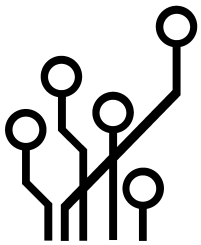


PROJECT ORIGIN

projectorigin2023@gmail.com

Ricerca OpenID

Versione	0.1.0
Responsabile	Ibra Elton
Redattori	Bobirica Andrei Cristian
Verificatori	Lotto Riccardo
Uso	Interno
Destinatari	*Project Origin* Prof. Vardanega Tullio Prof. Cardin Riccardo

Descrizione
Documento in cui vengono approfonditi il funzionamento e l'utilizzo dello standard The OpenID for Verifiable Credentials

Registro delle modifiche

Vers.	Data	Autore	Ruolo	Descrizione
0.1.0	2023-05-21	Lotto Riccardo	Verificatore	Verifica documento
0.0.1	2023-05-16	Bobirica Andrei	Analista	Redazione documento

Indice

1 Informazioni generali **3**
1.1 Terminologia . . . 3
1.2 OpenID for Verifiable Credential Issuance (OID4VCI) . . . 3
1.2.1 Credential Issuer . . . 3
1.2.2 Authorization Code Flow . . . 4
1.2.3 Credential Offer . . . 4
1.2.4 Authorization Endpoint . . . 5
1.2.5 Token Endpoint . . . 5
1.2.6 Credential Endpoint . . . 5
1.2.7 Credential Issuer metadata . . . 6
1.2.8 Sicurezza . . . 7
1.3 OpenID for Verifiable Presentations (OID4VP) . . . 7
1.3.1 Schema di Presentazione di una credenziale Same Device Flow . . . 7
1.3.2 Schema di presentazione di una credenziale Cross Device Flow . . . 8
1.3.3 Authorization Request . . . 8
1.3.4 Response . . . 9
1.3.5 Response Mode direct_post . . . 10
1.3.6 Firma e Crittografia della Response . . . 10
1.3.7 VP Token Validation . . . 10
1.3.8 Wallet Metadata . . . 10
1.4 Libraries . . . 10
1.4.1 Waltid ssikit . . . 10
1.4.2 SpruceID oidc4vci-rs . . . 11
1.4.3 Spheron . . . 11

2 Riferimenti **11**

1 Informazioni generali

1.1 Terminologia

- **Verifiable Credential (VC)**: Definizione che si collega alla definizione di W3C Verifiable Credential Data Model; si definisce VC una credenziale che può essere verificata attraverso procedure crittografiche.
- **End-User**: Utente (Holder).
- **Wallet**: Entità che riceve, archivia, presenta e organizza credenziali dell'Utente.
- **Verifier**: Entità che verifica e consuma la credenziale di un utente per fornirgli un servizio.
- **Credential Issuer**: Entità che fornisce una credenziale all'utente.

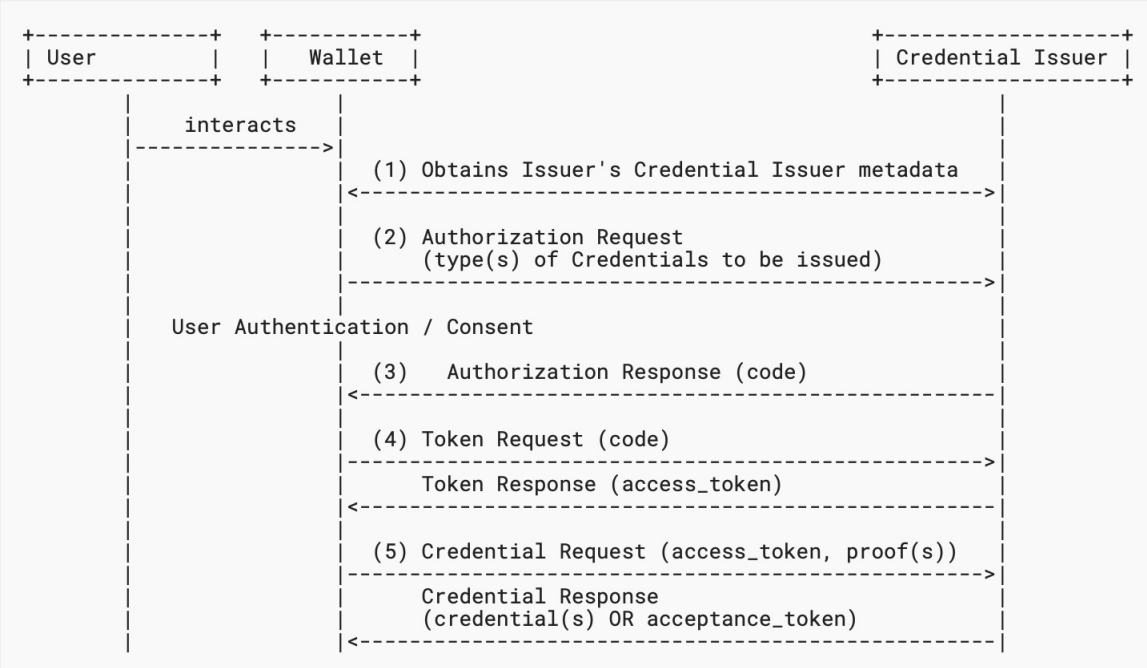
1.2 OpenID for Verifiable Credential Issuance (OID4VCI)

OpenID per fornitura di credenziali verificabili (OID4VCI) è uno standard che definisce una API usata per fornire credenziali verificabili (Verifiable Credentials).
Le Verifiable Credentials seguono uno schema ben definito e sono collegate ad un certo utente (Holder) attraverso un collegamento crittografico (Binding). Questo assicura la possibilità di presentare la credenziale ad un Verifier senza chiamare l'Issuer per ulteriori verifiche.
L'accesso a questa API è autorizzata attraverso la OAuth 2.0 che il Wallet usa per autorizzarsi e ricevere Verifiable Credentials. L'utilizzo della OAuth 2.0 garantisce sicurezza, semplicità e flessibilità.

1.2.1 Credential Issuer

API legate alla fornitura delle credenziali da parte del Credential Issuer.
Un issuer ha i seguenti endpoint, categorizzabili come interfacce, che offrono diversi servizi.
Un endpoint da cui una credenziale può essere fornita.
Un endpoint opzionale da cui si possono fornire più credenziali insieme.
Un meccanismo di metadata che offre al Issuer la possibilità di dichiarare in partenza quali credenziali può offrire, è un concetto simile a quello visto nello SPID.
I certificati hanno la possibilità opzionale di essere abbinati ad una chiave crittografica per garantire anche un Proof Of Posessione.
Un Credential Isuer utilizza il sistema OAuth 2.0 per autorizzare l'accesso alle proprio piattaforme e richiedere le credenziali.

1.2.2 Authorization Code Flow



1.2.3 Credential Offer

L'Issuer invia una Offerta di credenziale attraverso il protocollo HTTP GET oppure attraverso HTTP redirect, e lo invia al Credential offer Endpoint. L'offerta di una credenziale è un oggetto JSON identificato da un URI. L'Issuer può anche fornire un Codice QR che che può essere scannerizzato con il Wallet dell'Holder. I parametri nel Credential Offer sono credential_issuer, credentials, credentials_supported. Un esempio può essere:

```
{
   "credential_issuer": "https://credential-issuer.example.com",
   "credentials": [
      "UniversityDegree_JWT",
      {
         "format": "mso_mdoc",
         "doctype": "org.iso.18013.5.1.mDL"
      }
   ],
   "grants": {
      "authorization_code": {
         "issuer_state": "eyJhbGciOiJSU0Et...FYUaBy"
      },
      "urn:ietf:params:oauth:grant-type:pre-authorized_code": {
         "pre-authorized_code": "adhjhdjajkdkhjhdj",
         "user_pin_required": true
      }
   }
}
```

1.2.4 Authorization Endpoint

Usando una Authorization Request si può ottenere i permessi per accedere al Credential Endpoint. Una possibilità per ottenere i permessi è fornire un authorization_details fornendo nel dettaglio il tipo di credenziale che si desidera di tipo openid_credential.
Un esempio può essere :

```
[
   {
      "type": "openid_credential",
      "locations": [
         "https://credential-issuer.example.com"
      ],
      "format": "jwt_vc_json",
      "credential_definition": {
         "type": [
            "VerifiableCredential",
            "UniversityDegreeCredential"
         ]
      }
   }
]
```

Se l'esito della richiesta avrà esito positivo si ricevera un pachetto HTTP contenente i dettagli sulla richiesta effettuata insieme ad un Authorization Code necessario per i passaggi successivi.

1.2.5 Token Endpoint

Usando l'Authorizaiton Code in scambio attraverso il Token Endpoint si potrà ottenere un Access Token. Di seguito un esempio di una richiesta HTTP da fare per ottenere l'Access Token.

```
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
&code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
&redirect_uri=https%3A%2F%2FWallet.example.org%2Fcb
```

1.2.6 Credential Endpoint

Una volta ottenuto l'Access Token si può fare richiesta al Credential Endpoint per ottenere la credenziale desiderata. Si può richiedere la stessa credenziali molteplici volte per associarla a diversi DIDs (Decentralized Identifiers).
Se l'Access Token permette il reperimento di credenziali multiple è discrezione implementativa quali e in che ordine richiederle. La credenziale necessita di essere abbinata crittograficamente al Enduser Identifier. Questo abbinamento permette al Verifier di essere sicuro che chi presenta la credenziale è la stessa persona che ne ha fatto richiesta.
Un esempio di una richiesta di credenziale può essere la seguente:

```
POST /credential HTTP/1.1
Host: server.example.com
Content-Type: application/json
Authorization: BEARER czZCaGRSa3F0MzpnWDFmQmF0M2JW
```

```
5
6  {
7     "format": "jwt_vc_json",
8     "type": [
9        "VerifiableCredential",
10       "UniversityDegreeCredential"
11    ],
12    "proof": {
13       "proof_type": "jwt",
14       "jwt": "eyJraWQiOiJkaWQ6ZXhhbXBsZTplYmZlYjFmNzEyZWJjNmYxYzI3
             NmUxMmVjMjEva2V5cy8
15       xIiwiYWxnIjoiRVMyNTYiLCJ0eXAiOiJKV1QifQ.eyJpc3
             MiOiJzNkJoZFJrcXQzIiwiYXVkIjoiaHR
16       0cHM6Ly9zZXJ2ZXIuZXhhbXBsZS5jb20iLCJpYXQiOiIyMDE4LTA5LTE0VDIxOjE5
             OjEwWiIsIm5vbm
17       NlIjoidFppZ25zbkZicCJ9.ewdkIkPV50iOeBUqMXCC_aZKPxgihac0aW9EkL1nOzM
             "
18    }
19 }
```

Una volta che il Client invia la richiesta per la credenziale l'Issuer può inviarla istantaneamente oppure per diversi motivi fornirla in un secondo momento, in questo ultimo caso fornirà sul momento un acceptance_token che il Client userà successivamente per accettare la credenziale.

1.2.7 Credential Issuer metadata

Di seguito un esempio di un metadata di un issuer, dal quale si può capire il tipo di credenziali che fornisce e dettagli tecnici per averla.

```
1  {
2      "format": "jwt_vc_json",
3      "id": "UniversityDegree_JWT",
4      "type": [
5          "VerifiableCredential",
6          "UniversityDegreeCredential"
7      ],
8      "cryptographic_binding_methods_supported": [
9          "did:example"
10     ],
11     "cryptographic_suites_supported": [
12         "ES256K"
13     ],
14     "display": [
15         {
16             "name": "University Credential",
17             "locale": "en-US",
18             "logo": {
19                 "url": "https://exampleuniversity.com/public/logo.png",
20                 "alt_text": "a square logo of a university"
21             },
22             "background_color": "#12107c",
23             "text_color": "#FFFFFF"
24         }
25     ],
26     "credentialSubject": {
```

```
        "given_name": {
            "display": [
                {
                    "name": "Given Name",
                    "locale": "en-US"
                }
            ]
        },
        "last_name": {
            "display": [
                {
                    "name": "Surname",
                    "locale": "en-US"
                }
            ]
        },
        "degree": {},
        "gpa": {
            "display": [
                {
                    "name": "GPA"
                }
            ]
        }
    }
}
```

1.2.8 Sicurezza

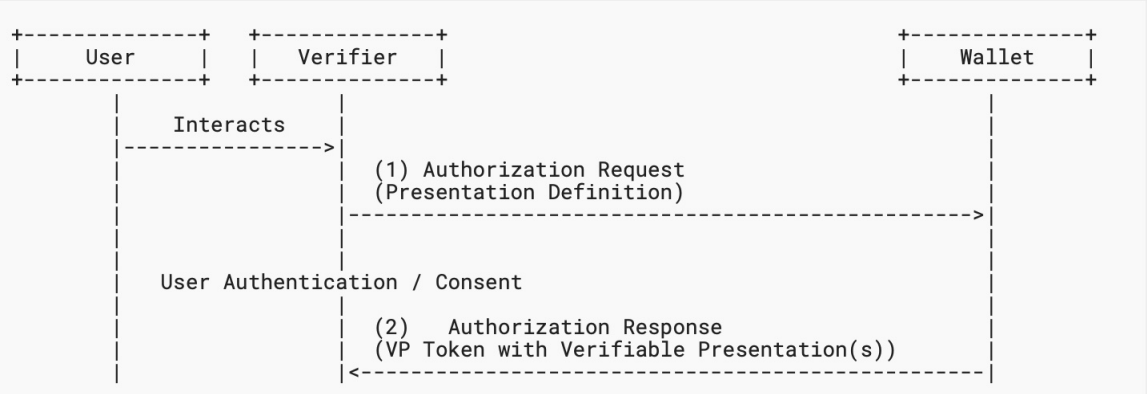
Per garantire una maggiore sicurezza l'Issuer ha bisogno di sapere a quale Wallet fornisce le credenziali. In particolare ha bisogno della certezza che il sudetto wallet sia sotto una certa regolamentazione statale o commerciale e che abbia parametri di sicurezza in grado di proteggere le sue chiavi private. In questa maniera l'Issuer non fornisce le credenziali a qualcuno che non ne è il propietario.
Un'altra considerazione utile è che le comunicazioni dovrebebro avvenire sotto il protoccolo TLS.

1.3 OpenID for Verifiable Presentations (OID4VP)

Questo standard definisce i meccanismi, basati sul OAuth 2.0 che permettono la presentazione di credenziali verificate ad un Verifier.

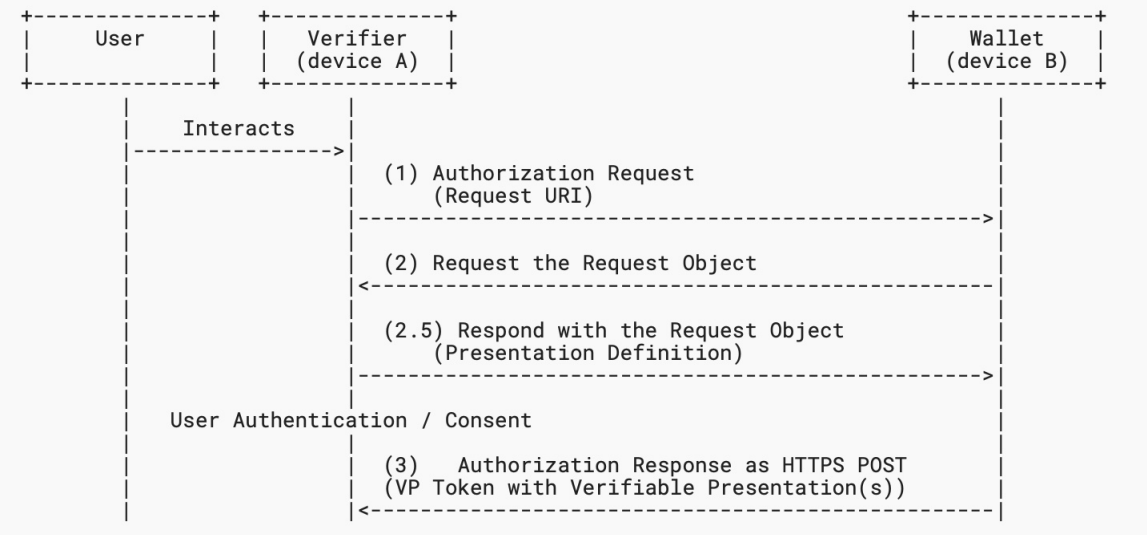
1.3.1 Schema di Presentazione di una credenziale Same Device Flow

Di seguito uno schema di interazione tra l'Holder e il Verifier, in cui interagiscono il Wallet e la parte che richiede una certificazione del Verifier, entrambe sullo stesso Device.



1.3.2 Schema di presentazione di una credenziale Cross Device Flow

Di seguito uno schema di interazione tra l'Holder con il proprio Wallet sul proprio device e il Verifer su un'altro device.



1.3.3 Authorization Request

Un Verifier attraverso un presentation_definition definisce la presentazione che richiede.
Di seguito un esempio:

```
{
    "id": "vp token example",
    "input_descriptors": [
        {
            "id": "id card credential",
            "format": {
                "ldp_vc": {
                    "proof_type": [
                        "Ed25519Signature2018"
                    ]
                }
            },
```

```
            "constraints": {
                "fields": [
                    {
                        "path": [
                            "$.type"
                        ],
                        "filter": {
                            "type": "string",
                            "pattern": "IDCardCredential"
                        }
                    }
                ]
            }
        }
    ]
}
```

Durante la richiesta di Autorizzazione se è completata con successo il Verifier fornisce un VP Token al Holder.

1.3.4 Response

Successivamente alla Authorization Request è fornito da parte del Verifier un VP Token. Può essere ritornato al Authorization Response.
Di seguito un esempio di un VP Token Response:

```
{
    "@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": "Stromberg",
                "birthdate": "1949-01-22"
            },
            "proof": {
```

```
                "type": "Ed25519Signature2018",
                "created": "2021-03-19T15:30:15Z",
                "jws": "eyJhb...JQdBw",
                "proofPurpose": "assertionMethod",
                "verificationMethod": "did:example:issuer#keys-1"
            }
        }
    ],
    "id": "ebc6f1c2",
    "holder": "did:example:holder",
    "proof": {
        "type": "Ed25519Signature2018",
        "created": "2021-03-19T15:30:15Z",
        "challenge": "n-0S6_WzA2Mj",
        "domain": "https://client.example.org/cb",
        "jws": "eyJhbG...IAoDA",
        "proofPurpose": "authentication",
        "verificationMethod": "did:example:holder#key-1"
    }
}
```

1.3.5 Response Mode direct_post

Esiste una modalità denominata Response Mode direct_post che permette al Wallet di inviare Authorization Request direttamente ad un endpoint del Verifier via parametro HTTP POST. Questo metodo si usa quando Holder e Verifer sono su device diversi.

1.3.6 Firma e Crittografia della Response

L'Authorization Response contenente la VP Token può avere una Segnature cioè una firma e può essere crittograffata a livello di applicazione nello schema TCP/IP.
Se si attuano queste pratiche attraverso la crittografia si può prevenire la leak di dati e con la firma si può avere la certezza che i dati al suo interno sinao originali.

1.3.7 VP Token Validation

Un Verifier deve validare un VP Token. Deve garantire integrità, autenticità e binding dell'Holder con qualsiasi presentazione che sta cercando di fornire. Verificare che la presentazione rispetti i formatti richiesti.

1.3.8 Wallet Metadata

Il Verifier può accedere ad informazioni come il formato delle credenziali, tipo di proof, ed in generale gli algoritmi supportati.

1.4 Libraries

1.4.1 Waltid ssikit

Libreria con la quale si può realizzare Issuer, Verifier e Wallet. Questa libreria è sviluppata da WaltID e ai fini del progetto può interessarci la sua implementazione attraverso REST API.
Nel caso il progetto abbia codice in Java/Kotlin questa libreria offre anche una implementazione tramite dependency di Grandle/Maven, tuttavia dalle specifiche del capitolato, dato che sarà necessario la creazione di una WebApp, questo dettaglio implementativo rende difficile utilizzare ques'ultima soluzione.

Offrono una buona documentazione dal proprio sito insieme anche ad esempi e container Docker direttamente con cui fare il deploy.
Ha una licenza di tipo Apache License, Version 2.0.

1.4.2 SpruceID oidc4vci-rs

Questa libreria può essere usata per implementare lo standard OIDC4VCI, quindi solo lato Issuer.
Questa libreria è stata realizzata con Rocket utilizzando il linguaggio di programmazione Rust. Oltre al suo dettaglio implementativo offre degli endpoint raggiungibile tramite chiamate HTTP, una volta fatto il deploy si può interaggire infatti attraverso chiamate HTTP. Offre una utile demo di un Issuer.
Ha una licenza di tipo Apache License, Version 2.0.

1.4.3 Spheron

Questa libreria ha diverse implementazioni. Offre la possibilità di implementare Wallet ed Issuer, solo Wallet, Wallet e Verifer, così appare da una breve analisi del sito OpenID.
Per quanto riguarda il primo modo rispetta lo standard OpenID4VCI, ed un dettaglio importante è che utilizza il Pre-authorized Code Flow, un metodo di autorizzazione non descritto in questo documento ma abbastanza comprensibile.
Tramite questa libreria si possono implementare tutti i 3 gli attori, sia Wallet, Issuer e Verifer. Tramite il suo SSI SDK si può implementare anche OID4VC ed OID4VP. Questa libreria è realizzata utilizzando Typescript, e questo è un dettaglio implementativo buono tenendo in considerazione che il gruppo dovrà realizzare una WebApp. Offre una discreta documentazione insieme anche a strumenti di testing.

2 Riferimenti

Lista delle repo ufficiali per lo SPID:
Documentazione OpenID:
https://openid.net/openid4vc/
Presentazione utile a capire il modello
https://www.slideshare.net/TorstenLodderstedt/openid-for-verifiable-credentials-iiw-36
Libreria WaltID SSIKIT
https://github.com/walt-id/waltid-ssikit
Libreria SpruceID
https://github.com/spruceid/oidc4vci-rs
https://github.com/spruceid/oidc4vci-issuer
Libreria Spheron
https://github.com/Sphereon-Opensource/OID4VCI
https://github.com/Sphereon-Opensource/SIOP-OID4VP
https://github.com/Sphereon-Opensource/ssi-sdk