Learner Credential Wallet Specification

Editors: Kim Hamilton Duffy, Ulrich Gallersdörfer, James Goodell, Matt Lisle, Brandon Muramatsu, Philipp Schmidt

This document constitutes deliverable Task 2, Sub-task 2.1 B as specified in the Department of Education contract "Develop Open Source Standard for Student Credential Wallet" between the Department of Education and MIT.

Introduction

Background

Digital wallets appear increasingly in our lives to handle a range of digital assets. These have expanded beyond currency; we already use digital wallets such as Apple Wallet or other applications to hold assets such as airline tickets; insurance cards; tickets to enter events such as concerts, movies, or conferences; and, in some cases, drivers licenses or identity cards such as student ID cards. There is also increased interest in digital wallets as a convenient way to store and manage digital learning or work records.

While digital wallets offer some conveniences in managing such a range of assets, lack of standardization and interoperability among wallets presents risks, including friction for users and issuers, reduced utility, and vendor lock-in.

Interoperability is the primary goal

This document is written to describe the design and implementation of the learner digital wallet ("wallet"), as outlined in "Develop Open Source Standard for Student Credential Wallet" deliverable of the US Department of Education OSS Wallet Application.

The approach outlined in this document builds on ongoing standards work in W3C, IEEE, and the Decentralized Identity Foundation focused on interoperable verifiable credentials, decentralized ecosystems enabling their secure exchange, and digital wallet standards enabling interoperability for credential holders. These design choices promote broad interoperability and relevance (even beyond learning and employment credentials).

Scope and Intended Audience

This specification describes the necessary wallet features and technical requirements enabling individuals to curate and present their learning and employment records to others -- for example, as applicants to educational programs or employers -- in an interoperable manner.

The specification is written for a technical audience seeking information about interoperable credential wallets. Credentials related to educational, training and professional development are the primary focus.

Aspects that are important to the overall goals of a learner-focused wallet but are beyond the scope of this document include security, detailed privacy, and UI/UX considerations. Relevant efforts are discussed in Intended Use and Next Steps.

Deliverables

The specification is the primary artifact fulfilling the "Develop Open Source Standard for Student Credential Wallet" deliverable. It describes:

1. Wallet functional requirements
2. Conceptual wallet flows supporting flexible use of relevant standards and data models
3. Foundational, extensible wallet design based on, and in support of, (2).

4. Design and implementation choices for wallet standards and credential data models for initial wallet application.
5. Interoperability requirements in sufficient detail to be implemented in software code
6. Overview of extensibility mechanisms

This effort supported other standardization efforts such as the Universal Wallet 2020 Interoperability Specification and Modeling Educational Verifiable Credentials. See Intended Use and Next Steps for a detailed list of those and other resources relating to the continued development of this project.

Concepts and Foundations

Important Concepts

Important concepts used in this document come from the W3C Verifiable Credentials (VC) Data Model[1]. Brief descriptions and clarifications in the scope of this document are provided below. See the VC Data Model and other normative references for full normative definitions.

Terms

Table 1: VC Data Model Terms

Term	**Definition**
Credential	A set of one or more claims made by an issuer.
Issuer	The entity asserting claims.
Subject	A thing about which claims are made. In this document, the subject is the student (or learner) possessing the wallet
Identifier	A way to refer unambiguously to an entity, used here in the same sense as in the VC data model.[2] The subject of a credential is identified with the `credentialSubject.id` property in a VC. The identifier data type in the VC data model is a Uniform Resource Identifier (URI).
Verifiable Credential (VC)	Tamper-evident credential with cryptographically verifiable authorship.
Presentation	Data derived from one or more VCs, issued by one or more issuers, that is shared with a specific relying party by the holder.

[1] https://www.w3.org/TR/vc-data-model/
[2] https://www.w3.org/TR/vc-data-model/#identifiers

Verifiable Presentation (VP)	Tamper-evident presentation with cryptographically verifiable authorship (cf. VC)
Holder	A role an entity might perform when possessing one or more VCs and generating presentations from them. While the VC holder is often the same entity as the VC subject, the VC allows these roles to differ. Examples where they differ: ● The VC subject is a minor; the VC holder is the minor’s guardian ● The VC issuer or some trusted party holds the VC on behalf of the subject In this document, the holder is almost always the same entity as the subject. Any exceptions are called out.
Verifier / Relying Party (RP)	A role an entity performs by receiving one or more VCs (or VP). The VC data model allows the verifier role to be distinct from the relying party, allowing for deployments where a relying party trusts another party to perform verification on their behalf. In this document, this distinction is not important, and the term *relying party* (abbreviated RP) is preferred for clarity.
Verify/Verification	The process whereby a relying party checks whether they will accept Verifiable Credential or Verifiable Presentation. Unless specified otherwise, this is shorthand for the complete acceptance process described in Verification, Validation, and Veracity.

Table 2: Other Terms and Concepts

Term	**Definition**
Learner	Individual who acquires skills and capabilities throughout their life, such as in traditional or non-traditional learning contexts or employment contexts. *Learner* is the preferred shorthand term in this document to refer to the individual holding the wallet.
Learning and Employment Records (LERs)	Digital record of learning and work that can be linked to an individual and combined with other digital records for use in pursuing educational and employment opportunities. Use of *LER* in this document refers specifically to the format described in the LER Wrapper and Wallet Specification[3], which is a specific kind of VC.
Record; learning and employment record “ler” (note use of lower case)	Records that may or may not conform to the VC data model (including the LER wrapper specification), yet may be handled by the wallet. These other records may serve as a proxy for credential-like employment experience. Note use of lower-case *ler* to distinguish from LER-conformant records. A diagram depicting these terms is in Figure 1.
Decentralized Identifier	A special type of identifier that enables verifiable, decentralized digital identity, as defined in Decentralized Identifiers (DIDs) v1.0

[3] https://cdn.filestackcontent.com/preview/FeqEJI3S5KelmLv8XJss

Identity Proofing	The process with the goal of answering the question “Does the record apply to a known person?”[4].
Holder and Subject Binding	Refers to aspects of identity proofing as it relates to the VC data model, summarized from Presentation Exchange v1.0.0 “Holder and Subject Binding”[5] and LER Wrapper and Wallet Specification Annex D “Verification and Authentication Details”[6] Definition: The process of allowing a relying party to establish that a credential (or credentials) are bound to a specific holder or subject Purpose: Before accepting a credential or presentation as legitimate, a relying party may wish to confirm that it is bound to the party presenting it. This may be achieved through proof of control over an identifier, knowledge of a secret value, or biometrics.
Proof of Identifier Control	A specific mechanism for holder and subject binding, achieved through providing cryptographic proof of control of the identifier embedded in the VC or VP[7]. This approach to holder and subject binding is used in this document, but it is not intended to be restrictive. Note: the LER Wrapper and Wallet Specification uses the term *identifier authentication* for this concept. An illustration of how this is achieved is included in Annex F: Proof of Identifier Control, Illustration.

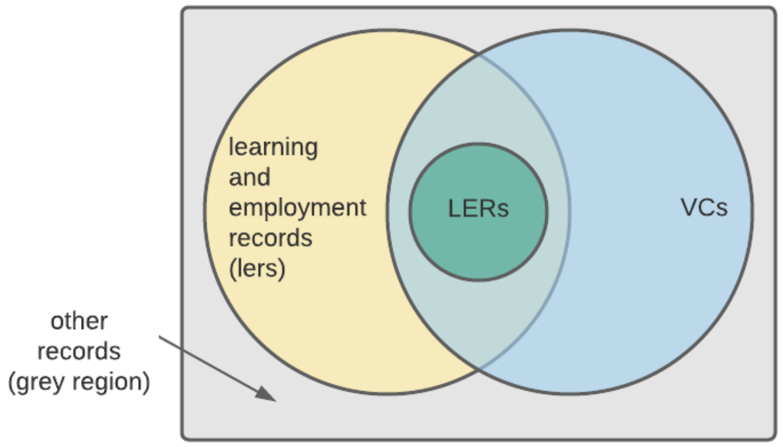


Figure 1: Diagram of key credential-related terminology in this document. An LER is a specific kind of VC[8].

[4] LER Wrapper and Wallet Specification, ANNEX D – VERIFICATION AND AUTHENTICATION
[5] https://identity.foundation/presentation-exchange/#holder-and-subject-binding
[6] https://cdn.filestackcontent.com/preview/FeqEJI3S5KelmLv8XJss
[7] https://identity.foundation/presentation-exchange/#proof-of-identifier-control
[8] The wallet may store or handle (per implementor extensions) other records. Lower-case *ler* refers to records related to learning and employment that do not follow the VC specification

Roles and Ecosystem

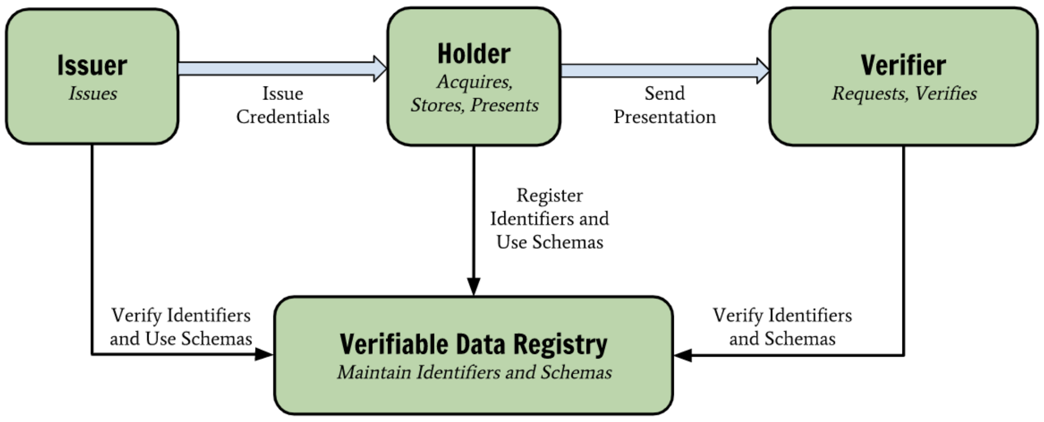

Issuer
Issues
Issue Credentials
Holder
Acquires, Stores, Presents
Send Presentation
Verifier
Requests, Verifies
Register Identifiers and Use Schemas
Verify Identifiers and Use Schemas
Verifiable Data Registry
Maintain Identifiers and Schemas
Verify Identifiers and Schemas

Figure 2: VC Ecosystem diagram from the VC Data Model[9].

Use Cases

The following use cases are referenced in this document

Table 3: Use Cases

Number	Subject/Holder Relationship	Record Types	Description
1	Subject = Holder	VCs (including LERs)	A learner (= subject = holder) uses their wallet to aggregate a collection of VCs (including LERs) matching some search criteria into a bundle, packaged as a VP. During this process, the learner may use their wallet to embed proof of identifier control.
2	Subject = Holder	All records	A learner uses their wallet to aggregate a collection of records, which may include records not conforming to the VC Data Model, matching some search criteria, into a bundle. In contrast with the above use case, identity proofing may not be supported on the non-conforming records (at least as defined here), but may be passed to the relying party for subsequent handling.
3	Subject <> Holder	VCs (including LERs)	A holder who is not the subject -- such as a trusted party the issuer or subject trusts to holds VCs on their behalf -- uses VPs to transmit the subject's VCs to another holder (enabling a chain of custody and holder identity proofing)

[9] While the VC model allows the subject to be different from the holder, the majority of uses mentioned here use the case that subject = holder, and exceptions are called out. Also note that this document uses the term *relying party* as a more general purpose term than *verifier*.

Foundations

The following specifications are fundamental to this approach.

Table 4: Foundational Specifications

Specification Name	Description
W3C Verifiable Credentials (VCs) Data Model	A lightweight, interoperable standard for expressing a wide variety of tamper-evident claims whose authenticity can be verified. The VC data model is associated with a range of emerging standards around the request and transfer of credentials, as well as identifier verification. The VC data model functions as an interoperable, secure wrapper around a variety of content (diplomas, transcripts, badges, competencies)[10]. Credentials stored in the wallet described here assume a VC or Verifiable Presentation (VP) data model.
LER Wrapper and Wallet Specification	Informs wallet functional requirements and provides methods for wrapping payload data models in the envelope, compatible with the VC data model.
Universal Wallet 2020	Provides an interoperable digital wallet standard and implementation. This specification is based on the universal wallet interop specification and describes how the wallet relates to it.

Normative References

- BBS+ Signatures 2020 [https://w3c-ccg.github.io/ldp-bbs2020]
- Comprehensive Learner Record Standard 1.0 [https://www.imsglobal.org/activity/comprehensive-learner-record]
- Confidential Storage 0.1 [https://identity.foundation/confidential-storage/]
- Credential Handler API 1.0 [https://w3c-ccg.github.io/credential-handler-api/]
- Decentralized Identifiers (DIDs) v1.0 [https://w3c.github.io/did-core/]
- DID Method Rubric [https://w3c.github.io/did-rubric/]
- did:web Decentralized Identifier Method Specification [https://w3c-ccg.github.io/did-method-web/]
- Encrypted Data Vaults 0.1 [https://digitalbazaar.github.io/encrypted-data-vaults/]
- Identity Hub github repository [https://github.com/decentralized-identity/identity-hub]
- JSON [https://tools.ietf.org/html/rfc8259]
- JSON Web Signatures (JWSs) [https://tools.ietf.org/html/rfc7515]
- JSON Web Token (JWT) [https://tools.ietf.org/html/rfc7519]

[10] The examples above apply to learning and employment contexts (the focus of this paper), but VCs can also be used for content such as health records, and even cases where the subject of the credential is not human,e.g. bills of lading for shipping, inventory in supply chains, and sensor data packets sent among self-driving vehicles.

- JSON-LD [https://cdn.filestackcontent.com/preview/FeqEJI3S5KelmLv8XJss]
- LER Wrapper and Wallet Specification [https://cdn.filestackcontent.com/preview/FeqEJI3S5KelmLv8XJss]
- Linked Data Proofs [https://w3c-ccg.github.io/ld-proofs/]
- Linked Data Signatures for JWS [https://w3c-ccg.github.io/lds-jws2020/]
- Modeling Educational Verifiable Credentials [https://w3c-ccg.github.io/vc-ed-models/]
- Open Badges Standard [https://www.imsglobal.org/activity/digital-badges]
- OpenID Connect Core [https://openid.net/specs/openid-connect-core-1_0.html]
- OpenID Connect Discovery [https://openid.net/specs/openid-connect-discovery-1_0.html]
- OpenID Connect Credential Provider [https://mattrglobal.github.io/oidc-client-bound-assertions-spec/]
- Presentation Exchange [https://identity.foundation/presentation-exchange/]
- Schema.org [http://schema.org/]
- Self-Issued OpenID Connect Provider (SIOP) DID Profile v0.1 [https://identity.foundation/did-siop]
- The did:key Method v0.7 [https://w3c-ccg.github.io/did-method-key/]
- Universal Wallet 2020 [https://w3c-ccg.github.io/universal-wallet-interop-spec/]
- VC HTTP API [https://w3c-ccg.github.io/vc-http-api/]
- VC Status List 2021 [https://w3c-ccg.github.io/vc-status-list-2021/]
- Verifiable Credentials Data Model 1.0 [https://www.w3.org/TR/vc-data-model/]
- Verifiable Presentation Request Specification [https://w3c-ccg.github.io/vp-request-spec/]
- Wallet and Credential Interactions (WACI) [https://identity.foundation/waci-presentation-exchange/]
- WebKMS v0.1 [https://w3c-ccg.github.io/webkms/]

Informative References

- Introducing OIDC Credential Provider [https://medium.com/mattr-global/introducing-oidc-credential-provider-7845391a9881]
- Verifiable Credentials Flavors Explained [https://www.lfph.io/wp-content/uploads/2021/02/Verifiable-Credentials-Flavors-Explained.pdf]
- Verification, Validation, and Veracity [https://github.com/digitalcredentials/docs/blob/main/verification/verify_credential.md]

How this differs from the LER Wrapper and Wallet Specification

This work builds on, and complements, the LER Wrapper and Wallet Specification that was published by the T3 Innovation Network in two ways.

First, the LER Wrapper and Wallet Specification outlined a set of functional requirements for a learner wallet. This effort builds on that foundation, specifying ***how*** those requirements can be fulfilled.

Second, the LER Wrapper and Wallet Specification provided a critical fast track to interoperability that leverages the W3C Verifiable Credential data model, where payloads based on multiple existing standards may be encoded flexibly in their original structure. This achieves interoperability at the envelope layer, which is critical for systems seeking to reuse transport, storage, and verification methods across a variety of credential types. However, this approach requires the need for payload format-specific processors/parsers at multiple points in the credential exchange flow, e.g., wallets and relying party systems each have to know how to decode the payload.

In contrast, this specification builds on the ongoing effort in the W3C VC-EDU task force to represent native-VCs, enabling deeper semantic interoperability and alignment through linked data. This document also provides guidance on an approach for cross-payload semantic interoperability and data linking beyond the use cases that are in scope for VC-EDU payloads.

How this differs from the Universal Wallet

This specification builds on, but is different from the W3C Credentials Community Group Universal Wallet Interoperability Specification. Specifically, this document focuses on wallet use cases applied to the education domain. As such:

- This specification addresses topics such as progressive trust that would not make sense in the case of a currency wallet.
- Currency-specific functions such as direct exchange do not apply here.
- New use cases are supported that have to do with different kinds of learning and employment records encoded using different standards, such as discovery of transcript and employment records mapped to common skills through linked data.
    - E.g., PESC transcript in JSON and HR Open employment record in JSON; both reference the same skill and a wallet's support for a query to "give me all things that reference that skill".

Requirements

Functional Requirements

The LER Wrapper and Wallet Specification Functional Requirements (Section 1.4) provide the basis of our functional requirements, which are further refined in this section.

1. Request Credential

    a. The wallet must be able to (on behalf of the holder) request a credential from an issuer
        i. The credential request must allow the request to enable holder and subject binding
        ii. The wallet must be able to request a credential in response to a holder action
    b. The wallet may be able to request a credential using a subscribe model in which VCs representing earned credentials from one or more issuers are requested/received/persisted so that the wallet stays up-to-date with available credentials from those issuers
    c. The wallet may be able to request other records that serve as a proxy for credential-like employment experience
2. Receive Credential
    a. The wallet must be able to receive credentials.
    b. The wallet must be able to decline credentials.
    c. The wallet must be able to persist credentials and store the appropriate metadata (see Persist Credential)
    d. The wallet may be able to unpack the credential payload, but it is not required to do so.
    e. The wallet may be able to request, listen for, or subscribe to credential updates, if offered, and if the holder chooses to enable.
        i. The holder must be able to decline a credential received via subscription.
    f. The wallet may be able to persist other records that serve as a proxy for credential-like employment experience
3. Persist Credential
    a. Wallet must be able to persist credentials with native format encoding from multiple standards.
    b. Stored credentials must be persisted with sufficient metadata to allow the wallet to execute the minimal functions described in these requirements.
    c. Credentials may be stored in both the native format and/or a processed format preferred by the wallet so long as the wallet can fully produce the original record intact (see Send Credentials).
    d. Wallet must be able to respond to a holder's request to remove a credential and stop persisting that credential.
4. Select Credentials by querying the wallet data store
    a. The wallet must be able to discover LERs at least by the name property and description property in the LER.
    b. The wallet may discover content by other parameters.
    c. The wallet may expose an external API.
5. Send Presentation
    a. The wallet must have a mechanism to create and submit a Verifiable Presentation to a relying party in response to

        i. A wallet owner action
        ii. A request for a VP obtained from an RP through a push or pull action, if approved by the wallet holder
    b. The wallet may have a mechanism for receiving and processing presentation requests
    c. The wallet must allow the presentation to include holder and subject binding
    d. The wallet may support pre-packaged presentation bundling options for convenience to the user, depending on parameters such as the type of relying party, credentials requested, etc.

6. Log Activity
    a. The wallet may be able to log activity (e.g., credentials and presentations sent and received for privacy auditing)
7. Holder and Subject Binding
    a. The wallet may be able to generate identifiers enabling proof of identifier control
        i. Examples include pairwise decentralized identifiers, other decentralized identifiers, and other methods resulting in a URI identifier that can serve as subject in a Verifiable Credential or a holder in a Verifiable Presentation
    b. The wallet may be able to generate proofs of identifier control
8. Management functions
    a. The wallet may be able to manage identity and identifier data. This includes the ability to:
        i. perform CRUD operations on decentralized identifier methods (not just create, from previous). I.e., update key material, read, delete
        ii. Group identifiers, attributes, and credentials into “persona” or profiles for use in different contexts
    b. The wallet may be able to manage connections (e.g. to issuers, RPs, and other parties)
    c. The wallet may be able to manage privacy and sharing settings

Identifiers and Authentication

Before accepting a credential or presentation as legitimate, a relying party may wish to confirm that it is bound to the party presenting it. This may be achieved through proof of control over an identifier, knowledge of a secret value, or biometrics[11].

This document will focus on the first one -- proof of identifier control -- which is commonly used in the VC ecosystem. The VC subject may be represented as a DID that the subject controls, enabling cryptographic proof of control of the identifier[12].

[11] See “Holder and Subject Binding” in Presentation Exchange
[12] See “Proof of Identifier Control in Presentation Exchange

As of early 2021, the emerging OIDC Credential Provider specification offers another promising approach to cryptographically verifiable URIs, which allows -- but does not require -- DIDs. This approach enables wallets to request credentials from OpenID providers (OP) such that they are reusable ("re-provable") to RPs without requiring the OP's subsequent involvement[13].

Note: the LER Wrapper and Wallet Specification uses the term identifier authentication, for this concept.

Additional Design Decisions and Goals

- Minimize technical and infrastructure requirements for learners and issuers
- Ensure requirements are applicable/adaptable to mobile and web wallets
- Enable integration into existing systems (including authentication solutions)
- Ensure design is applicable/extensible to emerging protocols (such as authentications built with awareness of decentralized identifiers)
- Use JSON-LD data formats where possible due to advantages of Linked Data in the educational data standards ecosystem, but support integration of other data formats, including JSON-formatted VCs, non-VCs (including unstructured data)
- Support range of proof mechanisms (JSON-LD, JWT, data minimization proofs), with initial prioritization of JSON-LD proof mechanisms (based on initial use cases).

The wallet design uses interfaces and plugins to achieve the flexibility described above. The initial reference wallet implementation will use the following specific protocols:
- HTTP/REST services
- Auth: OIDC and SAML supported by default

This document also describes how to integrate support for other protocols.

Learner wallet implementers should ensure the wallet is accessible to the largest number of users. While a full consideration of this topic is outside of the scope of this document, some relevant factors are listed below:

- Accessibility guidelines
- Functionality across web and mobile
- Supporting deployment on a broad range of devices

Implementers should consider the following storage requirements:

[13] See Introducing OIDC Credential Provider

- When using on-device storage in a mobile app, the presence of large, minimally compressible data in the credentials (such as images) is likely to dominate space requirements
- Wallet implementers may provide storage estimates to their users based on typical use (e.g. 100 credentials, each with a 2MB embedded image, might require around 210 MB).
- Wallet implementers may enable off-device storage, so that learners may choose the storage location.

Wallet Flows and Procedures

Conceptual Wallet Flows

This section describes conceptual end-to-end wallet flows relevant to requirements listed in the previous section. The abstractions identified in this conceptual view simultaneously inform a wallet design enabling adaptation to a range of standards, technical stacks, issuer deployment and wallet implementation choices.

While wallet-focused flows and standards are the focus of this specification, representative issuer-focused considerations are included in Annex E: Issuer-Side Credential Issuance Flow for completeness.

Issuance Request/Response Flow

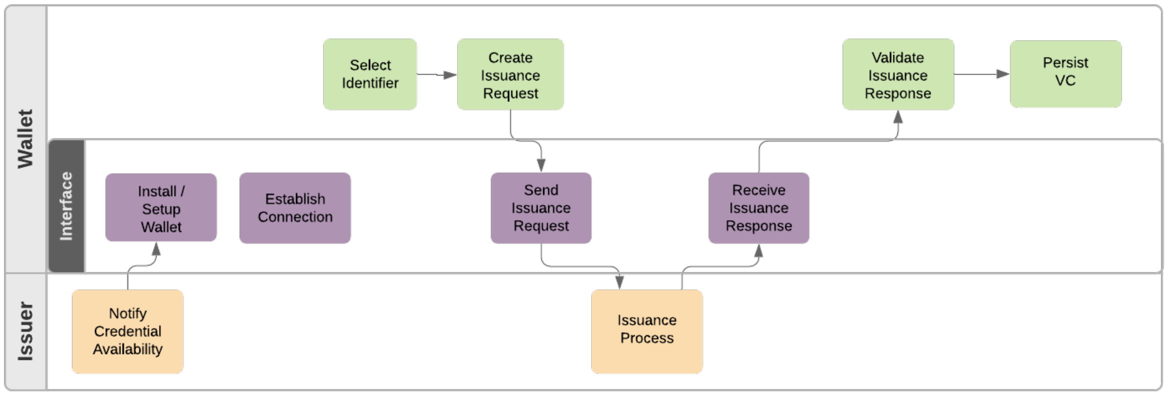


Figure 3: Issuance End-to-End Flow (Conceptual)

The first 3 steps are only briefly discussed[14] for the following reasons:

- Notify Credential Availability is largely up to the issuer; it could simply be implemented as a personalized email to the learner

[14] Nonetheless, the conceptual flows support these steps and allow for variation they require

- Install/Setup Wallet is a one-time prerequisite
- Establish Connection manifests in widely varying ways and orderings in relation to the other steps.

Issuance flow steps are labeled with one of the following categories:
- Issuer: Informed by standards, but realized entirely by issuer decisions
- Wallet / Interface: Interface points between wallet and issuer
- Wallet: Wallet-internal operations, which areabstractions supporting a variety of message formats and proof mechanism understood by external parties

Table 5: Issuance Flow Steps

Step Name	Category	Description
Notify Credential Availability	Issuer	School or issuing institution typically initiates the process by sending a notification (such as an email) informing the learner that they are eligible to receive digital credentials, with relevant prompts to proceed (along with links to install compatible wallets, if not done already). The prompt to receive a credential may appear as a deep link into the wallet, possibly encoded as a QR code, depending on the learner/device (mobile, web) interaction pattern.
Install / Setup Wallet	Wallet/ Interface	Learner must perform one-time wallet installation / setup. For mobile credential wallets, this involves installing a mobile app.
Establish Connection	Wallet/ Interface	Encapsulates the ways a learner may connect to an issuer or relying party, including peer-to-peer connections and well-known authentication protocols already used by the issuer/relying party. While depicted as an explicit step, the manner in which this occurs varies widely depending on the protocols/deployments used. This may be used to mutually exchange identifiers. This step ensures the wallet can access the relevant issuer services.
Select Identifier	Wallet	The wallet allows the learner to create and manage identifiers to be associated with credentials. This abstraction supports use of identifiers enabling proof of identifier control (such as decentralized identifiers) as well as other digital identities.
Create Issuance Request	Wallet	Message generated by the wallet to specify credential(s) the wallet holder wishes to be issued along with desired subject identifier. This abstraction supports a range of issuance request message formats.
Send Issuance Request	Wallet/ Interface	Encapsulates the range of transport protocols that deliver the Issuance Request to the issuer.
Issuance Process	Issuer	Procedure in which the issuer issues verifiable credential(s) and conveys them to the holder via the wallet.

Receive Issuance Response	Wallet/ Interface	Encapsulates the range of transport protocols that deliver the Issuance Response to the wallet. Response contains verifiable credential(s), generally wrapped with additional metadata.
Validate Issuance Response	Wallet	Verify and validate the response, including the embedded VC.
Persist VC	Wallet	Persist the credential in secure, flexible, accessible storage.

The following demonstrates a representative detailed procedure achieving the core wallet steps shown above.

Table 6: Example Wallet Steps in Issuance Request/Response Flow

Example Wallet Steps in Issuance Request/Response Flow	
Assumptions: 1. Student wallet has been installed and/or setup 2. Connection has been established or occurs as part of the issuance flow 3. Wallet is opened (via deep link, user action, etc) and provided with inputs below.	
Inputs:	
1. `vc_request_endpoint`: Credential issuance request endpoint (may include additional parameters)	
2. `domain`: Any string or URI, provided by the issuer	
3. `challenge`: Signing challenge. Should be a randomly-generated string[15]	
4. Authentication discovery information (or already authenticated)	
5. (optional) Expected subject identifier	
Step:	**Brief Description:**

[15] https://www.w3.org/TR/vc-imp-guide/#presentations

1. Select Identifier[16]	Wallet generates or allows selection of an `identifier` (depending on the DID methods it supports). If an expected subject identifier is provided, wallet confirms it has control of the expected subject identifier. Some protocols may also include a way to express only certain DID methods.
2. Create Issuance Request	a. Wallet forms an issuance request[17], populated with `identifier`, `domain`, and `challenge` b. Wallet signs the issuance request with the key material corresponding to `identifier`
• *Send Issuance Request: e.g., wallet sends the request to the issuer at `vc_request_endpoint` (may involve auth step)* • *Issuance Process: Issuer processes request, issues VC(s), sends* • *Receive Issuance Response*	
3. Validate Issuance Response	Wallet unwraps the VC(s) in the issuance response, and validates/verifies it
4. Persist VC	Wallet persists VC(s) securely based on pre-configured storage option or user selection.

Presentation Exchange Flow

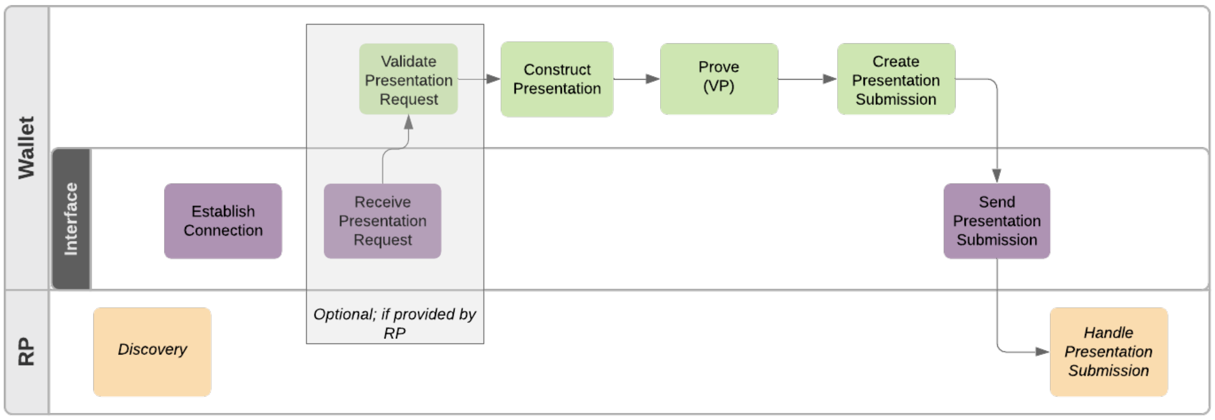

Wallet
Interface
RP
Validate Presentation Request
Construct Presentation
Prove (VP)
Create Presentation Submission
Establish Connection
Receive Presentation Request
Send Presentation Submission
Discovery
Optional; if provided by RP
Handle Presentation Submission

Figure 4: Presentation Exchange Flow

[16] Coverage of identifier selection best practices is outside the scope of this specification, but deserves special ongoing attention from the learning and employment community. The DID specification describes privacy considerations (such as correlation risk) resulting from identifier reuse, as well as mitigation strategies. Current practices range from reuse of identifiers for non-sensitive data to use of pairwise unique identifiers.

[17] The request may provide additional details about the types of credentials the learner wishes to receive (if supported by the issuer), such as: level of granularity or preferred credential format

The first 2 steps are only briefly discussed[18] for the following reasons:

- How the RP enables discovery of Presentation Requests and/or submission endpoints is influenced by standards but is largely up to the RP
- Establish Connection manifests in widely varying ways and orderings in relation to the other steps.

Further, Receive/Validate Presentation Request steps occur only if the RP makes a Presentation Request available. The remaining steps in the sequence may also occur in response to a wallet holder-initiated action.

Category Guide:

- Relying Party: Informed by standards, but realized entirely by RP decisions
- Wallet / Interface: Interface points between wallet and RP
- Wallet: Wallet internal operations; abstractions supporting a variety of message formats and proof mechanisms understood by external parties

Table 7: Example Wallet Steps in Presentation Exchange Steps

Step Name	**Category**	**Description**
Discovery	Relying Party	Widely variable method that the RP enables a holder to initiate the exchange process
Establish Connection	Wallet/ Interface	(Same as above)
Receive Presentation Request	Wallet/ Interface	The wallet may receive a presentation request; abstraction allows flexible formats
Validate Presentation Request	Wallet	Encapsulates ways a wallet may validate the RP's request for a VP (if a request is provided). This may include payload validation, checking whether the wallet supports the request, and the types of identifiers or proof mechanisms the RP supports.
Construct Presentation	Wallet	Encapsulates the process by which the wallet software selects credentials and/or derivations[19] consistent with the presentation request criteria (if provided) or with the wallet holder's search criteria. The wallet software (and/or associated agents) has a role in credential selection for improved convenience to the wallet holder, but wallet holder consent is a critical feature. This may be derived from wallet holder preference settings (e.g., credentials of this type may always be shared) and/or explicit approval before sending.
Prove (VP)	Wallet/	Apply a proof, typically a cryptographic signature, to the VP. Abstraction enables a range of proof mechanisms.

[18] Nonetheless, the conceptual flows support these steps and allow for variation they require
[19] Such as enabled by selective disclosure proof mechanisms

		This step may achieve proof of identifier control[20]. If a Presentation Request is provided, it may specify acceptable proof mechanisms.
Create Presentation Submission	Wallet/ Internal	Message generated by the wallet to specify credential(s) the wallet holder wishes to be issued along with desired subject identifier. This abstraction supports a range of issuance request message formats.
Send Presentation Submission	Wallet/ External	Deliver the Presentation Submission to the RP. This abstraction enables support of a range of transport protocols
Handle Presentation Submission	Relying Party	The RP verifies and validates the VP and continues with business processes

The following demonstrates a representative detailed procedure achieving the core wallet steps shown above.

Table 8: Example Wallet Steps in Presentation Request/Response Flow

Example Wallet Steps in Presentation Request/Response Flow
Assumption: Wallet has been installed and opened (via deep link, user action, etc) and provided with inputs below. This may be initiated by the issuer sending a notification or other prompt indicating that the learner is eligible to receive a credential and how to proceed, along with instructions about setting up the learner wallet (if not already done).
Inputs:
1. `presentation_submission_endpoint`: Credential issuance request endpoint (may include additional parameters)
2. `domain`: (Optional) Any string or URI, provided by the issuer
3. `challenge`: (Optional) Signing challenge. Should be a randomly-generated string[21]
4. Authentication discovery information (or already authenticated)
5. (optional) Expected subject identifier

[20] See Annex F: Proof of Identifier Control, Illustration
[21] https://www.w3.org/TR/vc-imp-guide/#presentations

Step:	**Brief Description:**
1. Validate presentation request (optional)	If the wallet receives a presentation request, validate to determine whether to accept.
2. Construct Presentation	a. Run query against credential storage. Query comes from presentation request (if provided) or from wallet holder criteria b. Builds presentation containing: ○ Matching credentials and/or derivations ○ `identifier` (newly generated or matching presentation request) ○ `domain` and `challenge`, if provided
3. Prove	Prove, resulting in a VP (which may used as proof of identifier control)
4. Create presentation submission	Wrap in additional metadata required depending on specification, such as Presentation Submission

Consent and Usability

Consent is a critical aspect of a learner-focused wallet, but usability remains careful in achieving *cognizant* consent. Overwhelming the user with information -- for example, by requesting the learner to review the detailed (machine-targeted) credential data before submitting -- is just as unsatisfactory as a cursory review step.

Addressing this problem will be an ongoing effort. The following examples show how a wallet might better merge the concerns of consent and usability:

- A wallet holder might choose to allow a trusted web application to receive all learner and employment records, but would like to review the request and explicitly "accept" before passing them along
- A wallet holder might decide certain records are acceptable to share with anyone, and doesn't want to review the request
- Requests for consent should be phrased to concisely convey when sensitive data occurs in a record. For example: "[Relying party] is requesting to be able to see your date of birth, your full transcript, etc. Do you consent?"
  - The name/type of record may not clearly communicate to the wallet holder what range of sensitive data appears in the record; a wallet with deeper semantic awareness of data fields has the opportunity to help.
  - The wallet may also present records in a manner that omits sensitive data the RP doesn't require (assuming the record is issued in a manner amenable to selective disclosure or minimal disclosure techniques).

In-Wallet Discovery

The previous “LER Wrapper and Wallet Specification” specified a minimum set of wallet functions and related metadata for discoverability and selection of LER payloads within a wallet. This specification carries forward those in wallet discovery functions, e.g., selection of LER’s based on name and description, and supports additional discoverability by other properties. The previous specification did not address a wallet’s ability to upack or semantically interpret the standards-based payload. This specification addresses the semantic interpretation and in wallet discovery based on payload data. It also points to emerging and anticipated standards and infrastructure to address translation across payloads. See the following section on “Data Schema Interoperability” for more on this topic.

Wallet Design

Universal Wallet Interop Specification

The goal of the Universal Wallet Interop Specification is to provide interoperability for a wide range of digital wallets, from credential wallets to currency wallets.

The specification defines data models and interfaces -- some relevant to all digital wallets, others applicable to specific domains or use cases, such as credential wallets or currency wallets. The specification’s associated open source codebase serves as a reference implementation, or wallet SDK, and includes:

- Data Model for expressing wallet entities
- Interfaces for expressing wallet operations
- Reference implementation

While these elements are currently collocated in a single repository, they will be split as the efforts mature through the standardization process.

The rest of this section provides details about how the Universal Wallet serves as a foundational layer to the learner wallet, and how its extensibility mechanisms work in the context of the learner wallet. Ongoing evolution will occur in the Universal Wallet and associated repositories listed in Intended Use and Next Steps.

Core Wallet Interfaces and Extensions

Base interfaces expose cross-cutting functionality relevant to all wallets. Examples include:

- Adding and removing items
- Locking and unlocking the wallet
- Importing and exporting from backup

Interfaces exposing specific functionality are layered on top. Examples include:

- Currency wallets can transfer money
- Credential wallets can sign, issue, and prove credentials

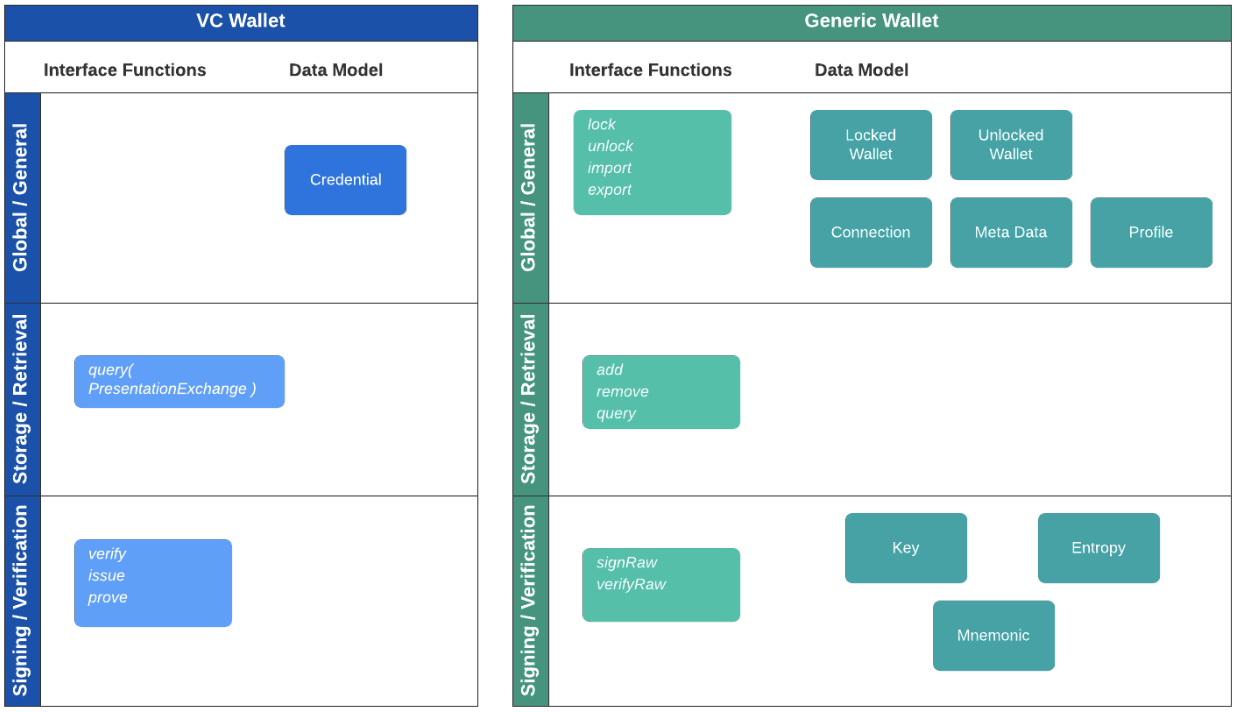


Figure 5: Depiction of Universal Wallet Interface and Data Model Layers

Wallet implementers can similarly achieve further specialization by adding the ability to read/interpret contents of different types of credential payloads, offering richer, domain-specific functionality. This could also provide the ability to translate across VC payloads for the purpose of semantic query alignment.

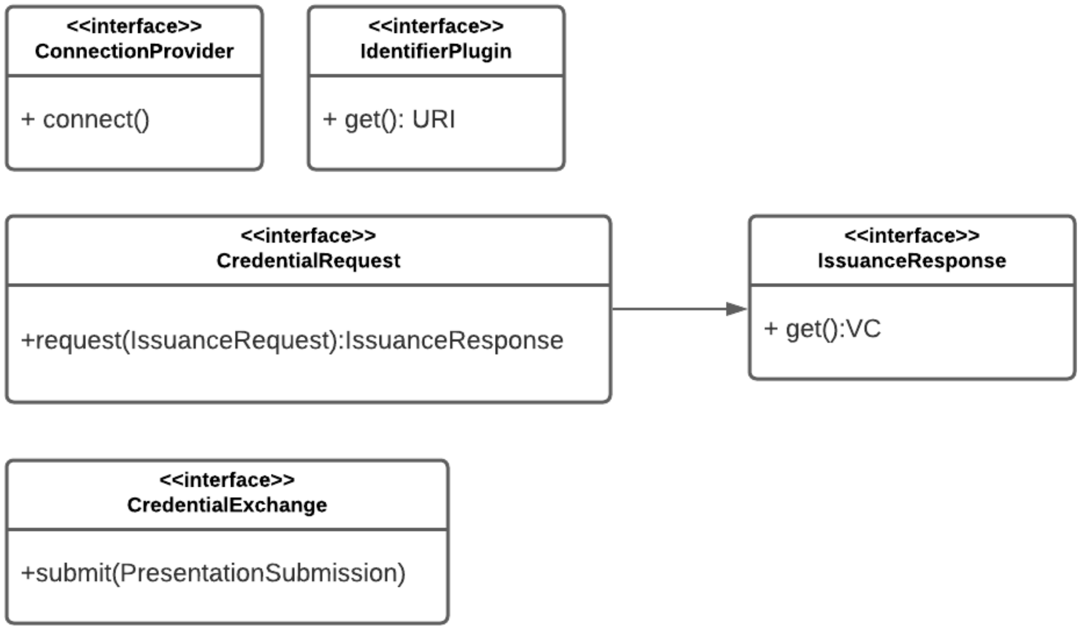
SDK Interfaces and Data Models

Basic interfaces and data models relevant to the initial implementation of the student wallet are:

- Universal Wallet Core
- VC Wallet Layer
- Additional Interfaces (signatures below)
    - ConnectionProvider
    - IdentifierPlugin
    - CredentialRequest
    - CredentialExchange

The additional interfaces will support Issuance and Presentation Exchange flows generalized way for implementers of the wallet and will be candidates for inclusion in the Universal Wallet as they mature.

Default implementations of these interfaces are described in Layered Design Decisions, and implementers extend to support additional identifier methods, storage mechanisms, request/exchange implementations, etc.



<<interface>> ConnectionProvider
+ connect()
<<interface>> IdentifierPlugin
+ get(): URI
<<interface>> CredentialRequest
+request(IssuanceRequest):IssuanceResponse
<<interface>> IssuanceResponse
+ get():VC
<<interface>> CredentialExchange
+submit(PresentationSubmission)

Example Pseudocode

The following pseudocode demonstrates how a wallet implementer might interact with the SDK as part of a credential request flow.

```
//
// CREATE ISSUANCE REQUEST
//

// generate new identifier
did = identifierPlugin.get() => URI (e.g., DID)

// create and prove VP
wallet.prove(did, options) => VP

// create issuance request
```

```
ir = new IssuanceRequest(VP)

// send request
wallet.request(ir) => IssuanceResponse
//
// HANDLE RESPONSE
//

// obtain VP response
response.getVC() => VC

// verify
wallet.verify(VC)

// store
wallet.store(VC)
```

Plugins and Extensions

When using the universal wallet reference implementation, wallet implementations select a set of plugins, where each plugin implements a set of interfaces. The plugin/interface model allows integration of a range of backing storage options, exchange protocols, etc. Developers decide which interfaces and plugins (or develop new ones) to add to wallets.

For example, the universal wallet defines a required interface `Storage` to support pluggable storage implementations. For a mobile wallet, a default option might be on-device storage, but wallet implementations may provide additional storage plugin implementations and/or allow selection by the user, either through configuration, or on-demand selection.

Migration

Migration across wallets is eased through wallet manifests expressing capabilities, permissions, requirements, and interfaces it supports. For example, moving from an LER specific wallet to a general VC wallet, could get warning about loss of richer support. This also allows expression of what level of security the wallet is appropriate for.

Representative Use Case of Extensibility

Through the extensibility models described above, the wallet could allow the learner to perform meaningful queries across their records (including VCs, LERs, and lers), enabling efficient curation actions (such as when building a collection of records to present to a potential employer).

Consider a simple use case in which a learner wants to prepare a portfolio of important learning, training, and employment milestones, sorted in reverse chronological order of effective date. At scale (as learners collect more records), this presents a challenge, as the field names vary across schemas and organizations.

Among ler/LER schemas, CEDS "Credential Definition Date Effective" corresponds to:

Schema	Class/Type	Element/Property
CTDL	`Credential`	`dateEffective`
HR Open	`CertificateType`	`firstIssued`
	`EducationDegreeType`	`date`
PESC	`Student.AcademicRecord.AcademicAward`	`AcademicAwardDate`
Schema.org	`EducationalOccupationalCredential`	`dateCreated`

The wallet's extensibility/plugin mechanisms enable queries across different payloads of all records (including VCs, LERs, and lers). A translation plugin/service (with updates to ensure up-to-date coverage of new schemas) enables a rename translation of these date fields required to perform this operation.

Verification and Trust

Progressive Trust - Is "More Trust" All that the Market Requires?

While the technical community seeks methods for absolute verifiability of digital credentials, the status quo that it hopes to replace is both binary--either a verifier takes action to verify a credential or they don't--and flawed.

In comparison to digital financial transactions, the market currently applies lower expectations for academic and employment credentials. In the case of education credentials 32% of small organizations don't bother to contact a verification service or PS Institution that issued a candidate's credential. There is of-course some risk of academic credential fraud, but many organizations don't see the risk as great enough to pay for verification services or the time it would take to verify on their own. So if verifiable credentials even incrementally reduce that risk it is providing benefit to the marketplace.

> "Despite the reported benefits of screening, many small organizations don't do as much as they could, possibly leaving themselves vulnerable. When screening a job candidate, respondents reported that criminal searches (97 percent) and identity verification (81 percent) were the most common checks. But only 58 percent of respondents verify previous employment history, and only 32 percent verify education credentials."[22]

While the goal of 100% verifiability of academic credentials is worthy of current efforts in the technical community, the market has signaled that a low-friction path to greater trustability may have greater benefits than a higher friction solution offering 100% trust. If the perceived cost/effort of using a perfectly verifiable (binary) solution is no less than making a phone call or sending an email, then many organizations will not want it. As an alternative we propose a progressive trust model that uses multiple verifiable facts to build increasing levels of trust without making work for the verifier or issuer. Instead of a binary trusted vs. non-trusted credential, the verifier would receive a verifiable credential with a trustability index, analogous to levels of assurance allowing trust requirements to vary depending on the context of its use.

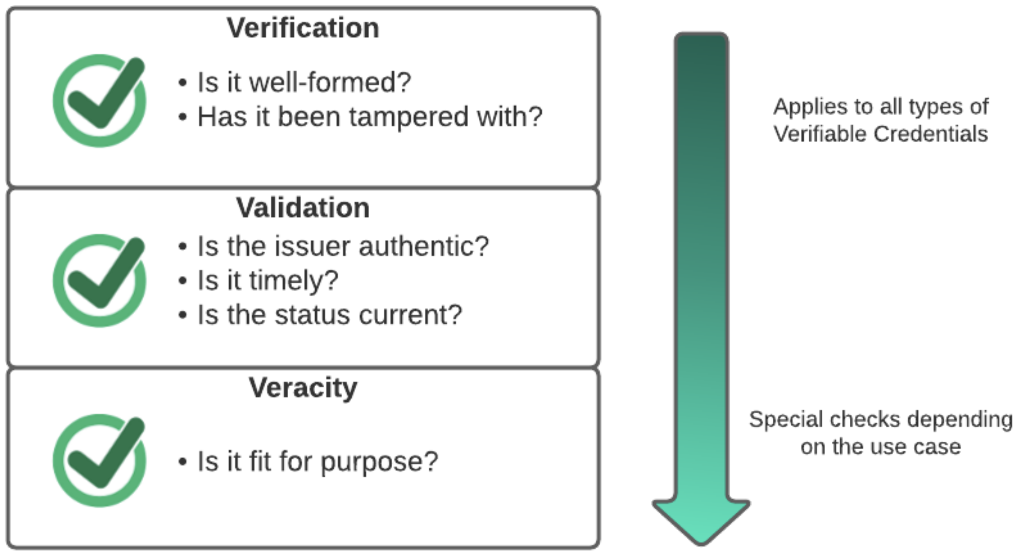
Multiple factors go into trustability, which the Verifiable Credentials ecosystem proposes to categorize as discussed in the next section.

Verification, Validation, and Veracity

Fortunately, the "verification" aspect of Verifiable Credentials does not prescribe a single verification mechanism that's meant to apply to all credentials; it is extensible to adapt to a range of requirements. The "Verification, Validation, and Veracity" process for Verifiable Credentials (3V) describes three categories of checks in establishing whether a relying party or verifier may trust a given credential.

These categories, and examples, are pictured below:

[22] https://www.shrm.org/resourcesandtools/hr-topics/risk-management/pages/screening-small-businesses-risk.aspx


Verification
• Is it well-formed?
• Has it been tampered with?
Validation
• Is the issuer authentic?
• Is it timely?
• Is the status current?
Veracity
• Is it fit for purpose?
Applies to all types of Verifiable Credentials
Special checks depending on the use case

In brief:

- **Verification** is the set of checks, with corresponding mechanisms, meant to apply to all types of verifiable credentials. These checks assure conformance to standards.
- **Validation** is a set of checks that are conceptually cross-cutting but vary greatly in implementation. A subset of these are handled by verification libraries through callbacks; the rest are handled by callers
- **Veracity** is the set of checks that would not be performed by common VC libraries, but are performed by RPs/verifiers in assessing whether to accept the credential.

The veracity category may be used to restrict and/or relax the type of credentials accepted:

- **Example of Restrict**: a relying party will only accept credential definitions described in the Credential Engine Registry, and only those issued by the set of issuers authorized to issue instances of that credential definition
- **Example of Relax**: a consortium or clearinghouse will accept un-revoked credentials that were issued by schools that are now out of business.

Further details are available in the draft specification Verification, Validation, and Validity, which will be the ongoing source for describing this process.

Who Verifies Credentials and Presentations

Each role in the VC ecosystem may wish to verify VCs and VPs[23]. For example:

- **Relying Party**: performs verification upon receipt from of a VP from a holder in order to:

[23] Further, as discussed in Verification, Validation, and Veracity, each party may apply different criteria when assessing the same credential, particularly at the Veracity layer

    - Verify the authenticity and integrity of the VCs it contains
    - Optionally perform proof of identifier control on the holder (which may be the same as the subject(s) in the VC(s)
- **Issuer**: may perform verification upon receipt of a VP presented by the learner (via the wallet) as proof of identifier control
- **Holder/Subject (via the wallet)**: performs verification before accepting a VC or VP from an issuer
    - This gives confidence that the credential(s) will be useful to the learner
    - Note: the issuer may deliver a VP as a wrapper for a bundle of VCs and/or as part of a signing challenge provided by the subject

Verify VCs and VPs

This section describes how the verification procedures for the wallet implementation will perform by default. The checks described here fall into the categories of *Verification* and *Validation* categories; see descriptions in Verification, Validation, and Validity.

While default implementations of additional inputs (signature suites, document loaders, and proof purpose) will be provided, the extensibility mechanisms enable implementer customization, including additional *Veracity* checks.

A VP contains:

- a list of VCs, and
- may contain a separate proof (enabling proof of identifier control)

Accordingly, the Verify VP procedure uses the Verify VC procedure.

Table 9: Verify VC

Verify VC
Inputs:
1. `vc`: A Verifiable Credential
2. `checkStatus`: Optional function for credential status
Steps:
1. Check well-formed according to the VC Data Model 2. Check proof   a. Credential hasn't been tampered with

   b. Issuer `id` (identifier)/key check, per caller-provided loaders*
      i. Signed with an expected issuer identifier/key
      ii. Signed by key authorized for the purposes of signing
   c. Expected proof properties are present
3. Check timeliness and status, per caller-provided `checkStatus` function**

* Default document loaders and DID resolvers will support methods specified in Implementation Choices
** Default function will perform a basic timeliness check using the usual interpretations of `issuanceDate` and `expirationDate` fields and a status check using methods specified in Implementation Choices.

Table 10: Verify VP

Verify VP

Inputs:

1. `vp`: A Verifiable Presentation
2. `domain`: (optional; if the requesting party provided one) Any string or URI
3. `challenge`: (optional; if the requesting party provided one) Expected signing challenge. Should be a randomly-generated string[24]

Steps:

1. VP Check
   a. Check VP well-formed according to the VC Data Model
   b. Check proof
      i. Presentation hasn't been tampered with
      ii. Holder `id` (identifier)/key check, per caller-provided loaders*
         1. Signed with an expected holder identifier/key
         2. Signed by key authorized for the purposes of signing
      iii. Expected proof properties are present
   c. If the requesting party provided a `domain` and `challenge`, check the VP contains the expected values
2. For each VC in VP: verify(VC)

* Default document loaders and DID resolvers will support methods specified in Implementation Choices.

Prove VP

Proving a VP is analogous to signing/issuing a VC, except the signature is applied at the presentation layer.

[24] https://www.w3.org/TR/vc-imp-guide/#presentations

A common wallet use case for this is providing proof of identifier control, in which case, the wallet forms a presentation as follows:

1. Add to `holder` field an identifier the wallet controls
2. Generate a proof (often a cryptographic signature), resulting in a VP

Note that this is a VP that doesn't contain credentials, as permitted by the VC data model.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1"
  ],
  "type": "VerifiablePresentation",
  "holder": "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "proof": {
    "type": "RsaSignature2018",
    "created": "2020-09-14T21:19:10Z",
    "proofPurpose": "assertionMethod",
    "domain": "http://www.example.com",
    "challenge": "e506f108-9c74-4c52-b84c-a7019c83328c",
    "verificationMethod": "did:example:ebfeb1f712ebc6f1c276e12ec21#keys-1",
    "jws": "pYw8XNi1..Cky6Ed="
  }
}
```

Data Schema Interoperability

Ecosystem Supporting Cross-Standards Interoperability within the Universal Wallet

Universal learner wallets may contain data payloads that represent many kinds of assertions about the learner based on experiences at all academic levels as well as workplace and informal learning contexts. Data standards developed for traditional educational contexts use different schemas developed in different communities of practice than standards for workplace learning or other contexts. This "Tower of Babel" problem is being addressed in this Universal Wallet Specification.

To be future-proof, a universal wallet must be able to accomodate data schema that don't yet exist, documenting learning experiences and competency definitions for fields that have not been developed.

It is unlikely that any one standard or even one standards organization can ever cover current and future schemas needed to capture evidence of all learning experiences, competencies and credential assertions. However, the Universal Wallet can be universal if it is extensible and integrated into an ecosystem that is continuously updated with the information needed to accommodate new standard data definitions and schemas, and metadata needed to make sense of the various schemas.

There are two parts to the recommended approach to make wallets universal across standards and future-proof.

1. Extensibility using a plugin mechanism
2. Metadata ecosystem used by translation services and wallet plugins

Extensible Plugin Mechanism
(See Plugins and Extensions of this document.)

Open Metadata Ecosystem
The envisioned distributed metadata network will provide metadata of maps and transformation rules across and between various wallet payload standards. For the purposes of this specification we will provide a simple proof-of-concept example of the kind of metadata to accomplish this mapping and transformation of data.

The metadata ecosystem will take time to mature. In the meantime, such as with pilots of this specification, isolated sets of translation metadata may be used with a wallet ‘translation plugin’ to accomplish an isolated demonstration of the approach.

Intended Use and Next Steps

This document constitutes deliverable Task 2, Sub-task 2.1 B as specified in the Department of Education contract “Develop Open Source Standard for Student Credential Wallet” between the Department of Education and MIT.

The approach and recommendations described in this document will undergo continued development[25], with credit to the U.S. Department of Education (Contract Number: 91990020C0105), in the following locations:

Group / Standard	Item

[25] To also include interoperability test suites

Universal Wallet 2020 [https://w3c-ccg.github.io/universal-wallet-interop-spec/]	Wallet Data Model, Interfaces, and SDK
Digital Credentials Consortium github repo [https://github.com/digitalcredentials]	● Wallet Mobile Application (based on Universal Wallet SDK) ● Issuance and Verification Libraries (conformant to vc-http-api)
W3C VC-EDU Task Force [https://w3c-ccg.github.io/vc-ed/]	● Wallet Requirements ● Wallet Flows ● Linked Data Contexts, Draft Protocols, and Query Extensions ● Credential Models and Examples; specifically "Modeling Educational Verifiable Credentials" [https://w3c-ccg.github.io/vc-ed-models/]
Vc-http-api [https://github.com/w3c-ccg/vc-http-api]	Verification Guidelines
Decentralized Identity Foundation Wallet Security Working Group [https://identity.foundation/]	Wallet Security

As these efforts mature, they may be submitted for further standardization at the appropriate body. This may include a W3C or IEEE Working Group[26], as appropriate.

Annex A: Acknowledgements

Funding for this research report was provided by the U.S. Department of Education (Contract Number: 91990020C0105). The opinions expressed herein do not necessarily represent the positions or policies of the U.S. Department of Education, and no official endorsement by the U.S. Department of Education should be inferred.

Additionally, the US Department of Education's funding supported the development of the following specifications referenced in this document:

- Universal Wallet 2020
- vc-http-api (Verification, Validation, and Validity)
- Modeling Educational Verifiable Credentials

Editors:

- Kim Hamilton Duffy
- Ulrich Gallersdörfer
- James Goodell
- Matt Lisle

[26] For example, the IEEE Learning Technology Standards Committee (LTSC) [https://sagroups.ieee.org/ltsc/] may be appropriate for learning-related standards.

- Brandon Muramatsu
- Philipp Schmidt

Technical Contributions and Feedback On This And Supporting Specifications:

- Phil Barker
- Anthony Camilleri
- Sam Curren
- Jeff Dieffenbach
- Taylor Kendal
- Adam Lemmon
- Kerri Lemoie
- Phillip Long
- Nate Otto
- Sudesh Shetty
- Jacksón Smith
- Manu Sporny
- Orie Steele
- Nathan Tonani

This builds on ongoing work of the following organizations:

- W3C Credentials Community Group [https://www.w3.org/community/credentials/]
- Decentralized Identity Foundation [https://identity.foundation/]
- IEEE Learning Technology Standards Committee (LTSC) [https://sagroups.ieee.org/ltsc/]
- Internet Identity Workshop [https://internetidentityworkshop.com/]
- Rebooting Web of Trust [https://www.weboftrust.info/]

Annex B: Mapping Conceptual Data Flows to Standards

This section maps the conceptual wallet flows from the previous section to emerging specifications and protocols. To enable application to the widest range of relevant standards, (non-normative) terminology is introduced.

While some of the specifications/protocols listed below are in draft status, none are prerequisites for the wallet SDK; rather the wallet SDK abstractions can be mapped as they mature.

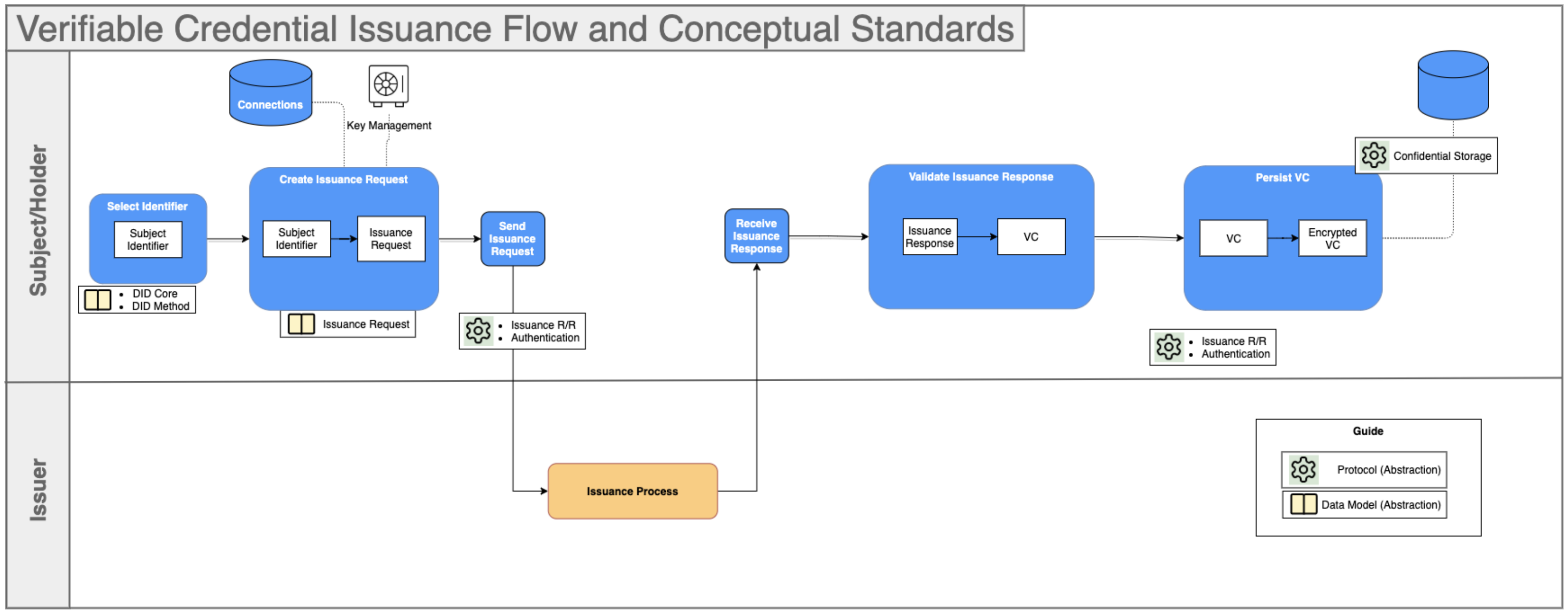


Figure B1: Issuance Request/Response Flow, Refined

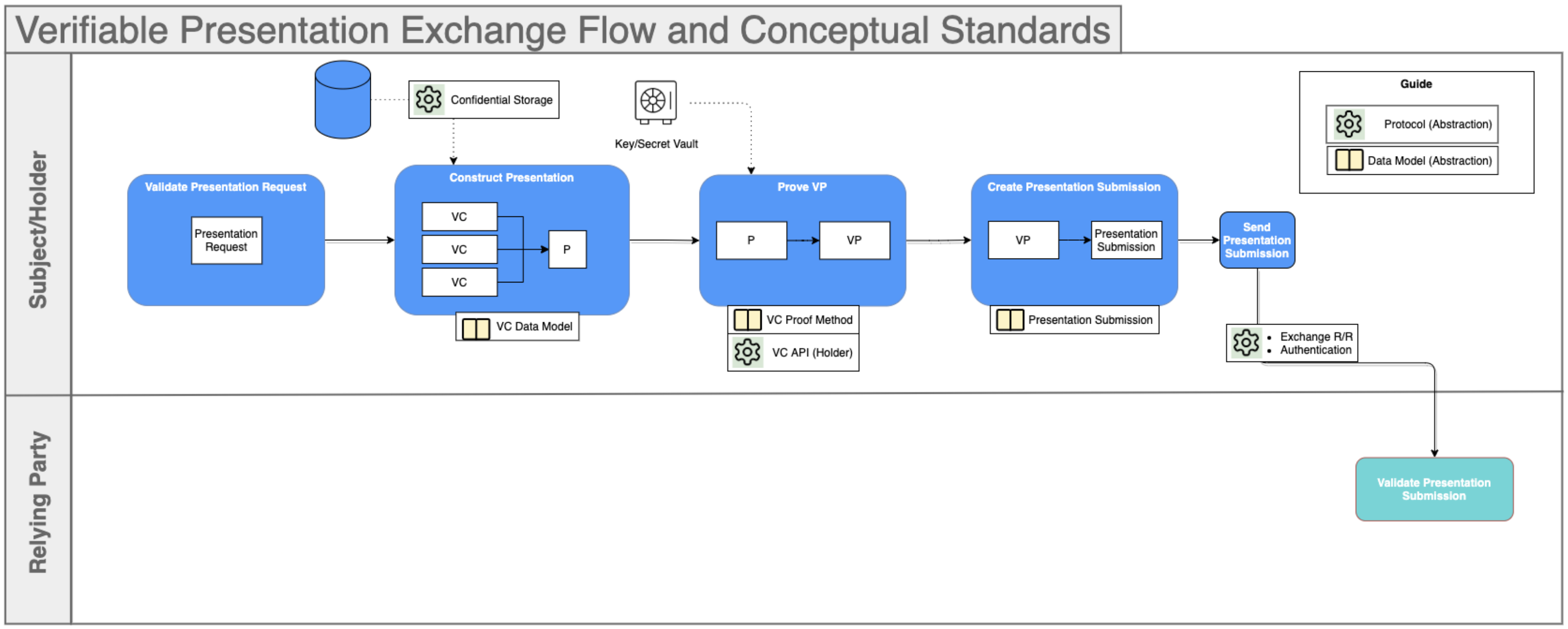


Figure B2: Presentation Exchange Flow, Refined

Table B1: Conceptual Data to Standards Map and Initial Implementation Choices

Concept Name	Type	Specification Examples	Initial Implementation Choices
Issuance Request / Response	Protocol	● OpenID Connect (OIDC) Credential Provider	● [Spec/Location pending] ● [Extension] OpenID Connect (OIDC) Credential Provider
Presentation Request / Response	Protocol	● Credential Handler API 1.0 ● Wallet And Credential Interactions (WACI)	● [Spec/Location pending]
Issuance Request	Data Model	● Verifiable Presentation Request Specification ● OIDC Credential Provider "Credential Request"	● Verifiable Presentation for purpose of proof of identifier control[27] ● OIDC Credential Provider "Credential Request"
Issuance Response	Data Model	● Verifiable Presentation or Verifiable Credential ● OIDC Credential Provider "Credential Response"	● Verifiable Presentation[28] ● OIDC Credential Provider "Credential Response"
Presentation Request	Data Model	● Presentation Exchange "Presentation Definition" ● Verifiable Presentation Request Specification	● Verifiable Presentation Request Specification (Query)[29] ● [Extension] Presentation Definition
Presentation Response	Data Model	● Presentation Exchange "Presentation Submission" ● Verifiable Presentation	● Verifiable Presentation ● [Extension] Presentation Submission
Identifier (Issuer and Subject)	Data Model	Decentralized Identifier	Generation: ● did:key Resolve: ● did:key ● did:web ● [Extension] did:sidetree

[27] See VP Request specification Example 4 https://w3c-ccg.github.io/vp-request-spec/#example-4-a-didauth-response

[28] Using the VP to wrap the issued VC(s) supports multiple VCs as well as holder signature for cases where the holding party conveying the credentials may not be the issuing party

[29] https://w3c-ccg.github.io/vp-request-spec/#format

Authentication	Protocol	Variety of authentication techniques, ranging from traditional methods such as OIDC and SAML to DID-based (or DID-aware) methods such as SIOP-DID or OIDC Credential Provider	• OIDC, SAML (using existing issuer/RP relationship with learner) • [Extension] OIDC Credential Provider
Confidential Storage	Data Models and Protocols	Emerging alignment in Confidential Storage[30]	Local secure storage with secure backup/export options
Key / Secret Vault	Protocol	WebKMS	Local secure storage with secure backup/export options
Verification/Validation	Standard-based procedures	VC/VP Verification	VC/VP Verification
VC APIs	API specification	vc-http-api, which includes endpoints for: • VC Issuance • VC Proof • VP Verification • VP Submission	vc-http-api
Proof Mechanism[31]	Data Model, Procedures	• Linked Data Proofs 1.0 • JWTs	Prove: • Ed25519Signature2020 LinkedData Proof Suite • [Extension] JWT • [Extension] BBS+ Signatures 2020 Resolve: • Linked Data Proofs 1.0 • JWT
Status Checking Mechanism	Data Model, Procedures	Status List 2021	Status List 2021

Table B2: Data Format Extensions

Type	Description
LER	Flavor of Verifiable Credential with LER semantic support

[30] The Confidential Storage draft specification represents emerging alignment among different decentralized identity-based secure storage specifications, such as Encrypted Data Vaults and Identity Hubs.
[31] See Verifiable Credentials Flavors Explained for a thorough overview of VCs and proof mechanisms, including those supporting selective disclosure

Open Badge v2.0 (example)	Open Badge v2, used to demonstrate non-VC data support and interoperability

Annex C: External Interface Calls and Payloads

Non-normative; variable per wallet implementation. This section describes reference implementation, which establishes a pattern other wallet implementations may follow.

Deep Link Example

This example demonstrates how the reference implementation uses a deep link to open the wallet to an issuance request screen, pre-populated with the necessary parameters. Parameter names are non-standard.

```
dccrequest://request?       // deep link to open wallet
    &auth_type=<auth_type>  // code | saml
    &issuer=<issuer>        // depends on previous param; for OIDC the issuer URI[32]
    &vc_request_url=<vc_request_url>   // credential request endpoint
    &challenge=<challenge>  // VP challenge (GUID)
```

Example Issuance Request

Example of an Issuance Request payload that is generated by the wallet and sent to the issuer. The `holder` field contains the identifier generated by the wallet, and the proof is signed by the corresponding identifier, enabling the issuer to ensure the submitter "controls" the identifier ("proof of identifier control")

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1"
  ],
  "type": "VerifiablePresentation",
  "holder": "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "proof": {
    "type": "RsaSignature2018",
    "created": "2020-09-14T21:19:10Z",
    "proofPurpose": "assertionMethod",
    "domain": "http://www.example.com",
    "challenge": "e506f108-9c74-4c52-b84c-a7019c83328c",
    "verificationMethod": "did:example:ebfeb1f712ebc6f1c276e12ec21#keys-1",
    "jws": "pYw8XNi1..Cky6Ed="
```

[32] See OIDC Provider configuration: https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig

```
  }
}
```

Example of request to issuer:

```
POST /<request_endpoint>
    Content type: application/json
    Authorization header
    Parameter:
       - issuanceRequest (VerifiablePresentation)
```

Example Issuance Response

Example of an Issuance Response payload that is created by the issuer and conveyed to the wallet. The VC is wrapped in a VP to allow for multiple VCs to be returned, as well as to accommodate scenarios where the entity delivering the credentials is different from the issuer.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1"
  ],
  "type": "VerifiablePresentation",
  "holder": "did:web:digitalcredentials.github.io",
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://www.w3.org/2018/credentials/examples/v1",
        "https://w3id.org/security/jws/v1",
        "https://w3id.org/dcc/v1"
      ],
      "id": "https://digitalcredentials.github.io/samples/certificate/1fe91f0f-4c64-48c8-
bfc8-7132f75776fe/",
      "type": [
        "VerifiableCredential",
        "LearningCredential"
      ],
      "issuer": {
        "type": "Issuer",
        "id": "did:web:digitalcredentials.github.io",
        "name": "Sample Issuer",
        "url": "https://digitalcredentials.github.io/samples/"
      },
      "issuanceDate": "2021-01-19T18:22:34.772810+00:00",
      "credentialSubject": {
        "type": "Person",
        "id": "did:example:456",
        "name": "Percy",
        "hasCredential": {
          "type": [
            "EducationalOccupationalCredential",
            "ProgramCompletionCredential"
          ],
          "name": "DCC Program Completion Credential",
          "description": "Awarded on completion of the digital credential program",
          "awardedOnCompletionOf": {
            "type": "EducationalOccupationalProgram",
            "identifier": "program-v1:Sample",
            "name": "Digital Credential Program",
            "description": "Educational program teaching how work with digital credentials",
            "numberOfCredits": {
              "value": "1"
            }
          }
        }
      },
      "proof": {
          ...
      }
    }
  ],
  "proof": {
      ...
  }
}
```

Annex D: Issuer-Side Credential Issuance Flow

This annex demonstrates how the wallet protocols can be adapted into issuer flows.

Issuer Sequence and Adaptations

Figure D1 expands the credential Issuance flow to include a representative issuer sequence. This also demonstrates a critical distinction of decentralized identity issuance flows from traditional methods from an issuer perspective -- the need to collect the learner identifier before issuing the credential.

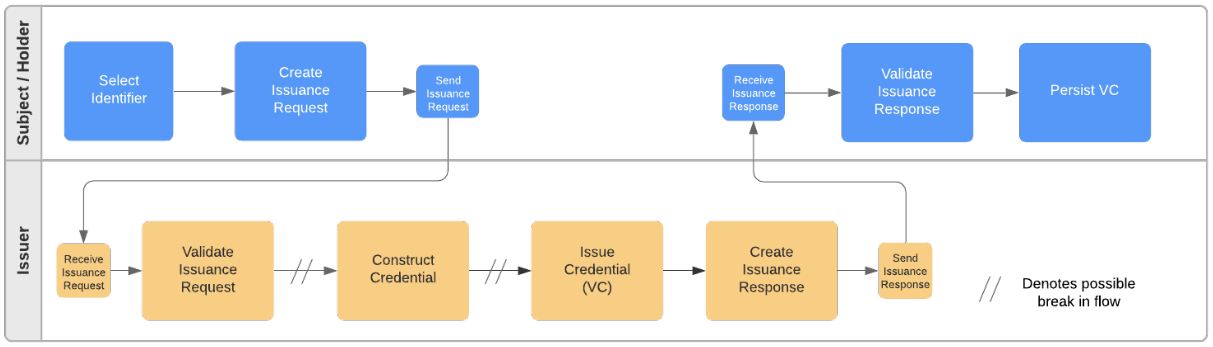
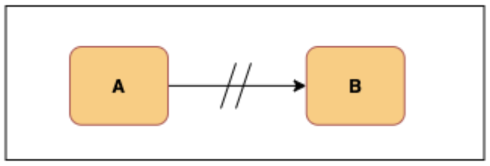


Figure D1: Expanded Credential Issuance Flow with Issuer Steps

The “break in flow” referenced in Figure D1 accommodate two categories of adaptations:

1. Asynchronous workflows (in which the wallet may not expect an immediate response)
2. Out-of-band triggers, including scenarios such as:
    a. Issuer already knows the subject identifier
    b. Issuer pre-populates credentials by batch, but adds the identifier and issues the credential on-demand

Figure D2 demonstrates the “break in flow” more precisely.



...shorthand for

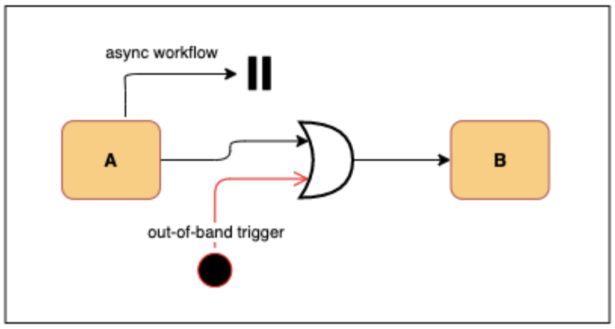


Figure D2: Expansion of “break in flow”

Data Map

Figure D3 demonstrates how standard inputs into a credential (including those coming from a student information system or other record source) become merged with the subject identifier, and then issued as a Verifiable Credential.

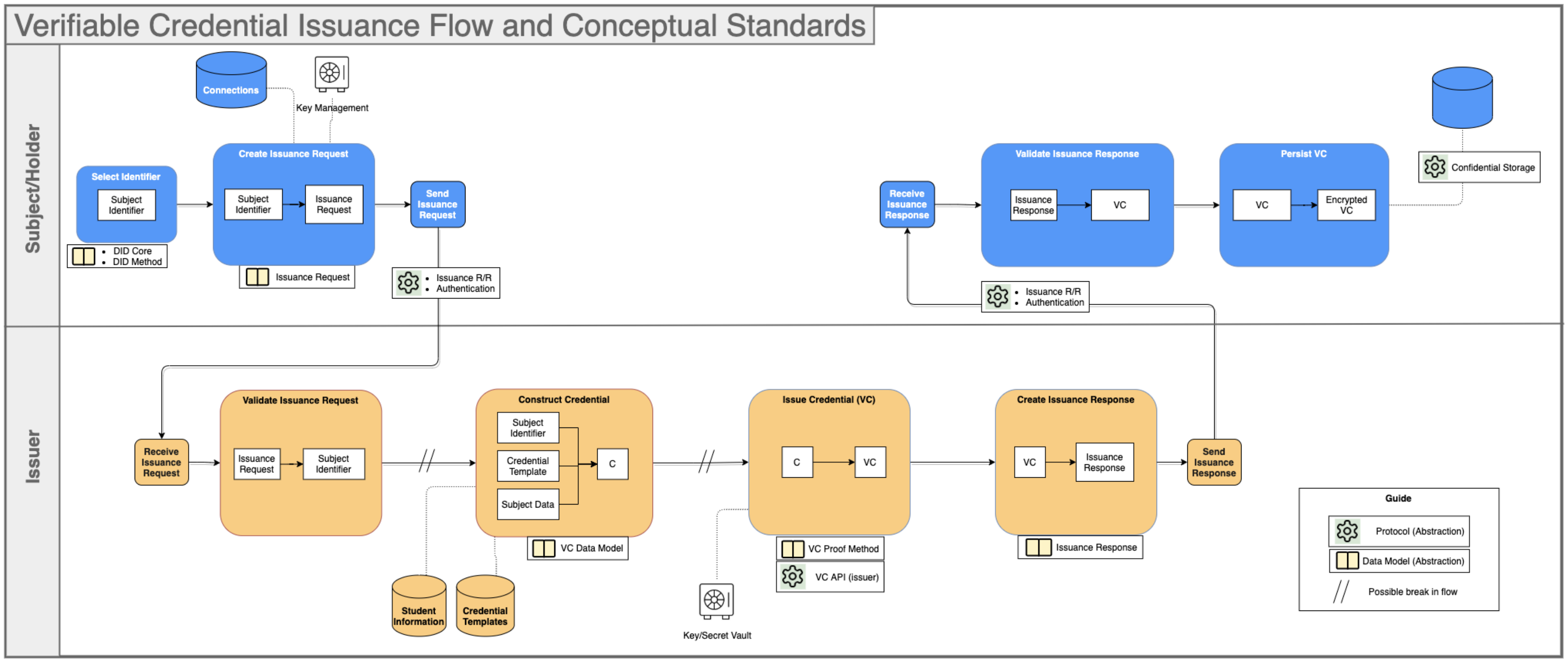


Figure D3: Expanded Issuance Request/Response Flow with Issuer Step

Annex E: Design Decision Factors

Decentralized Identifiers

Decentralized Identifiers offer an interoperable, portable, and individual-controlled form of identifiers for use in VCs. There are many emerging DID methods (implementations of DIDs) with different characteristics, which the wallet can support flexibly[33] based on the interoperable IdentifierPlugin data model.

For purposes of this work, these were the desired characteristics for selecting which DID methods to include as part of the initial wallet implementation[34]:

- Low or no cost
- Portable, learner-controlled, decentralized
- Standards-compliant, has multiple implementations
- Cryptographic keys are rotatable OR issuer supports credential reissuance

As of the time of publication, did:key provided the best fit despite the lack of ability to rotate cryptographic keys; the workaround is that issuers must reissuance for such deployments.

Many DID methods were considered. Those that came close to satisfying the above desired characteristics (they had challenges at the time of publication but are promising near-term alternatives):

- did:sidetree: specification was not yet stable, but expected to be soon (with multiple implementations)
- did:peer: Suitable when peer relationship is already established (with RP), which would impose additional prerequisites

For issuer identifiers, did:web was selected, largely a matter of trust and convenience in our early implementations for issuers with bootstrap trust from their web domains (and corresponding established processes for maintaining updates).

[33] Including non-DID (URI) identifiers
[34] The DID Method Rubric (https://w3c.github.io/did-rubric/) offers guidelines for how to choose DID methods.

Annex F: Proof of Identifier Control, Illustration

This section illustrates a flow establishing proof of identifier control, as well as the wallet's role.

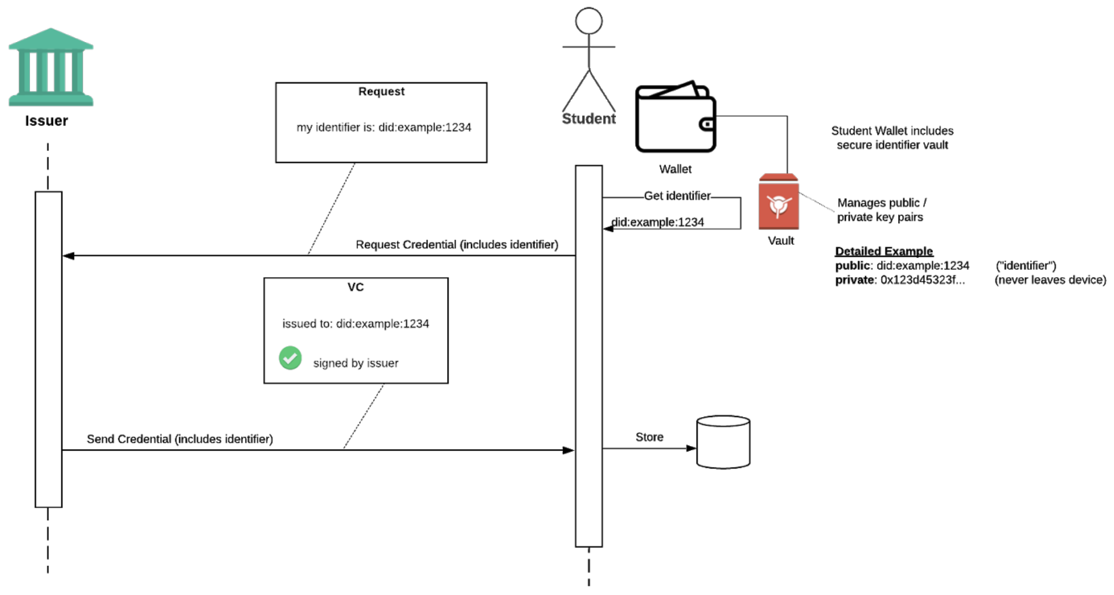
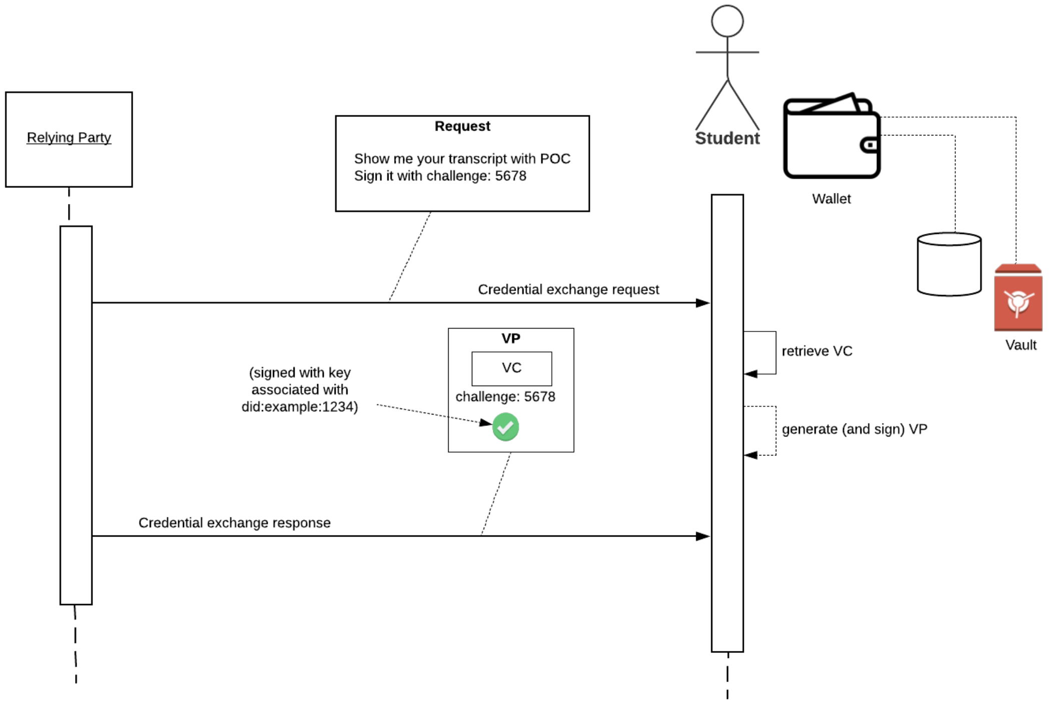

Issuer
Request
my identifier is: did:example:1234
Student
Wallet
Student Wallet includes secure identifier vault
Get identifier
did:example:1234
Vault
Manages public / private key pairs
Request Credential (includes identifier)
Detailed Example
public: did:example:1234 ("identifier")
private: 0x123d45323f... (never leaves device)
VC
issued to: did:example:1234
signed by issuer
Send Credential (includes identifier)
Store

Figure E1: At issuance time

1. At issuance time, the learner's wallet, via the wallet's secure identifier vault, generates an identifier that it will be able to prove control over. In the initial wallet implementation, this is achieved as follows:
    a. The wallet has a secure vault that manages identifiers. These identifiers are backed by (and bound to) cryptographic keys.
    b. At issuance time, the wallet sends a credential request that includes the identifier
    c. The issuer includes that identifier in the VC as the subject of the VC



Relying Party
Request
Show me your transcript with POC
Sign it with challenge: 5678
Student
Wallet
Vault
Credential exchange request
retrieve VC
VP
VC
challenge: 5678
(signed with key associated with did:example:1234)
generate (and sign) VP
Credential exchange response

Figure E2: At exchange time

2. When providing a credential to a relying party, the wallet is able to:
    a. Look up the cryptographic key material in its vault that's associated with the subject identifier of the credential(s).
    b. Create a cryptographic signature with that key material, proving control over the identifier.