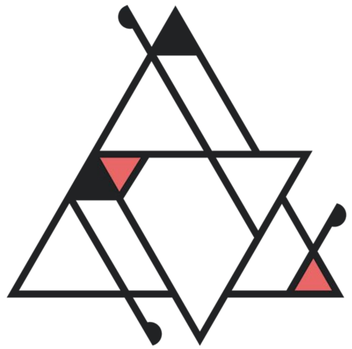
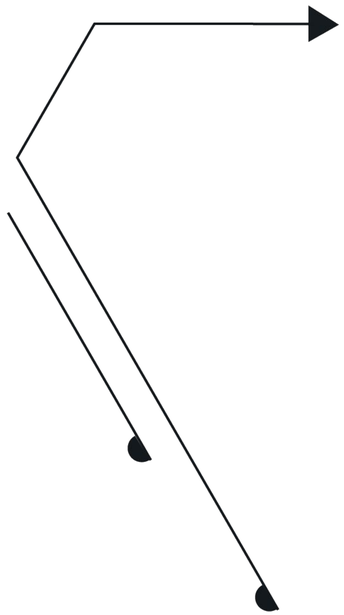
Hyperledger Aries Framework JavaScript
0.4.0



ANIMO

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

Index



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

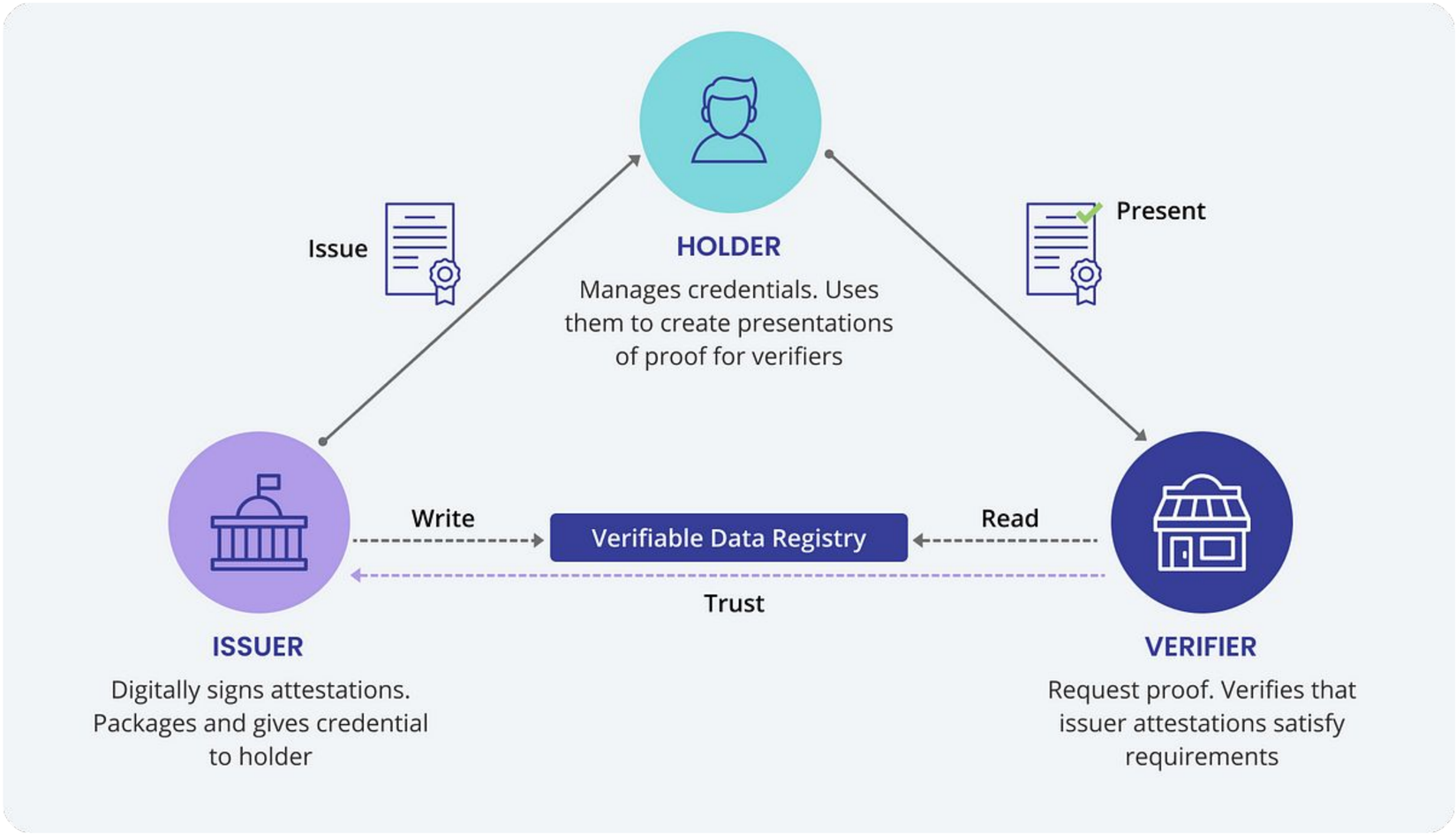
Introduction

Hyperledger Aries Framework JavaScript

- 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



Release Overview

ANIMO

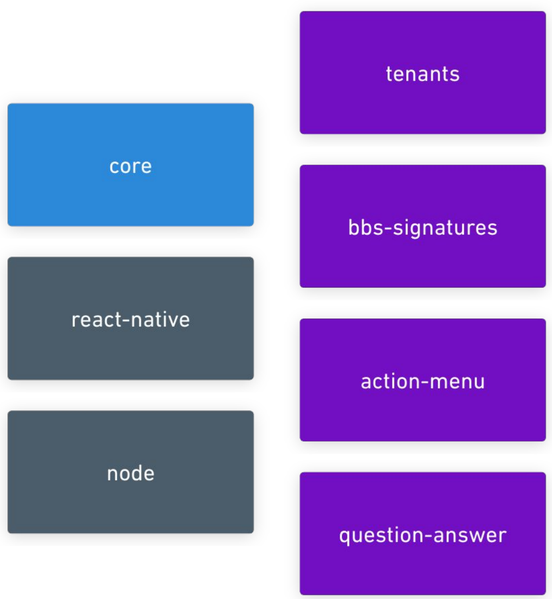
2060

Modularization

ANIMO

2060

0.3.0



0.4.0


core
react-native
node
tenants
bbs-signatures
action-menu
question-answer
anoncreds
anoncreds-rs
askar
cheqd
indy-vdr
openid4vc-client
indy-sdk-to-askar-migration
Present in 0.3.0
Moved out of core
New functionality
Shared components

Indy SDK

- 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

- 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

basic messages
common
connections
credentials
didcomm
dids
discover features
generic records
indy
ledger
oob
problem reports
proofs
routing
vc

Equivalent setup with Indy SDK

@aries-framework/core

basic messages
common
connections
credentials
didcomm
dids
discover features
generic records
oob
problem reports
proofs
routing
vc

@aries-framework/anoncreds

@aries-framework/indy-sdk

Equivalent setup with shared components

@aries-framework/core

basic messages
common
connections
credentials
didcomm
dids
discover features
generic records
oob
problem reports
proofs
routing
vc

ANIMO

@aries-framework/anoncreds

@aries-framework/anoncreds-rs

@aries-framework/askar

@aries-framework/indy-vdr

Pluggable Wallet and Storage

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.

Improvements in DIDs 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 DIDs 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,
      keyTpe: KeyType.Ed25519,
    },
  ],
})
```

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],
})
```

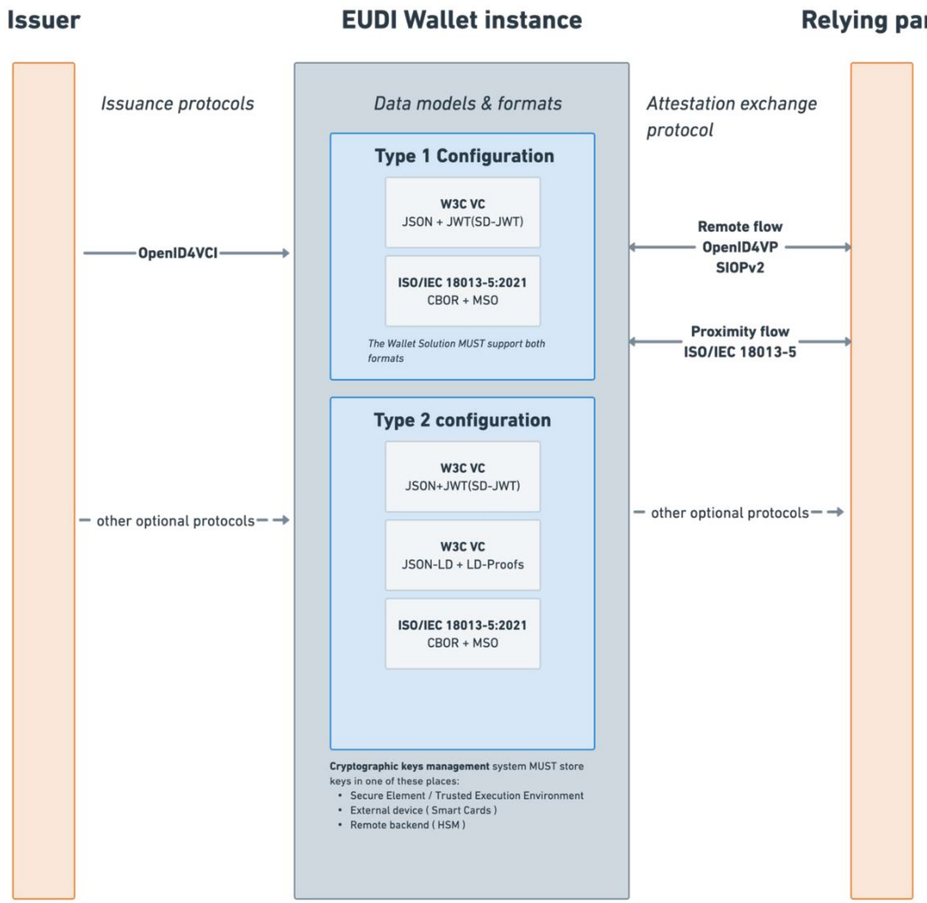
ANIMO 2060

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)

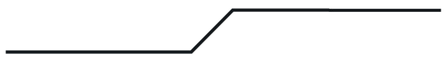


The value of going beyond Aries

- 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

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



Ledger-Agnostic AnonCreds

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

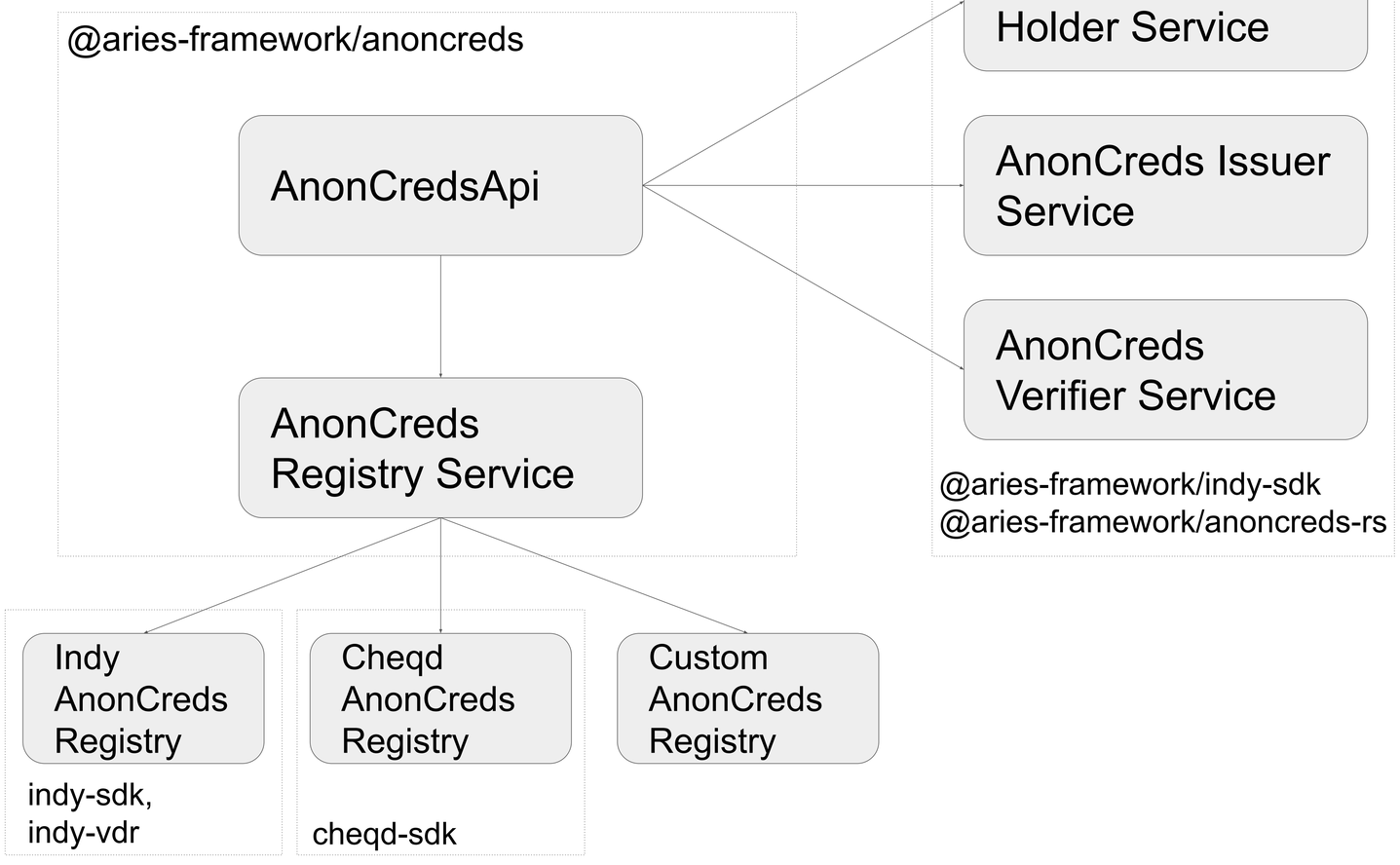
Aries Framework JavaScript 0.4.0 fully* supports them, providing backwards compatibility with legacy Indy credentials

* Issuance of revocable credentials in progress

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



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.

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

Demo

```ts
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,
  }) => { ... },
})
```

Some real code

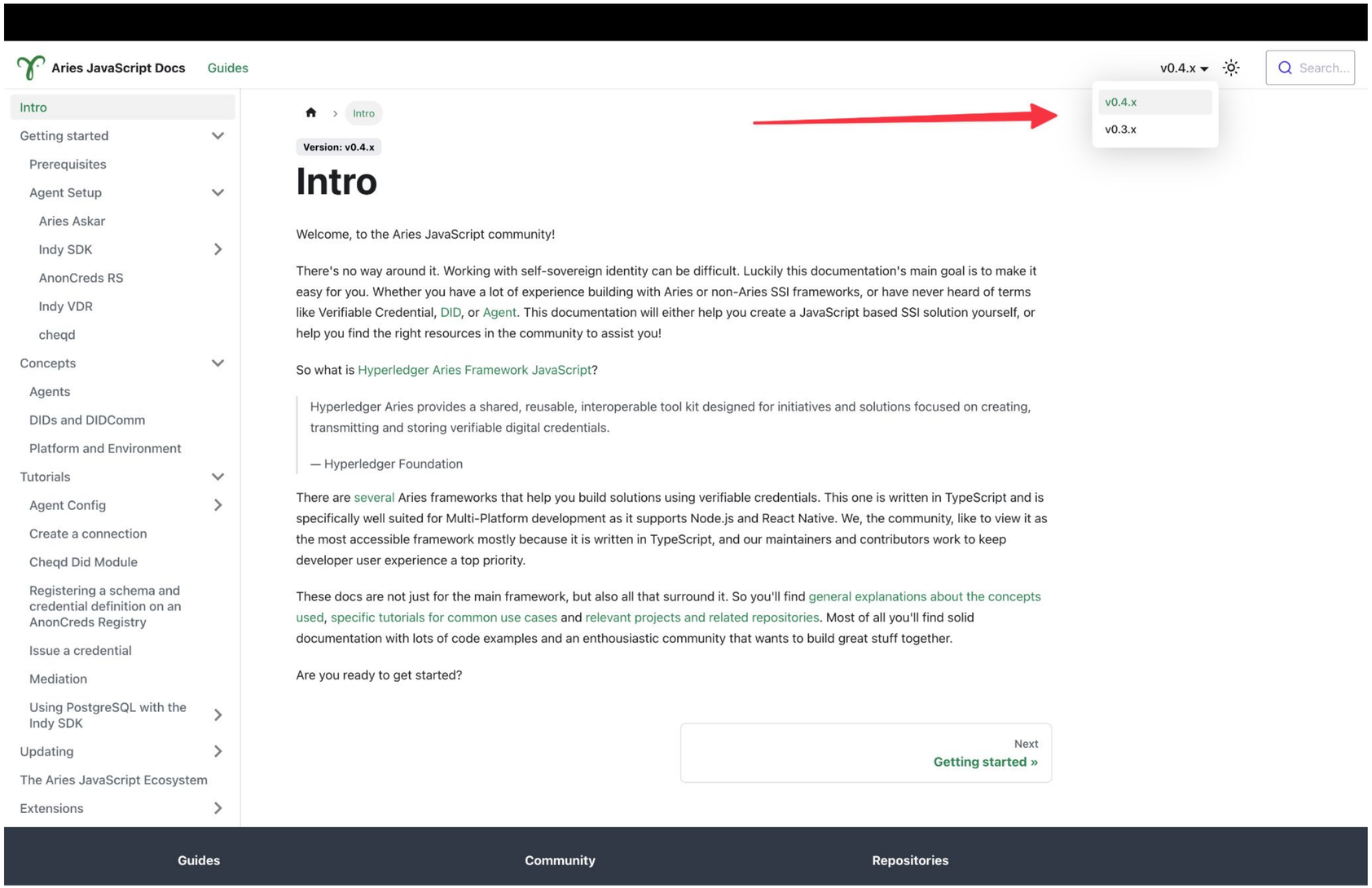
https://github.com/animo/hyperledger-afj-040-release-workshop

ANIMO

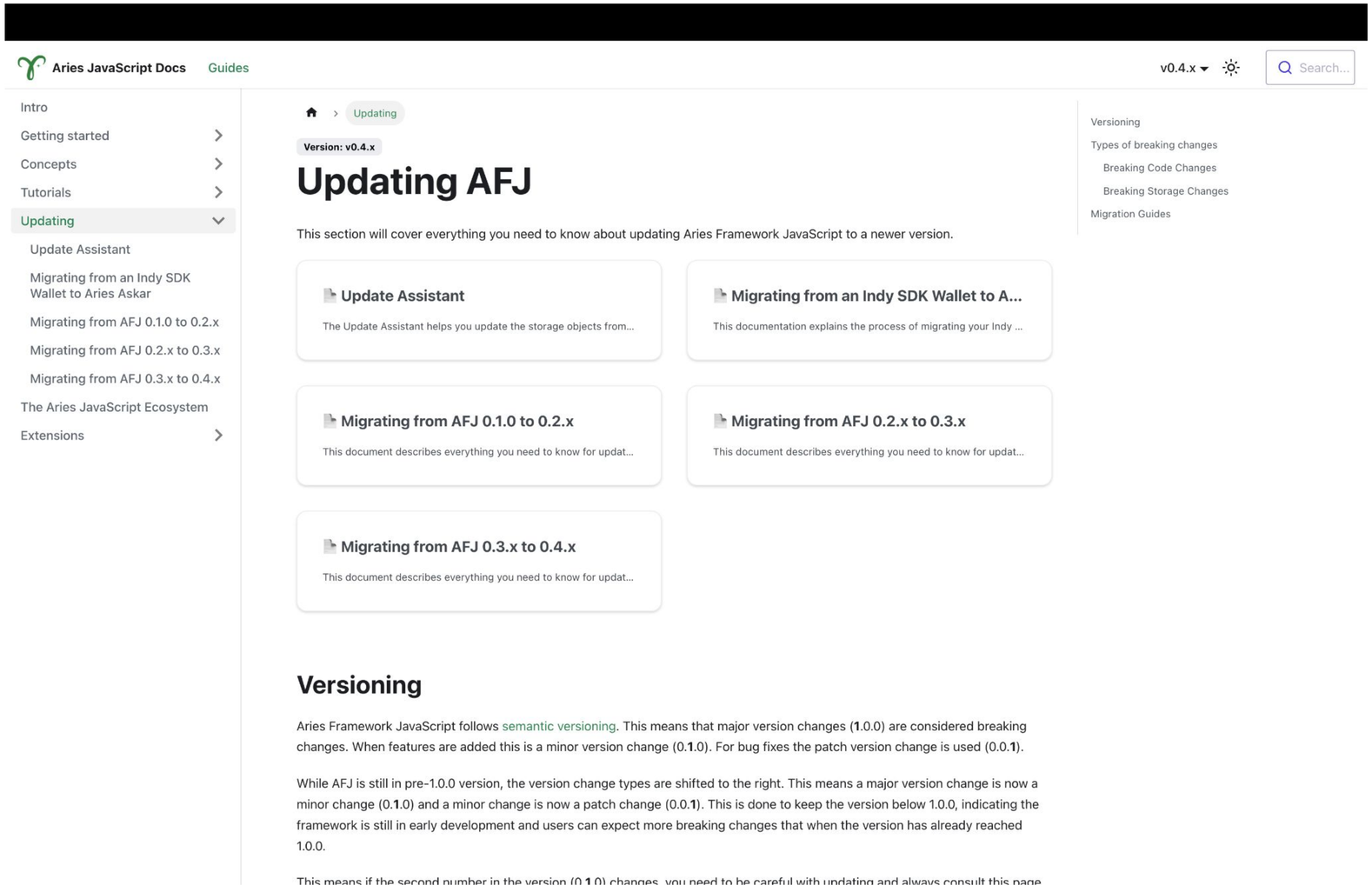
2060

Documentation

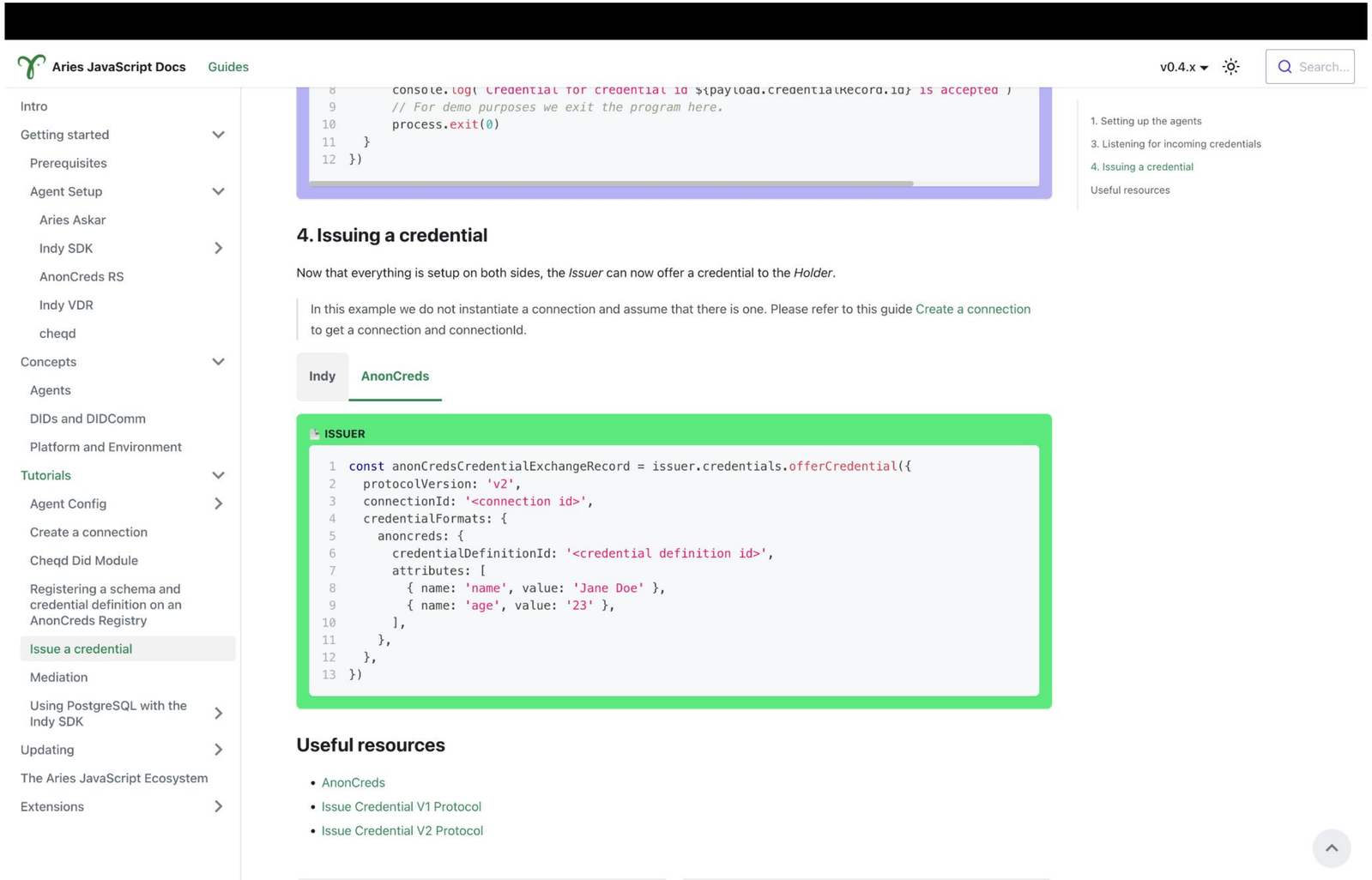
Versioned documentation



Migration documentation



Protocol specific documentation



What’s Next?

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

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

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

Architecture Reference Framework Support

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

Easier to set up

- Wallet rework
- Smaller core
- Pre configured agents

ANIMO

2020

Q/A