.org:3000/client_api/authresp/uhn",
   "iat":1615910538,
   "exp":1615911138,
   "sub":"did:ion:EiC6Y9_aDaCsITlY06HId4seJjJ...b1df31ec42d0",
   "auth_time":1615910535,
   "verifiable_presentations":[
      {
         "format":"vp_jwt",
         "presentation":{
            "@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"
                  ],
                  "id":"https://example.com/credentials/1872",
                  "type":[
                     "VerifiableCredential",
                     "IDCardCredential"
                  ],
                  "issuer":{
                     "id":"did:example:issuer"
                  },
                  "issuanceDate":"2010-01-01T19:23:24Z",
                  "credentialSubject":{
                     "given_name":"Fredrik",
                     "family_name":"Str&#246;mberg",
                     "birthdate":"1949-01-22"
                  },
                  "proof":{
                     "type":"Ed25519Signature2018",
                     "created":"2021-03-19T15:30:15Z",

"jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..PT8yCqVjj5ZHD0W36zsBQ4
7oc3El07WGPWaLUuBTOT48IgKI5HDoiFUt9idChT_Zh5s8cF_2cSRWELuD8JQdBw",
                     "proofPurpose":"assertionMethod",
                     "verificationMethod":"did:example:issuer#keys-1"
                  }
               }
            ],
            "id":"ebc6f1c2",
            "holder":"did:example:holder",
            "proof":{
               "type":"Ed25519Signature2018",
               "created":"2021-03-19T15:30:15Z",
               "challenge":"()&)()0__sdf",

"jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..GF5Z6TamgNE8QjE3RbiDOj
3n_t25_1K7NVWMUASe_OEzQV63GaKdu235MCS3hIYvepcNdQ_ZOKpGNCf0vIAoDA",
               "proofPurpose":"authentication",
               "verificationMethod":"did:example:holder#key-1"
            }
         }
      }
   ],
   "nonce":"960848874",
```

```
    "sub_jwk":{
       "crv":"P-384",
       "kty":"EC",
       "x":"jf3a6dquclZ4PJ0JMU8RuucG9T1O3hpU_S_79sHQi7VZBD9e2VKXPts9lUjaytBm",
       "y":"38VlVE3kNiMEjklFe4Wo4DqdTKkFbK6QrmZf77lCMN2x9bENZoGF2EYFiBsOsnq0"
    }
}
```

9.3. Authorization Code Flow with Verifiable Presentation in ID Token

Below are the examples when W3C Verifiable Credentials are requested and returned inside ID Token as part of Authorization Code flow. ID Token contains a `verifiable_presentations` element with the Verifiable Presentations data.

9.3.1. Authentication Request

```
  GET /authorize?
    response_type=code
    &client_id=s6BhdRkqt3
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &scope=openid
    &claims=...
    &state=af0ifjsldkj
    &nonce=n-0S6_WzA2Mj HTTP/1.1
  Host: server.example.com
```

9.3.1.1. Claims parameter

Below is a non-normative example of how the `claims` parameter can be used for requesting verified presentations signed as JWT.

```
{
   "id_token":{
      "acr":null,
      "verifiable_presentations":{
         "credential_types":[
            {
               "type":"https://did.itsourweb.org:3000/smartcredential/Ontario-Health-
Insurance-Plan"
            }
         ]
      }
   }
}
```

9.3.2. Authentication Response

```
HTTP/1.1 302 Found
  Location: https://client.example.org/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj
```

9.3.3. Token Request

```
  POST /token HTTP/1.1
  Host: server.example.com
  Content-Type: application/x-www-form-urlencoded
  Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

  grant_type=authorization_code
  &code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

9.4. Authorization Code Flow with Verifiable Presentation returned from the UserInfo endpoint

Below are the examples when verifiable presentation is requested and returned from the UserInfo endpoint as part of OpenID Connect Authorization Code Flow. UserInfo response contains a `verifiable_presentations` element with the Verifiable Presentation data.

9.4.1. Authentication Request

```
  GET /authorize?
    response_type=code
    &client_id=s6BhdRkqt3
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &scope=openid
    &claims=...
    &state=af0ifjsldkj
    &nonce=n-0S6_WzA2Mj HTTP/1.1
  Host: server.example.com
```

9.4.1.1. Claims parameter

Below is a non-normative example of how the `claims` parameter can be used for requesting verified presentations signed as JWT.

```
{
   "userinfo":{
      "verifiable_presentations":{
         "credential_types":[
            {
               "type":"https://did.itsourweb.org:3000/smartcredential/Ontario-Health-
Insurance-Plan"
            }
         ]
      }
   },
   "id_token":{
      "auth_time":{
         "essential":true
      }
   }
}
```

9.4.2. Authentication Response

```
HTTP/1.1 302 Found
  Location: https://client.example.org/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj
```

9.4.3. Token Request

```
  POST /token HTTP/1.1
  Host: server.example.com
  Content-Type: application/x-www-form-urlencoded
  Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

  grant_type=authorization_code
  &code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

9.4.4. Token Response

9.4.4.1. id_token

```
{
  "iss": "http://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "auth_time": 1615910535
}
```

9.4.5. UserInfo Response

Below is a non-normative example of a UserInfo Response that includes a `verifiable_presentations` claim:

```
  HTTP/1.1 200 OK
  Content-Type: application/json

  {
   "sub": "248289761001",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
    "verifiable_presentations":[
      {
         "format":"vp_jwt",
         "presentation":"ewogICAgImlzcyI6Imh0dHBzOi8vYm9vay5pdHNvdXJ3ZWIub...IH0="
      }
   ],
  }
```

JWT inside the `verifiable_presentations` claim when decoded equals to a verifiable presentation in Self-Issued OP with Verifiable Presentation in ID Token, Authentication Response section.

9.5. Authorization Code Flow with Verifiable Presentation returned from the UserInfo endpoint (LDP)

9.5.1. Claims parameter

Below is a non-normative example of how the `claims` parameter can be used for requesting verified presentations signed as Linked Data Proofs.

```
{
   "userinfo":{
      "verifiable_presentations":{
         "credential_types":[
            {

"type":"https://www.w3.org/2018/credentials/examples/v1/IDCardCredential",
               "claims":{
                  "given_name":null,
                  "family_name":null,
                  "birthdate":null
               }
            }
         ]
      }
   },
   "id_token":{
      "auth_time":{
         "essential":true
      }
   }
}
```

9.5.2. Token Response

9.5.2.1. id_token

```
{
  "iss": "http://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "auth_time": 1615910535
}
```

9.5.3. UserInfo Response

Below is a non-normative example of a UserInfo Response that includes `verifiable_presentations` claim:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
 "sub":"248289761001",
 "name":"Jane Doe",
 "given_name":"Jane",
 "family_name":"Doe",
 "verifiable_presentations":[
    {
       "format":"vp_jwt",
       "presentation":{
          "@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"
                ],
                "id":"https://example.com/credentials/1872",
                "type":[
                   "VerifiableCredential",
                   "IDCardCredential"
                ],
                "issuer":{
                   "id":"did:example:issuer"
                },
                "issuanceDate":"2010-01-01T19:23:24Z",
                "credentialSubject":{
                   "given_name":"Fredrik",
                   "family_name":"Str&#246;mberg",
                   "birthdate":"1949-01-22"
                },
                "proof":{
                   "type":"Ed25519Signature2018",
                   "created":"2021-03-19T15:30:15Z",

"jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..PT8yCqVjj5ZHD0W36zsBQ4
7oc3El07WGPWaLUuBTOT48IgKI5HDoiFUt9idChT_Zh5s8cF_2cSRWELuD8JQdBw",
                   "proofPurpose":"assertionMethod",
                   "verificationMethod":"did:example:issuer#keys-1"
                }
             }
          ],
          "id":"ebc6f1c2",
          "holder":"did:example:holder",
          "proof":{
             "type":"Ed25519Signature2018",
             "created":"2021-03-19T15:30:15Z",
             "challenge":"()&)()0__sdf",

"jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..GF5Z6TamgNE8QjE3RbiDOj
3n_t25_1K7NVWMUASe_OEzQV63GaKdu235MCS3hIYvepcNdQ_ZOKpGNCf0vIAoDA",
             "proofPurpose":"authentication",
             "verificationMethod":"did:example:holder#key-1"
          }
       }
    }
```

```
    ]
}
```

9.6. SIOP with vp_token

This section illustrates the protocol flow for the case of communication through the front channel only (like in SIOP).

9.6.1. Authentication request

The following is a non-normative example of how an RP would use the `claims` parameter to request claims in the `vp_token`:

```
  HTTP/1.1 302 Found
  Location: openid://?
    response_type=id_token
    &client_id=https%3A%2F%2Fclient.example.org%2Fcb
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &scope=openid
    &claims=...
    &state=af0ifjsldkj
    &nonce=n-0S6_WzA2Mj
    &registration_uri=https%3A%2F%2F
      client.example.org%2Frf.txt%22%7D
```

9.6.1.1. claims parameter

```
{
   "vp_token":{
      "credential_types":[
         {
            "type":"https://www.w3.org/2018/credentials/examples/v1/IDCardCredential",
            "claims":{
               "given_name":null,
               "family_name":null,
               "birthdate":null
            }
         }
      ]
   }
}
```

9.6.2. Authentication Response (including vp_token)

The successful authentication response contains a `vp_token` parameter along with `id_token` and `state`.

```
  HTTP/1.1 302 Found
  Location: https://client.example.org/cb#
    id_token=eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso
    &vp_token=...
    &state=af0ifjsldkj
```

9.6.2.1. id_token

This example shows an ID Token containing a `vp_hash`:

```
{
   "iss":"https://book.itsourweb.org:3000/wallet/wallet.html",
   "aud":"https://book.itsourweb.org:3000/client_api/authresp/uhn",
   "iat":1615910538,
   "exp":1615911138,
   "sub":"urn:uuid:68f874e2-377c-437f-a447-b304967ca351",
   "auth_time":1615910535,
   "vp_hash":"77QmUPtjPfzWtF2AnpK9RQ",
   "nonce":"960848874",
   "sub_jwk":{
      "crv":"P-384",
      "ext":true,
      "key_ops":[
         "verify"
      ],
      "kty":"EC",
      "x":"jf3a6dquclZ4PJ0JMU8RuucG9T1O3hpU_S_79sHQi7VZBD9e2VKXPts9lUjaytBm",
      "y":"38VlVE3kNiMEjklFe4Wo4DqdTKkFbK6QrmZf77lCMN2x9bENZoGF2EYFiBsOsnq0"
   }
}
```


9.6.2.2. vp_token content

```
[
   {
      "format":"vp_ldp",
      "presentation":{
         "@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"
               ],
               "id":"https://example.com/credentials/1872",
               "type":[
                  "VerifiableCredential",
                  "IDCardCredential"
               ],
               "issuer":{
                  "id":"did:example:issuer"
               },
               "issuanceDate":"2010-01-01T19:23:24Z",
               "credentialSubject":{
                  "given_name":"Fredrik",
                  "family_name":"Str&#246;mberg",
                  "birthdate":"1949-01-22"
               },
               "proof":{
                  "type":"Ed25519Signature2018",
                  "created":"2021-03-19T15:30:15Z",

"jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..PT8yCqVjj5ZHD0W36zsBQ4
7oc3El07WGPWaLUuBTOT48IgKI5HDoiFUt9idChT_Zh5s8cF_2cSRWELuD8JQdBw",
                  "proofPurpose":"assertionMethod",
                  "verificationMethod":"did:example:issuer#keys-1"
               }
            }
         ],
         "id":"ebc6f1c2",
         "holder":"did:example:holder",
         "proof":{
            "type":"Ed25519Signature2018",
            "created":"2021-03-19T15:30:15Z",
            "challenge":"()&)()0__sdf",

"jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..GF5Z6TamgNE8QjE3RbiDOj
3n_t25_1K7NVWMUASe_OEzQV63GaKdu235MCS3hIYvepcNdQ_ZOKpGNCf0vIAoDA",
            "proofPurpose":"authentication",
            "verificationMethod":"did:example:holder#key-1"
         }
      }
   }
]
```


9.7. Authorization Code Flow with vp_token

This section illustrates the protocol flow for the case of communication using frontchannel and backchannel (utilizing the authorization code flow).

9.7.1. Authentication Request

```
  GET /authorize?
    response_type=code
    &client_id=s6BhdRkqt3
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &scope=openid
    &claims=...
    &state=af0ifjsldkj
    &nonce=n-0S6_WzA2Mj HTTP/1.1
  Host: server.example.com
```

9.7.1.1. Claims parameter

```
{
   "vp_token":{
      "credential_types":[
         {
            "type":"https://www.w3.org/2018/credentials/examples/v1/IDCardCredential",
            "claims":{
               "given_name":null,
               "family_name":null,
               "birthdate":null
            }
         }
      ]
   }
}
```

9.7.2. Authentication Response

```
HTTP/1.1 302 Found
  Location: https://client.example.org/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj
```

9.7.3. Token Request

```
  POST /token HTTP/1.1
  Host: server.example.com
  Content-Type: application/x-www-form-urlencoded
  Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

  grant_type=authorization_code
  &code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

9.7.4. Token Response (including vp_token)

```
{
   "access_token":"SlAV32hkKG",
   "token_type":"Bearer",
   "refresh_token":"8xLOxBtZp8",
   "expires_in":3600,
   "id_token":"eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso",
   "vp_token":[
      {
         "format":"vp_ldp",
         "presentation":{
            "@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"
                  ],
                  "id":"https://example.com/credentials/1872",
                  "type":[
                     "VerifiableCredential",
                     "IDCardCredential"
                  ],
                  "issuer":{
                     "id":"did:example:issuer"
                  },
                  "issuanceDate":"2010-01-01T19:23:24Z",
                  "credentialSubject":{
                     "given_name":"Fredrik",
                     "family_name":"Str&#246;mberg",
                     "birthdate":"1949-01-22"
                  },
                  "proof":{
                     "type":"Ed25519Signature2018",
                     "created":"2021-03-19T15:30:15Z",

"jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..PT8yCqVjj5ZHD0W36zsBQ4
7oc3El07WGPWaLUuBTOT48IgKI5HDoiFUt9idChT_Zh5s8cF_2cSRWELuD8JQdBw",
                     "proofPurpose":"assertionMethod",
                     "verificationMethod":"did:example:issuer#keys-1"
                  }
               }
            ],
            "id":"ebc6f1c2",
            "holder":"did:example:holder",
            "proof":{
               "type":"Ed25519Signature2018",
               "created":"2021-03-19T15:30:15Z",
               "challenge":"()&)()0__sdf",

"jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..GF5Z6TamgNE8QjE3RbiDOj
3n_t25_1K7NVWMUASe_OEzQV63GaKdu235MCS3hIYvepcNdQ_ZOKpGNCf0vIAoDA",
               "proofPurpose":"authentication",
               "verificationMethod":"did:example:holder#key-1"
            }
         }
      }
   ]
}
```

9.7.4.1. id_token

```
{
  "iss": "http://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "vp_hash": "77QmUPtjPfzWtF2AnpK9RQ"
}
```

Appendix A. IANA Considerations

TBD

Appendix B. Acknowledgements

TBD

Appendix C. Notices

TBD

Appendix D. Document History

[[To be removed from the final specification]]

-00

- initial revision

Appendix E. Related Issues

- https://bitbucket.org/openid/connect/issues/1206/how-to-support-ld-proofs-in-verifiable#comment-60051830

Authors' Addresses

Oliver Terbu
ConsenSys Mesh
Email: oliver.terbu@mesh.xyz

Torsten Lodderstedt
yes.com
Email: torsten@lodderstedt.net

Kristina Yasuda
Microsoft
Email: kristina.yasuda@microsoft.com

Adam Lemmon
Convergence.tech
Email: adam@convergence.tech

Tobias Looker
Mattr
Email: tobias.looker@mattr.global