Our experience building with AFJ

Why AFJ? Why not!

Reasons for choosing AFJ over other frameworks

- Due to JavaScript:
    - Direct integration in React Native mobile apps
    - Ability to use the same codebase for mobile and cloud environments
    - Popular language, availability of libraries, easier learning curve
- Due to its architecture:
    - Modular architecture
    - Easy to follow and extend functionality
    - Fairly simple to embed into existing Node applications
- Due to its community
    - Continuously growing
    - Pretty active and willing to improve day by day
    - Interesting discussions in WG calls, open to everybody

How we are using it

Our project involves the integration of different services to the SSI world.

For this purpose, we are designing protocols over DIDComm to fit their needs (currently satisfied by more ‘traditional’ APIs).

These flows are used to connect services with other cloud-based services but also regular users running mobile or desktop applications. Some examples:

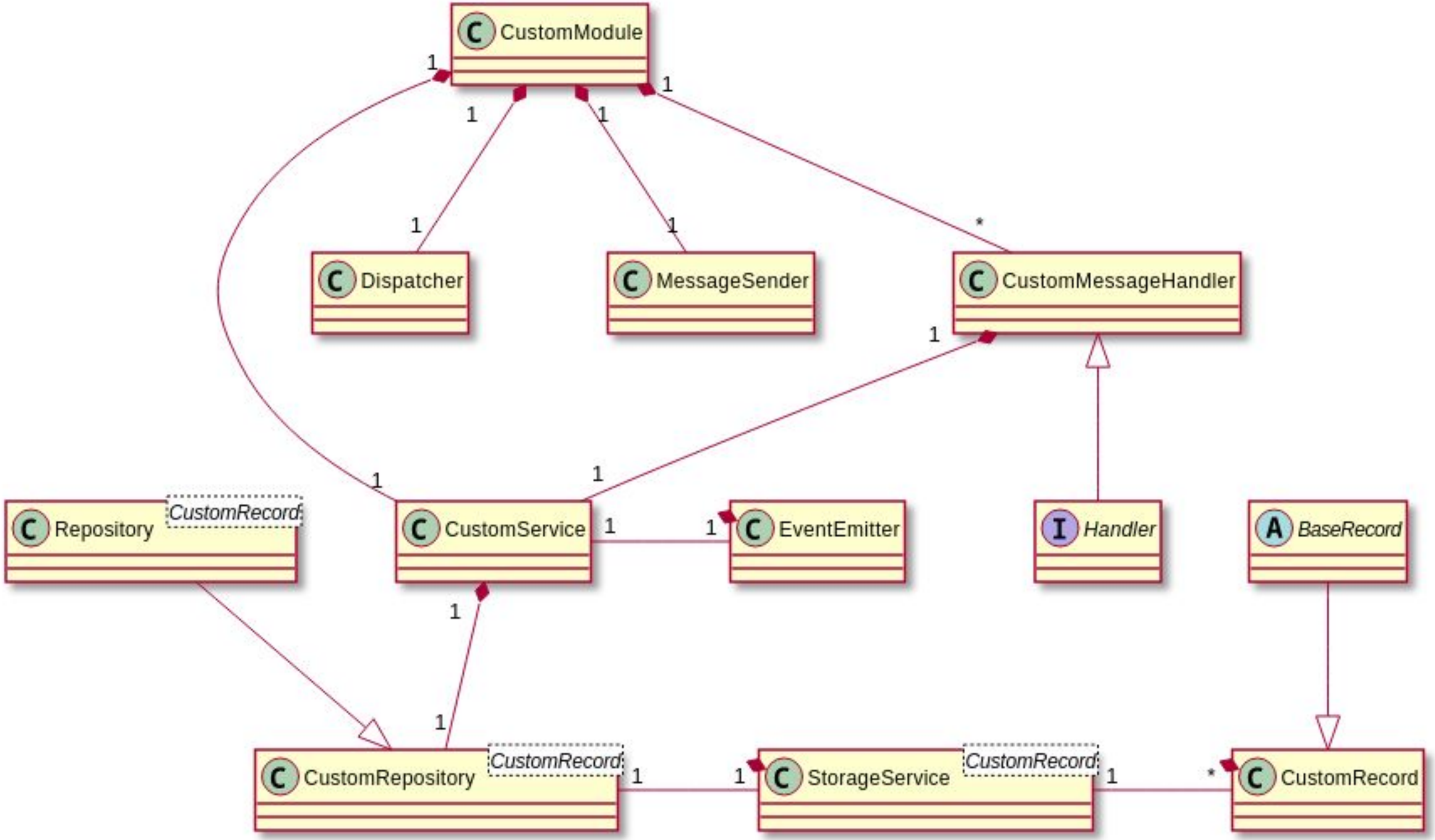
- Identity Verification
- User Registration / onboarding
- Chat (text, signalling for calls, document sharing, etc.)

How we are using (cont.)

For each protocol, we implement an extension module that provides an API to manage protocol flows. It registers handlers in Agent’s Dispatcher and is capable of emitting events like any other core module.

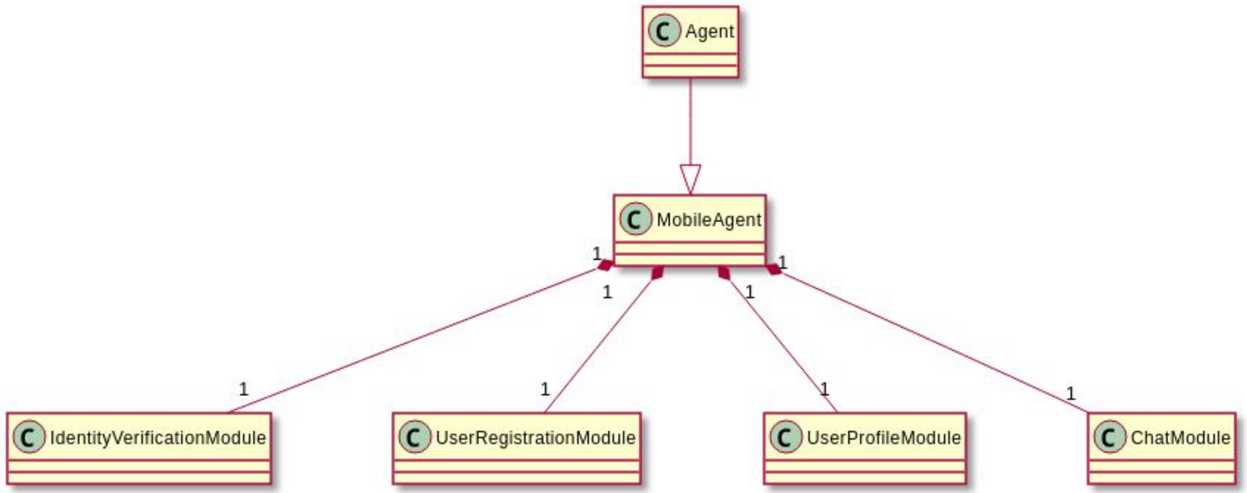
Extensions modules are not only used to support custom DIDComm protocols but also to add some other capabilities such as:

- Store arbitrary data in the wallet (user profile, settings, etc.)
- Add specific tags or metadata to records from existing core modules


CustomModule
Dispatcher
MessageSender
CustomMessageHandler
Repository
CustomRecord
CustomService
EventEmitter
Handler
BaseRecord
CustomRepository
CustomRecord
StorageService
CustomRecord
CustomRecord
1
1
1
1
1
1
*
1
1
1
1
1
1
1
1
1
1
*

How we are using (cont.)

We define different ‘custom Agents’ depending on the actor using it (e.g. end-users, verification services, organizations, agencies, etc.) that **inject** these extension modules and add some other project-specific capabilities.



What’s missing - PRs to come!

- Protocol nesting
- Persistent message queue
- Pluggable VDR/wallet backend (i.e. independence from Indy SDK)