# Open Source SSI and Inclusive Safety Communities

1

## Topics

- Aries Framework JavaScript
- Inclusive Safety Communities

## Aries Framework JavaScript

Aries Framework JavaScript is a framework for building SSI Agents and DIDComm services that aims to be compliant and interoperable with the standards defined in the Aries RFCs.

https://github.com/hyperledger/aries-framework-javascript

## Cross Platform

- Server (Node.JS) -https://github.com/animo/aries-agent-demo-nodejs
- Desktop (Electron) - https://github.com/blu3beri/aries-agent-electron
- Mobile (React Native) - https://github.com/hyperledger/aries-mobile-agent-react-native

## Popular Language

- JavaScript is the most popular language according to the StackOverflow Developer Survey 2020
- TypeScript on top of JavaScript takes away a lot of the JavaScript headaches

## Ease of Use

```
npm install @aries-framework/core @aries-framework/react-native
```

```
// agent.ts
import { HttpOutboundTransporter, Agent } from '@aries-framework/core'
import { agentDependencies } from '@aries-framework/react-native'
import { BCGOV_TEST_GENESIS } from './utils'

const agentConfig = {
  label: 'Aries Framework JavaScript',
  walletConfig: { id: 'AriesFrameworkJavaScript' },
  walletCredentials: { key: 'AriesFrameworkJavaScript' },
  genesisTransactions: BCGOV_TEST_GENESIS,
  autoAcceptConnections: true,
  mediatorConnectionsInvite: 'https://mediator.animo.id?c_i=eyXXXX'
}

const agent = new Agent(config, agentDependencies)
agent.setOutboundTransporter(new HttpOutboundTransporter())
await agent.initialize()

// Scanned using QR scanner
const url = 'https://agent.com?c_i=eyXXX'

const { id } = await agent.connection.receiveInvitationFromUrl(url)
const connection = await agent.connections.returnWhenIsConnected(id)
```

```
import React, { useEffect, useLayoutEffect } from 'react'
import { View, Text, StyleSheet } from 'react-native'
import { CredentialsSelectors, CredentialsThunks, initializeStore } from '@aries-framework/redux-store'

const { useAppDispatch } = initializeStore(agent)

export const Credentials = () => {
  const dispatch = useAppDispatch()
  const credentials = useAgentSelector(CredentialsSelectors.credentialRecordsSelector)

  useEffect(() => {
    dispatch(CredentialsThunks.getAllCredentials())
  }, [])

  return (
    <View style={StyleSheet.absoluteFill}>
      {credentials.map((c) => (
        <View key={c.id}>
          <Text>{c.id}</Text>
        </View>
      ))}
    </View>
  )
}
```

7

## Some projects

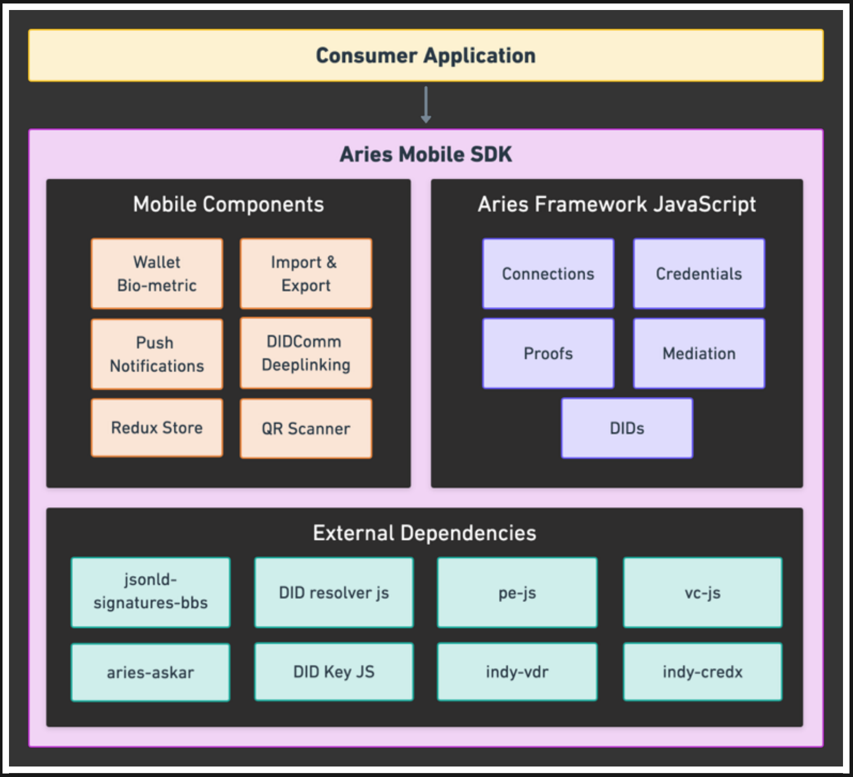
- Cardea - Site
- UNICC - Article
- Aries Mobile Agent React Native - Repo
- Several Student Projects (HU)
- Inclusive Safety Communities

## Current Status

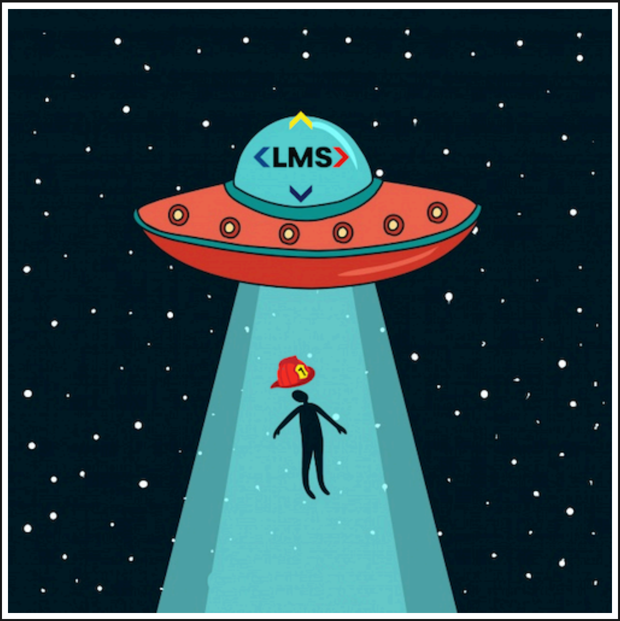
- ✅ React Native, Node.JS
- ✅ Connections, Issue Credential, Present Proof
- ✅ Basic Messages
- ✅ Indy Credentials (with did:sov support)
- ✅ HTTP, WebSocket Transport
- ✅ Mediator Coordination Protocol
- ✅ Dependency Injection
- 🚧 Breaking up into multiple packages
- 🚧 Smart auto-acceptance of credentials and proofs
- 🚧 Connection-less Issuance and Verification
- 🚧 Revocation of Indy Credentials

9

## Road Ahead


Consumer Application
Aries Mobile SDK
Mobile Components
Wallet Bio-metric
Import & Export
Push Notifications
DIDComm Deeplinking
Redux Store
QR Scanner
Aries Framework JavaScript
Connections
Credentials
Proofs
Mediation
DIDs
External Dependencies
jsonld-signatures-bbs
DID resolver js
pe-js
vc-js
aries-askar
DID Key JS
indy-vdr
indy-credx

## Inclusive Safety Communities



Animo is working with the LMS (a national collaboration of emergency dispatch offices) to facilitate citizen volunteering in emergency situations

- Odyssey Hackathon
- Challenge set out by the Dutch Police and the Dutch Public Safety Alliance

https://odyssey.org/past-editions/odyssey-2020/public-safety-and-security/inclusive-safety-communities

## The Problem

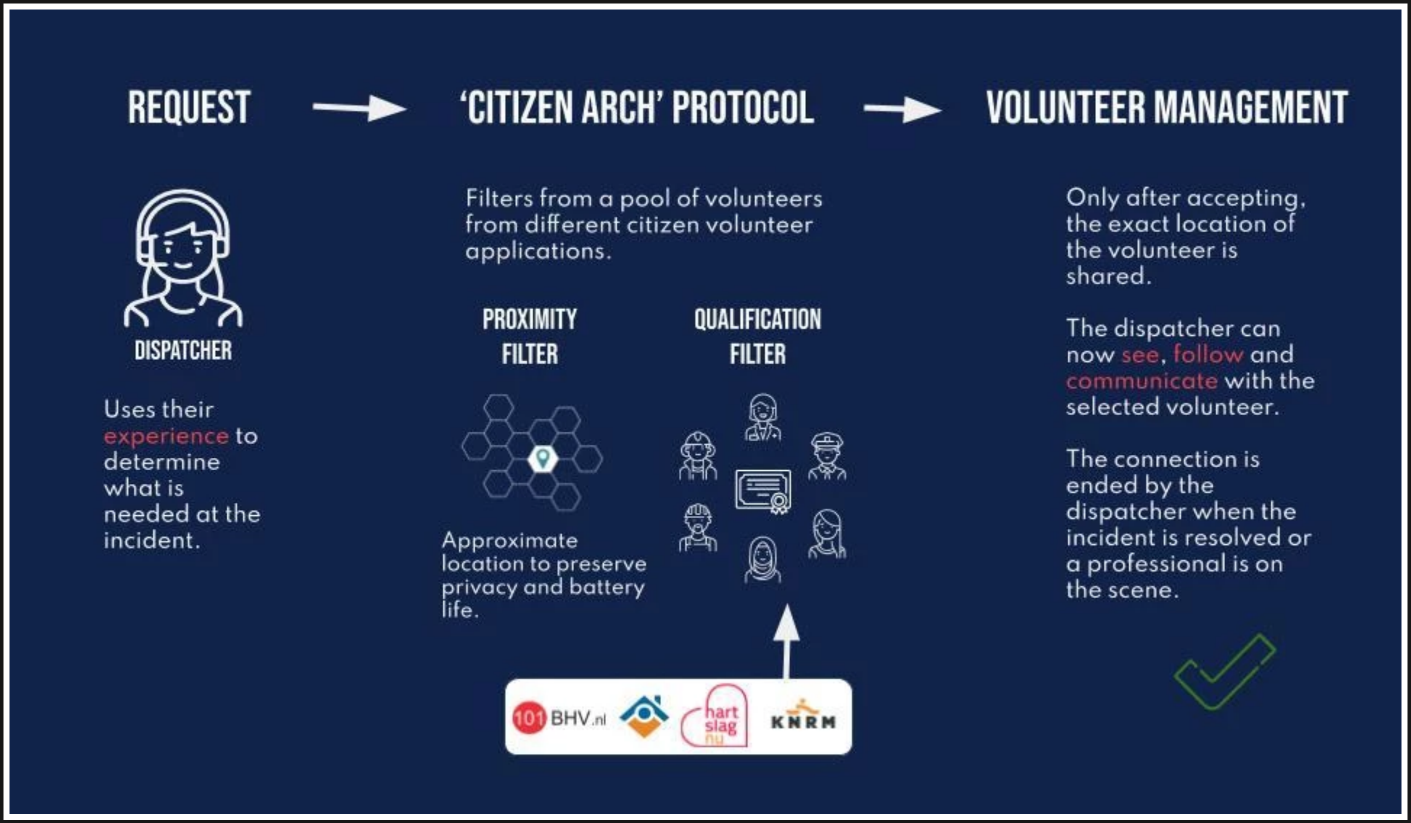
> *How can the 112-dispatch (911) center find and connect to qualified citizen volunteers, who are able to assist or even solve emergency situations while the professionals are on their way.*

- It can take up to 8 minutes for the fire department to arrive (In the netherlands, a small densely populated country)
- Smoke inhalation will kill in 3 min

## The Solution

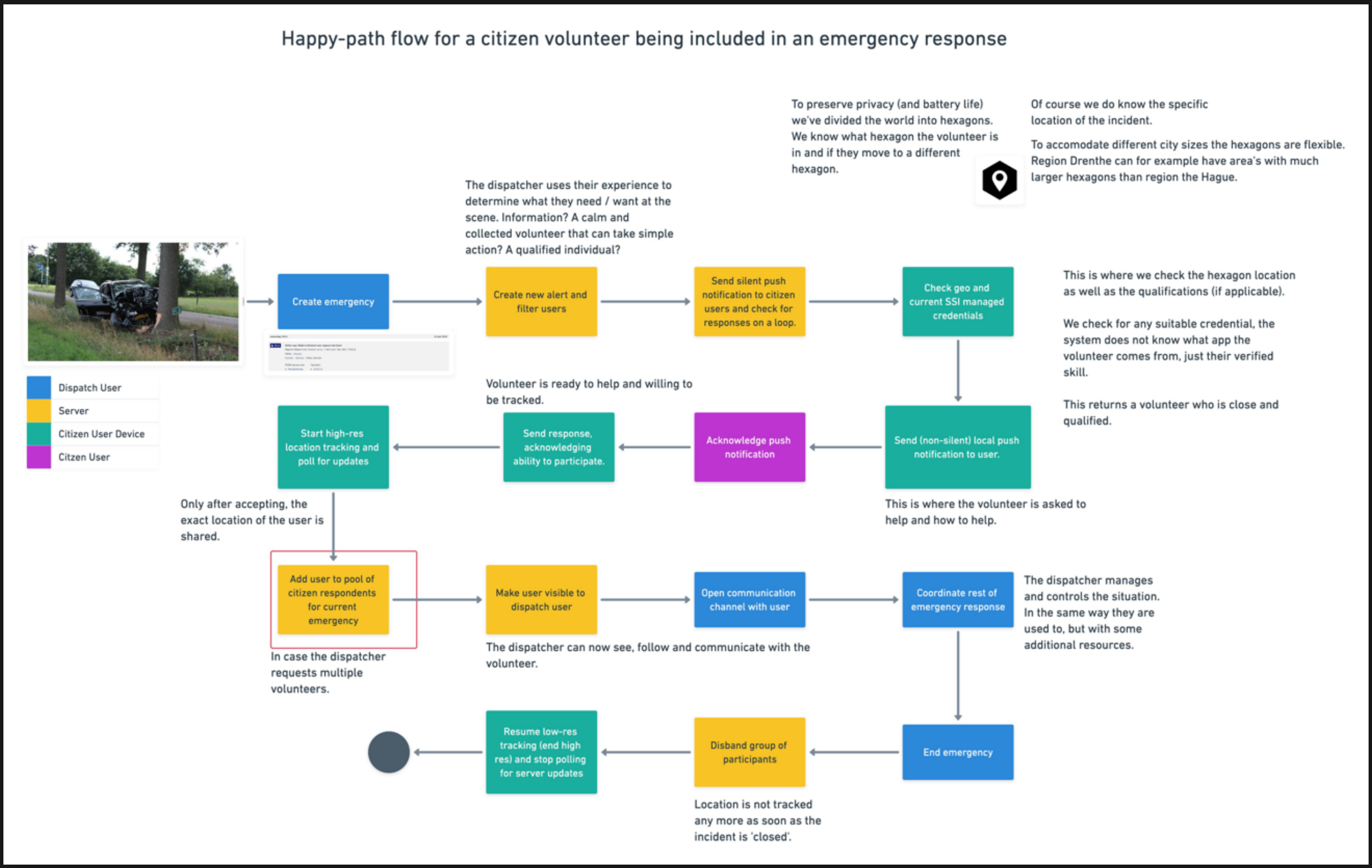
> *We connect the dispatch to a pool of volunteers from several citizen volunteer applications. This allows dispatchers to manage qualified volunteers in much the same way they currently manage emergency services using systems and communities already in place. Getting the right volunteer help to the right location quickly and safely.*

## High Level Overview


REQUEST
'CITIZEN ARCH' PROTOCOL
VOLUNTEER MANAGEMENT
DISPATCHER
Uses their experience to determine what is needed at the incident.
Filters from a pool of volunteers from different citizen volunteer applications.
PROXIMITY FILTER
QUALIFICATION FILTER
Approximate location to preserve privacy and battery life.
101 BHV.nl
hart slag nu
KNRM
Only after accepting, the exact location of the volunteer is shared.
The dispatcher can now see, follow and communicate with the selected volunteer.
The connection is ended by the dispatcher when the incident is resolved or a professional is on the scene.

15

## In Depth Flow


Happy-path flow for a citizen volunteer being included in an emergency response
Dispatch User
Server
Citizen User Device
Citzen User
Create emergency
Create new alert and filter users
Send silent push notification to citizen users and check for responses on a loop.
Check geo and current SSI managed credentials
Send (non-silent) local push notification to user.
Acknowledge push notification
Send response, acknowledging ability to participate.
Start high-res location tracking and poll for updates
Add user to pool of citizen respondents for current emergency
Make user visible to dispatch user
Open communication channel with user
Coordinate rest of emergency response
End emergency
Disband group of participants
Resume low-res tracking (end high res) and stop polling for server updates

To preserve privacy (and battery life) we've divided the world into hexagons. We know what hexagon the volunteer is in and if they move to a different hexagon.

Of course we do know the specific location of the incident.

To accomodate different city sizes the hexagons are flexible. Region Drenthe can for example have area's with much larger hexagons than region the Hague.

The dispatcher uses their experience to determine what they need / want at the scene. Information? A calm and collected volunteer that can take simple action? A qualified individual?

This is where we check the hexagon location as well as the qualifications (if applicable).

We check for any suitable credential, the system does not know what app the volunteer comes from, just their verified skill.

This returns a volunteer who is close and qualified.

Volunteer is ready to help and willing to be tracked.

This is where the volunteer is asked to help and how to help.

Only after accepting, the exact location of the user is shared.

In case the dispatcher requests multiple volunteers.

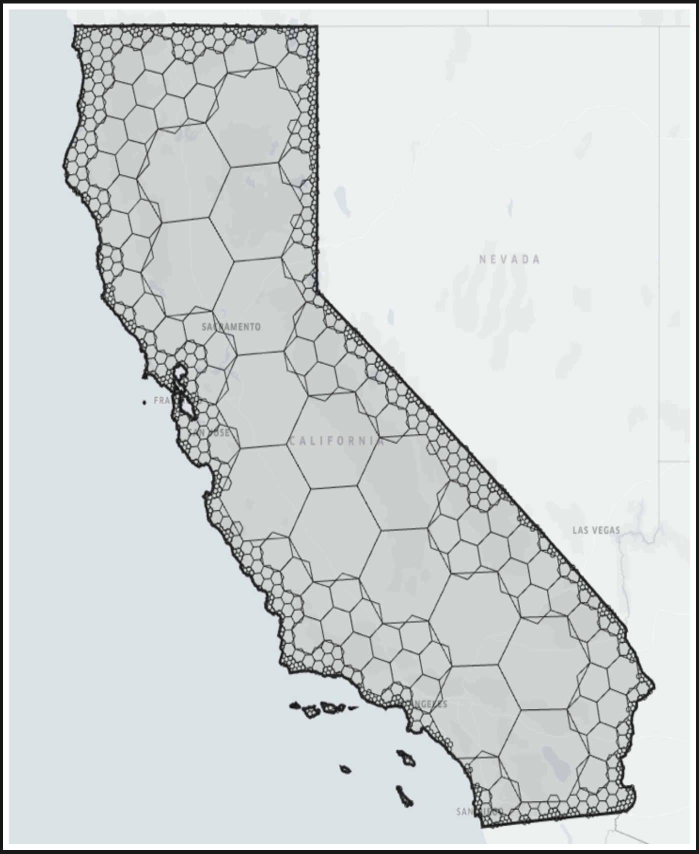
The dispatcher can now see, follow and communicate with the volunteer.

The dispatcher manages and controls the situation. In the same way they are used to, but with some additional resources.

Location is not tracked any more as soon as the incident is 'closed'.

## Approximate Location Filtering

https://h3geo.org/

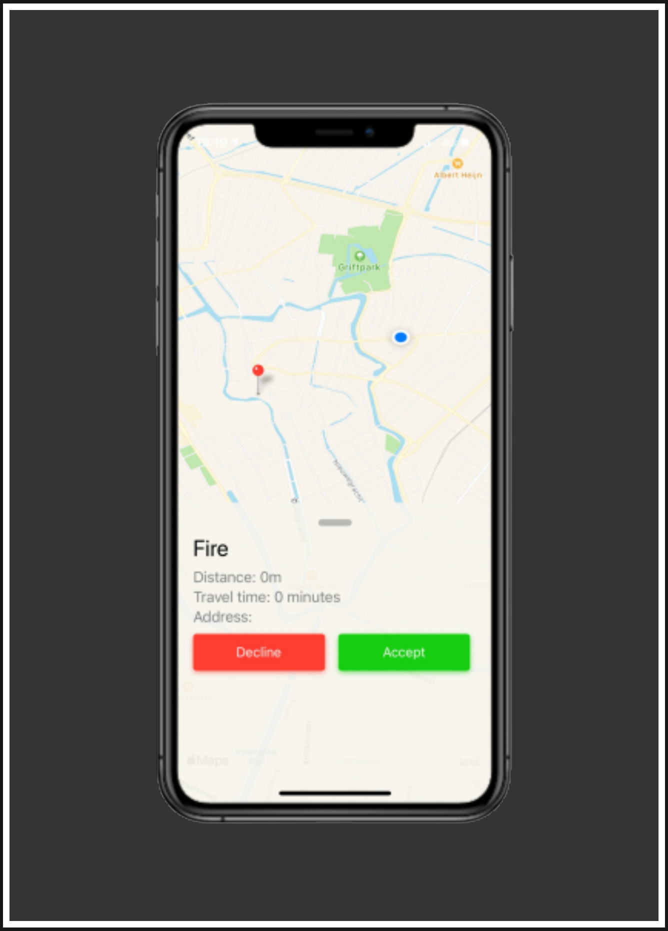


## Pilot

- Regional fire department and regional dispatch office
- Low populated area in the Netherlands
- Suffer the most from long traveling time to emergency
- Start with off duty fire fighters

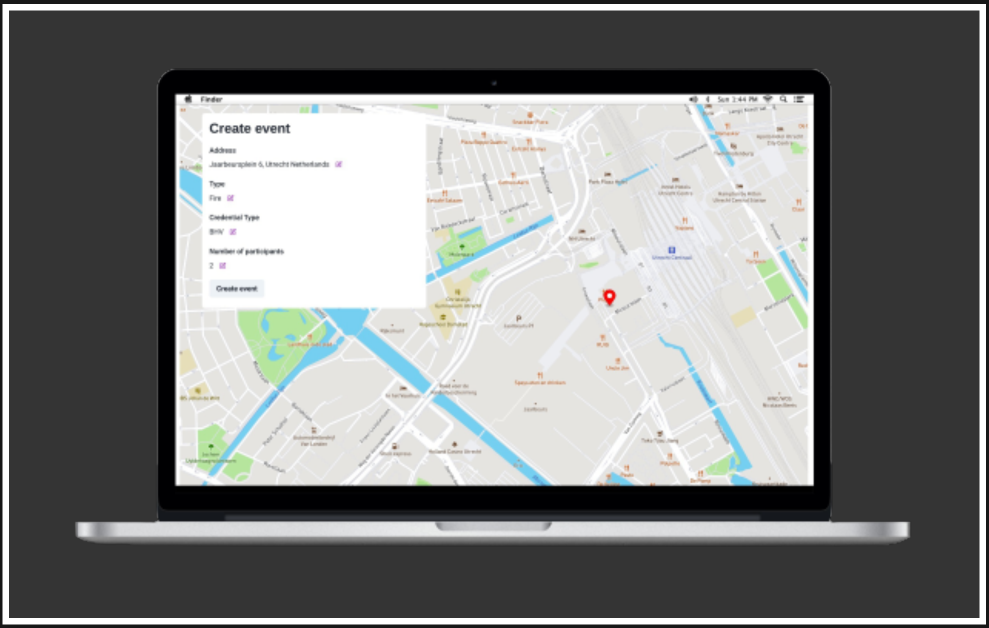
## Pilot Flow

- Fire Fighters receive credentials based on their qualifications and training levels
- Issued by the fire department where they work
- Credentials are stored in a custom holder application
- Dispatch matches existing low-risk emergency codes to qualifications
- Custom DIDComm protocols facilitate location sharing and communication



Fire

Distance: 0m
Travel time: 0 minutes
Address:



Create event

Address

Jaarbeursplein 6, Utrecht Netherlands

Type

Fire

Credential Type

BHV

Number of participants

2

21

## Challenges

- Storing the minimal amount of data that could be used to locate citizens
    - Push Notifciation Identifiers
    - Approximate Location
- Approximate, but still usefull location data
- How to integrate a wallet into existing volunteer mobile applications?
- How to scale
    - One of the applications for CPR volunteers has more than 170.000 volunteers

## Why SSI / Aries

- DIDComm allows for secure communication channel between dispatch center and volunteers
- Protocols allow for an open standard, other volunteer apps can integrate.
- Credentials allow the dispatch center to verify skills before sending them to the emergency, without storing data about its citizens

## The benefit of AFJ for this use case

- Provides a framework that we can use for issuer, holder and verifier side
- Open source foundation for other applications to integrate
- Accesible language means higher chance the government can mantain and extend it on a national scale
- Implement custom protocols once, use it on both mobile and server side

## Next

- Start practical implementation of pilot
    - Approx. 6 months to build and test first pilot version
- Keep working on AFJ (AIP 2.0, BBS+, DIDComm V2, etc…)
    - Will provide more efficient flow for users and dispatch
    - Multiple credential formats and signature suites will make it easier to add more and more qualifications

## Discussion