Issuing Credentials

Suggest Edits

Identity providers can issue portable and decentralized Verifiable Credentials to their users. Verifiable Credentials represent provable claims associated with an identity. They can represent any number of claims -- proof of KYC, credit score, etc.

Verifiable Credentials are issued to a recipient-controlled identifier (such as a DID), which increases the ability for the recipient to use the credential in different contexts throughholder/subject binding techniques that prove control over an identifier.

Collecting that identifier in a secure and convenient way for the user requires different interaction flows; the following pattern is one way an issuer can enable this process for an authenticated user.

Issuance Flow

In this example, an individual's financial institution offers the ability to request a KYC or credit score credential to be held in the individual's mobile identity wallet. The flow consists of the exchange of three distinct JSON objects:

1. A Credential Offer object published by the Issuer that describes the credential and how to apply for it
2. A Credential Application object submitted by the wallet, containing information that, when appropriately validated and/or verified, will form the contents of those credentials, and
3. A Credential Fulfillment envelope returned by the Issuer that contains the final credential(s) to be stored by the wallet and presented later to unknown future verifiers.

Sample JSON objects

- Credential Offer
- Credential Manifest
- Credential Application
- Credential Fulfillment

Wallet-Bound Credential Issuance

In more detail, here is a flow for issuing a DID-bound (i.e., identity wallet-bound) credential to an identity wallet:

1. A User navigates to the Issuer site
2. The Issuer presents a QR code.
3. User scans the QR code with their wallet.
4. Wallet parses the QR code, which encodes a JSON object with a
5. challengeTokenUrl
6. property.
7. Wallet performs a GET request at that URL to return a Credential Offer, a wrapper around aCredential Manifest
8. , with three supplementary properties:* The issuer DID.
9.
    - A URL for the wallet to submit a Credential Application.
10.
    - A challenge to sign.
11. The wallet prompts the user to proceed. The Credential Manifest includes descriptive properties, e.g. in the Verite demo app a title and description of the credential are shown.
12. Once the recipient proceeds, the wallet prepares a signedCredential Application
13. ,* If the wallet doesn't have a DID, it generates one.
14.
    - The wallet creates a Credential Application for the DID.
15.
    - The application is signed along with the challenge in the Credential Offer
16.
    - The Verite library exposes a convenience method
17.
    - createCredentialApplication
18.
    - for this purpose.
19. Wallet submits the Credential Application to the URL found in the Credential Offer.
20. The Issuer creates a Verifiable Credential and returns it to the wallet as aCredential Fulfillment
21. .
22. Wallet persists the credential.

Address-Bound Credential Issuance

In the case of credentials bound to a blockchain address, issuance is necessarily bespoke to a given wallet implementation, as few cryptocurrency wallets have standardized verification credential capabilities or even standardized secure storage. In many use-cases, address-bound credentials are only recommended for addresses that are securely limited to a trusted wallet (whether hardware-secured, multi-signature, or EOA) whose signing control cannot be transfered or exfiltrated.

As wallet standards emerge, guidance and potentially even full interface specifications will be forthcoming for how wallets can receive VCs, store them, and sign over them in live presentations.

For now, Verite recommends creating verifiable and replayable receipts of all off-chain wallet interactions (including issuance and delegated signing/presentation operations). The Sign-In With Ethereum specification (EIP-4361) is one such way of harmonizing on off-chain behavior, an approach being extended to other VMs in theCASA byCAIP-122 . Receipts of these events can be logged asCACAO s, for uniform verifiability. Updated3 months ago *Table of Contents * * Issuance Flow * * * Sample JSON objects * * *Wallet-Bound Credential Issuance * * *Address-Bound Credential Issuance