Welcome !

Fabric Application Developer Community Call

October 3rd 2019

Agenda

HYPERLEDGER FABRIC

Today's Agenda :

- Wallets in Fabric SDK 2.0 – Mark Lewis (slides further down)
- Tutorial Updates – Paul O'Mahony
  - React – Typescript E2E tutorial now published – https://developer.ibm.com/tutorials/integrate-typescript-smart-contract-with-react-based-dashboard-app/
- AOB

Community Support

- **Questions & Open Community Support (Stack, Mailing List, Twitter)**

Contracts: http://stackoverflow.com/questions/tagged/hyperledger-fabric

http://stackoverflow.com/questions/tagged/hyperledger-chaincode

SDKs: http://stackoverflow.com/questions/tagged/hyperledger-fabric-sdk-js

(etc ie *'language suffix'*)

Twitter: https://twitter.com/hyperledger

Mailing List: details next page

Community Chat

- **Developer Chat (RocketChat) – channel names**

  https://chat.hyperledger.org/channel/

  #fabric-chaincode-dev

  #fabric-sdk-node

  #fabric-sdk-java

  #fabric-sdk-go

Application Developer Community Support (Mailing List)

- **Hyperledger Mailing List**

To subscribe or unsubscribe, visit
https://lists.hyperledger.org/g/fabric

or, via email, send a message with subject or body 'help' to
fabric@lists.hyperledger.org

Lists (Subgroups): https://lists.hyperledger.org/g/main

Help: fabric+help@lists.hyperledger.org

You can change your settings once you log in at
https://lists.hyperledger.org/g/main

Further Links

- **Fabric Developer - Community Home Page**
  https://wiki.hyperledger.org/display/fabric/Fabric+Application+Developer+Community+Calls
- **Fabric Developer Community – resources**
  https://wiki.hyperledger.org/display/fabric/Fabric+App+Developer%3A+New+Programming+Model+resources

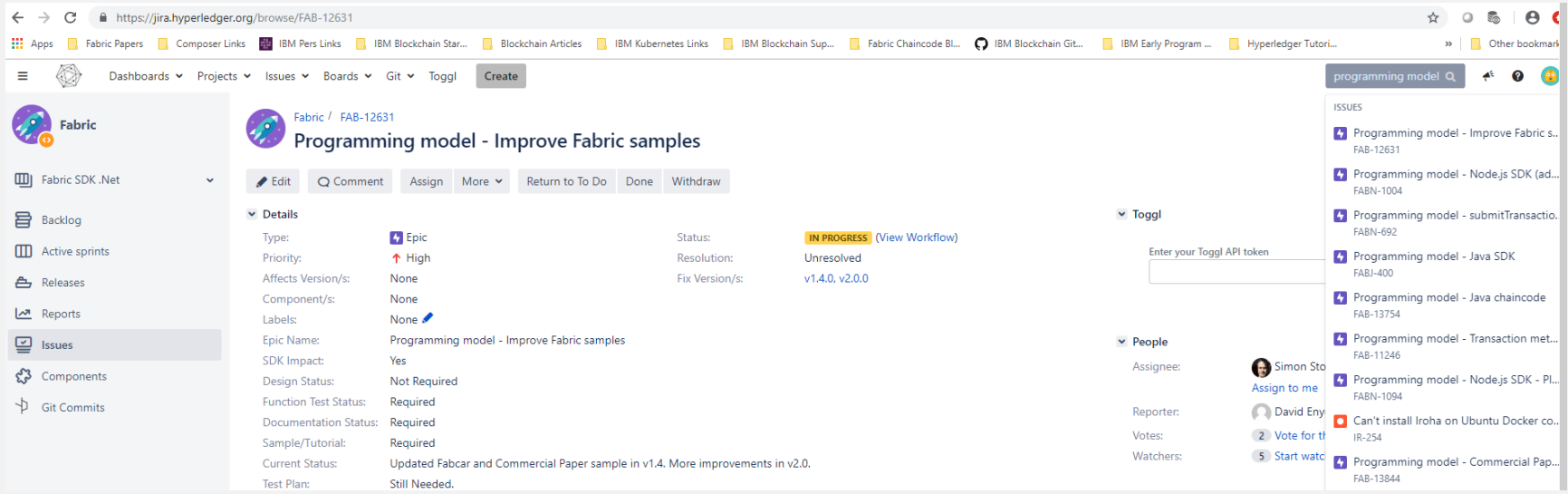
- [ ] **Developing Applications using Hyperledger Fabric** (using the new programming model)
  https://hyperledger-fabric.readthedocs.io/en/latest/developapps/developing_applications.html
- [ ] **Fabric Application Developer samples (Contracts, SDK, Tutorials):**
  - **Fabric samples Repo (eg Commercial Paper, Fabcar)**: : https://github.com/hyperledger/fabric-samples/tree/release-1.4/
  - **'Animal Tracking' TypeScript sample Contract/App Client (1.4)** https://github.com/mahoney1/animaltracking **with**
  - **Tutorial:** https://github.com/mahoney1/docs/blob/master/animaltracking-tutorial.md
- [ ] **Go developer API (prototype) example** - (see README): https://github.com/awjh-ibm/fabric-go-developer-api
- [ ] **Resources, tips and best practices** to share with the community
  https://github.com/ampretia/fabric-application-examples

Fabric Team encourages you to review/comment on current Fabric JIRAs

(ie stories, epics, requests relating to work-in-progress)

Eg. JIRA search - 'programming model' – (results below)

JIRAs – feel free to comment, give your input



Reference

HYPERLEDGER FABRIC

New Programming Model (1.4.x) :

- JIRA references - Fabric Programming Model Info:
  - https://jira.hyperledger.org/projects/FABN/issues/FABN-692
  - https://jira.hyperledger.org/browse/FAB-11246

THANKS !

HYPERLEDGER FABRIC

CIAO FOR NOW !

Addendum

Wallets in Fabric SDK 2.0

October 3rd 2019

Wallets in SDK 2.x

Mark S. Lewis

@bestbeforetoday

Wallet responsibilities

- Persist identity information
- Use identities for Fabric connections

- Allow user-defined persistent store
- Compatibility across SDK languages

Old wallet challenges

- Complex persistent format
- Hard to implement new stores
- Single identity type per wallet

Separate concerns

- Wallet interface
- Persistent store
- Interpret identity information

Using a wallet to connect

```
// Create wallet using default store: in-memory, file system or CouchDB
const wallet = await Wallets.newFileSystemWallet(walletDirectoryPath);

const gatewayOptions: GatewayOptions = {
    identity: 'user@example.org',
    wallet,
};
const gateway = new Gateway();
await gateway.connect(commonConnectionProfile, gatewayOptions);
```

Using a custom store

```
// Custom wallet store implementation
class MyCustomWalletStore implements WalletStore {
      async delete(label: string): Promise<void> { … }
      async get(label: string): Promise<Buffer | undefined> { … }
      async list(): Promise<string[]> { … }
      async put(label: string, data: Buffer): Promise<void> { … }
}

// Create wallet using custom store
const store = new MyCustomWalletStore();
const wallet = new Wallet(store);
```

Importing identity information

```
// Identity type defined by TypeScript
const identity: X509Identity = {
    credentials: {
        certificate: 'PEM format certificate string',
        privateKey: 'PEM format private key string',
    },
    mspId: 'wonderland',
    type: 'X.509',
};

await wallet.put('alice', identity);
```

Using a Hardware Security Module

```
// Provider holds details of HSM for HSM-managed identities in wallet
const hsmProvider = new HsmX509Provider({
    lib: '/path/to/hsm-specific/pkcs11/library',
    pin: '1234567890',
    slot: 0,
});
wallet.getProviderRegistry().addProvider(hsmProvider);

const identity: HsmX509Identity = {
    credentials: { certificate: 'PEM format certificate string' },
    mspId: 'org1',
    type: 'HSM-X.509',
};
await wallet.put('bob', identity);
```

Implementation detail

JSON persistent data format (IdentityData)

```
{
    credentials: {
        certificate: "…",
        privateKey: "…",
    },
    mspId: "…",
    type: "X.509",
    version: 1
}
```

Providers understand identity types

```
interface IdentityProvider {
    readonly type: string;
    fromJson(data: IdentityData): Identity;
    toJson(identity: Identity): IdentityData;
    setUserContext(client: Client,
        identity: Identity,
        name: string): Promise<void>;
}
```

Project changes

- New wallet code written in TypeScript
  - TypeScript definitions no longer hand-written
  - Typing aids code correctness, navigation and refactoring

- ***fabric-network*** source code now in **fabric-network/src**
  - Allows mixed JavaScript and TypeScript

- Compiled output in **fabric-network/lib** to minimize test code impact