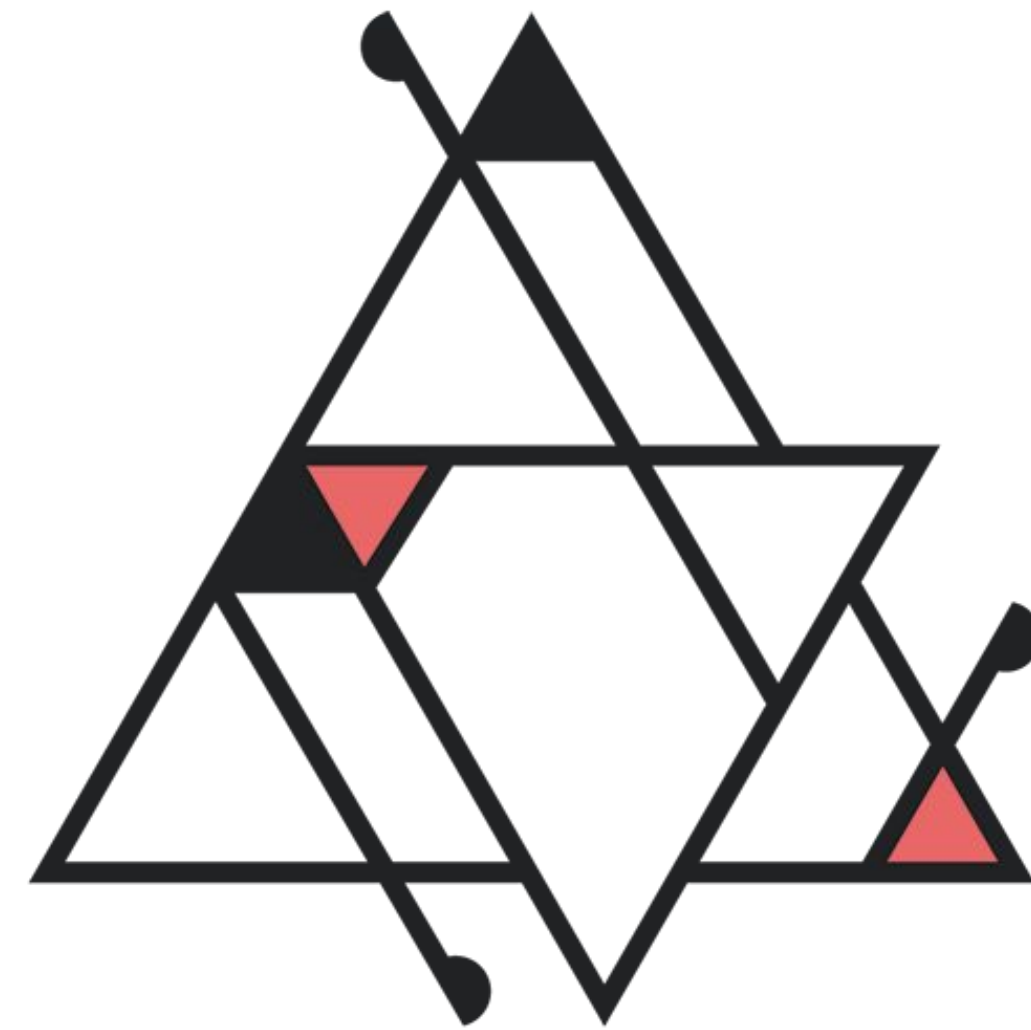




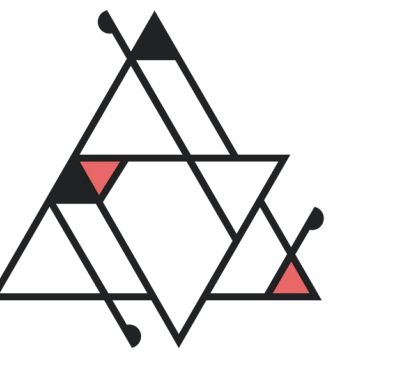
# Hyperledger Aries Framework JavaScript 0.4.0



A N I M O

Ariel Gentile (2060), Karim Stekelenburg (Animo), Berend Sliedrecht (Animo)  
11/07/2023





ANIMO

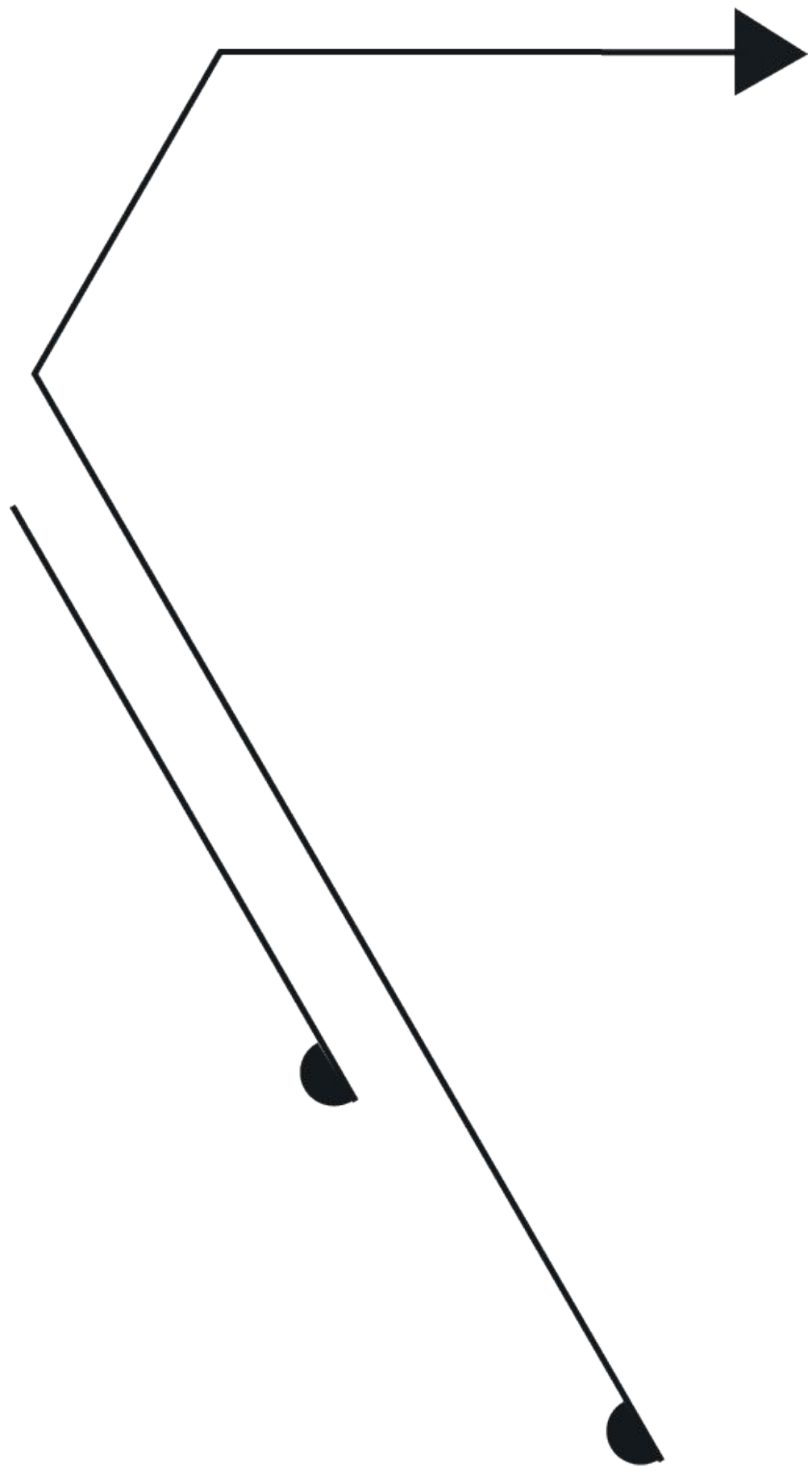
2060

# Index

---

---

- Introduction
- Release overview
- Demo
- Code examples
- Documentation
- What's next?
- Q/A





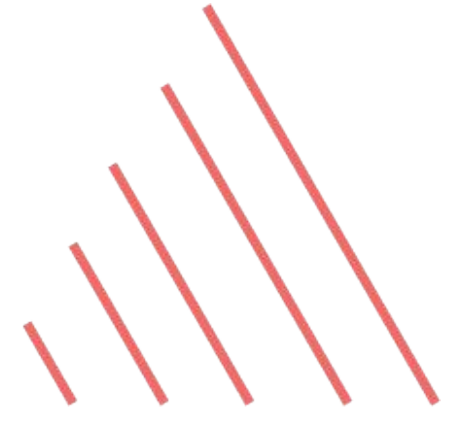
ANIMO

2060

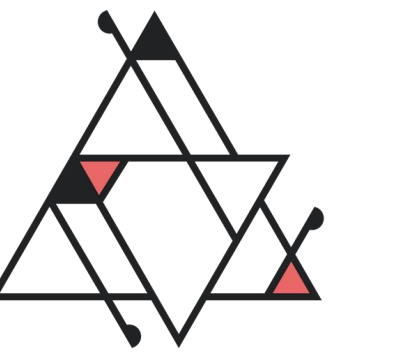


# Introduction





# Hyperledger Aries Framework JavaScript



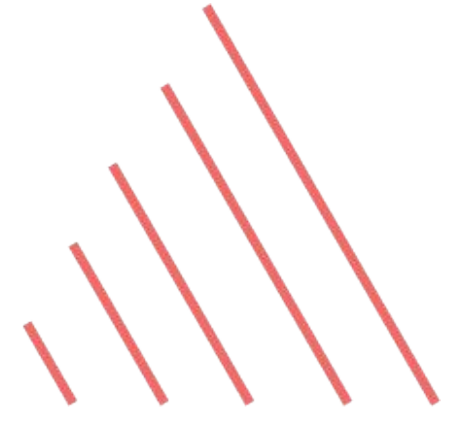
ANIMO



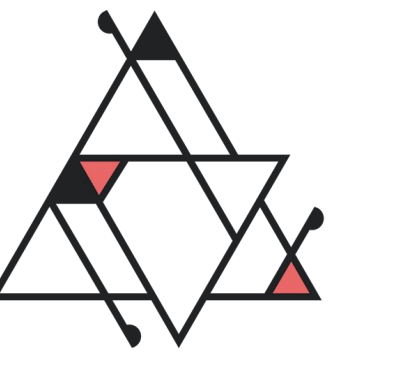
- A framework for implementing self-sovereign/decentralized identity solutions
- Based on the Hyperledger Aries standards
  - Hyperledger Aries came from the Hyperledger Indy project
  
- Multi-platform, meaning it can run:
  - Server-side (Node.JS)
  - On mobile devices (React Native)



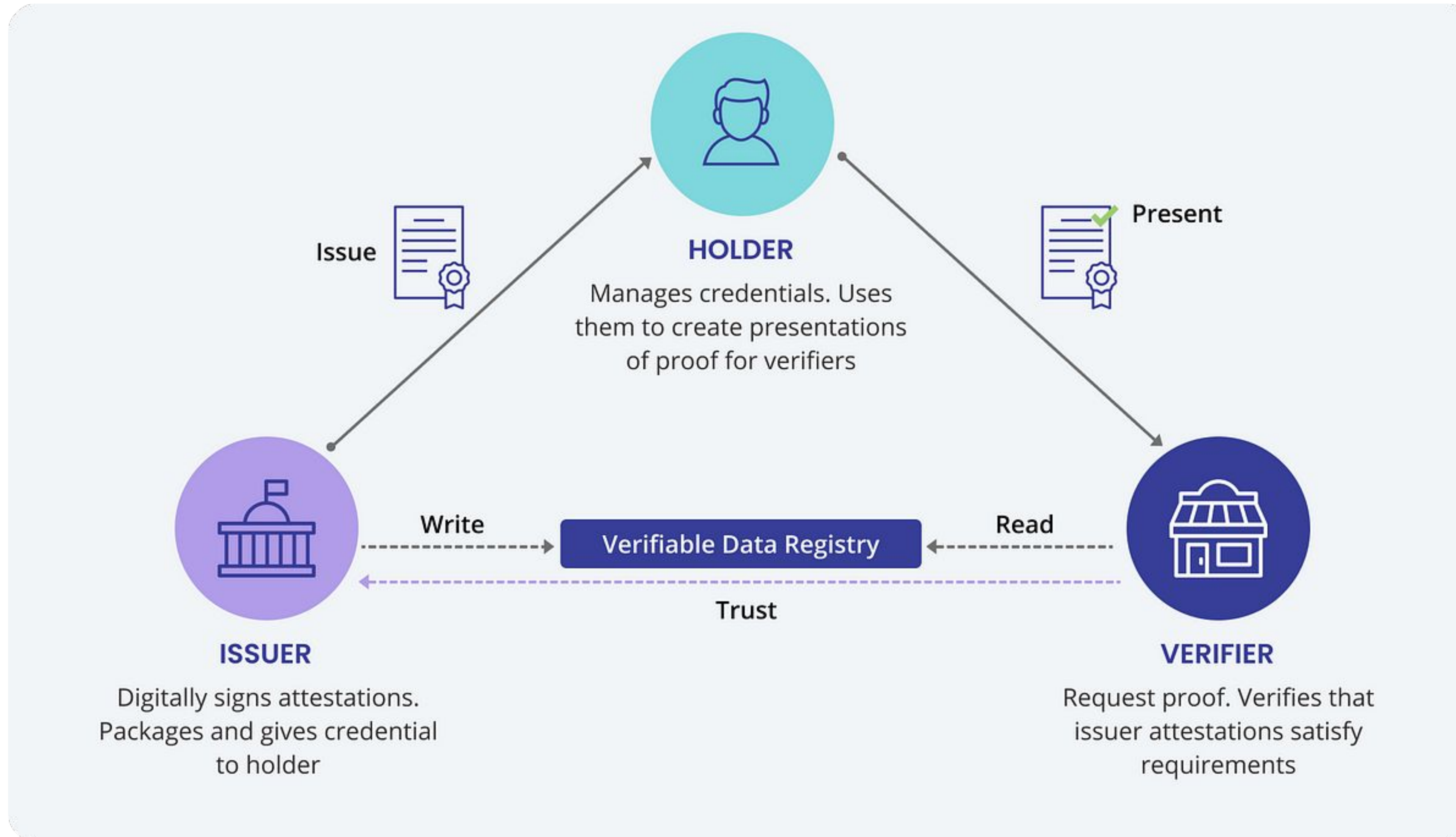




# Covers all roles in the ecosystem



ANIMO





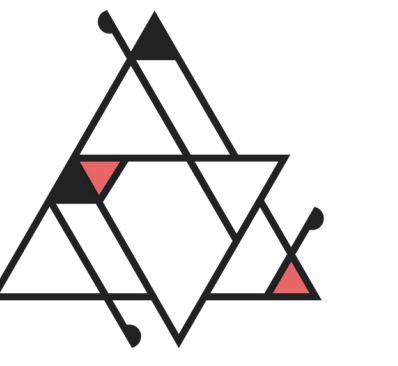
ANIMO

2060



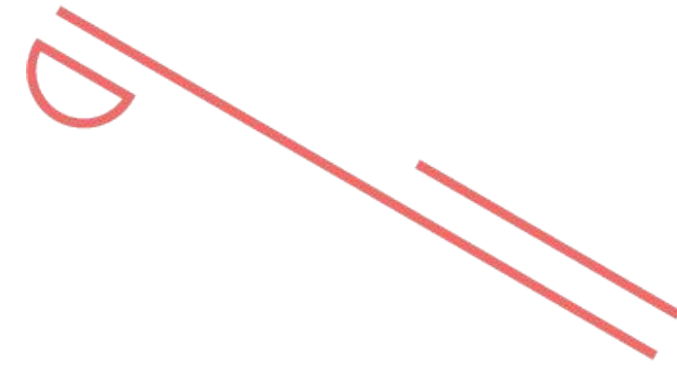
# Release Overview



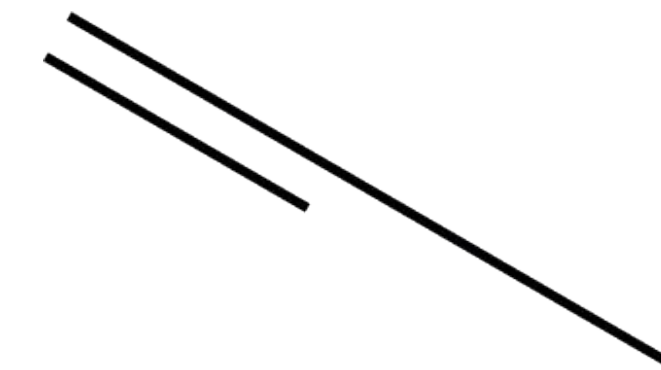


ANIMO

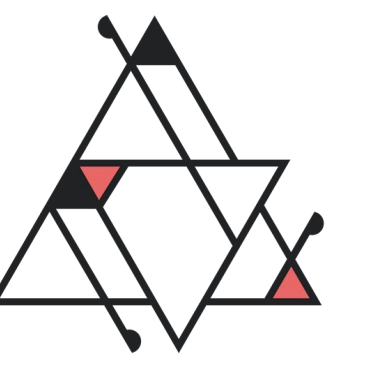
2060



# Modularization



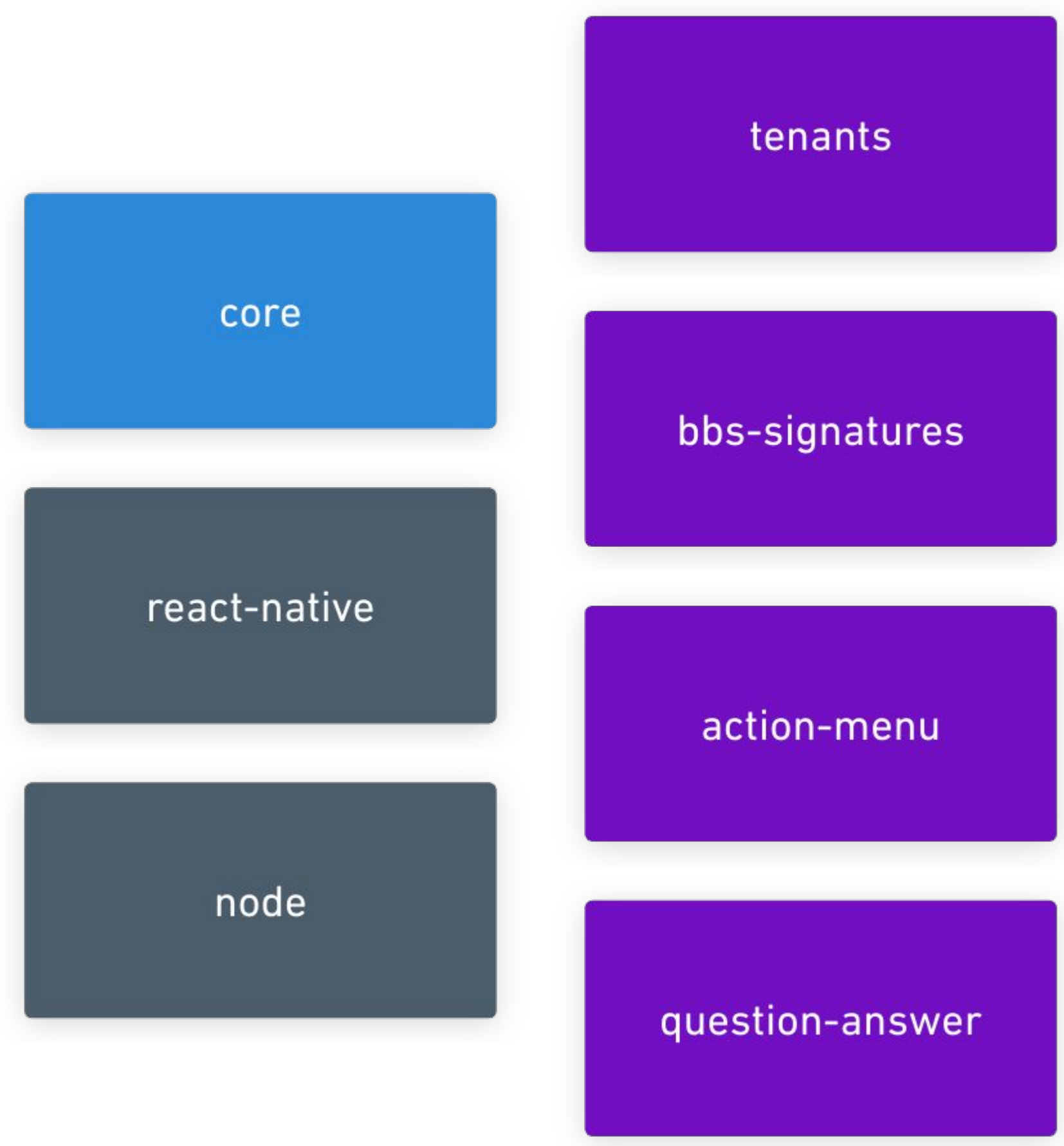




ANIMO



# 0.3.0

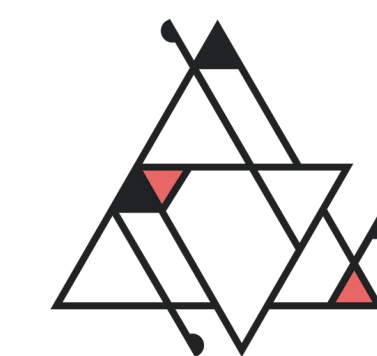
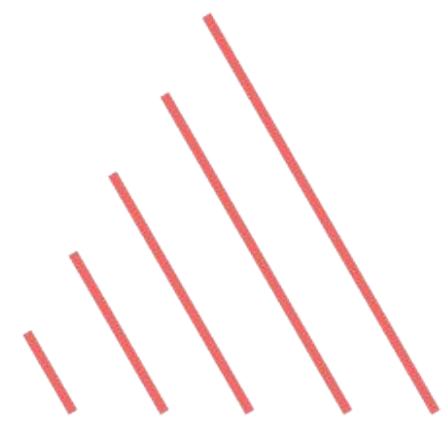


# 0.4.0



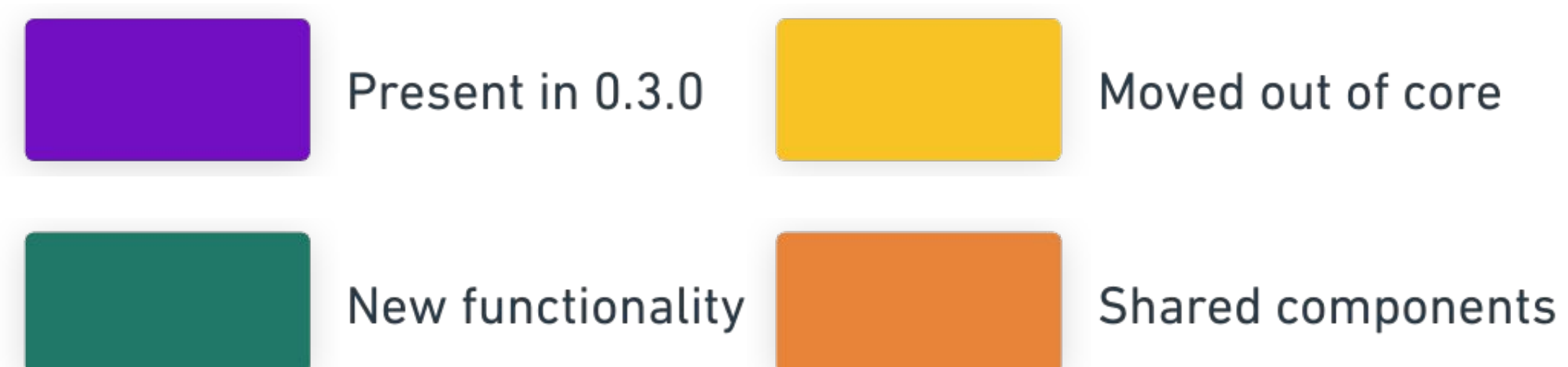
	Present in 0.3.0		Moved out of core
	New functionality		Shared components

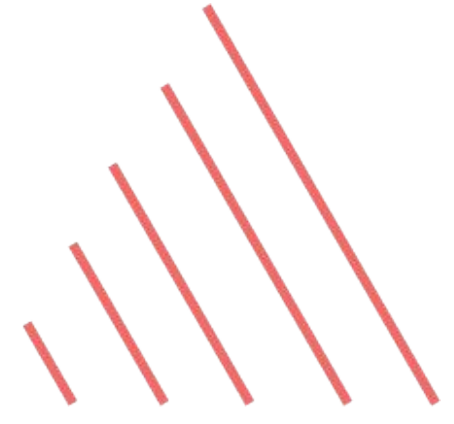




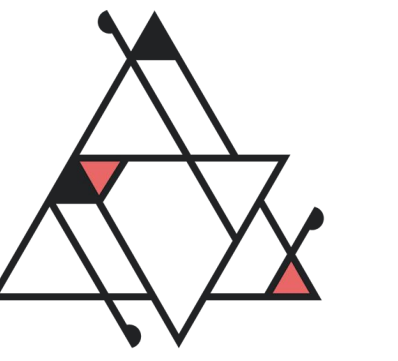
ANIMO

2060





# Indy SDK

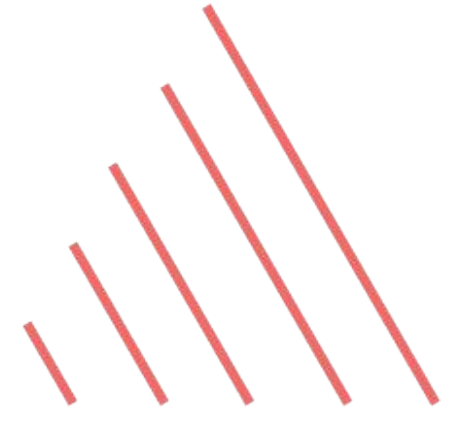


ANIMO

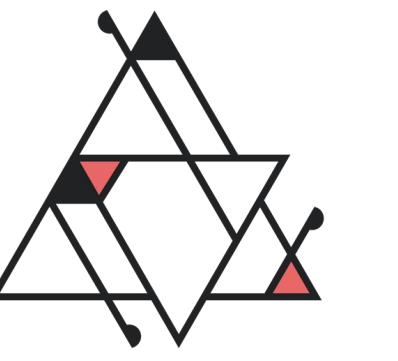


- Indy SDK used to be part of the core of the framework, and used for:
  - Secure storage
  - Cryptography (powered by Ursa)
  - AnonCreds related functionality
  - Communication with Indy network
  
- In the meantime...
  - AnonCreds became a standalone standard
  - Support for other credential formats (e.g. W3C JSON-LD) has been added
  
- As a result:
  - Users were forced to include AnonCreds related logic
  - Users were forced to include Indy related logic





# Shared Components



ANIMO

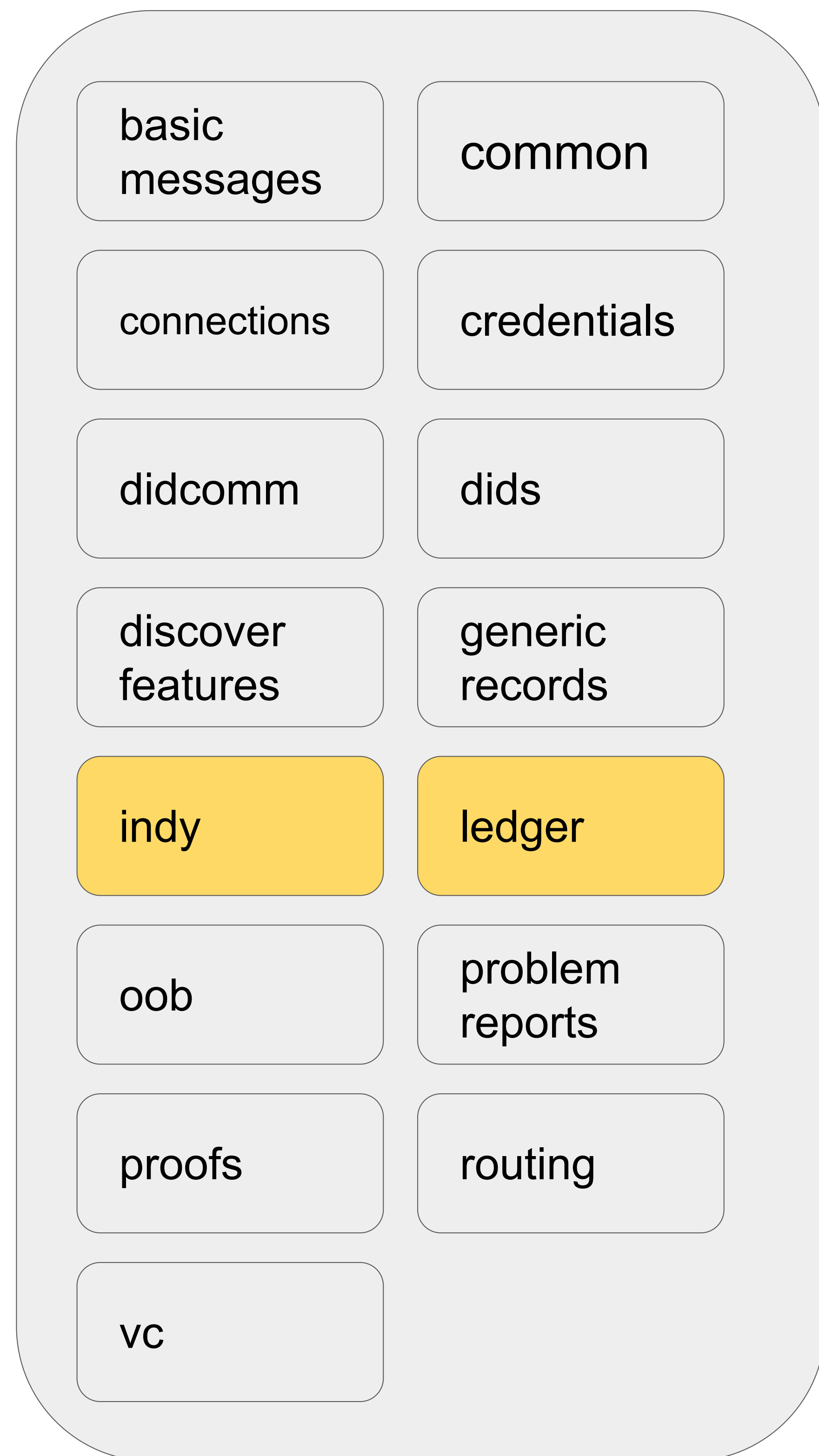


- To solve this problem, three new components were introduced:
  - Aries Askar (storage & cryptography)
  - Indy VDR (ledger communication)
  - AnonCreds (credential format)



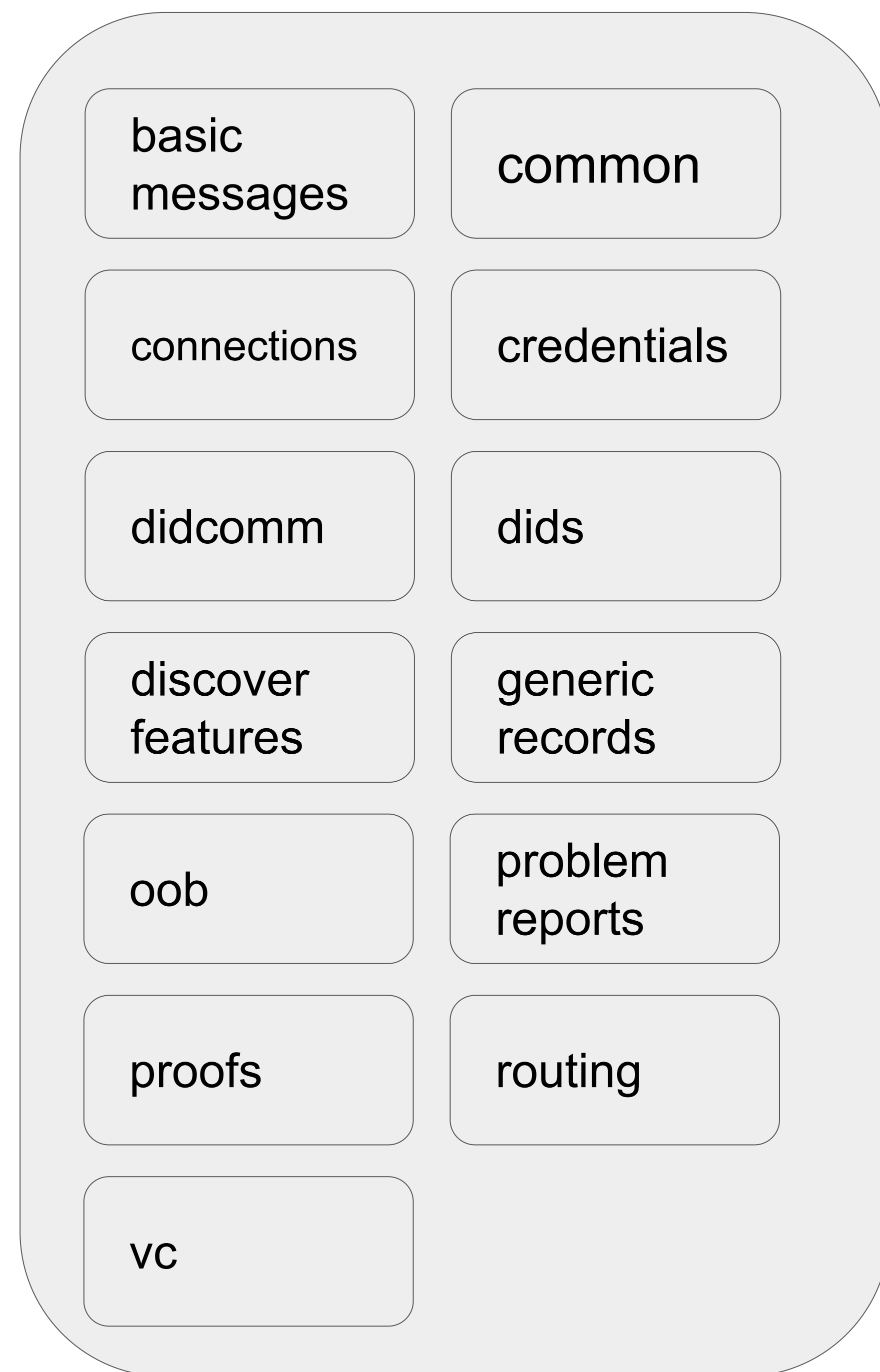
0.3.0

@aries-framework/core



Equivalent setup with Indy SDK

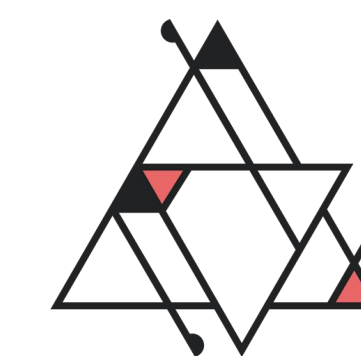
@aries-framework/core



@aries-framework/anoncreds

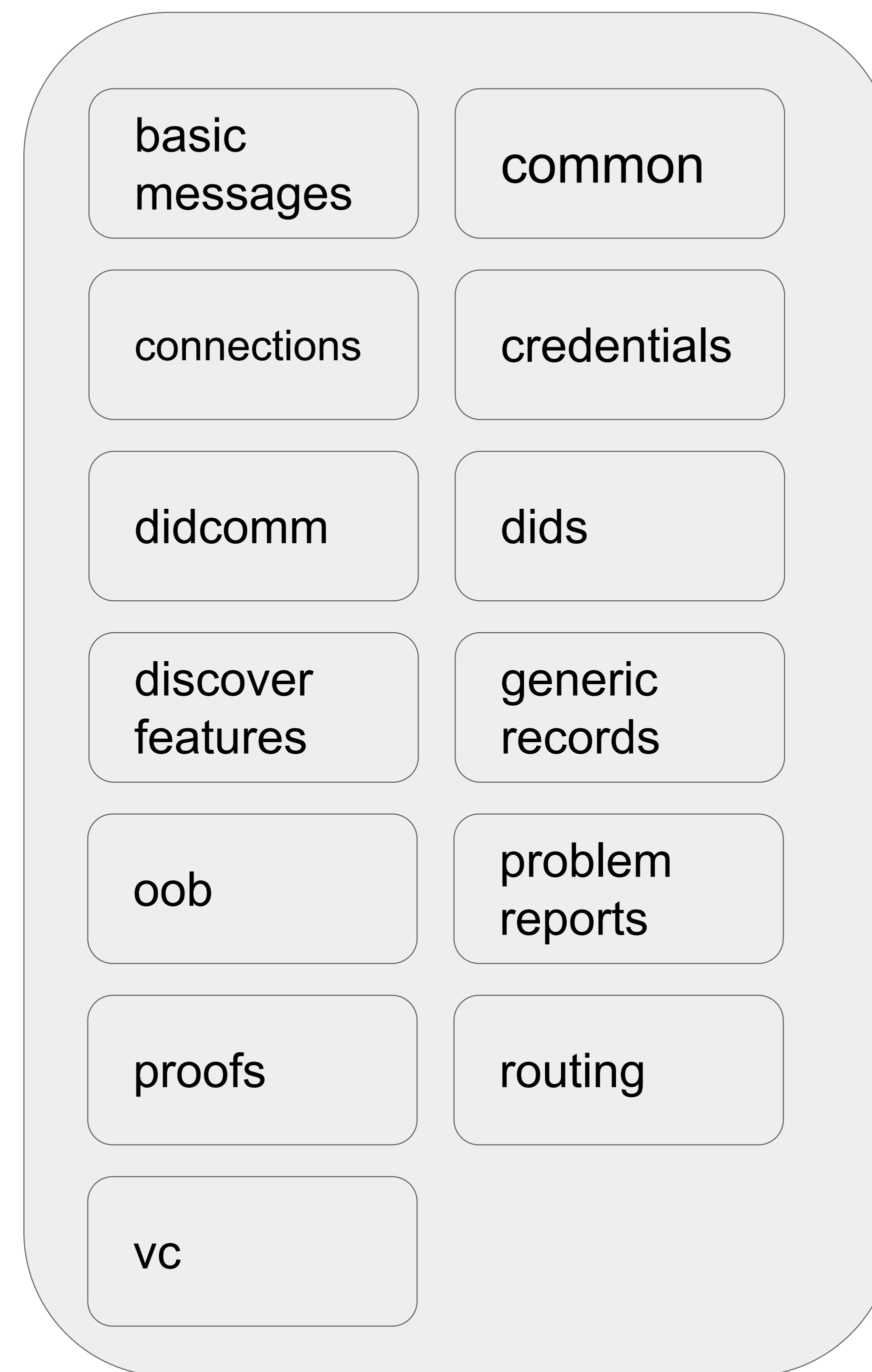
@aries-framework/indy-sdk

Equivalent setup with shared components



ANIM

@aries-framework/core



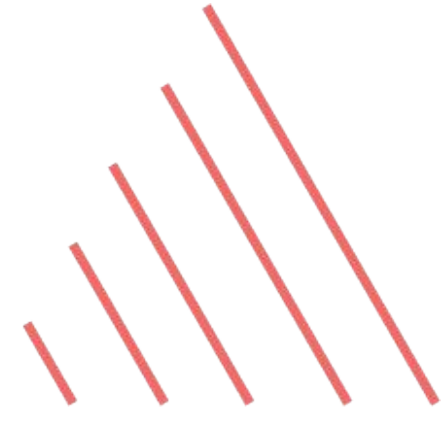
@aries-framework/anoncreds

@aries-framework/anoncreds-rs

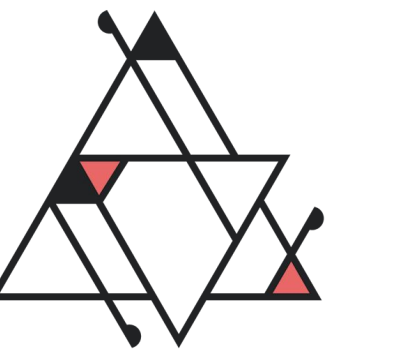
@aries-framework/askar

@aries-framework/indy-vdr





# Pluggable Wallet and Storage



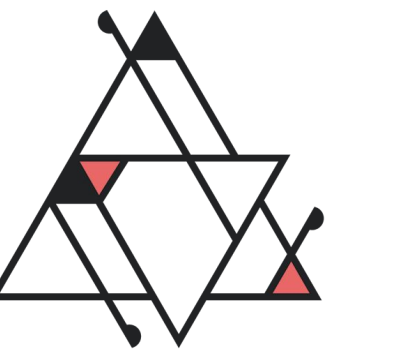
ANIMO



Removal of Indy SDK implied that there is not a default Wallet and Storage Service anymore. This means that, in order to instantiate an Agent, it is needed to provide a module that registers suitable instances and classes for these vital components.

@aries-framework/askar, based on Aries Askar secure storage, provides both of them, although it is possible to implement others, such as an in-memory Wallet for demonstration purposes or a Wallet that can run on a Browser.





ANIMO



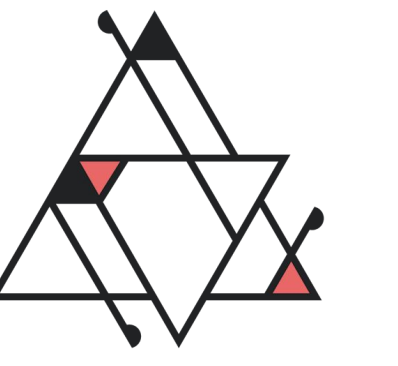
# Improvements in DID management

Previously, Agent instances could define `publicDidSeed` and `publicDid` properties intended to be used for accessing Indy ledgers.

In 0.4.0 we added an import functionality to DID module, allowing to store private data of a DID created elsewhere and use it for any relevant task

```
const privateKey = TypedArrayEncoder.fromString(
  'a-sample-seed-of-32-bytes-in-tot'
)
const did = 'did:key:26MkjEayvPpjVJKFLirX8SomBTPDboHmIXSckUev2M4siQty'
await agent.dids.import({
  did,
  privateKeys: [
    {
      privateKey,
      keyType: KeyType.Ed25519,
    },
  ],
})
```





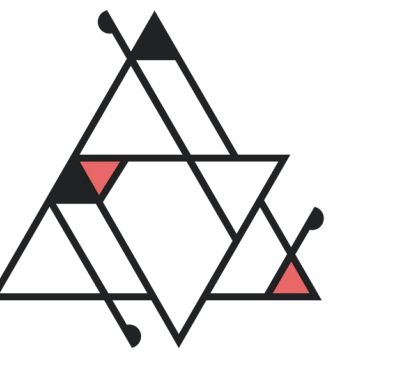
ANIMO



## Connect using Public DIDs

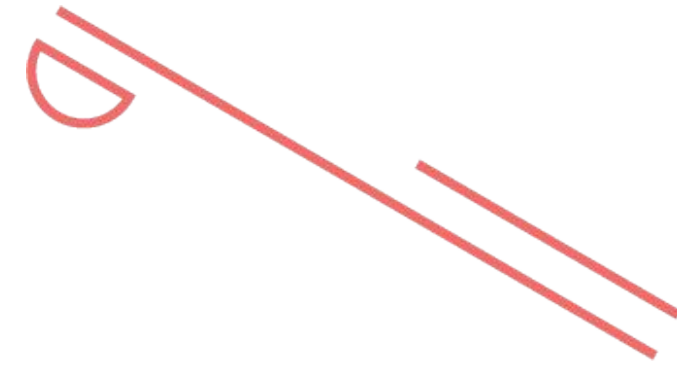
In order to connect to public entities, it is now possible to simply provide a resolvable DID and let AFJ perform the connection mechanism as if it was an 'implicit invitation'.

```
await agent.oob.receiveImplicitInvitation({
  did: 'did:web:faber.com',
  alias: 'Faber public',
  label: 'Alice',
  handshakeProtocols:
    [HandshakeProtocol.DidExchange],
```

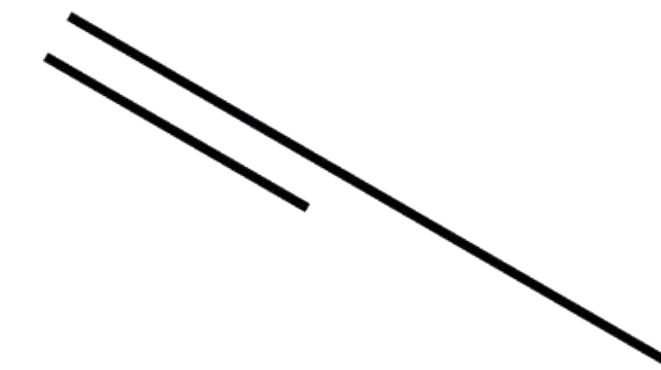


ANIMO

2060



# Beyond Aries

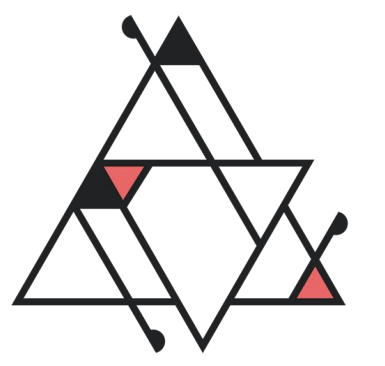
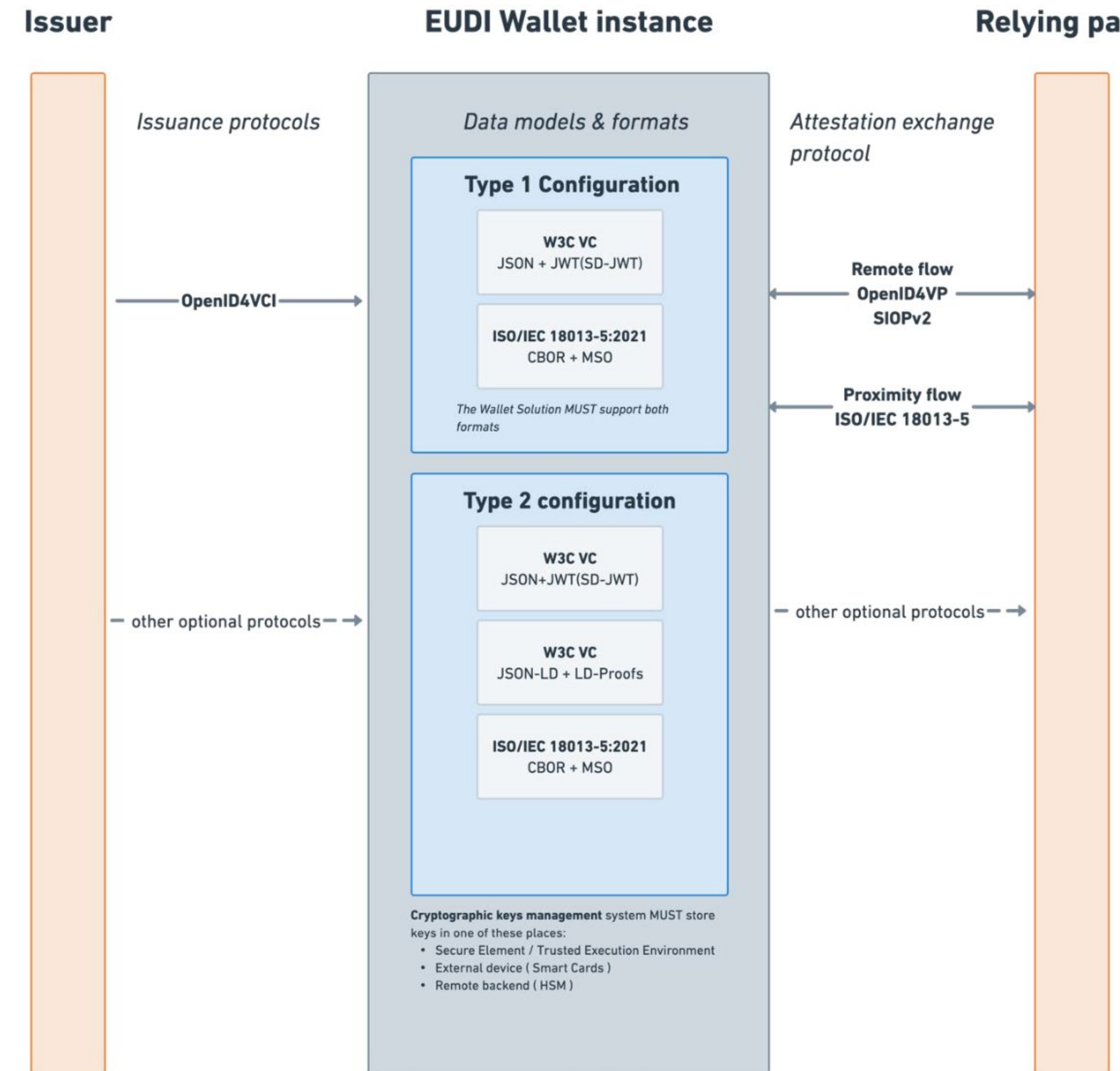




# The space is evolving

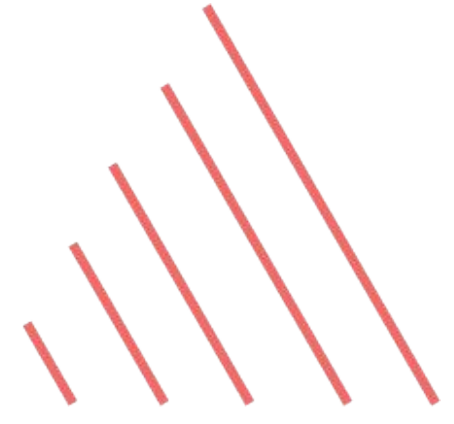
Both the European Commission (EC) and the Department of Homeland Security (DHS) have recently hinted on the standards they will adopt.

# Example (ARF)

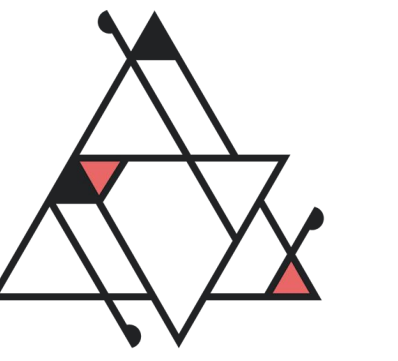


ANIMO





# The value of going beyond Aries

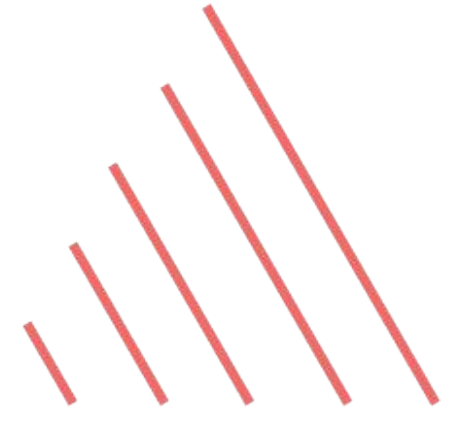


ANIMO

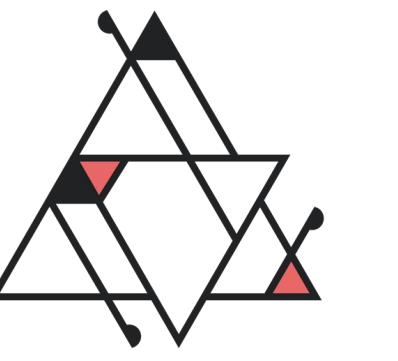


- Enables utilization across a broader range of projects
- Ensures its relevance outside the
- Ensures longevity and relevance in a rapidly evolving landscape
- Could facilitate interoperability between different technologies
  - For example, issue a credential over OpenID4VCI, and verify it over DIDComm





# In other words

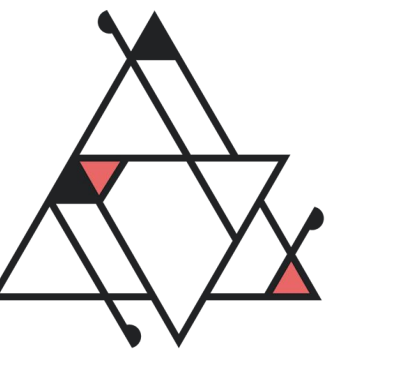


ANIMO



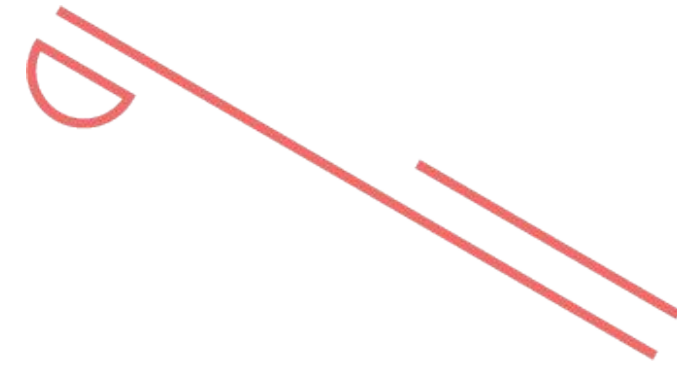
We envision Aries Framework JavaScript as a general, all-purpose toolkit for self-sovereign identity solutions.



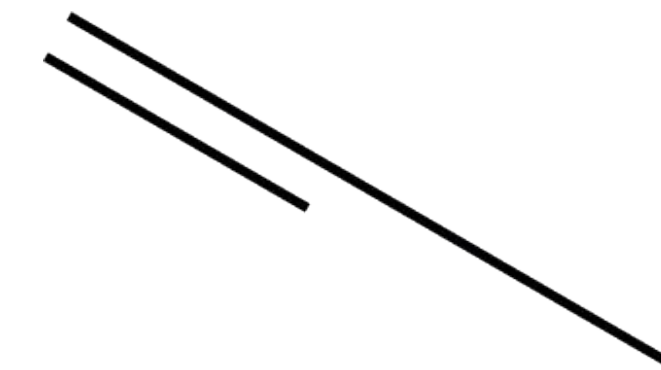


ANIMO

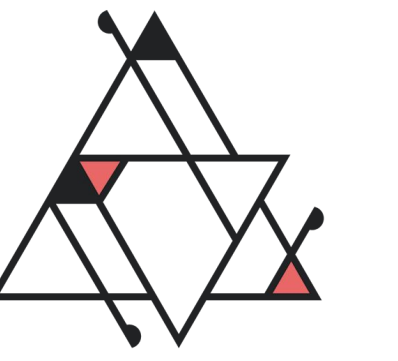
2060



# Ledger-Agnostic AnonCreds







ANIMO



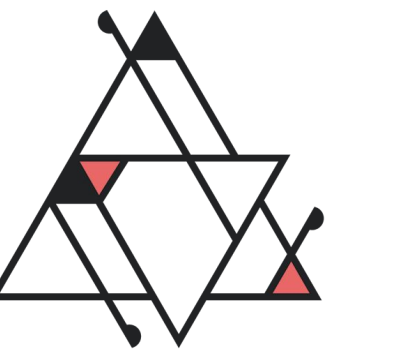
# Ledger-agnostic AnonCreds

Originally, AnonCreds were technically tied to Indy. In 2022, they became a dedicated project at the Hyperledger Foundation.

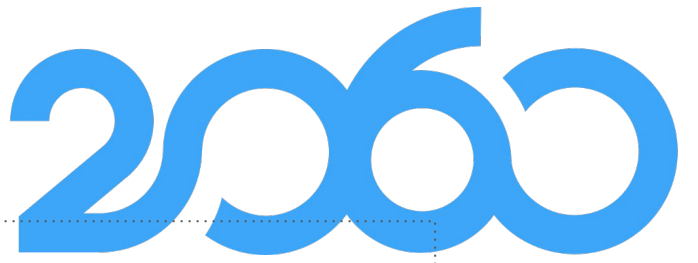
- [AnonCreds Specification v1.0](#) was written
  - Formal definitions of AnonCreds objects and algorithms based on previous implementation from Indy
  - Some adjustments done in order to allow storing objects in other *Verifiable Data Registries* than Indy
- A complete open source implementation in Rust has been created
  - Wrappers available for Python and JavaScript

Aries Framework JavaScript 0.4.0 fully\* supports them, providing backwards compatibility with legacy Indy credentials

\* Issuance of revocable credentials in progress



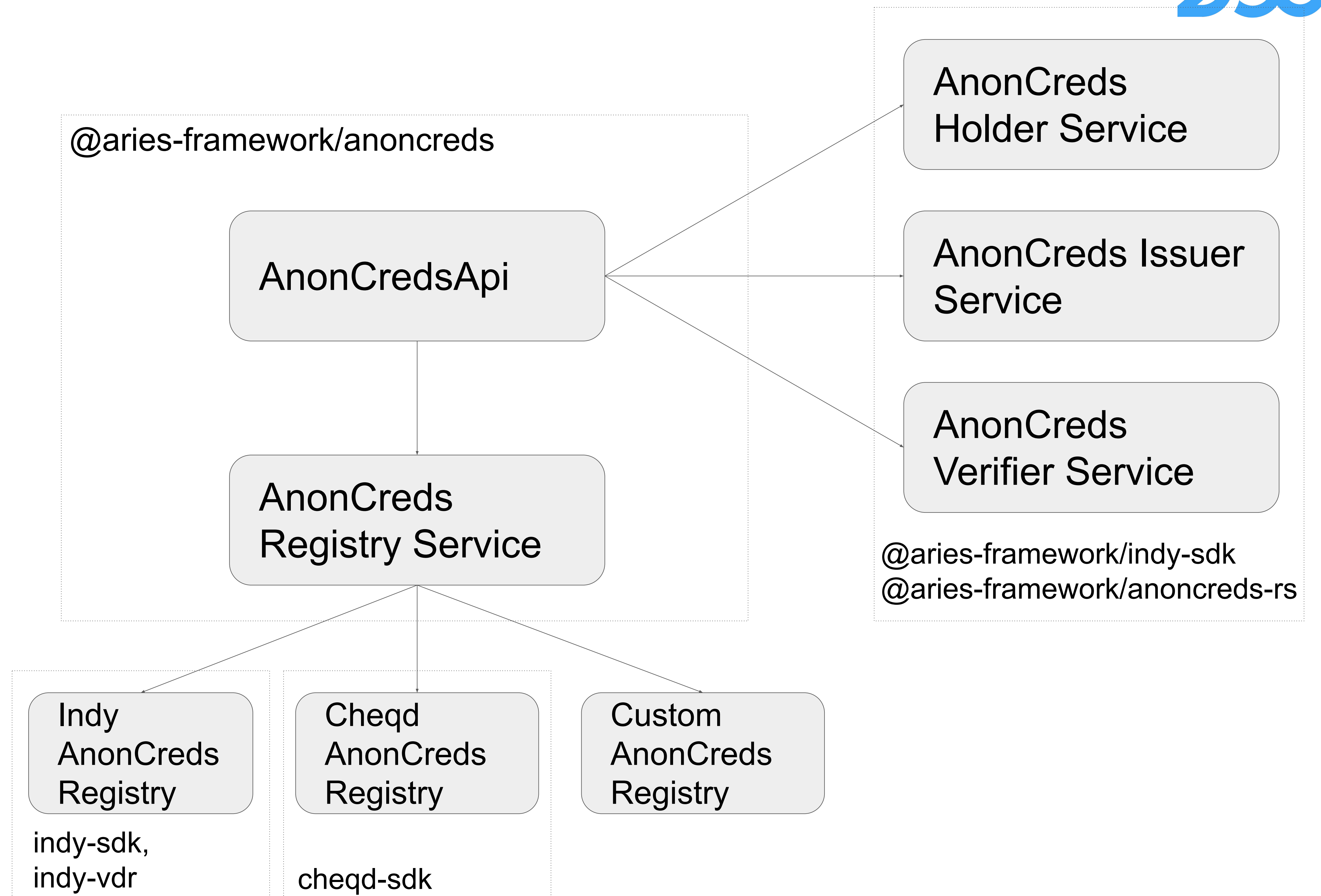
ANIMO

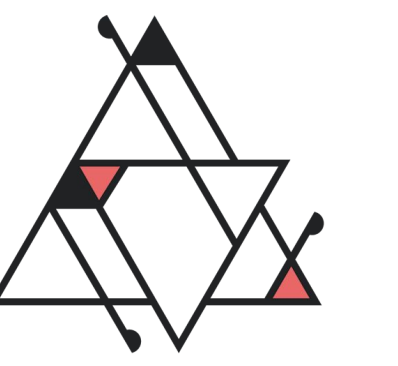


# AnonCreds modules architecture

**anoncreds** module provides the main API to manage VDR objects and interfaces to create actual implementations

- **AnonCreds services** are registered as singletons and provide bindings to AnonCreds libraries such as **indy-sdk** and **anoncreds-rs**
- **AnonCreds Registries** provide means to register and resolve AnonCreds objects. Multiple registries can be supported at the same time





ANIMO



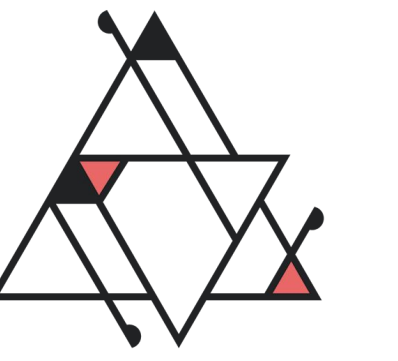
# Supported AnonCreds methods

AFJ main repository contains packages supporting two methods out of the box:

- Indy (<https://hyperledger.github.io/indy-did-method/#other-indy-ledger-object-identifiers>) through **@aries-framework/indy-vdr** and legacy **@aries-framework/indy-sdk**
- Cheqd (<https://docs.cheqd.io/identity/guides/anoncreds>) using **@aries-framework/cheqd-sdk** contributed by cheqd team

Both provide the classes needed to plug into anoncreds module to use their DID resolvers, registrars and AnonCreds registries.





ANIMO



# Adding more AnonCreds methods

Support for a different [AnonCreds method](#) is as simple as:

- Implementing **AnonCredsRegistry** interface with the bindings for accessing the VDR
- Implementing **DidResolver** and **DidRegistrar** interfaces for the underlying DID method (if not already supported by the framework)
- Registering them to **anoncreds** and **dids** modules

Some methods supported by the community:

- [Cardano](#) (by RootsId)
- did:web (by 2060.io): <https://github.com/2060-io/aries-javascript-didweb-anoncreds>





ANIMO

2060



# Demo





ANIMO

2060



```
const agent = new Agent({
  config: { label: "openid4vc-client" },
  modules: { openid4vc: new OpenId4VcClientModule() },
  dependencies: agentDependencies,
})

await agent.initialize()

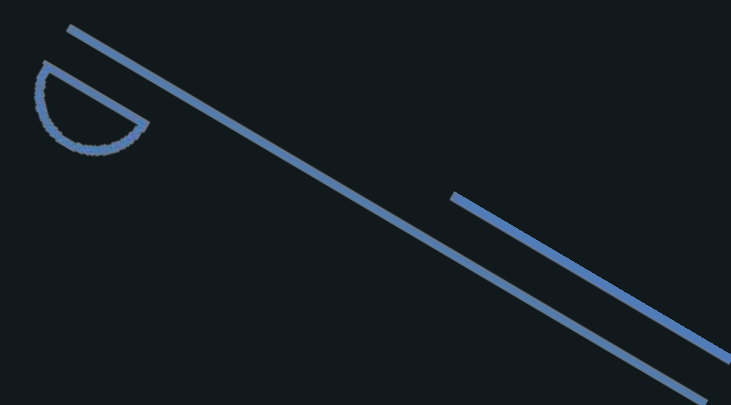
agent.modules.openid4vc.requestCredentialUsingPreAuthorizedCode({
  issuerUri: qrCodeData,
  verifyCredentialStatus: false,
  proofOfPossessionVerificationMethodResolver: async ({
    supportedDidMethods,
    keyType,
  }) => { ... },
})
```





ANIMO

2060



# Some real code





ANIMO

2060

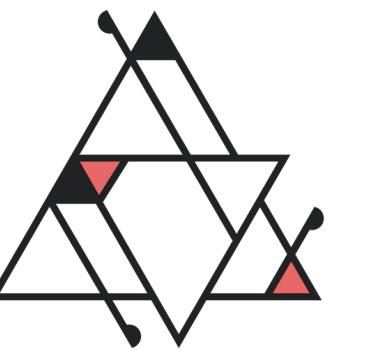


# Documentation





# Versioned documentation



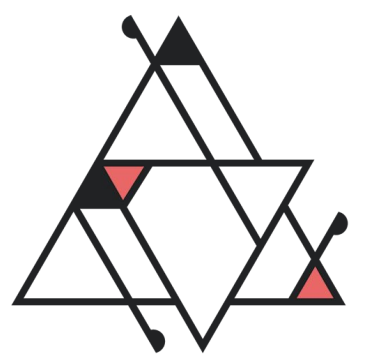
ANIMO



The screenshot shows the Aries JavaScript Docs website. The top navigation bar includes the site logo, "Aries JavaScript Docs", and "Guides". On the right, there is a search bar and a version selector currently set to "v0.4.x". A dropdown menu is open, showing options for "v0.4.x" and "v0.3.x". A red arrow points from the "Intro" page title to the version dropdown. The main content area displays the "Intro" page for version "v0.4.x", with a breadcrumb trail "Intro" and a "Version: v0.4.x" indicator. The page content includes a welcome message, an introduction to the Hyperledger Aries Framework JavaScript, and a "Next Getting started »" button.



# Migration documentation



ANIMO



Aries JavaScript Docs Guides v0.4.x Search...

- Intro
- Getting started >
- Concepts >
- Tutorials >
- Updating** ▾
  - Update Assistant
  - Migrating from an Indy SDK Wallet to Aries Askar
  - Migrating from AFJ 0.1.0 to 0.2.x
  - Migrating from AFJ 0.2.x to 0.3.x
  - Migrating from AFJ 0.3.x to 0.4.x
- The Aries JavaScript Ecosystem
- Extensions >

Updating

Version: v0.4.x

## Updating AFJ

This section will cover everything you need to know about updating Aries Framework JavaScript to a newer version.

- Update Assistant**  
The Update Assistant helps you update the storage objects from...
- Migrating from an Indy SDK Wallet to A...**  
This documentation explains the process of migrating your Indy ...
- Migrating from AFJ 0.1.0 to 0.2.x**  
This document describes everything you need to know for updat...
- Migrating from AFJ 0.2.x to 0.3.x**  
This document describes everything you need to know for updat...
- Migrating from AFJ 0.3.x to 0.4.x**  
This document describes everything you need to know for updat...

### Versioning

Aries Framework JavaScript follows [semantic versioning](#). This means that major version changes (1.0.0) are considered breaking changes. When features are added this is a minor version change (0.1.0). For bug fixes the patch version change is used (0.0.1).

While AFJ is still in pre-1.0.0 version, the version change types are shifted to the right. This means a major version change is now a minor change (0.1.0) and a minor change is now a patch change (0.0.1). This is done to keep the version below 1.0.0, indicating the framework is still in early development and users can expect more breaking changes that when the version has already reached 1.0.0.

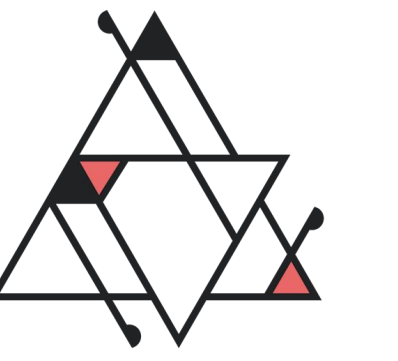
This means if the second number in the version (0.1.0) changes, you need to be careful with updating and always consult this page

Versioning

- Types of breaking changes
  - Breaking Code Changes
  - Breaking Storage Changes
- Migration Guides



# Protocol specific documentation



ANIMO



Aries JavaScript Docs Guides v0.4.x Search...

- Intro
- Getting started
  - Prerequisites
  - Agent Setup
    - Aries Askar
    - Indy SDK
    - AnonCreds RS
    - Indy VDR
    - cheqd
  - Concepts
    - Agents
    - DIDs and DIDComm
    - Platform and Environment
  - Tutorials
    - Agent Config
    - Create a connection
    - Cheqd Did Module
    - Registering a schema and credential definition on an AnonCreds Registry
    - Issue a credential**
    - Mediation
    - Using PostgreSQL with the Indy SDK
  - Updating
  - The Aries JavaScript Ecosystem
  - Extensions

```
8 console.log( `Credential for credential id ${payload.credentialRecord.id} is accepted ` )
9 // For demo purposes we exit the program here.
10 process.exit(0)
11 }
12 }
```

### 4. Issuing a credential

Now that everything is setup on both sides, the *Issuer* can now offer a credential to the *Holder*.

In this example we do not instantiate a connection and assume that there is one. Please refer to this guide [Create a connection](#) to get a connection and connectionId.

Indy AnonCreds

```
ISSUER
1 const anonCredsCredentialExchangeRecord = issuer.credentials.offerCredential({
2   protocolVersion: 'v2',
3   connectionId: '<connection id>',
4   credentialFormats: {
5     anoncreds: {
6       credentialDefinitionId: '<credential definition id>',
7       attributes: [
8         { name: 'name', value: 'Jane Doe' },
9         { name: 'age', value: '23' },
10      ],
11     },
12   },
13 })
```

### Useful resources

- AnonCreds
- Issue Credential V1 Protocol
- Issue Credential V2 Protocol

1. Setting up the agents  
3. Listening for incoming credentials  
4. Issuing a credential  
Useful resources



ANIMO

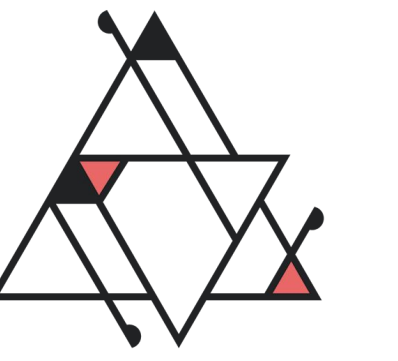
2060



# What's Next?







ANIMO

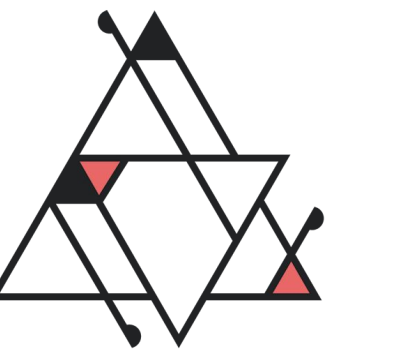


# Server-side AFJ

Although Aries Framework JavaScript is mostly known in mobile environments, thanks to its bindings to Node JS it is absolutely possible to use it in both ends to create Aries agents.

Among the usually server-side features fully supported by AFJ we can mention:

- **Mediation:** agents can act as DIDComm V1 mediators out of the box, using an in-memory message queue that can be replaced by more sophisticated mechanisms, add push notifications, etc.
- **Credential issuance:** besides acting as holder or verifier, AFJ supports issuing credentials in various formats



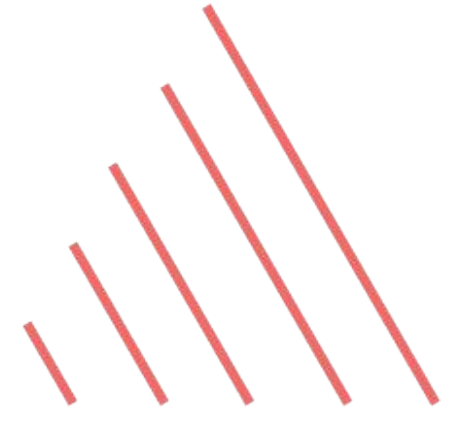
ANIMO



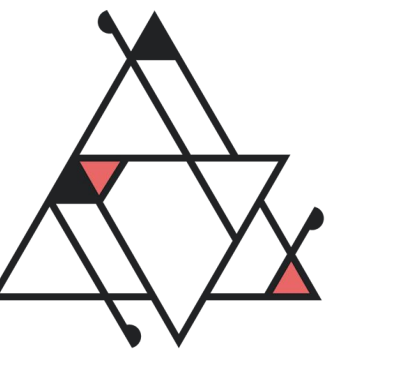
# Why server-side AFJ?

There are various advantages of using AFJ in the server:

- No need of mastering two different programming languages for client and server
- Write extensions only once:
  - AnonCreds registries
  - DID resolvers and registrars
  - Custom DIDComm protocols
- Storage backend flexibility
- Ready to use [REST interface](#)



# Web Support

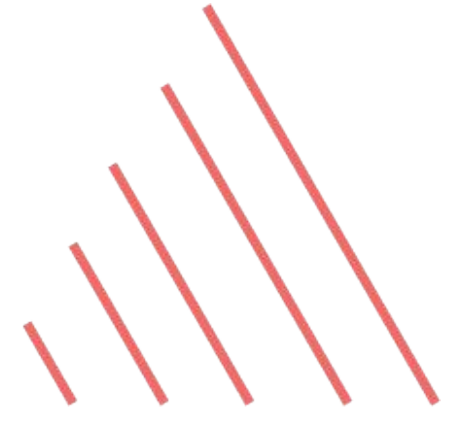


ANIMO

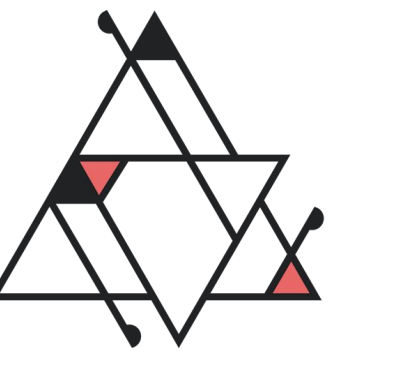


- Demo environments
- WASM (WASI) storage
- DWN storage





# Architecture Reference Framework Support



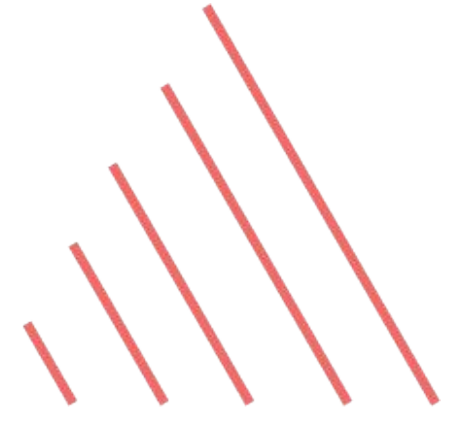
ANIMO



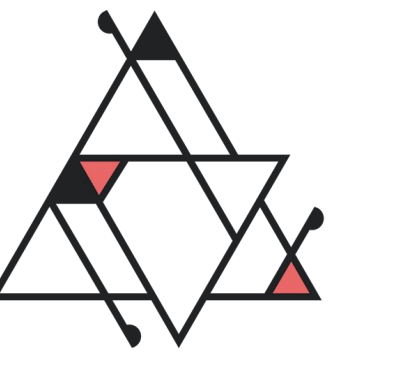
- SD-JWT
- ISO/IEC 18013-5:2021
- More OID4VC support
- Secure enclave support







# Easier to set up



ANIMO



- Wallet rework
- Smaller core
- Pre configured agents





ANIMO

2060



Q/A

