# Wallet and Credentialing Framework for Interoperability

[**Wallet and Credentialing Framework for Interoperability**](#_83qi8j6gfvz8) **1**

[**Objectives of this document**](#_zhgooyti5o7h) **1**

[**Components**](#_97a85iq9zmq5) **2**

[**Identifier**](#_szl5adjai3m1) **3**

[**Credential Structure**](#_z5qszvmsao9m) **3**

[**Credential Schemas**](#_kftc6wf2w6mc) **3**

[**Schemas**](#_14e4aye62run) **3**

[**Signatures**](#_j7m3es144ylm) **3**

[**Verification Method**](#_gkuttpen2ja8) **3**

[**Wallet to Wallet communication**](#_rachryss7hwv) **4**

[**Messaging Standard for PI Verification**](#_j7kmu8z66acz) **4**

[**Interaction between Identity Wallet and PI Verification messaging service (VRS)**](#_f5aahvkt2gdn) **4**

[Wallet configuration](#_qgswlf68nkf5) 4

[Configuration of Credential Issuer’s wallet](#_3y39zznfc1if) 5

[Configuration of Trading Partner’s wallet](#_gz7w6pedlonz) 5

[**Interactions between Credential Issuer’s wallet and Trading Partner’s wallet**](#_b9fsvzivh688) **6**

[Credential Issuer’s wallet offers/issues “CompanyIdentityVerificationCredential” to Trading Partner’s wallet](#_24nufuupl1zn) 6

[Credential Issuer’s wallet requests/obtains presentation of “CompanyIdentityVerificationCredential” from Trading Partner’s wallet](#_n6wperulaq9t) 6

[**Interaction between Trading Partner’s wallet and PI Verification messaging service (VRS)**](#_w888bjv2k5mo) **7**

[Create ATP VP (Verifiable Presentation)](#_jahddt85v5e6) 7

[Verify ATP VP (Verifiable Presentation)](#_m1ertg6gbtbb) 8

[Verify Signed Verified Presentation (JWT)](#_o4c1mr2i10fl) 9

# Version

Version 0.8

Last update: March 31th

# Objectives of this document

This documentation has the objective to support interoperability among ATP solutions. Based on these specifications, stakeholders like trading partners, service providers or standard organizations have an insight into the used components of the ATP Pilot implementation.

This documentation is based on the “Product Verifications” use case, but shall also support the evaluation of the applicability of the used standards for further use cases - within the DSCSA regulatory framework and beyond.

# Components

Components that are required to ensure interoperability in the context of ATP:

|  |  |
| --- | --- |
| **Component** | **Realization** |
| Identifier | [W3C DIDs](https://www.w3.org/TR/did-core/)   * [ETHR DID Method Spec](https://github.com/decentralized-identity/ethr-did-resolver/blob/master/doc/did-method-spec.md) * [DID:WEB](https://w3c-ccg.github.io/did-method-web/) |
| Credential Structure | [W3C VCs](https://www.w3.org/TR/vc-data-model/) |
| Credential Schemas | [Credential Schema documentation](https://docs.google.com/document/d/1DP8IwI1iFxRXYp9EVmqLgZUVK-MItTaIDji5FvP6dLU/edit):   1. Identity Credential 2. ATP Credential |
| Schemas | GS1 Web Vocab |
| Signatures | [JSON Web Signature 2020](https://w3c-ccg.github.io/lds-jws2020/)   * using Secp256k1 curve |
| Verification Method | [JSON Web Key 2020](https://w3c-ccg.github.io/lds-jws2020/#json-web-key-2020)   * using Secp256k1 curve |
| Wallet to Wallet communication | * Aries RFCs (key ones)   + [Issue Credential Protocol v2](https://github.com/hyperledger/aries-rfcs/tree/master/features/0453-issue-credential-v2)   + [Credential Manifest](https://github.com/hyperledger/aries-rfcs/tree/master/features/0511-dif-cred-manifest-attach)   + [Present Proof Protocol v2](https://github.com/hyperledger/aries-rfcs/tree/master/features/0454-present-proof-v2)   + [Presentation Exchange](https://github.com/hyperledger/aries-rfcs/tree/master/features/0510-dif-pres-exch-attach) * Decentralized Identity Foundation specs   + [DIDComm Messaging](https://identity.foundation/didcomm-messaging/spec/) v2   + [Credential Manifest](https://identity.foundation/credential-manifest/)   + [Presentation Exchange](https://identity.foundation/presentation-exchange/) |
| Messaging Standard to exchange ATP Credentials | GS1 Lightweight Messaging Standard for PI Verification |
| Credential issuance for ATPs | Open APIs |
| PI Verification with ATP Credentials | Open APIs |

# Identifier

A DID is a globally unique identifier developed specifically for decentralized systems as defined by the [**W3C DID specification**](https://w3c-ccg.github.io/did-spec/). DIDs enable interoperable decentralized Self-Sovereign Identity management. A DID is associated with exactly one DID Document.

# 

# Credential Structure

A Credential that includes a proof from its issuer. Typically this proof is in the form of a digital signature. Based on the definition provided by the [**W3C Verifiable Claims Working Group**](https://www.w3.org/2017/vc/WG/).

# 

# Credential Schemas

The credential schema are documented in the ATP Pilot document: “[**Documentation of the credential schemas designed for the ATP Pilot**](https://docs.google.com/document/d/1JZTDvlP0B9MA4khNEj2_8oMjZHNVdRNwKp_fPJ43EZM/edit?usp=sharing)“

# 

# Schemas

JSON Schemas for aforementioned credentials to be defined and anchored using GS1 Web Vocab

# 

# Signatures

The proofs (including signatures) of Verifiable Credentials and Verifiable Presentation to be generated and verified in conformance with [JSON Web Signature 2020](https://w3c-ccg.github.io/lds-jws2020/) that is one of relevant LD Signature Suites to the date that are registered at [Linked Data Cryptographic Suite Registry](https://w3c-ccg.github.io/ld-cryptosuite-registry/#jsonwebsignature2020).

# 

# Verification Method

1. Signing and verification of Verifiable Credentials to be achieved using key pairs based on Secp256k1 curve.
2. Public keys to be represented as [JWK](https://tools.ietf.org/html/rfc7517) that is [a supported format by DID Doc spec](https://www.w3.org/TR/did-core/#verification-method-properties).
3. Public keys to be referenced in a respective DID Document and associated with specific proof purposes via [the "Verification Relationship" concept that is defined in DID Core spec](https://www.w3.org/TR/did-core/#verification-relationships) for further verification.

# Wallet to Wallet communication

Wallet to wallet communication between Credential Issuer and Trading Partner Identity Wallet is based on a set of interoperable and DID method agnostic Aries RFCs that find its roots in working items of Decentralized Identity Foundation. This [High Level](https://drive.google.com/file/d/1ml6VNmhsef-Bw1oPoCP6OsfM9-EGDrKF/view?usp=sharing) chart can be used as guidance for the subject workflow.

All messages to be packed (signed/encrypted) in JWM envelopes using [DIDComm v2](https://identity.foundation/didcomm-messaging/spec/) spec.

* Aries RFCs (key ones)
  + [Issue Credential Protocol v2](https://github.com/hyperledger/aries-rfcs/tree/master/features/0453-issue-credential-v2)
  + [Credential Manifest](https://github.com/hyperledger/aries-rfcs/tree/master/features/0511-dif-cred-manifest-attach)
  + [Present Proof Protocol v2](https://github.com/hyperledger/aries-rfcs/tree/master/features/0454-present-proof-v2)
  + [Presentation Exchange](https://github.com/hyperledger/aries-rfcs/tree/master/features/0510-dif-pres-exch-attach)
* Decentralized Identity Foundation specs
  + [DIDComm Messaging](https://identity.foundation/didcomm-messaging/spec/) v2
  + [Credential Manifest](https://identity.foundation/credential-manifest/)
  + [Presentation Exchange](https://identity.foundation/presentation-exchange/)

# 

# Messaging Standard for PI Verification

To exchange verifiable presentations of VCs in form of a JSON Web Token, the VRS providers integrating the Spherity Credentialing Service are able attach the JWT to the header of the GS1 Lightweight Messaging Standard for PI Verifications.

# Credential Issuance for ATPs

The full documentation of the APIs can be found here.

<https://documenter.getpostman.com/view/11378415/T17FAToR>

# 

# Interactions between Credential Issuer’s wallet and Trading Partner’s wallet

In order to issue credentials, **Credential Issuer** will be utilizing the implementation of [*“issue-credential” protocol v2*](https://github.com/hyperledger/aries-rfcs/tree/master/features/0453-issue-credential-v2) *using* [*DID:WEB*](https://w3c-ccg.github.io/did-method-web/) *and/or* [*DID:ETHR*](https://github.com/decentralized-identity/ethr-did-resolver/blob/master/doc/did-method-spec.md) *identifiers.*

## **Credential Issuer’s** walletoffers/issues “**CompanyIdentityVerificationCredential**”to **Trading Partner’s** wallet

**Note:** The latest Spheirty Wallet API docs for Credential Offer flow can be found [here](https://docu.spherity.com/#686f089a-9630-4b76-b4e1-feacf0351a25).

Please find the flow overview below.

1. Initiate communication from **Credential Issuer** side
   1. **Credential Issuer** sends "offer-credential" message to **Trading Partner** AND provide custom VC ID as "credInput.id" in the request body
2. **Trading Partner** responds with "request-credential" message in an automatic manner (message filter should be set up during configuration of a **Trading Partner**)
3. **Credential Issuer** sends "issue-credential" message to **Trading Partner** (a manual step on **Credential Issuer** side)
4. **Trading Partner** stores a credential from an incoming "issue-credential" message in an automatic manner (message filter should be set up during configuration of a **Trading Partner**)

## **Credential Issuer’s** wallet requests/obtains presentation of “**CompanyIdentityVerificationCredential**” from **Trading Partner’s** wallet

**Note:** The latest Spherity Wallet API docs for Request Presentation flow can be found [here](https://docu.spherity.com/#73237d1a-e580-468d-9288-607a5afe48d5).

Please find the flow overview below.

1. Initiate communication from **Credential Issuer** side
   1. **Credential Issuer** requests “CompanyIdentityVerificationCredential” from **Trading Partner**
      1. Two API endpoints MUST be triggered one-be-one
         1. Create a message filter to save the presentation in case "presentation" message is received from a particular DID of **Trading Partner** whom you did send "request-presentation" message beforehand
         2. **Credential Issuer** sends "request-presentation" message to **Trading Partner**
2. **Trading Partner** responds with "presentation" message in an automatic manner (message filter should be set up during configuration of a **Trading Partner**)
3. **Credential Issuer** stores a presentation from an incoming "presentation" message in an automatic manner (message filter should be set up simultaneously at the time of sending "request-presentation" message)

# Interaction between Trading Partner’s wallet and PI Verification messaging service (VRS)

**Note:** The latest Spherity Wallet API docs for ATP VP Issuance and ATP VP Verification can be found [here](https://documenter.getpostman.com/view/11378415/T17FAToR#e01839c4-fb3e-4c7c-b852-7f9e0e47ec44).

## **API - Create** **ATP** **Credential** **Verifiable Presentation**

This API describes the generation of a signed Verifiable Presentation in form of a JWT. To audit trail all PI messages the piMessageHash is combined with the presentation of the Verifiable Credential.

Verifiable Credential types that can be fetched from the Trading Partners wallet are:

* DSCSA Wholesaler ATP Credential
* DSCSA Manufacturer ATP Credential
* DSCSA Dispenser ATP Credential

Please consider that the Trading Partner needs to have a respective credential in its wallet to successfully create a JWT.

Within the API the following steps are done:

1. Create basic *log* *record* for an incoming request
   1. “***holder***” property from request body
   2. “***vcType***” property from request body
   3. “***corrUUID***” property from request body
   4. “***piMessageHash***” property from request body
   5. “***creator***” property from decoded JWT(“***email***” property from Bearer token)
2. Fetch ATP VC from database via Holder DID (“***holder***”) & VC Type (“***vcType***”)
3. Throw **404** when no ATP VCs found
4. Filter the latest ATP VC via Issuance Date (“***issuedOn***” property that holds UNIX Epoch milliseconds)
5. Update *log* *record* with “***verifiableCredential***” property based on found ATP VC
6. Verify ATP VC based on a *verification* *scope* that was provided in the request body **OR** skip in case verification scope for VC is not provided in request’s query params.
   1. Update *log* *record* with received ATP VC verification result
   2. Throw **500** when ATP VC verification result include errors **OR** return **200** in case query params include **gracefulErrors=true**
7. Find wallet record that holds provided Holder DID (“**holder**”)
8. Pick the first keypair of Holder DID (“**holder**”)
9. Issue ATP VP (JWT) with the latest ATP VC & “***piMessageHash****”* as “*nonce” inside*
10. Update *log* *record* with “***verifiablePresentation***” property based on issued ATP VP
11. Verify ATP VP based on a *verification* *scope* that was provided in the request body **OR** skip in case verification scope for VP is not provided in request’s query params.
    1. Update *log* *record* with received ATP VP verification result
    2. Throw **500** when ATP VP verification result include errors **OR** return **200** in case query params include **gracefulErrors=true**
12. Return **200** and the latest *log record* inside response body

This API describes the generation of a signed Verifiable Presentation in form of a JWT. To audit all PI messages the PI MessageHash is combined with the presentation of the Verifiable Credential.

The PI Message Hash is created by the VRS by calculating: SHA256(GTIN+SERIAL+LOT+EXPIRY.substring(0,4))

Next to that it also takes a few request parameters.

## **API - Verify ATP Credential Verifiable Presentation**

This API describes the verification of a received JWT. The VRS provider needs to send the JWT together with the respective corrUUID to the verifier wallet.

In the verification API the following will be checked::

1. the signature on the JWT
2. the signature on the DSCSA ATP Credential issuer
3. the credential expiration date
4. the revocation registry

The API will respond with error messages if one of these steps fails. Please find error messages and codes in the examples of the [API documentation](https://documenter.getpostman.com/view/11378415/T17FAToR).

Within the API the following steps are done:

1. Create basic *log* *record* for an incoming request
   1. “***verifiablePresentation***” property from request body
   2. “***corrUUID***” property from request body
   3. “***creator***” property from decoded JWT(“***email***” property from Bearer token)
2. Decode JWT i.e. the value of “***verifiablePresentation***”
3. Update *log* *record* with “***piMessageHash***” property from decoded ATP VP’s “***nonce***”
4. Find ATP VC within decoded ATP VP data
5. Update *log* *record* with “***verifiableCredential***” property based on found ATP VC
6. Import counterparty into your wallet based on:
   1. Subject DID (“***credentialSubject.id***”) value of ATP VC as “***did***”
   2. Company name (“***credentialSubject.[“Subject Company Name”*]**) value from ATP VC as “***didName***”
7. Verify ATP VP based on a *verification* *scope* that was provided in the request body **OR** skip in case verification scope for VP is not provided in request’s query params.
   1. Update *log* *record* with received ATP VP verification result
   2. Throw **500** when ATP VP verification result include errors **OR** return **200** in case query params include **gracefulErrors=true**
8. Verify ATP VC based on a *verification* *scope* that was provided in the request body **OR** skip in case verification scope for VC is not provided in request’s query params.
   1. Update *log* *record* with received ATP VC verification result
   2. Throw **500** when ATP VC verification result include errors **OR** return **200** in case query params include **gracefulErrors=true**
9. Return **200** and the latest *log record* inside response body