

Appendix A

User Manual

To run the Healthcare PoC Application, users have to install the prerequisites and set up the environment. Besides, in this chapter, the instructions for using the app are given.

A.1 Pre-requisites

The application can be installed on Ubuntu or MacOS. The operating system we used for the project is Ubuntu Linux 16.04.7 LTS. We recommend you have at least 4GB of memory. The prerequisites given by the official document are listed below(17):

- Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
- Docker Engine: Version 17.03 or higher
- Docker-Compose: Version 1.8 or higher
- Node: 8.9 or higher (note version 9 is not supported)
- npm: v5.x
- git: 2.9.x or higher
- Python: 2.7.x
- A code editor of your choice, we recommend VSCode.

The prerequisites for Ubuntu can be downloaded from the following commands:

```
curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh  
chmod u+x prereqs-ubuntu.sh
```

It should be noticed that you should log in as a normal user rather than root, and don't use *su* to root. When running the script by following the command, it will use *sudo* during its execution and the password will be required.

```
./prereqs-ubuntu.sh
```

More instructions about the installation of prerequisites on Mac OS can be found in the official document (17).

The Hyperledger Composer does not support Node version 9 and higher. Node version 8 is recommended and it can be installed and switched to using the following commands:

```
nvm install v8
nvm use 8
```

Hyperledger Fabric is required to run the PoC application. The Hyperledger Fabric runtime we used is called **fabric-dev-servers** from the official. The following step is to create a directory for it and to get the **.tar.gz** file that contains the tools to install Hyperledger Fabric, and download a local Hyperledger Fabric v1.2 runtime:

```
mkdir ~/fabric-dev-servers && cd ~/fabric-dev-servers
curl -O https://raw.githubusercontent.com/hyperledger/composer-
tools/master/packages/fabric-dev-servers/fabric-dev-servers.tar.gz
tar -xvf fabric-dev-servers.tar.gz
export FABRIC_VERSION=hlfv12
./downloadFabric.sh
```

A.2 Set up environment

The Hyperledger runtime can be started or stopped using a set of scripts in **fabric-dev-servers**.

To start up a new runtime and create a PeerAdmin card:

```
cd ~/fabric-dev-servers
export FABRIC_VERSION=hlfv12
./startFabric.sh
./createPeerAdminCard.sh
```

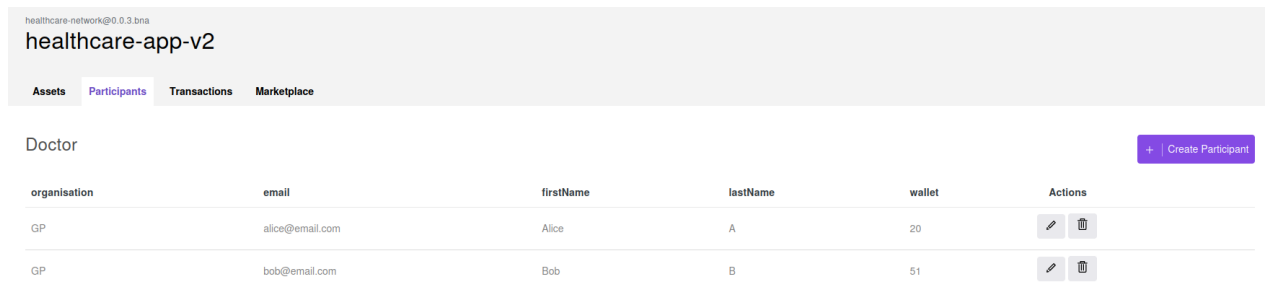
After finishing running the application, the Hyperledger Fabric runtime can be stopped and all the Docker containers can be removed by following commands:

```
cd ~/fabric-dev-servers
./stopFabric.sh
./teardownFabric.sh
docker kill $(docker ps -q)
docker rm $(docker ps -aq)
docker rmi $(docker images dev-* -q)
```

Then a Fabric v1.2 runtime is ready, and we can deploy the healthcare network on it. The required files are in the **healthcare-network** folder. We use 0.0.6 version **bn**a file to deploy the network, and import the network administrator identity as a usable business network card :

```
composer network install --archiveFile healthcare-network@0.0.6.bna --card
PeerAdmin@healthcare-network
composer network start --networkName healthcare-network --networkVersion
0.0.6 --card PeerAdmin@healthcare-network --networkAdmin admin
--networkAdminEnrollSecret adminpw
composer card import --file networkadmin.card
```

The deployed healthcare network can be checked by the following command:



organisation	email	firstName	lastName	wallet	Actions
GP	alice@email.com	Alice	A	20	
GP	bob@email.com	Bob	B	51	

Figure A.1: Example: Doctor Participant List

```
composer network ping --card admin@healthcare-network
```

Then generate the Restful Server for interacting with the composer network:

1.

```
composer-rest-server
```

2. Enter **'admin@healthcare-network'** as the card name.
3. Select **never use namespaces** when asked whether to use namespaces in the generated API.
4. Select **No** when asked whether to secure the generated API.
5. Select **Yes** when asked whether to enable event publication.
6. Select **No** when asked whether to enable TLS security.

To start the application, enter the **'healthcare-app-v2'** directory and use the command below:

```
npm start
```

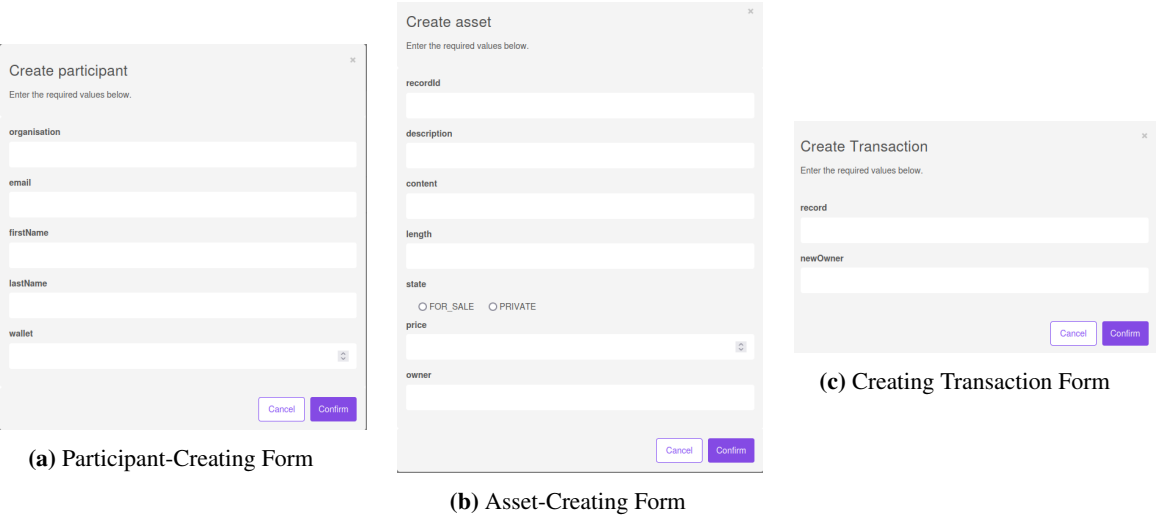
Finally, the healthcare application is running against the REST API at **http://localhost:4200**.

A.3 Instructions for using App

Since every medical record has an owner, at least one participant should be created first. Users can use the participant-creating form as shown in Figure A.2a on the Patient, Doctor, Healthcare Provider, or Government page to create a specific type of participant. The difference between Patient participants and others is that there is no **organisation** value in the patient-creating form. Note that every participant is identified by the value of **email**, so it cannot be duplicated in the system.

After the creation, all the existing participants are listed. Figure A.1 is an example screenshot.

Now the participants are ready to be assigned to their own medical records. On the Assets Page, there is an asset-creating form as shown in Figure A.2b to create the medical records. The **state** value can be **'FOR_SALE'** to publish the record in the marketplace or **'PRIVATE'** for the non-publishing record. The **price** is an optional value, and it can be used when there is a need to publish the record to the marketplace.



AssetsParticipantsTransactionsMarketplace							
recordId	description	content	length	state	price	owner	Actions
RECORD9	test record	Some prescriptions given by Alice	30	FOR_SALE	10	resource:org.healthcare.network.Doctor#alice@email.com	Buy

Figure A.3: Example: Marketplace

Note that the value of **owner** should be the participant name with a complete namespace. The namespace is **'org.healthcare.network.'**, adding the participant class and user email. For example, for a Doctor participant with an identified email **'bob@email.com'**, its full value should be **'org.healthcare.network.Doctor#bob@email.com'**.

As existing participants, all the records are listed on the Assets Page once they are created. For the records with a **'FOR_SALE'** state, they are also listed on the Marketplace Page as the example shown in Figure A.3.

All the existing participants and assets can be updated and deleted. When updating, the identified email for the participant cannot be changed, and the identified **'recordId'** for the asset cannot be changed.

There are two ways to transfer a data asset. One way is to fill out the transaction creating form shown in Figure A.2c to transfer it directly without a money transaction. As mentioned before, the value of **'newOwner'** should be an identified user email with its complete namespace. Another way is to buy it on the Marketplace page. Except for transferring the data asset to the new owner, the buying function also checks if the new owner has enough money to pay, changes the seller and buyer's wallets properly, and deletes the record from the marketplace.

Appendix B

Maintenance Manual

To run the application, developers should meet the prerequisites in Section A.1 and follow the steps in A.2 to set up the Hyperledger Fabric environment. Then maintenance and modification of the PoC application can be done from two aspects - the blockchain network deployment and the Angular web application.

B.1 Blockchain Network Deployment

We recommend using the Hyperledger Composer extension for VSCode to edit the code and do the application development. After launching VSCode, press the **Extensions** button on the vertical left toolbar, and type **composer** into the search bar to install the **Hyperledger Composer** extension. Reloading is required after the finish of installing to activate the extension.

Install the utility for generating application assets:

```
npm install -g generator-hyperledger-composer@0.20
```

Install the tool for generating the application:

```
npm install -g yo
```

Note that these tools are not necessary for the development. It helps developers to generate a simple application automatically, but developers can also use their own customised application.

Another tool that is not essential but can help with testing the business network before deploying is the composer playground, and it can be installed by the following command:

```
npm install -g composer-playground@0.20
```

There are three files to define the healthcare network, including **logic.js** file in **lib** directory, **org.healthcare.network.cto** file in **models** directory, and a **permissions.acl** file.

The logic file is a JavaScript file that defines the transaction function. In our case, it defines the function to transfer the medical record to the new owner, change the seller and buyer's wallets, and change the 'state' attribute of the record. This file should be modified if the function of transferring needs to be changed. The **cto** file is the model file where the structures of all the assets, participants, and transactions are defined. If the attributes of the existing assets, participants, and transactions need to be changed or a new category of asset, participant or transactions needs to

be added, the developer should modify the model file. The **acl** file is the file defining the access control rules. If the developer wants to change the permission for participants, this file should be edited.

These files are packaged into one **healthcare-network@0.0.6.bna** file for the healthcare network deployment. If the definition of the network needs to be changed, the developer should package the modified files into a new **bna** file and upgrade the healthcare network.

After modifying the code, the developer should open the **package.json** file in **healthcare-network directory**, and change the **version** from **0.0.6** to a new one (e.g. 0.0.7). Then package them into a new **bna** file, install and upgrade the healthcare network using following commands (We use version 0.0.7 as an example):

```
composer archive create -t dir -n .  
composer network install -a healthcare-network@0.0.7.bna -c peeradmin@hlfv1  
composer network upgrade -c peeradmin@hlfv1 -n healthcare-network -V 0.0.7
```

Finally, the healthcare network should be successfully upgraded.

B.2 Angular Application

All the files to set up the Angular application are in the '**healthcare-app-v2**' directory. To modify the function or the user interface for the web application, the developer should change the scripts under the path '**healthcare-network/healthcare-app-v2/src/app/**'. Each page corresponds to a directory. The **Record** directory corresponds to the Asset Page, and the **Transfer** directory corresponds to the Transaction Page.

In the Angular application, the routers are used for navigation. Each page is exported as a component and is imported in the '**app-routing.module.ts**' file to define the router. If new pages are added, the developer should import them as components and define the path for them in the routing file.

In each component directory, there are a CSS file and an HTML file for the design of the page. The functions are written in the **ts** file using TypeScript language. And there is a service **ts** file that helps with establishing a service that makes the component can communicate with others.

To transform the class from Hyperledger Composer to the class in the Angular application, the '**org.hyperledger.composer.system.ts**' file and the '**org.healthcare.network.ts**' file containing the code for the class export.

In the project, we achieved the function of publishing the for-sale medical records to the marketplace on the application level rather than the network layer. So, the functions related to publishing assets to the marketplace and deleting the sold assets are defined in the application scripts.