Te Tari Taiwhenua
Internal Affairs

Verifiable Credentials in Action

Detailed Technical Whitepaper v3.0

Contents

Introduction

Document purpose

Verifiable Credentials will play a pivotal role in driving economic and social prosperity in New Zealand. These credentials (and the wallets which hold them) will significantly improve access to services for New Zealanders and empower citizens with greater privacy, choice, and control than ever before. The purpose of the document is to:

- Describe detailed technical overview for issuing Verifiable Credentials and presenting Verifiable Credentials with the Verifiers (aka relying parties) to obtain Verifier service access or apply for entitlements, etc.
- Test the technical overview and technical choices with Issuers, Verifiers and Market Participants within digital identity ecosystem.
- Gather feedback from policy, security, privacy, legal and technical teams and identify gaps and any potential alternative options in the proposed technical overview.

It is essential to point out that the proposed technical overview articulated in this whitepaper is not the final ecosystem architecture and tries to bring Issuers (e.g. DIA and other agencies), Verifiers and Market Participants onto the same page. The technical whitepaper may require an update and depends on the feedback received.

We are sensitive to and mindful of the future Digital Identity Services Trust Framework (DISTF) and intend to align with the DISTF when this is established.

Background

The Department of Internal Affairs (DIA) oversees the RealMe platform, which serves today as the New Zealand government's core digital identity provider. RealMe services have been recently migrated to the cloud platform to allow RealMe to be fit for the future and DIA will continue to operate RealMe services.

The Department is committed to supporting the Future State New Zealand Identity Ecosystem and intends to transition to Verifiable Credentials over the coming years. DIA believes its role as custodian of much of New Zealand's core life and identity data and its extensive experience in managing and maintaining high-integrity identity records positions well to provide this vital infrastructure.

The Department has written a white paper on transiting to verifiable credentials and has received feedback from various stakeholders, including Verifiers and Market Participants. A key piece of feedback concerned the need for more clarity about technical implementation and choices. The Department has decided to develop this detailed technical overview white paper to address the feedback.

Glossary of terms

The following terms are used in this document:

Term	Detail
API	API is the acronym for Application Programming Interface, a software intermediary that allows two applications to transfer personal information.
Authenticator	An authenticator is a means used to confirm a user's identity, that is, to perform digital authentication. A person authenticates to a computer system or application by demonstrating that he or she has possession and control of an authenticator.
Client Organisations	Client organisations are the existing clients of RealMe, or the receivers of the DIA issued Verifiable Credentials through Holder App.
Credential	A set of one or more claims related to the customer made by an issuer.
Credential Issuance Service	Credential service issues Verifiable Credentials to the Holder App.
Credential Repository	A Credential Repository is a storage vault deployed on a personal device or cloud service that stores and protects access to the holder's Verifiable Credentials.
Customer	A member of the public or user who enrols with the Client Organisation services for applying for entitlements or creating an account. Note that the customers can be enterprise workforce to get Verifiable Credentials from the issuers.
DID	Decentralised Identifier, A DID is a simple text string consisting of three parts: 1) the DID URI scheme identifier, 2) the identifier for the DID method, and 3) the DID method-specific identifier.
EIC	Electronic Identity Credential (EIC) represents the verified identity information, including Full Name, Date of Birth, Place of Birth, Photo and Registered Sex and is regulated by the Electronic Identity Verification Act 2012.
Issuer	An issuer is a role an entity can perform by asserting claims about one or more subjects, creating a Verifiable Credential from these claims, and transmitting the Verifiable Credential to a holder.

Holder	A holder is a role an entity might perform by possessing one or more Verifiable Credentials and generating presentations from them. Holders store their credentials in credential repositories.
Holder App	Holder App is an example credential repository. Holder App acts as an agent for the Holder that receives, stores, presents, and manages Credentials and key material of the customer. There is no single deployment model for a Holder App. Credentials and keys can be stored/managed locally by the customer or by using a remote self-hosted or third-party service.
Life data attributes	They include full name, dob, gender, place of birth, passports, citizenship status, etc.
ODIC	OpenID Connect is an identity layer on top of the OAuth 2.0 protocol. It enables the Verifiers or client organisations to verify the identity of the customer based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the customer in an interoperable and REST-like manner.
Relying Party	The relying party is a service provider offering customers digital or in-person services and relies on external identity providers for customer authentication and information.
RealMe FIT	RealMe Federated Identity Tag is a unique identifier per which represents the association between the Customer and the Client Organisation’s privacy domain.
SAML	Security Assertion Markup Language is an open standard for exchanging authentication and authorization data between an identity provider and a service provider aka relying party or client organisation.
URI	A Uniform Resource Identifier is a unique sequence of characters that identifies a logical or physical resource used by web technologies.
Verifier	A verifier is a role an entity performs by receiving one or more Verifiable Credentials, optionally inside a Verifiable Presentation for processing. Other specifications might refer to this entity as a relying party or OAuth client.

Verifiable Credential	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.
Verifiable Credential Lifecycle Management	The Verifiable Credential lifecycle involves changing the credential status based on changes to the verified identity record or a notification from the Holder App regarding any changes to the device authentication etc.
Verifiable Credential Status Check	This is one of the future capabilities proposed as part of this white paper. Verifiers query Issuer's (i.e. DIA) status check capability to confirm the status of Verifiable Credential that they have received from the Holder App.
Verifiable Data Registry	A role a system might perform by mediating the creation and verification of identifiers, keys, and other relevant data, such as Verifiable Credential schemas, revocation registries, issuer public keys, and so on, which might be required to use Verifiable Credentials. Example of verifiable data registries includes trusted databases, decentralized databases, government ID databases, and distributed ledgers. Often there is more than one type of verifiable data registry utilized in an ecosystem.
Verifiable Identity Credential	It is a Verifiable Identity Credential with a credential type as "Identity Credential" and verified identity information of an individual, which includes Full Name, Date of Birth, Place of Birth, Photo and Registered Sex.
Verifiable Presentation	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.
Verified Identity Store	DIA's register for saving the verified identity record for the customers. The customer's verified identity record contains their full name, place of birth, date of birth, registered sex and verified photo, and links to the authenticator (s).

Table 1: Glossary of Terms

Verifiable Credentials Conceptual View

The customers want to have **Control** over their identity information, **Choice** about when, how and to whom it is asserted as proof of identity and **the Portability** of those attributes and credentials for ease of use, and **Trust** that their information is private and secure. Issuing the life data attributes as Verifiable Credentials to the customers is a fundamental anchor for the digital services ecosystem. It enables customers to bind their identity attributes to credentials they might want from other private and public sector agencies.

The Department plans to issue the authoritative Verifiable Credential for the customers with their core life data information (name, date of birth, place of birth, registered sex, and photo) to enable a customer-centric identity ecosystem and the economy to function productively and make customers' lives easier. Issuing Verifiable Credentials with identity information goes through a rigorous identity-proofing process, ensuring the information is accurate and binding it to the genuine customer's biometric information is a vital element of the issuance process.

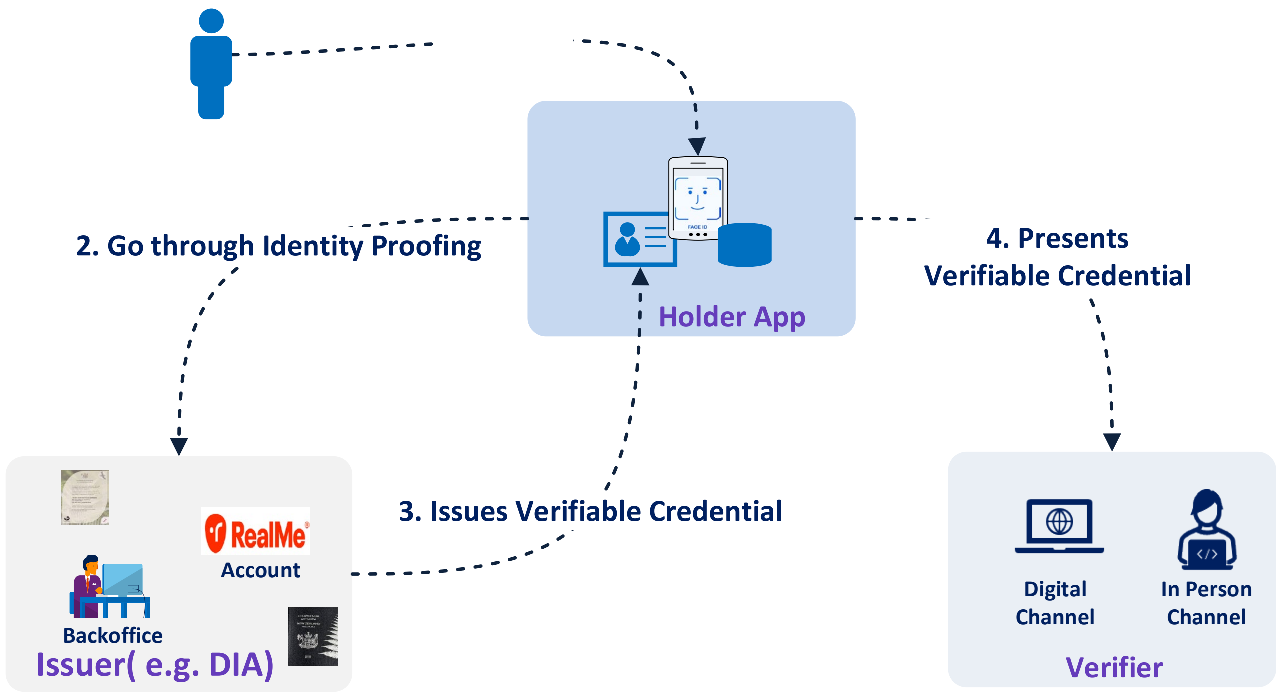

Holder App
2. Go through Identity Proofing
4. Presents Verifiable Credential
3. Issues Verifiable Credential
RealMe
Account
Backoffice
Issuer(e.g. DIA)
Digital Channel
In Person Channel
Verifier

Figure 1: Verifiable Credentials Issuance and Presentation – Key Roles

Note: other agencies can be issuers of Verifiable Credentials based on the customer's attribute data that they hold, and they can use their own identify proofing methods or rely on external services such as RealMe to issue Verifiable Credentials.

Key Roles

- **Holder/ Customer:** A customer or holder is a person who initiates the transactions for obtaining Verifiable Credentials from issuers and provides consent for presenting Verifiable Credentials to the verifiers.
- **Issuer**: For example, DIA acts as an issuer by creating a Verifiable Credential from the RealMe verified identity or its authoritative information and transmitting the Verifiable Credential to the customer's preferred wallet. It has the potential to be fully automated and/or may require back office manual checks. Other issuers follow their own proofing process to issue Verifibale Credentials to the customers.
- **Holder App**: A Holder App is a mobile application with one or more Verifiable Credentials and can generate presentations of the credentials, as appropriate. Holders store their credentials in credential repositories (i.e. device or cloud-based identity hubs). There is an assumption that the market will offer the holder app to the customers.
- **Verifier**: A verifier is a relying party that receives a Verifiable Credential from the holder, optionally inside a Verifiable Presentation for processing.

What is Verifiable Credential?

The Verifiable Credential is a key enabler for the **Future State New Zealand Identity Ecosystem**, enabling **Portability**, **Control** and **Choice** for the customer of their identity and other attributes.   DIA's intends to support the **Future State New Zealand Identity Ecosystem** and key customer outcomes by issuing Verifiable Credentials with the authoritative life data attributes we hold for New Zealanders.

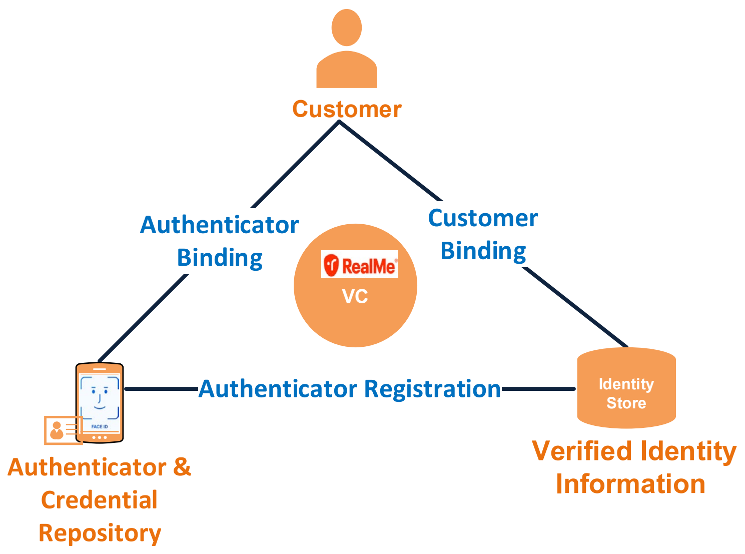

Customer
Authenticator Binding
Customer Binding
RealMe
VC
Authenticator Registration
Identity Store
Verified Identity Information
Authenticator & Credential Repository

Figure 2: Example RealMe Verifiable Credential

The example diagram represents the connection between the Customer, Authenticator (represented by a Holder App) and Verified Identity Information (represented by the identity store).

The Customer and Verified Identity information relationship is marked as Customer binding. The Verified Identity Information and Authenticator relationship is marked as Authenticator Registration. The Authenticator and the Customer relationship is marked as Authenticator Binding. The triangulation of these entities represents the **Verifiable Credential**.

The following are the key points regarding **the Example RealMe Verifiable Credential**:

- The customer's RealMe verified identity record or authoritative information is stored in DIA's identity register as per current state.
- The customer is bound to their RealMe identity record through DIA's identity proofing process which involves knowledge, possession, and biometric checks in alignment with the NZ Identification Management Standards.
- The Holder App is an authenticator linked to the RealMe verified identity (claimed identity record) and bound to the customer.
- The Verifiable Credential with identity claims is persistent on the customer's device and cannot be altered as DIA signs the Verifiable Credential using DIA's signing key.

Note: other agencies can be issuers of Verifiable Credentials based on the customer's attribute data that they hold, and they can use their own identify proofing methods or rely on external services such as RealMe to issue Verifiable Credentials.

Technical Overview

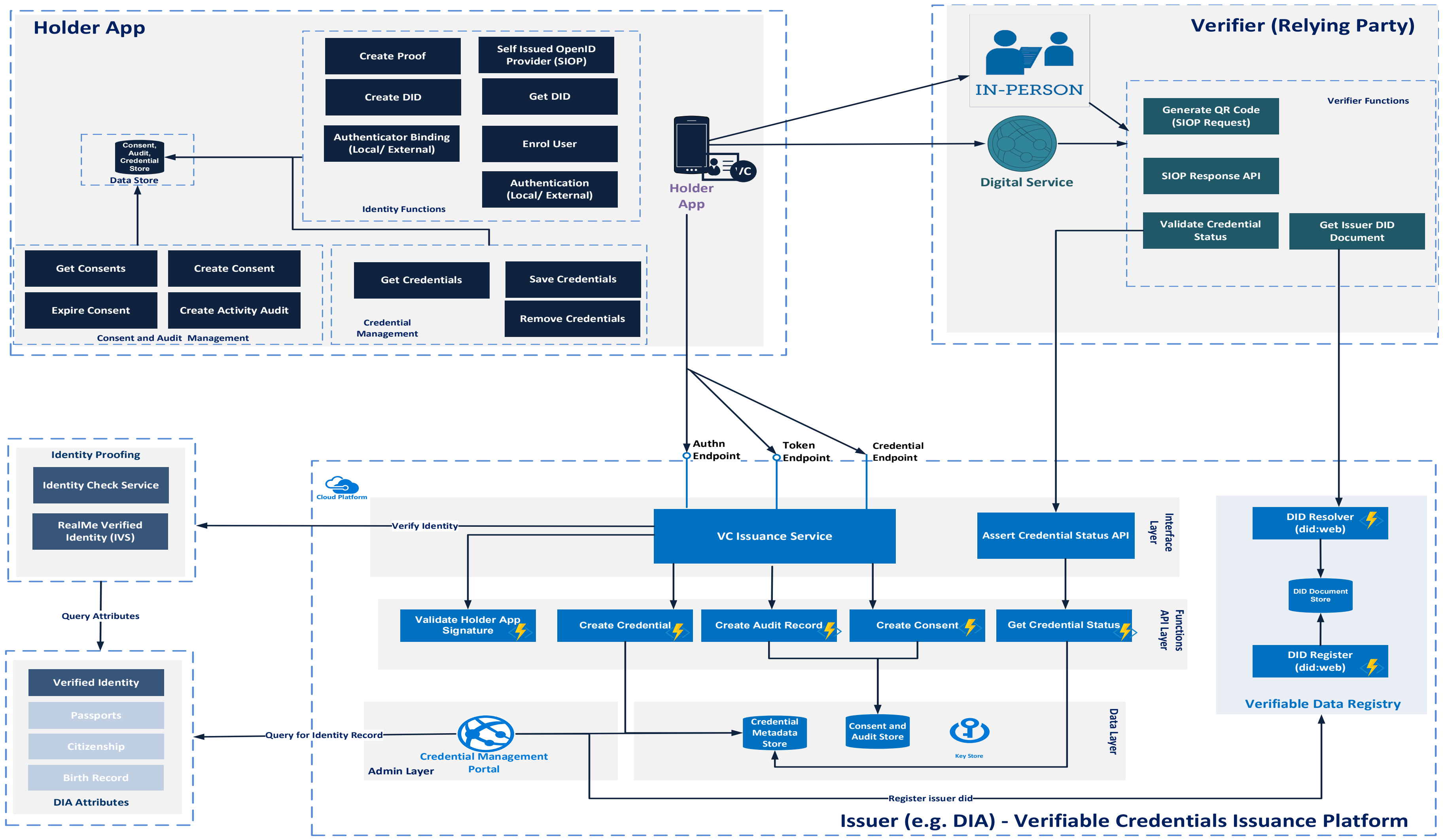


Figure 3: Key Roles Functions - Technical Overview

Technical Choices

The following are the key choices around design aspects DIA has identified based on the best current knowledge. DIA is interested in feedback from other Issuers and Market Participants on these choices.

- DIA intends to support selective disclosure and believes this can be achieved by issuing a separate Verifiable Credential for each identity claim.
- DIA will implement OpenID Connect for Verifiable Credentials for issuing Verifiable Credentials to the Holder App.
- DIA 's DID document will be stored in the did:web endpoint. DIA will host did:web ./well-known endpoint in a public cloud environment to share its DID document.
- DIA will host the Verifiable Credential Assertion Status endpoint for the Verifiers to confirm the validity of the Verifiable Credential.
- The Holder App stores received Verifiable Credentials in the secured location of the device.
- The Holder App shares the Verifiable Credentials with the Verifiers using the Self Issued Open ID Provider Verifiable Presentation profile.
- The example Verifiable Credentials message format in this document is "ldp_vc".

Issuer Functional Capabilities

Issuer capabilities help issue Verifiable Credential(s) to customers, which are trustworthy and acceptable to the Holder Apps, Verifiers, and Identity ecosystem. The Issuer should manage the Verifiable Credential(s) lifecycle, which includes updating, suspending, and revoking the Verifiable Credential(s) to maintain its integrity.

Note: DIA expects the following capabilities to issue Verifiable Credentials to the Customer. DIA is interested in feedback from the other Issuers and Market Participants.

Verifiable Credential (VC) Issuance Service

DIA is an issuer of Verifiable Credentials with identity claims and issues Verifiable Credentials based on the binding of a person to a record held by DIA, which include:

- RealMe verified identity,
- NZ Passport or
- NZ Citizenship by Grant records.

VC Issuance Service is an OpenID Connect (OIDC) standard-based interface to orchestrate customer functional flows for issuing Verifiable Credentials based on the Person record held by DIA to the Holder App(s). The VC Issuance Service offers the following OIDC endpoints for the Holder App:

- **Authorization Endpoint**: The Holder App redirects the customer with an authorization request to the Authorization Endpoint of VC Issuance Service. The Authorization Endpoint validates the authorization request and verifies the identity of the customer, which typically includes user authentication at RealMe or real-time binding of the Person to a record (i.e. NZ Passports or NZ Citizenship records) and gathers user consent on successful identity proofing for issuing Verifiable Credentials. Upon successfully receiving identity assertion from the DIA identity proofing services, the Authorization Endpoint returns an authorization response with the Authorization Code to the Hoder App. The customer's verified identity details will be stored in a temporary cache, which maps to the authorization code and access token.

  Optionally the Authorization Endpoint can also obtain an attestation along with the passkey from an inbuilt device FIDO authenticator to allow the customers to login to RealMe using their device as an authenticator.
- **Token Endpoint**: The Holder App sends a token request to the Token Endpoint with the Authorization Code through a back channel. Upon successfully validating the Authorization Code, the Token Endpoint returns an Access Token in the Token Response.
- **Credential Endpoint**: The Holde App sends a credential request to the Credential Endpoint with the Access Token and the proof of possession of the public key to which Verifiable Credential will be bound. Upon successfully validating the Access Token and signature proof, the Credential Endpoint returns the Verifiable Credentials with identity claims in the Credential Response.

Note: refer to the OIDC message specification section for VC issuance.

Identity Proofing

Identity proofing is a process of establishing an identity of the person based on the binding of a person with the records held at the Issuer or relying on third-party Identity providers such as RealMe.

DIA uses the following for an identity proofing to issue Verifiable Credentials.

- RealMe Verified Identity or
- Binding of a person to the NZ Passports Record in real-time or
- Binding of a person to the NZ Citizenship Record in real-time.

Create Consent Function API

The VC Issuance Service authorisation endpoint captures the customer consent for issuing Verifiable Credentials with identity claims to the Holder App as part of the Verifiable Credential issuance journey. The VC Issuance invokes Create Consent API to write a consent record in the audit and consent datastore.

DIA as an issuer saves the following consent details:

- Consent Purpose for issuing Verifiable Credentials with identity attributes.
- Consented Attributes (i.e. full name, date of birth, place of birth, gender, verified photo)
- Consented relying party id (Holder App id)

- Consented Date Time
- Identity Record ID
- Holder App DID (did:key or did:jwk)

Note: the consent event will be stored in consent and audit datastore.

Create Audit Record Function API

The VC Issuance Service invokes the Create Audit Record Function API to create an audit record of every step involved in the VC issuance process to support non-repudiation requirements.

DIA as an issuer saves the following audit details:

- Audit Event Creation Date
- Audit Text
- Audit Type (user, system etc)
- Relying Party ID (Holder App id)
- Identity Record ID
- Holder App DID (did:key or did:jwk)

Note: the audit event will be stored in consent and audit datastore.

Validate Holder App Signature Function API

The VC Issuance Service Credential Endpoint receives a credential request from the Holder App with the signature proof as a signed JSON Web Token (JWT). The Holder App signs the JWT using its private key. The VC Issuance Credential Endpoint calls this API to validate the Holder App signature of JWT. The public key of the Holder App can be part of the issuer claim and expressed as either did:jwk or did:key format.

Note: Refer to the OIDC message specification for issuance.

Create Credential Function API

On successful verification of JWT signature, VC Issuance Credential Endpoint invokes this API for creating a Verifiable Credential. The API request contains the following parameters:

- Messaging standard format (JSON-LD or JWT)
- JSON Claims

The API creates the Verifiable Credential(s) and the following are the key points regarding Verifiable Credential(s):

- Verifiable Credential contains user claims and issuer's signature.
- Issuer signs Verifiable credential(s) using their signature key. The issuer's signing public key can be provided through did:web endpoint.
- Verifiable Credential contains Issuer identifier as did:web:${issuer did endpoint name}, e.g. did:web:diddoc.identityservices.dia.govt.nz.
- Issuer can issue multiple Verifiable Credentials to support selective disclosure use cases.

- The following credential metadata information will be saved in the credentials metadata store:
    - Credential Issuance Date – Verifiable Credential issued date.
    - Credential Status (Active, Suspended, Revoked)
    - Credential ID – the identifier of the issued Verifiable Credential
    - Identity Record ID – the identifier of the verified identity record of the person.
    - Credential Attributes – Attribute name or comma separated attribute names.
    - Holder App DID (did:key or did:jwk)

Credential Management Portal

The Issuer (e.g. DIA) Administration team log in to the Credential Management portal using their enterprise login credentials. The key objective of the Credential Management portal is to manage the status of the issued Verifiable Credential. The Issuer should define new operating model for credential management process which includes access, integration support, technical support and contact centre support etc.

The Credential Management portal has the following key technical functions:

- **Authentication and Access**

  The Administrators access the Credential Management portal using their enterprise login through the Issuer's IDAM federation. For example, DIA administrators access the portal using their DIA credentials through Azure AD Federation.

- **DID Document Management**

  The Issuer Administration team registers or updates the Issuer's Decentralised Identity (DID) Document with a Verifiable Data Registry through the Credential Management portal.

- **Search User**

  The Issuer Administrator team searches for the user using their identity details. The portal retrieves an identity record from the identity repository. On successful identity retrieval, the portal queries the credential metadata store using the identity record ID. If identity and associated credential metadata are found, the user summary page is displayed with the identity and associated credential metadata details.

- **View Consent and Audit History**

  The Administrator team views the consent and audit history associated with the Identity record. The Administrator team uses this information to confirm the caller's identity (i.e. customer).

- **Update Credential Status**

  The Administrator Team suspends or revokes the Verifiable Credential in case of suspicious activity or fraud to maintain the integrity of the Verifiable Credentials. The Administrator team can also suspend or revoke the Verifiable Credential if the person calls the Contact Centre team for a lost device. The portal updates the Verifiable Credential status in the Credential Metadata datastore.

Assert Credential Status API

The API receives Verifiable Credential(s) which were issued by the Issuer to provide the status of the Verifiable Credential(s). This API will enable Verifiers to meet regulatory obligations. The API can provide unique pseudonymous identifier such as RealMe FIT and Verifiable Credential status (i.e., active, revoked, etc.). The API can also provide Verifiable Credential(s) status based on query from the Holder App.

Key Store

Issuer Key Store component persists resources keys, shared secrets, and database connection strings. The certificates and shared keys require an update, and the Issuer support team manages the certificates and keys. The Administrator team updates the DID document in the Verifiable Data registry (i.e. did:web endpoint) with a new signing public key using the Credential Management portal.

Issuer Credential Data Model

Issuer Credential Data Model contains the following entities:

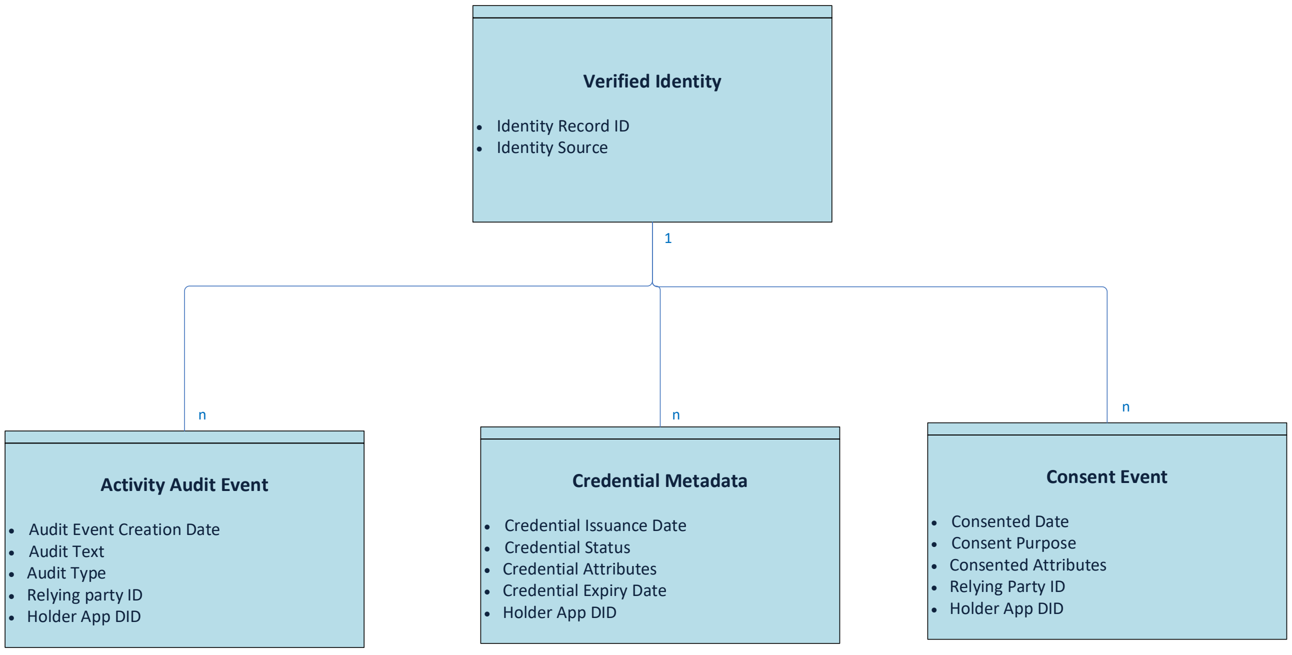


Figure 4: Issuer Credential Data Model

Holder App Functional Capabilities

The Holder App enables customers to have **Control** over their identity information, **Choice** about when, how and to whom it is asserted as proof of identity and **the Portability** of those attributes and credentials for ease of use. There is an assumption that the market will offer the holder app to the customers.

Defining the capabilities of the Holder App would ensure trust and completeness across the end-to-end Verifiable Credential technical view. The following are the key functions of the Holder App:

Identity Function: Enrol User (Authenticator Setup)

The customer downloads the Holder App and installs it on their mobile device. One of the options the Holder App can offer is face as an Authenticator. The customer sets up their face as an Authenticator as part of the set-up. To enrol face as an authenticator, the Holder App confirms the customer as a live human through liveness. The Holder App uses local or external biometrics capabilities to enrol the customer's live face as an authenticator. The face template will be created and stored on successful customer enrolment in the Holder App datastore. The Hoder App saves identity record with the following details:

- Creation Date
- Face Template
- Holder App DID (did:key or did:jwk)

This white paper preferred Face as an Authenticator as it is one of the strongest authentication methods. NZ Trust Framework rules will govern the Holder App providers to implement appropriate Authenticator options.

Identity Function: Create Decentralised Identifier (DID)

On successful customer enrolment, the Holder App creates private and public key pair using the Elliptic Curve Digital Signature Algorithm with the P-256 curve and the SHA-256 hash function (ES256). It is an asymmetric algorithm that uses the ECDSA private key to generate signature proof. The Holder App saves the private key in device secured location.

The Holder App creates did:key or did:jwk using the public key. The Holder App saves did:key or did:jwk against the customer’s face template in the Holder App datastore. The Holder App DID will be provided to the Issuer’s VC issuance service to obtain the Verifiable Credential. The Holder App DID is subject of the Verifiable Credential.

Identity Function: Authenticator Binding

One of the options to bind the customer to the Holder App is through their Face. It can be through Device-based Face Authentication like Passkey or custom biometric implementation supported by the Holder App. The Holder App should confirm the liveness of the customer before enrolling their Face if they support custom biometric implementation. The liveness evaluation includes:

- photo quality check for the customer enrolment,
- evaluation of liveness frames which includes anti-spoofing protection by using a presentation attack detection algorithm. Tracking between frames in the video is used to detect suspicious variations in facial location.

- creates face template of successful live face.

The Holder App uses its biometric libraries (local) or integrates with external biometric services to confirm the liveness.

The Holder App can also use multifactor authentication through user knows, and user has factors.

Identity Function: Authentication

The Holder App can support multiple authentication options to authenticate the customer. The Authentication options include:

- Multifactor Authentication using the user knows and user has factors.
- Face biometric authentication (Local or External): The Holder App confirms the liveness before matching the live image against the enrolled face template to verify the authenticity of the customer accessing the Holder App.
- Passkey and other authentication method etc.

Identity Function: Get DID

The Holder App uses this function to get DID associated with the Holder App. The Holder App DID retrieves DID from the Identity record.

Identity Function: Create Proof

The Holder App uses this function to create signature proof by issuing signed JSON Web Token (JWT) to the obtain Verifiable Credential(s) from the Issuer's VC Issuance Service Credential Endpoint.

Note: Refer to the OIDC Message Specification section for details.

Identity Function: Self-Issued OpenID Provider (SIOP)

The Holder App acts as a Self-Issued OpenID Provider (Self-Issued OP), an OP controlled by the Customer. Using SIOP, the Customer authenticates themselves with Self-Issued ID Tokens signed with the private key under the Customer's control and presents Verifiable Credential(s) as a VP (Verifiable Presentation) Token directly to the Verifiers.

Note: Refer to the OIDC Message Specification section for details.

Consent and Audit Function: Create Consent

The Holder App captures the customer consent for sharing Verifiable Credentials with the Verifiers as part of SIOP request for presenting Verifiable Credential(s). The Holder App uses Create Consent function to write a consent record in the Holder App datastore. The consent record is a specialised audit event with the following details:

- Consent Purpose for sharing Verifiable Credentials with the Verifier
- Consented Credential Claims (i.e. full name, date of birth, place of birth, gender, verified photo, Address etc)
- Consented relying party id (Verifier id)
- Consented Date Time
- Consent Expiry Date Time

Consent and Audit Function: Expire Consent

The Holder App uses this function to expire enduring consents if the customer wants to withdraw them. The Expire Consent Function updates the expiry date of the enduring consent.

Consent and Audit Function: Get Consents

The Holder App uses this function to display existing consents associated with the customer by querying the Holder App store.

Consent and Audit Function: Create Activity Audit

The Holder App uses this function to create an audit record of every step involved in the VC issuance and VC sharing transactions to support non-repudiation requirements. The audit event consists of the following details:

- Audit Event Creation Date
- Audit Text
- Audit Type (user, system etc)
- Relying Party ID (Holder App ID)
- Issuer ID

The audit event will be stored in the Holder App datastore.

Credential Management Function: Save Credentials

On receiving Verifiable(s) from the issuer, the Holder App uses this function to save Verifiable Credential(s) to the Holder App datastore.

Credential Management Function: Get Credentials

On successful customer authentication, the Hoder App retrieves the Verifiable Credential(s) from the Holder App Datastore and reads the claims from the Verifiable Credential(s) and displays the claims to the customer.

Credential Management Function: Remove Credential

The Holder App uses this function to remove Verifiable Credential(s) from the Holder App datastore when:

- the customer performs an action to remove Verifiable Credential,
- any changes to the Holder App DID,
- any changes to the device such as jailbroken or rooted etc.

Holder App Data Model

Holder App Data Model contains the following data entities:

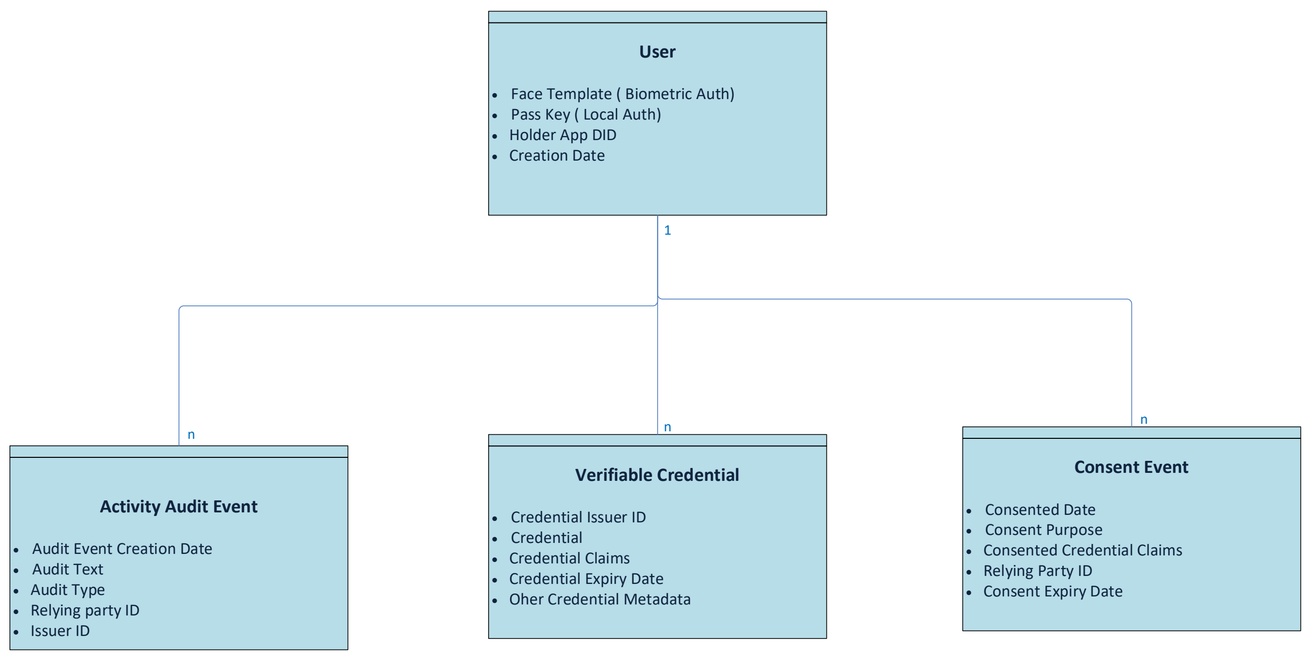


Figure 5: Holder App Data Model

Verifiable Data Registry Functional Capabilities

The Verifiable Data Registry is a trust capability that will be deployed as did:web endpoint under the Issuer's VC Issuance Platform. The Verifiable Data Registry has the following functions:

Register DID Function API

The Issuer registers their Decentralised Identifier (DID) using the following information:

- Domain Name which resolves to did:web endpoint
- DID Document which is in JSON-LD format with the following details:
  - id – Decentralised Identifier of Issuer (e.g: did:web:realme.govt.nz)
  - Verification Method - contains the public key of the Issuer
    - id – identifier of key, e.g. "did:web:realme.govt.nz#key1".
    - type – the value must be set to "JsonWebKey2020".
    - controller – must be same as id, e.g. "did:web:identityservices.dia.govt.nz".
    - publicKeyJWK
      - kty – should be set to Elliptic Curve Algorithm, i.e "EC".
      - crv – should be set to "P-256" - denotes that this is a P256 curve.
      - x- value of the public key.
      - y- value of the public key.
  - AssertionMethod – public key identifier e.g. "did:web: identityservices.dia.govt.nz #key1".
  - Authentication – public key identifier e.g. "did:web: identityservices.dia.govt.nz #key1".

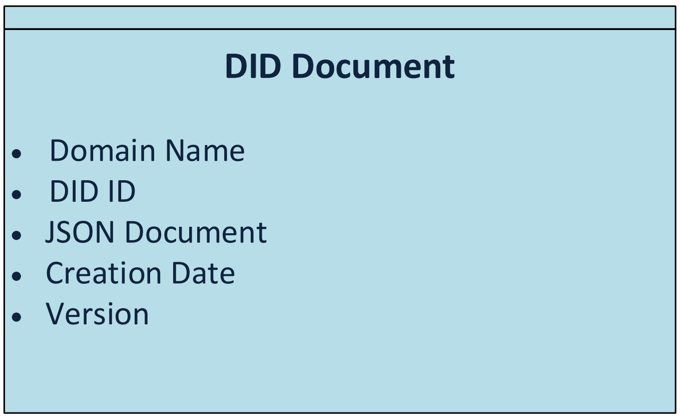


Figure 6: DID Document Entity

Resolve DID Function API

The Issuer DID resolve the domain and forms a well-known URI to access the Issuer's DID Document. For a DID "did:web:issuer.govt.nz", the Verifier will attempt to access the Issuer's DID document at https://issuer.govt.nz/.well-known/did.json. The well-known endpoint is publicly accessible.

Note: refer to the DID Document specification section for the DID Document details.

Verifier Functional Capabilities

Verifiers are service providers who need to confirm identity information from their customers. Verifiers can collect and verify customer information in a more convenient, cost-effective, private, and trustworthy way for them and their customers using Verifiable Credentials. Verifiers don’t need to retain a hard copy of identity documents or personal information. The customer can present their Verifiable Credential(s) to access services (digital or in-person) that the Verifiers offer. The Verifiable Credential(s) will reduce any risk of any privacy breaches or hacks designed to access and publicise personal information, creating identity theft opportunities.

Digital Service

The Verifier’s Digital Service is a web service which offers a service specific to the Verifier’s domain context. Customers access the service over the Internet and must prove themselves digitally. One of the options could be sharing their Verifiable Credential(s) to prove themselves digitally.

The Verifier’s Digital Service uses the Generate QR Code function to display the QR Code. The customer uses their Holder App to scan the QR Code. The Digital Service provides access to the customer after receiving the Verifiable Credential(s) claims from the SIOP Response API.

Generate QR Code Function

Verifiers create a QR code with Self Issued Open ID Provider authorisation request. The authorisation request will always be bound to the user’s session. The Holder App scans the QR code to retrieve authorisation parameters.

Note: refer to the OIDC message specification section for VC sharing.

SIOP Response API

On successful validation of OIDC authentication request and user consent, the Holder App creates an ID Token and VP Token with verifiable credentials as claims and shares them to the Verifier’s SIOP response API. On receiving the response from the Holder App, the SIOP response API:

- validates the ID Token signature, which is signed by the Holder App,
- validates the VP Token signature, which is signed by the Holder App,
- retrieves the Verifiable Credential(s) from the VP Token,
- gets the Issuer’s DID from the VP Token,
- uses Get Issuer DID Document function to get the Issuer’s assertion public key,
- validates the Verifiable Credential(s) signature using the Issuer’s assertion public key,
- optionally uses the Validate Credential Status Function to confirm the validity of the Verifiable Credential(s).
- reads the claims from the Verifiable Credential(s) on successful validation of the Verifiable Credential(s),
- updates the customer’s session with Verifiable Credential(s) claims,
- returns HTTP status code 200 to the Holder App,

Get Issuer DID Document Function

The following are key points regarding this function:

- uses Issuer’s DID, i.e. “did:web:issuer.govt.nz” and attempts to access the Issuer’s DID document at https://issuer.govt.nz/.well-known/did.json
- reads assertion public key from the Issuer’s DID document and
- shares assertion public key to the SIOP Response API.

Validate Credential Status Function

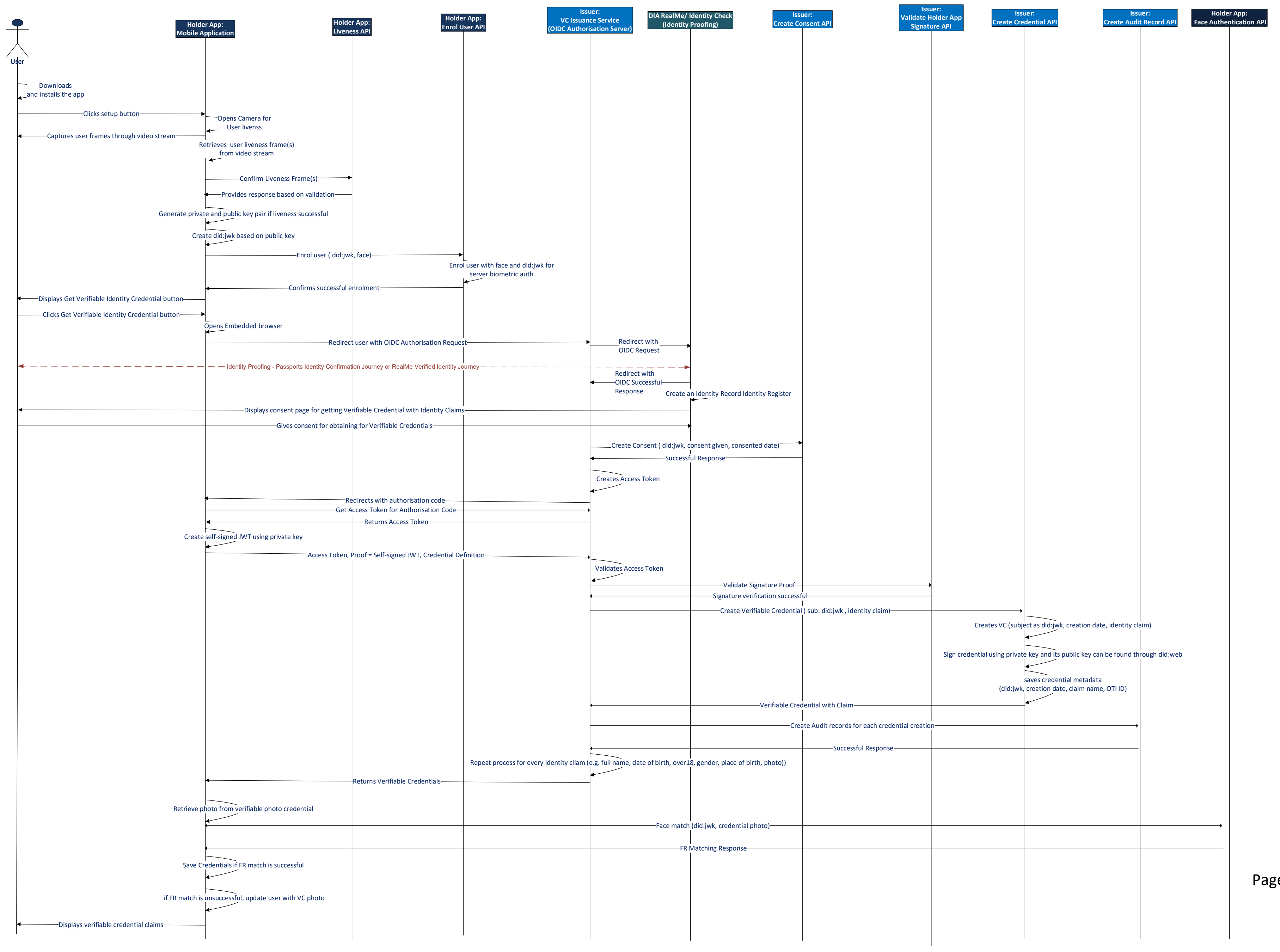
The following are key points regarding this function:

- queries the Issuer’s VC Status API to confirm the status of the Verifiable Credential(s).
- if the status is “active” then confirms the validity of the Verifiable Credential(s) to the SIOP Response API.
- If status is not “active” then returns error response.

Proposed Issuer Message Flow – Verifiable Identity Credentials Issuance by DIA

The following diagram depicts the Verifiable Credentials with Identity claims issued by the Department of Internal Affairs (DIA).

Note: This flow can differ for other Issuers who issue verified credentials to the customer.


User
Holder App: Mobile Application
Holder App: Liveness API
Holder App: Enrol User API
Issuer: VC Issuance Service (OIDC Authorisation Server)
DIA RealMe/ Identity Check (Identity Proofing)
Issuer: Create Consent API
Issuer: Validate Holder App Signature API
Issuer: Create Credential API
Issuer: Create Audit Record API
Holder App: Face Authentication API
Downloads and installs the app
Clicks setup button
Opens Camera for User livenss
Captures user frames through video stream
Retrieves user liveness frame(s) from video stream
Confirm Liveness Frame(s)
Provides response based on validation
Generate private and public key pair if liveness successful
Create did:jwk based on public key
Enrol user (did:jwk, face)
Enrol user with face and did:jwk for server biometric auth
Confirms successful enrolment
Displays Get Verifiable Identity Credential button
Clicks Get Verifiable Identity Credential button
Opens Embedded browser
Redirect user with OIDC Authorisation Request
Redirect with OIDC Request
Identity Proofing - Passports Identity Confirmation Journey or RealMe Verified Identity Journey
Redirect with OIDC Successful Response
Create an Identity Record Identity Register
Displays consent page for getting Verifiable Credential with Identity Claims
Gives consent for obtaining for Verifiable Credentials
Create Consent (did:jwk, consent given, consented date)
Successful Response
Creates Access Token
Redirects with authorisation code
Get Access Token for Authorisation Code
Returns Access Token
Create self-signed JWT using private key
Access Token, Proof = Self-signed JWT, Credential Definition
Validates Access Token
Validate Signature Proof
Signature verification successful
Create Verifiable Credential (sub: did:jwk , identity claim)
Creates VC (subject as did:jwk, creation date, identity claim)
Sign credential using private key and its public key can be found through did:web
saves credential metadata (did:jwk, creation date, claim name, OTI ID)
Verifiable Credential with Claim
Create Audit records for each credential creation
Successful Response
Repeat process for every identity claim (e.g. full name, date of birth, over18, gender, place of birth, photo))
Returns Verifiable Credentials
Retrieve photo from verifiable photo credential
Face match (did:jwk, credential photo)
FR Matching Response
Save Credentials if FR match is successful
if FR match is unsuccessful, update user with VC photo
Displays verifiable credential claims

The following are the key points regarding the proposed VC Issuance message flow:

- The customer (i.e., the user) downloads the Holder App and sets up their authenticator using its authentication options. One of the Authentication options could be the Face as an Authenticator. For Face Authenticator binding, the Holder App confirms the liveness of the customer. It generates a private-public key pair before enrolling the customer with a face template and did:jwk or did:key to the Holder App. The user enrolment details can be saved on the device or the Holder App cloud infrastructure.
- On successful enrolment, the customer clicks the Get Identity Credentials button, which redirects the customer to the DIA VC Issuance Service Authorisation Endpoint with OIDC Authorisation Request through the embedded browser. The DIA VC Issuance Service validates the Authorisation Request and redirects the customer to the Identity Check service or RealMe Verified Identity service with the OIDC Authorisation Request for the customer's Identity verification or confirmation.
- If the customer chooses the Identity Check service as an Identity Proofing service, the Identity Check service validates the customer's entered NZ passport or NZ citizenship identity details, validates the customer's liveness, matches the live image against the passport photo, and returns Identity Claims as part of an ID Token. If the customer is RealMe customer and chooses the RealMe as an Identity Proofing service, then RealMe returns Identity Claims as part of ID Token.
- The VC Issuance Service saves Verified Identity details in DIA's Verified Identity store (IVS). The VC Issuance service displays the consent page for customer's consent for issuing Verifiable Credential(s). The VC Issuance service creates a consent record and persists Identity claims in a temporary cache before redirecting the customer with the OIDC Authorisation Code to the Holder App.
- The Holder App invokes the VC Issuance service OIDC Token endpoint with an OIDC Authorisation Code, and the VC Issuance service returns an access token to the Holder App.
- The Holder App creates a Signed JWT using their private key. It invokes the VC Issuance service OIDC Credential Endpoint with Signed JWT as signature proof, Access Token, and required credentials (full name, date of birth, place of birth, over18, gender, photo) as a Credential definition.
- The VC Issuance service validates the signature proof and Access token and creates a separate Verifiable Credential for each requested claim to support selective disclosure transactions. The VC Issuance service saves Verifiable Credentials Metadata information in the Credential Metadatastore and returns Verifiable Credential(s) to the Holder App in response.
- The Holder App retrieves a photo from Photo Verifiable Credential and does one-to-one matching against the enrolled user's face print to confirm the same user. If the match succeeds, the Holder App saves Verifiable Credential(s) in the Holder App datastore. The Holder App datastore can be on the device or the Holder App cloud infrastructure.
- The Holder App displays the Verifiable Credentials claims to the customer.

Proposed Message Flow – Verifiable Identity Credentials Presentation by Holder App

The following diagram depicts the proposed Verifiable Credentials Presentation with the Verifier's Digital service by the Holder App.

Note: DIA is interested in feedback from the Verifiers and Market Participants.

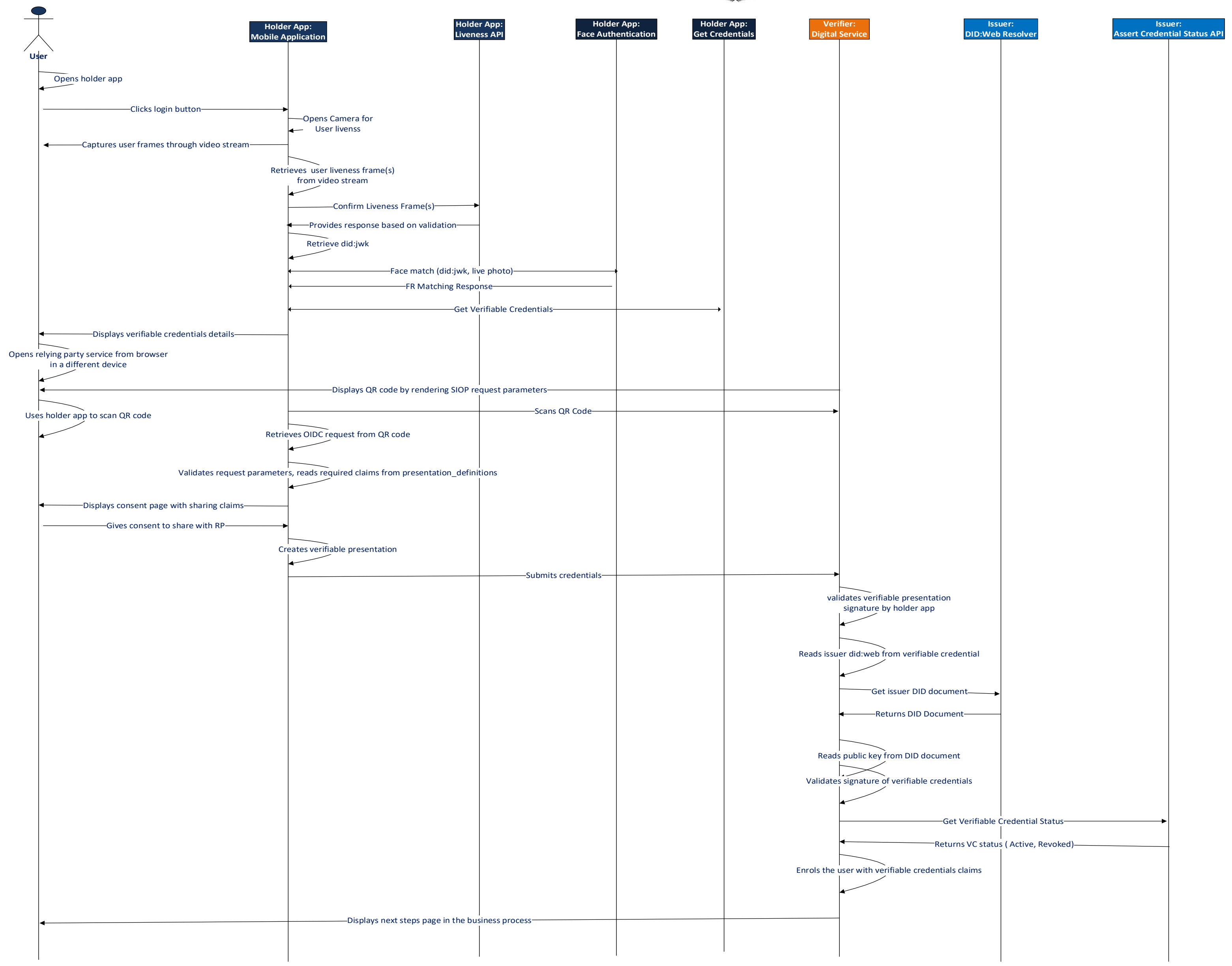

User
Holder App: Mobile Application
Holder App: Liveness API
Holder App: Face Authentication
Holder App: Get Credentials
Verifier: Digital Service
Issuer: DID:Web Resolver
Issuer: Assert Credential Status API
Opens holder app
Clicks login button
Opens Camera for User livenss
Captures user frames through video stream
Retrieves user liveness frame(s) from video stream
Confirm Liveness Frame(s)
Provides response based on validation
Retrieve did:jwk
Face match (did:jwk, live photo)
FR Matching Response
Get Verifiable Credentials
Displays verifiable credentials details
Opens relying party service from browser in a different device
Displays QR code by rendering SIOP request parameters
Uses holder app to scan QR code
Scans QR Code
Retrieves OIDC request from QR code
Validates request parameters, reads required claims from presentation_definitions
Displays consent page with sharing claims
Gives consent to share with RP
Creates verifiable presentation
Submits credentials
validates verifiable presentation signature by holder app
Reads issuer did:web from verifiable credential
Get issuer DID document
Returns DID Document
Reads public key from DID document
Validates signature of verifiable credentials
Get Verifiable Credential Status
Returns VC status (Active, Revoked)
Enrols the user with verifiable credentials claims
Displays next steps page in the business process

Figure 7: VC Presentation Message Flow

The following are the key points regarding the VC Presentation message flow:

- The customer opens the Holder App and clicks the “login” button. The Holder App confirms the liveness of the customer and does one-to-one matching against the enrolled user’s face print against the live image to ensure the same user. On successful face authentication, the Holder App retrieves the Verifiable Credentials from the Holder App datastore and displays the Verifiable Credentials to the customer.
- The customer wants to enrol with a Verifier’s Digital service and must digitally prove their identity and share other claims. The Verifier’s Digital service creates a Self-Issued OP request and renders it as a QR code as per Self Issued OpenID Provider (SIOP) protocol. The customer scans the QR code with the Holder App.
- The Holder App validates SIOP request and identifies the required claims from the SIOP request. The customer gives a consent to share claims from the Verifiable Credentials with the Verifier’s Digital service. The Holder App creates a consent record in the Holder App datastore.
- The Holder App creates a Verifiable Presentation token, defined in OpenID Connect for Verifiable Presentation protocol. The Verifiable Presentation token contains Verifiable Credentials and signs the Verifiable Presentation using its private key. The Holder App sends to the Verifier web service with Verifiable Presentation token.
- The Verifier Digital service invokes Issuer’s DID:web resolver API to obtain DIA’s DID document. The Verifiable Data Registry is a public cloud platform to save issuer DIDs (DID: web). The Verifier Digital service gets the issuer's public key from the issuer's DID document and verifies the signature of the Verifiable Credential using the issuer's public key.
- On successful signature verification, the Verifier’s Digital service invokes DIA’s Credential Status API to check the status of the Verifiable Credential. The Credential status API provides the credential status as “Active” or “Revoked” and a unique pairwise identifier (also known as RealMe FIT) which is the customer’s identifier related to the Verifier’s Digital service. The Verifier’s Digital service takes the customer to the next step in the enrolment process if the credential status is “Active”. The user takes the customer to the alternative identity verification options if the credential status is “revoked”.

VC Issuance – Proposed OIDC Message Specification

Note: The proposed message specification may require an update based on feedback from the verifiers and the Market Participants.

This section describes the intended message specification for the Department of Internal Affairs (i.e. Issuer) to issue Verifiable Credentials with Identity Claims. The Department of Internal Affairs exposes the VC Issuance Service for the Holder Apps to receive Verifiable Credentials with identity claims. The following three endpoints are involved in issuing Verifiable Credentials:

- OIDC Authorisation Endpoint
- OIDC Token Endpoint
- OIDC Credential Endpoint

OIDC Authorisation Endpoint

The Holder App redirects the user with an authentication request to the VC Issuance Service Authorisation Endpoint. The following table describes an authentication request parameter:

Parameter	Description
scope	The scope must contain: • openid • identitynamecredential • identitydobcredential • identitypobcredential • identitygendercredential • identityphotocredential • identityover18credential
authorization_details	The request parameter authorization_details MUST be passed to convey the details about the Credentials the Holder App wants to obtain if the request doesn't contain a scope parameter. Refer to the Table 3 for the details.
response_type	Response Type value that determines the authorization processing flow to be used, including what parameters are returned from the endpoints used. When using the authorisation code flow, this value is code.
client_id	The identifier of the Holder App provider.

code_challenge	It is recommended to pass this parameter when using Single Page Apps (SPA) or Mobile Apps. The Holder App creates a unique string value, code_verifier, which it hashes and encodes as a code_challenge. In place of the client_secret, while interacting with the VC Issuance service token endpoint, the Holder App uses code_verifier to authenticate itself with the token endpoint.
code_challenge_method	This parameter must be passed, and the value should be SHA256.
redirect_uri	Redirection URI to where the response will be sent, and the redirection URI MUST use the custom scheme as defined in native app redirect URI scheme. The URI must be myapp://callback.
state	Opaque value used to maintain state between the request and the callback. This will be passed back in the response and should be checked to confirm the value matches what was passed in the request. Typically, Cross-Site Request Forgery (CSRF, XSRF) mitigation is done by cryptographically binding the value of this parameter with a browser cookie.

Table 2: OIDC Authorisation Endpoint – Authentication Request

Request Parameter	Description
type	JSON string that determines the authorization details type. MUST be set to openid_credential.
format	JSON string representing the format in which the Credential is requested to be issued. MUST be set to either jwt_vc_json or ldp_vc.
types	It is an array of stings, must contain one of these values: • [“VerifiableCredential”, “IdentityNameCredential”], • [“VerifiableCredential”, “IdentityDoBCredential”] • [“VerifiableCredential”, “IdentityPoBCredential”] • [“VerifiableCredential”, “IdentityGenderCredential”] • [“VerifiableCredential”, “IdentityPhotoCredential”] • [“VerifiableCredential”, “IdentityOver18Credential”]

Table 3: authorisation_details

The following is an example of authentication request by the Holder App:

```
GET  /authorize?  scope=openid, IdentityNameCredential, IdentityDoBCredential, IdentityPoBCredential,
IdentityGenderCredential, IdentityPhotoCredential, IdentityOver18Credential

 &response_type=code

 &client_id=s6BhdRkqt3

 &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM

 &code_challenge_method=S256

 &redirect_ui=myapp://callback
```

OIDC Authentication Successful Response

Below is an example of a successful Authorization Response by Issuer Authentication Endpoint:

```
HTTP/1.1 302 Found

 Location: myapp://callback?code=SplxlOBeZQQYbYS6WxSbIA
```

OIDC Authentication Unsuccessful Response

Below is an example of an unsuccessful Authorization Response.

```
HTTP/1.1 302 Found

Location: myapp://callback?

   error=invalid_request

   &error_description=Unsupported%20response_type%20value
```

The following table provides key error codes and descriptions:

Error Code	Error Description
request_not_supported	Issuance Endpoint does not support the use of the request parameter supplied in the authentication request.
interaction_required	Issuance Endpoint requires customer interaction to proceed through authentication journey.
unsupported_response_type	Issuance Endpoint does not support obtaining an authorisation code using this method.
invalid_request	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed.
invalid_scope	The requested scope is invalid, unknown, or malformed.

server_error	Issuance Endpoint encountered an unexpected condition that prevented it from fulfilling the request.
access_denied	User exited from the issuance journey or Issuance service denied the request.
consent_required	Issuance Endpoint requires customer's consent to issue verifiable credential.

Table 4: Error Codes and Descriptions

OIDC Token Endpoint

On receiving the authorisation code, the Holder App invokes the VC Issuance Service Token Endpoint by sending the following request parameters using the "application/x-www-form-urlencoded" format with a character encoding of UTF-8 in the HTTP request entity-body. The Holder App sends the following request parameters to the **Token Endpoint**:

Request Parameter	Mandatory/ Optional	Description
code	*Mandatory*	The Holder App redeems the authorisation code with the Token Endpoint for the ID token.
grant_type	*Mandatory*	The value must be "authorization_code".
redirect_uri	*Mandatory*	The "redirect_uri" parameter value MUST be the same as the value that was included in the authorization request.
code_verifier	*Mandatory*	The same code_verifier used to obtain the authorization code. Required as PKCE was used in the authorization code grant request.

Table 5: Token Endpoint – Request Parameters

The following is a non-normative example of a Token Request:

```
POST /token HTTP/1.1
Host: issuer.govt.nz
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

 grant_type=authorization_code
 &code=SplxlOBeZQQYbYS6WxSbIA
 &code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
 &redirect_uri=myapp://callback
```

Token Endpoint Successful Response

After receiving and validating a valid and authorized Token Request from the Holder App, the Issuer Token Endpoint returns a successful response that includes an access token.

Response Parameter	Description
access_token	Access token issued for Credential Endpoint
token_type	The value must be Bearer
expires_in	Token expiry time in milliseconds

Table 6: Token Endpoint Successful Response

Token Endpoint Unsuccessful Response

The following table provides key error codes and descriptions:

Error Code	Error Description
request_not_supported	Token endpoint does not support the use of the request parameter supplied in the authentication request.
invalid_request	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed.
server_error	Token endpoint encountered an unexpected condition that prevented it from fulfilling the request.

Table 7: Error Codes and Error Descriptions

OIDC Credential Endpoint

Credential Request

The VC Issuance Service Credential Endpoint issues Verifiable Credential(s) to the Holder App as approved by the user upon presentation of a valid Access Token representing the approval. The endpoint issues the following verifiable credentials, and each credential must be requested separately:

- IdentityNameCredential
- IdentityDoBCredential
- IdentityPoBCredential
- IdentityGenderCredential
- IdentityPhotoCredential
- IdentityOver18Credential

Communication with the Credential Endpoint MUST utilize TLS.

The following table provides Credential Request parameters:

Parameter	Mandatory/ Optional	Description
Format	Required	JSON string representing the format in which the Credential is requested to be issued. This Credential format identifier determines further claims in the authorization details object specifically used to identify the Credential type to be issued. Must be set to jwt_vc_json or ldp_vc.
credential_ definition	Required	It is required for ldp_vc format. Refer to the table 9 for more details.
proof	Required	It must be Signed JWT. The following are the key parameters need to be passed: • "proof_type": "jwt", • "jwt": refer to the table 8 for more details.

Table 8: Credential Request Parameters

The following table provides credential definition parameters.

Parameter	Mandatory/ Optional	Description
@context	Required	It must be ["https://www.w3.org/2018/credentials/v1", "https://www.dia.govt.nz/2023/credentials/identitypoc/v1"]
types	Required	It is an array of stings and one of the values can be: • ["VerifiableCredential", "IdentityNameCredential"] or • ["VerifiableCredential", "IdentityDoBCredential"] or • ["VerifiableCredential", "IdentityPoBCredential"] or • ["VerifiableCredential", IdentityGenderCredential"] or • ["VerifiableCredential", "IdentityPhotoCredential"] or • ["VerifiableCredential", "IdentityOver18Credential"]

Table 9: Credential Definition Parameters

The following table provides required elements in signed JWT as a proof for VC Issuance Service Credential Endpoint.

JWT Claim	Header/ Body	Description
typ	JOSE Header	Must be openid4vci-proof+jwt
alg	JOSE Header	Must be ES256
jwk	JOSE Header	Must be containing the public key (i.e. did:jwk) material the new credential shall be bound to, i.e. Holder App.
iss	Body	did:jwk: {publickey}
aud	Body	The value of this claim MUST be the Credential Issuer URL of the Credential Issuer.
iat	Body	The value of this claim MUST be the time at which the proof was issued.
nonce	Body	The value type of this claim MUST be a string, where the value is a c_nonce provided by the Credential Issuer.

Table 10: Credential Request – Signed JWT

The following non-normative example of credential request:

```
POST /credential HTTP/1.1

Host: realme.govt.nz

Content-Type: application/json

Authorization: BEARER czZCaGRSa3F0MzpnWDFmQmF0M2JW


{

  "format":"ldp_vc ",

  "types":[“VerifiableCredential”, “IdentityNameCredential” ],

    "proof":{

         "proof_type":"jwt",

         "jwt":"eyJraWQiOiJkaWQ6ZXhhbXBsZTplYmZlYjFmNzEyZWJjNmYxYzI3NmUxMmVjMjEva2V5cy8

               xIiwiYWxnIjoiRVMyNTYiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJzNkJoZFJrcXQzIiwiYXVkIjoiaHR

               0cHM6Ly9zZXJ2ZXIuZXhhbXBsZS5jb20iLCJpYXQiOiIyMDE4LTA5LTE0VDIxOjE5OjEwWiIsIm5vbm

               NlIjoidFppZ25zbkZicCJ9.ewdkIkPV50iOeBUqMXCC_aZKPxgihac0aW9EkL1nOzM"

       }

}
```

Successful Credential Response

After receiving and validating a valid Credential Request from the Holder App, the Issuer Credential Endpoint returns a successful response that includes a Credential. The following table provides successful response parameters:

Parameter	Mandatory/ Optional	Description
format	Required	JSON string representing the format in which the Credential is requested to be issued. Must be set to jwt_vc_json or ldp_vc.
credential	Required	Contains issued Credential. MUST be present. It is one of the Identity Credentials.

Table 11: Successful Credential Response

The following non-normative example of successful credential response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
{
  "format": "ldp_vc",
  "credential": {
      Refer to Credential Section for Strucure.
  }
}
```

Unsuccessful Credential Response

The following table provides key error codes and descriptions:

Error Code	Error Description
invalid_request	Credential Request was malformed. One or more of the parameters (i.e. format, proof) are missing or malformed.
invalid_token	Credential Request contains the wrong Access Token, or the Access Token is missing.
unsupported_credential_type	requested credential type is not supported.
unsupported_credential_format	requested credential format is not supported.
invalid_or_missing_proof	Credential Request did not contain a proof, or proof was invalid.

Table 12: Error Codes and Descriptions

The following non-normative example of error response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store
{
  "error": "invalid_request"
}
```

Example Verifiable Credentials with Identity Credentials

Identity Name Credential Example

```
{
                        '@context': [
                                'https://www.w3.org/2018/credentials/v1',
                                'https://w3id.org/security/suites/jws-2020/v1', {
                                     'Date_of_Birth': 'ex:Date_of_Birth',
                                      'Gender': 'ex:Gender',
                                     'IdentityGenderCredential': 'ex:IdentityGenderCredential',
                                     'IdentityNameCredential': 'ex:IdentityNameCredential',
                                     'IdentityPhotoCredential': 'ex:IdentityPhotoCredential',
                                      'IdentityPoBCredential': 'ex:IdentityPoBCredential',
                                      'Over18': 'ex:Over18',
                                      'Photo': 'ex:Photo',
                                     'Place_of_Birth': 'ex:Place_of_Birth',
                                      'ex': 'https://realme.govt.nz/.wellknownendpoint/schema#ex',
                                      'format': 'ex:format',
                                      'givennames': 'ex:givennames',
                                      'identity': 'ex:identity',
                                       'surname': 'ex:surname'
                            }], // end of credential context
                        'credentialSubject': {
                                    "id":
"did:jwk:eyJjcnYiOiJQLTI1NiIsImt0eSI6IkVDIiwieCI6ImFjYklRaXVNczNpOF91c3pFakoydHBUdFJNN
EVVM3l6OTFQSDZDZEgyVjAiLCJ5IjoiX0tjeUxqOXZXTXB0bm1LdG00NkdxRHo4d2Y3NEk1TEtncmw
yR3pIM25TRSJ9",
                                'identity': {
                                        'givennames': 'Joe',
                    'surname': 'blogs'
                }
            },
            'id': 1c226e2c-0e43-43e7-b5d0-6f6c13acd0df',
            'issuanceDate': '2023-08-20T15:23:12',
            'issuer': 'did:web:realme.govt.nz',
            'proof': {
                'created': '2023-08-19T22:55:31',
                'jws':
'eyJiNjQiOiBmYWxzZSwgImNyaXQiOiBbImI2NCJdLCAiYWxnIjogIkVTMjU2In0..TbFwNTqMP6zm2k9_MzK_
DcdwivCq39LB7_UDXwh8DOZDJXhMWMPX5OnLYXLR_Xv8kqdui8uzS52gJu21Sb3hUg',
                'proofPurpose': 'assertionMethod',
                'type': 'JsonWebSignature2020',
                'verificationMethod': 'did:web:realme.govt.nz#key1'
            },
            'type': ['IdentityNameCredential', 'VerifiableCredential']
        }
```

Identity Date of Birth Credential Example

```
{
            '@context': [
                'https://www.w3.org/2018/credentials/v1',
                'https://w3id.org/security/suites/jws-2020/v1', {
                    'Date_of_Birth': 'ex:Date_of_Birth',
                     'Gender': 'ex:Gender',
                    'IdentityGenderCredential': 'ex:IdentityGenderCredential',
                    'IdentityNameCredential': 'ex:IdentityNameCredential',
                    'IdentityPhotoCredential': 'ex:IdentityPhotoCredential',
                     'IdentityPoBCredential': 'ex:IdentityPoBCredential',
                     'Over18': 'ex:Over18',
                     'Photo': 'ex:Photo',
                    'Place_of_Birth': 'ex:Place_of_Birth',
                     'ex': 'https://realme.govt.nz/.wellknownendpoint/schema#ex',
                     'format': 'ex:format',
                     'givennames': 'ex:givennames',
                        'identity': 'ex:identity',
                         'surname': 'ex:surname'
                    }], // end of credential context
                'credentialSubject': {
                        "id":
"did:jwk:eyJjcnYiOiJQLTI1NiIsImt0eSI6IkVDIiwieCI6ImFjYklRaXVNczNpOF91c3pFakoydHBUdFJNN
EVVM3l6OTFQSDZDZEgyVjAiLCJ5IjoiX0tjeUxqOXZXTXB0bm1LdG00NkdxRHo4d2Y3NEk1TEtncmw
yR3pIM25TRSJ9",
                'identity': {
                         'Date_of_Birth': ‘1990-01-01’,
                         'format': 'YYYY-MM-DD'
                    }
                },
                'id': 1c226e2c-0e43-43e7-b5d0-6f6c13acd0df',
                'issuanceDate': '2023-08-20T15:23:12',
                'issuer': 'did:web:realme.govt.nz',
                'proof': {
                    'created': '2023-08-19T22:55:31',
                    'jws':
'eyJiNjQiOiBmYWxzZSwgImNyaXQiOiBbImI2NCJdLCAiYWxnIjogIkVTMjU2In0..TbFwNTqMP6zm2k9_MzK_
DcdwivCq39LB7_UDXwh8DOZDJXhMWMPX5OnLYXLR_Xv8kqdui8uzS52gJu21Sb3hUg',
                    'proofPurpose': 'assertionMethod',
                    'type': 'JsonWebSignature2020',
                    'verificationMethod': 'did:web:realme.govt.nz#key1'
                },
                'type': [' IdentityDoBCredential', 'VerifiableCredential']
            }
```

Identity Place of Birth Credential Example

```
{
                '@context': [
                    'https://www.w3.org/2018/credentials/v1',
                    'https://w3id.org/security/suites/jws-2020/v1', {
                        'Date_of_Birth': 'ex:Date_of_Birth',
                         'Gender': 'ex:Gender',
                        'IdentityGenderCredential': 'ex:IdentityGenderCredential',
                        'IdentityNameCredential': 'ex:IdentityNameCredential',
                        'IdentityPhotoCredential': 'ex:IdentityPhotoCredential',
                         'IdentityPoBCredential': 'ex:IdentityPoBCredential',
                         'Over18': 'ex:Over18',
                         'Photo': 'ex:Photo',
                        'Place_of_Birth': 'ex:Place_of_Birth',
                         'ex': 'https://realme.govt.nz/.wellknownendpoint/schema#ex',
                         'format': 'ex:format',
                         'givennames': 'ex:givennames',
                         'identity': 'ex:identity',
                          'surname': 'ex:surname'
                  }], // end of credential context
                'credentialSubject': {
                        "id":
"did:jwk:eyJjcnYiOiJQLTI1NiIsImt0eSI6IkVDIiwieCI6ImFjYklRaXVNczNpOF91c3pFakoydHBUdFJNN
EVVM3l6OTFQSDZDZEgyVjAiLCJ5IjoiX0tjeUxqOXZXTXB0bm1LdG00NkdxRHo4d2Y3NEk1TEtncmw
yR3pIM25TRSJ9",
                'identity': {
                          'Place_of_Birth': 'Wellington, New Zealand'
                    }
                },
                'id': 1c226e2c-0e43-43e7-b5d0-6f6c13acd0df',
                'issuanceDate': '2023-08-20T15:23:12',
                'issuer': 'did:web:realme.govt.nz',
                'proof': {
                    'created': '2023-08-19T22:55:31',
                    'jws':
'eyJiNjQiOiBmYWxzZSwgImNyaXQiOiBbImI2NCJdLCAiYWxnIjogIkVTMjU2In0..TbFwNTqMP6zm2k9_MzK_
DcdwivCq39LB7_UDXwh8DOZDJXhMWMPX5OnLYXLR_Xv8kqdui8uzS52gJu21Sb3hUg',
                    'proofPurpose': 'assertionMethod',
                    'type': 'JsonWebSignature2020',
                    'verificationMethod': 'did:web:realme.govt.nz#key1'
                },
                'type': [' IdentityPoBCredential', 'VerifiableCredential']
           }
```

Identity Gender Credential Example

```
{
                    '@context': [
                           'https://www.w3.org/2018/credentials/v1',
                           'https://w3id.org/security/suites/jws-2020/v1', {
                                'Date_of_Birth': 'ex:Date_of_Birth',
                                 'Gender': 'ex:Gender',
                                'IdentityGenderCredential': 'ex:IdentityGenderCredential',
                                'IdentityNameCredential': 'ex:IdentityNameCredential',
                                'IdentityPhotoCredential': 'ex:IdentityPhotoCredential',
                                 'IdentityPoBCredential': 'ex:IdentityPoBCredential',
                                 'Over18': 'ex:Over18',
                                 'Photo': 'ex:Photo',
                                'Place_of_Birth': 'ex:Place_of_Birth',
                                 'ex': 'https://realme.govt.nz/.wellknownendpoint/schema#ex',
                                 'format': 'ex:format',
                                 'givennames': 'ex:givennames',
                                 'identity': 'ex:identity',
                                  'surname': 'ex:surname'
                       }], // end of credential context
                    'credentialSubject': {
                              "id":
"did:jwk:eyJjcnYiOiJQLTI1NiIsImt0eSI6IkVDIiwieCI6ImFjYklRaXVNczNpOF91c3pFakoydHBUdFJNN
EVVM3l6OTFQSDZDZEgyVjAiLCJ5IjoiX0tjeUxqOXZXTXB0bm1LdG00NkdxRHo4d2Y3NEk1TEtncmw
yR3pIM25TRSJ9",
                    'identity': {
                                  'Gender': 'Male'
                           }
                    },
                    'id': 1c226e2c-0e43-43e7-b5d0-6f6c13acd0df',
                    'issuanceDate': '2023-08-20T15:23:12',
                    'issuer': 'did:web:realme.govt.nz',
                    'proof': {
                           'created': '2023-08-19T22:55:31',
                                'jws':
'eyJiNjQiOiBmYWxzZSwgImNyaXQiOiBbImI2NCJdLCAiYWxnIjogIkVTMjU2In0..TbFwNTqMP6zm2k9_MzK_
DcdwivCq39LB7_UDXwh8DOZDJXhMWMPX5OnLYXLR_Xv8kqdui8uzS52gJu21Sb3hUg',
                                'proofPurpose': 'assertionMethod',
                                'type': 'JsonWebSignature2020',
                                'verificationMethod': 'did:web:realme.govt.nz#key1'
                        },
                        'type': [' IdentityGenderCredential', 'VerifiableCredential']
                }
```

Identity Over18 Credential Example

```
{
                        '@context': [
                                'https://www.w3.org/2018/credentials/v1',
                                'https://w3id.org/security/suites/jws-2020/v1', {
                                    'Date_of_Birth': 'ex:Date_of_Birth',
                                     'Gender': 'ex:Gender',
                                    'IdentityGenderCredential': 'ex:IdentityGenderCredential',
                                    'IdentityNameCredential': 'ex:IdentityNameCredential',
                                    'IdentityPhotoCredential': 'ex:IdentityPhotoCredential',
                                     'IdentityPoBCredential': 'ex:IdentityPoBCredential',
                                     'Over18': 'ex:Over18',
                                     'Photo': 'ex:Photo',
                                    'Place_of_Birth': 'ex:Place_of_Birth',
                                     'ex': 'https://realme.govt.nz/.wellknownendpoint/schema#ex',
                                     'format': 'ex:format',
                                     'givennames': 'ex:givennames',
                                     'identity': 'ex:identity',
                                      'surname': 'ex:surname'
                           }], // end of credential context
                        'credentialSubject': {
                                   "id":
"did:jwk:eyJjcnYiOiJQLTI1NiIsImt0eSI6IkVDIiwieCI6ImFjYklRaXVNczNpOF91c3pFakoydHBUdFJNN
EVVM3l6OTFQSDZDZEgyVjAiLCJ5IjoiX0tjeUxqOXZXTXB0bm1LdG00NkdxRHo4d2Y3NEk1TEtncmw
yR3pIM25TRSJ9",
                    'identity': {
                                'Over18': 'true'
                          }
                    },
                    'id': 1c226e2c-0e43-43e7-b5d0-6f6c13acd0df',
                    'issuanceDate': '2023-08-20T15:23:12',
                    'issuer': 'did:web:realme.govt.nz',
                    'proof': {
                          'created': '2023-08-19T22:55:31',
                          'jws':
'eyJiNjQiOiBmYWxzZSwgImNyaXQiOiBbImI2NCJdLCAiYWxnIjogIkVTMjU2In0..TbFwNTqMP6zm2k9_MzK_
DcdwivCq39LB7_UDXwh8DOZDJXhMWMPX5OnLYXLR_Xv8kqdui8uzS52gJu21Sb3hUg',
                          'proofPurpose': 'assertionMethod',
                          'type': 'JsonWebSignature2020',
                          'verificationMethod': 'did:web:realme.govt.nz#key1'
                    },
                    'type': [' IdentityOver18Credential', 'VerifiableCredential']
              }
```

Identity Photo Credential Example

```
{
                    '@context': [
                          'https://www.w3.org/2018/credentials/v1',
                          'https://w3id.org/security/suites/jws-2020/v1', {
                              'Date_of_Birth': 'ex:Date_of_Birth',
                               'Gender': 'ex:Gender',
                              'IdentityGenderCredential': 'ex:IdentityGenderCredential',
                              'IdentityNameCredential': 'ex:IdentityNameCredential',
                              'IdentityPhotoCredential': 'ex:IdentityPhotoCredential',
                               'IdentityPoBCredential': 'ex:IdentityPoBCredential',
                               'Over18': 'ex:Over18',
                               'Photo': 'ex:Photo',
                              'Place_of_Birth': 'ex:Place_of_Birth',
                               'ex': 'https://realme.govt.nz/.wellknownendpoint/schema#ex',
                               'format': 'ex:format',
                        'givennames': 'ex:givennames',
                        'identity': 'ex:identity',
                        'surname': 'ex:surname'
                }], // end of credential context
            'credentialSubject': {
                    "id":
"did:jwk:eyJjcnYiOiJQLTI1NiIsImt0eSI6IkVDIiwieCI6ImFjYklRaXVNczNpOF91c3pFakoydHBUdFJNN
EVVM3l6OTFQSDZDZEgyVjAiLCJ5IjoiX0tjeUxqOXZXTXB0bm1LdG00NkdxRHo4d2Y3NEk1TEtncmw
yR3pIM25TRSJ9",
            'identity': {
                        'photo': 'base64 encoded bytes',
                        'format': 'JPG'
                }
            },
            'id': 1c226e2c-0e43-43e7-b5d0-6f6c13acd0df',
            'issuanceDate': '2023-08-20T15:23:12',
            'issuer': 'did:web:realme.govt.nz',
            'proof': {
                'created': '2023-08-19T22:55:31',
                'jws':
'eyJiNjQiOiBmYWxzZSwgImNyaXQiOiBbImI2NCJdLCAiYWxnIjogIkVTMjU2In0..TbFwNTqMP6zm2k9_MzK_
DcdwivCq39LB7_UDXwh8DOZDJXhMWMPX5OnLYXLR_Xv8kqdui8uzS52gJu21Sb3hUg',
                'proofPurpose': 'assertionMethod',
                'type': 'JsonWebSignature2020',
                'verificationMethod': 'did:web:realme.govt.nz#key1'
            },
            'type': [' IdentityOver18Credential', 'VerifiableCredential']
    }
```

VC Presentation Cross Device – Proposed OIDC Message Specification

Note: The proposed message specification may require an update based on feedback from the verifiers and the Market Participants.

SIOP Authorisation Request

Verifier digital service creates a QR code with SIOP Authorisation request. The customer scans QR code to retrieve authorisation parameters. The OIDC Authorisation Request contains the following parameters:

Parameter	Description
scope	The scope must contain: • openid The scope should contain one of these: • IdentityNameCredential • IdentityDoBCredential • IdentityPoBCredential • IdentityGenderCredential • IdentityPhotoCredential • IdentityOver18Credential
response_type	Response Type value that determines the authorization processing flow to be used, including what parameters are returned from the endpoints used. When using the authorisation code flow, this value is id_token.
redirect_uri	Redirection URI to where the response will be sent, and the redirection URI MUST use the https scheme.
nonce	String value used to associate a verifier session with an ID Token, and to mitigate replay attacks. The value is passed through unmodified from the Authentication Request to the ID Token.
id_token_type	The types of ID Token the verifier want to obtain and the value must be subject_signed.
response_mode	Must be “post”.
client_metadata	Must be {"subject_syntax_types_supported":["did:jwk22"], "id_token_signed_response_alg":"ES256"}

Table 13: SIOP Request

```
GET siopv2://authorize?
   response_type=id_token
   &scope=openid, IdentityNameCredential,IdentityDoBCredential
   &id_token_type=subject_signed
   &client_id=https%3A%2F%2Fverifier.govt.nz%2Fcb
   &response_mode=post
   &redirect_uri=https%3A%2F%2Fverifier.govt.nz%2Fcb
   &nonce=n-0S6_WzA2Mj HTTP/1.1
  &client_metadata= %7B%22subject_syntax_types_supported%22%3A
   %5Bdid:jwk22%5D%2C%0A%20%20%20%20
   %22id_token_signed_response_alg%22%3A%22ES256%22%7D
```

SIOP Response

On successful validation of OIDC authentication request and user consent, the Holder App creates an ID Token and VP Token with verifiable credentials as claims. The following table describes the response parameters:

Parameter	Description
id_token	Holder App access as Self Issued Open ID Connect Provider and issues id_token to the verifier. Refer to the table 15 for its structure.
vp_token	JSON String or JSON object that MUST contain a single Verifiable Presentation or an array of JSON Strings and JSON objects each of them containing a Verifiable Presentations. Each Verifiable Presentation MUST be represented as a JSON string (that is a Base64url encoded value) or the format must be ldp_vp. Refer to the VP Token section for details.
presentation_submission	This is expressed via elements in the descriptor_map array, known as Input Descriptor Mapping Objects.

	These objects contain a field called path, which, for this specification, MUST have the value $ (top level root path) when only one Verifiable Presentation is contained in the VP Token, and MUST have the value $[n] (indexed path from root) when there are multiple Verifiable Presentations, where n is the index to select. The path_nested object inside an Input Descriptor Mapping Object is used to describe how to find a returned Credential within a Verifiable Presentation, and the value of the path field in it will ultimately depend on the credential format.

Table 14: SIOPV2 Response – Verifiable Token Response

The following is non-normative example of response.

```
HTTP/1.1 302 Found
  Location: https://verifier.govt.nz/cb#
    id_token=
    &presentation_submission=...
    &vp_token=...
```

The following table describes SIOPv2 id_token response:

ID Tpken Claim	Header/ Body	Description
typ	JOSE Header	Must be JWT
alg	JOSE Header	Must be ES256
jwk	JOSE Header	Must be containing the public key (i.e. did:jwk) material the new credential shall be bound to, i.e. Holder App.
iss	Body	Holder App's did and it must be did:jwk: {publickey}.
sub	Body	Holder App's did and it must be did:jwk: {publickey}.
aud	Body	The value of this claim MUST be the verifier's client id.
iat	Body	The value of this claim MUST be the time at which the proof was issued.
nonce	Body	The value type of this claim MUST be a string, it is same values as authentication request.

Table 15: SIOPV2 ID Token Response

The following is non-normative example:

```
{// header
  "jwk": {
    "kty": "EC",
    "crv": "P-256",
    "x": "u2AjxKaEh0dtsFPJQr5oiCceGtEW5UbIw0AmRwhMVRU",
    "y": "WfUkDxPprn-ZuW1WOsJyfp7-YgHkPCymdUJp2UrpJuw"
  },
  "alg": "ES256",
  "typ": "jwt"
}.
{// body
  "iss":
"did:jwk:eyJrdHkiOiJFQyIsImNydiI6IlAtMjU2IiwieCI6InUyQWp4S2FFaDBkdHNGUEpRcjVvaUNjZUd0RVc1VWJJdzBBbVJ3aE1WUlUiLCJ5IjoiV2ZVa0R4UHBybi1adVcxV09zSnlmcDctWWdIa1BDeW1kVUpwMlVycEp1dyJ9",
  "sub":
"did:jwk:eyJrdHkiOiJFQyIsImNydiI6IlAtMjU2IiwieCI6InUyQWp4S2FFaDBkdHNGUEpRcjVvaUNjZUd0RVc1VWJJdzBBbVJ3aE1WUlUiLCJ5IjoiV2ZVa0R4UHBybi1adVcxV09zSnlmcDctWWdIa1BDeW1kVUpwMlVycEp1dyJ9",
  "aud": "https://verifier.govt.nz/cb",
  "nonce": "117fbc5e-a869-48a6-9f52-c4c9296db47f"
}.
{// signature
  "kty": "EC",
  "crv": "P-256",
  "x": "u2AjxKaEh0dtsFPJQr5oiCceGtEW5UbIw0AmRwhMVRU",
  "y": "WfUkDxPprn-ZuW1WOsJyfp7-YgHkPCymdUJp2UrpJuw"
}
```

The following is the presentation_submission normative example:

```
        'presentation_submission': {
                'definition_id': 'Identity Name and Identity DoB ldp_vc',
                'id': 'identityname_dob_ldp_vc_presentation_submission',
                'descriptor_map': [{
                        'id': 'id_name_credential',
                        'path': '$',
                        'format': 'ldp_vp',
                        'path_nested': {
                                'format': 'ldp_vc',
                                'path': '$.verifiableCredential[0]'
                        }
                }, {
                        'id': 'id_dob_credential',
                        'path': '$',
                        'format': 'ldp_vp',
                        'path_nested': {
                                'format': 'ldp_vc',
                                'path': '$.verifiableCredential[1]'
                        }
                }]
        }
```

The following is normative example of VP_token:

```
'vp_token': {
                '@context': ['https://www.w3.org/2018/credentials/v1',
'https://w3id.org/security/suites/jws-2020/v1'],
                'type': ['VerifiablePresentation'],
                'verifiableCredential': [{
                        '@context': ['https://www.w3.org/2018/credentials/v1',
'https://w3id.org/security/suites/jws-2020/v1', {
                                'Date_of_Birth': 'ex:Date_of_Birth',
                                'Gender': 'ex:Gender',
                                'IdentityGenderCredential': 'ex:IdentityGenderCredential',
                                'IdentityNameCredential': 'ex:IdentityNameCredential',
                                'IdentityPhotoCredential': 'ex:IdentityPhotoCredential',
                                'IdentityPoBCredential': 'ex:IdentityPoBCredential',
                                'Over18': 'ex:Over18',
                                'Photo': 'ex:Photo',
                                'Place_of_Birth': 'ex:Place_of_Birth',
                                'ex': 'https://oti-mattr-
bridge.australiasoutheast.cloudapp.azure.com/api/vocab#ex',
                                'format': 'ex:format',
                                'givennames': 'ex:givennames',
                                'identity': 'ex:identity',
                                'surname': 'ex:surname'
                        }],
                        'credentialSubject': {
                                'id': '
did:jwk:eyJrdHkiOiJFQyIsImNydiI6IlAtMjU2IiwieCI6InUyQWp4S2FFaDBkdHNGUEpRcjVvaUNjZUd0RVc1V
WJJdzBBbVJ3aE1WUlUiLCJ5IjoiV2ZVa0R4UHBybi1adVcxV09zSnlmcDctWWdIa1BDeW1kVUpwMlVycEp1d
yJ9',
                                'identity': {
                                     'givennames': 'Joe',
                                     'surname': 'Blogs'
                                }
                        },
                        'id': '1c226e2c-0e43-43e7-b5d0-6f6c13acd0df',
                        'issuanceDate': '2023-08-20T15:23:12',
                        'issuer': 'did:web:realme.govt.nz',
                        'proof': {
                                'created': '2023-08-19T22:55:31',
                                'jws':
'eyJiNjQiOiBmYWxzZSwgImNyaXQiOiBbImI2NCJdLCAiYWxnIjogIkVTMjU2In0..TbFwNTqMP6zm2k9_MzK_
DcdwivCq39LB7_UDXwh8DOZDJXhMWMPX5OnLYXLR_Xv8kqdui8uzS52gJu21Sb3hUg',
                                'proofPurpose': 'assertionMethod',
                                'type': 'JsonWebSignature2020',
                                'verificationMethod': 'did:web:oti-mattr-
bridge.australiasoutheast.cloudapp.azure.com#key1'
                        },
                        'type': ['IdentityNameCredential', 'VerifiableCredential']
```

```
        }, {
            '@context': ['https://www.w3.org/2018/credentials/v1',
'https://w3id.org/security/suites/jws-2020/v1', {
                'Date_of_Birth': 'ex:Date_of_Birth',
                'Gender': 'ex:Gender',
                'IdentityGenderCredential': 'ex:IdentityGenderCredential',
                'IdentityNameCredential': 'ex:IdentityNameCredential',
                'IdentityPhotoCredential': 'ex:IdentityPhotoCredential',
                'IdentityPoBCredential': 'ex:IdentityPoBCredential',
                'Over18': 'ex:Over18',
                'Photo': 'ex:Photo',
                'Place_of_Birth': 'ex:Place_of_Birth',
                'ex': 'https://oti-mattr-bridge.australiasoutheast.cloudapp.azure.com#vocab',
                'format': 'ex:format',
                'givennames': 'ex:givennames',
                'identity': 'ex:identity',
                'surname': 'ex:surname'
            }],
            'credentialSubject': {
                'id': '
did:jwk:eyJrdHkiOiJFQyIsImNydiI6IlAtMjU2IiwieCI6InUyQWp4S2FFaDBkdHNGUEpRcjVvaUNjZUd0RVc1V
WJJdzBBbVJ3aE1WUlUiLCJ5IjoiV2ZVa0R4UHBybi1adVcxV09zSnlmcDctWWdIa1BDeW1kVUpwMlVycEp1d
yJ9',
                'identity': {
                    'Date_of_Birth': '1990-01-01',
                    'format': 'YYYY-MM-DD'
                }
            },
            'id': 1c226e2c-0e43-43e7-b5d0-6f6c13acd0df',
            'issuanceDate': '2023-08-20T15:23:15',
            'issuer': 'did:web:realme.govt.nz',
            'proof': {
                'created': '2023-08-19T22:55:31',
                'jws':
'eyJiNjQiOiBmYWxzZSwgImNyaXQiOiBbImI2NCJdLCAiYWxnIjogIkVTMjU2In0..-
r0TPhdqXKFMziq4mMegJBOuM3bicduetwjkSqRgrZOXwu314LQmD6tSRdmyuXlNfdk0gp3KZEhacTBSPXwY
8Q',
                                        'proofPurpose': 'assertionMethod',
                                        'type': 'JsonWebSignature2020',
                                        'verificationMethod': 'did:web:oti-mattr-
bridge.australiasoutheast.cloudapp.azure.com#key1'
                              },
                              'type': ['IdentityDoBCredential', 'VerifiableCredential']
                    } ],
                    'id': '1234568900',
                    'holder':
'did:jwk:eyJrdHkiOiJFQyIsImNydiI6IlAtMjU2IiwieCI6ImktUHotV3pxSUtZR1hfbHhvS05DWHhTVVJkemdHRk
RiSGZIeXpJUTZjRUUiLCJ5IjoiRUwyamJ6OUtrZTZkX29NejV4UEhWNWh1R0hpMzlLUHNySjlFa1JiZ3M0OCJ9'
,
                    'proof': {
                              'type': 'JsonWebSignature2020',
                              'created': '2023-08-21T07:47:14.591130',
                              'challenge': '531ade20-c978-43e0-8030-441ba8b9f41d',
                              'domain': 'https://verifier.govt.nz/cb',
                              'proofPurpose': 'authentication',
                              'verificationMethod':
'did:jwk:eyJrdHkiOiJFQyIsImNydiI6IlAtMjU2IiwieCI6ImktUHotV3pxSUtZR1hfbHhvS05DWHhTVVJkemdHRk
RiSGZIeXpJUTZjRUUiLCJ5IjoiRUwyamJ6OUtrZTZkX29NejV4UEhWNWh1R0hpMzlLUHNySjlFa1JiZ3M0OCJ9'
,
                              'jws':
'eyJiNjQiOmZhbHNlLCJjcml0IjpbImI2NCJdLCJhbGciOiJFUzI1NiJ9..w4K2o9RnoSxB8XnUvseW5GbIVfnfpqTV
gj3vj5xijrgOxrt2sg4dbMdOPyem-A63uiTCeKuUIqGfgk6OY0XiJA'
                    }
          }
```

What Next?

Next Steps

The Department of Internal Affairs will:

- Continue to engage partner agencies regarding the technical white paper and get their feedback.
- Progress Proof of Concept opportunities with partner agencies to test and refine/revise the technical approach.
- Continue refining and understanding our approach in preparation for implementation.