Before you start

This tutorial will walk you through building a Hyperledger Composer Blockchain solution from scratch.

In the space of a few hours you will be able to go from an idea for a disruptive blockchain innovation, to executing transactions against a real Hyperledger Fabric blockchain network and generating/running a sample Angular 2 application that interacts with a blockchain network.

The following steps are for use in the provided virtual machine. If you plan on going through the demo on your own machine with your own installation of Hyperledger, please make the proper changes in the path names and variable names.

The commands you will need to execute in the command line and the options you will have to choose in the command line are in red text. Everything else is in black text.

With this document in the form of a PDF, some single-commands might get divided into multiple lines. In this case, please copy the lines one after another in one command. The parts where such cases are likely are explicitly pointed out.

Machine Username: hyperledger
Machine Password: hyperledger

Steps for System Setup [NOT REQUIRED if you are using the VM]

Hyperledger prerequisites for Ubuntu

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

Composer prerequisites for Ubuntu

```
npm install -g composer-cli
npm install -g composer-rest-server
npm install -g generator-hyperledger-composer
npm install -g yo
npm install -g composer-playground
```

Fabric Installation (Every bullet is a single command)

- `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`
- `./downloadFabric.sh`

Docker Permission

```
sudo usermod -a -G docker $USER
sudo reboot
```

Starting fabric

Change directory into fabric-dev-servers by executing the following in the terminal

```
cd  /home/hyperledger/fabric-dev-servers
```

Start fabric

```
./startFabric.sh
```

Create credentials

```
./createPeerAdminCard.sh
```

Setting up your business network

To use the yo tool, execute the following commands in the terminal.

```
cd ~/
yo hyperledger-composer:businessnetwork
```

Set up the network as follows

Business network name: mynet
Description: nm
Author name: <Your name>
Author email: <Your email>
License: Apache-2.0
Namespace: org.example.mynet
Do you want to generate an empty template network?: No

Files will now be available in mynet

```
cd /home/hyperledger/mynet
```

As developers, the files important to us are in models, lib and permissions.acl

Defining the Business Network

Make the following changes to the files

To make the changes, you can either use a text editor like **Atom/gedit** to open the files, or you can open the files through the command line in **nano** by going to their location using cd. nano is a text editor for Unix systems that uses the command line interface. While it gets the job done, and while many programmers swear by it, the interface might not be as intuitive as any GUI based text editor you may have experience using.
The commands for nano will be given below. You do not need to use them if you choose to use Atom/gedit instead.

First, we will change the **org.example.mynet.cto** file.
The location of the file is /home/hyperledger/mynet/models.

If you are using Atom/gedit, you can right-click on the file and open it with Atom/gedit.
If are using nano,execute the following commands in the terminal:

```
cd /home/hyperledger/mynet/models
nano org.example.mynet.cto
```

Replace all the contents of org.example.mynet.cto with the following code snippet

```
/**
* My commodity trading network
*/
namespace org.example.mynet
asset Commodity identified by tradingSymbol {
      o String tradingSymbol
      o String description
      o String mainExchange
      o Double quantity
      --> Trader owner
}
participant Trader identified by tradeId {
      o String tradeId
      o String firstNameCt
      o String lastName
}
```

```
transaction Trade {
      --> Commodity commodity
      --> Trader newOwner
}
```

Save the file directly if working in **Atom/gedit**.
If working in **nano**, follow the following steps:
Save: Ctrl+O
Exit: Ctrl+X

Now, we will change the **logic.js** file in the models folder.
The location of the file is /home/hyperledger/mynet/lib.

If you are using Atom/gedit, you can right-click on the file and open it with Atom/gedit.
If are using nano, execute the following commands in the terminal:

```
cd /home/hyperledger/mynet/lib
nano logic.js
```

Replace all the contents of logic.js with the following code snippet

```
/**
* Track the trade of a commodity from one trader to another
* @param {org.example.mynet.Trade} trade - the trade to be processed
* @transaction
*/

async function tradeCommodity(trade) {
      trade.commodity.owner = trade.newOwner;
      let assetRegistry = await getAssetRegistry('org.example.mynet.Commodity');
      await assetRegistry.update(trade.commodity);
}
```

Save the file directly if working in **Atom/gedit**.
If working in **nano**, follow the following steps:
Save: Ctrl+O
Exit: Ctrl+X

Now, we will change the **permissions.acl** file in the models folder.
The location of the file is /home/hyperledger/mynet.

If you are using Atom/gedit, you can right-click on the file and open it with Atom/gedit.
If are using nano, execute the following commands in the terminal:

```
cd /home/hyperledger/mynet
nano permissions.acl
```

Replace all the contents of permissions.acl with the following code snippet

```
/**
* Access control rules for mynet
*/

rule Default {
      description: "Allow all participants access to all resources"
      participant: "ANY"
      operation: ALL
      resource: "org.example.mynet.*"
      action: ALLOW
}
rule SystemACL {
      description: "System ACL to permit all access"
      participant: "ANY"
      operation: ALL
      resource: "org.hyperledger.composer.system.**"
      action: ALLOW
}
```

Save the file directly if working in **Atom/gedit**.
If working in **nano**, follow the following steps:
Save: Ctrl+O
Exit: Ctrl+X

Generating the Business Network Archive file

Now that we have created the artefacts required for the project, we can now create the archive that can be deployed on the fabric. In the terminal execute

```
cd /home/hyperledger/mynet
```

composer archive create -t dir -n .

Deploying Business Network

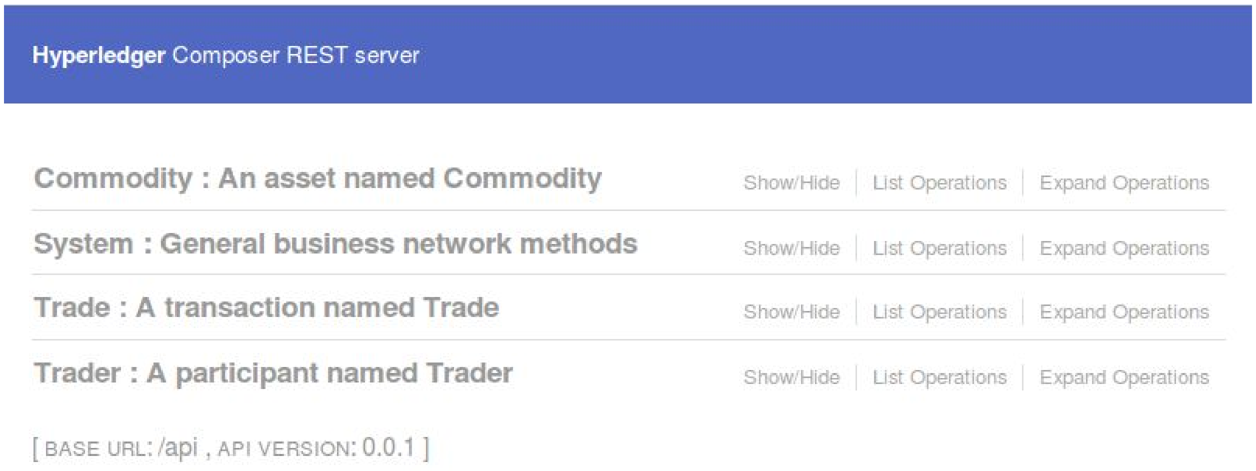
We can now install the archive on the fabric. Execute the following commands in the terminal.
IMPORTANT: The following commands are supposed to be on one line per bullet. If they show up as two different lines in the CLI, the command will not work as one line will execute before the other and will return some error. If that happens, please copy one line and paste the next line right next to it.

- cd /home/hyperledger/mynet
- composer network install --card PeerAdmin@hlfv1 --archiveFile mynet@0.0.1.bna
- composer network start --networkName mynet --networkVersion 0.0.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file networkadmin.card
- composer card import --file networkadmin.card
- composer network ping --card admin@mynet

Generate REST Server

We will now create a REST server. In the terminal, execute
composer-rest-server
Choose the following options/Enter the following data when prompted.
Enter the name of the business network card to use: admin@mynet
Specify if you want namespaces in the generated REST API: never use namespaces
Specify if you want to use an API key to secure the REST API: N
Specify if you want to enable authentication for the REST API using passport: N
Specify if you want to enable event publication over WebSockets: Y
Specify if you want to enable TLS security for the REST API: N
The REST server will now be installed.
Do not close this terminal window.

Now, go to your browser and open up http://localhost/3000
You should see something like this:



Hyperledger Composer REST server

Commodity : An asset named Commodity
Show/Hide | List Operations | Expand Operations

System : General business network methods
Show/Hide | List Operations | Expand Operations

Trade : A transaction named Trade
Show/Hide | List Operations | Expand Operations

Trader : A participant named Trader
Show/Hide | List Operations | Expand Operations

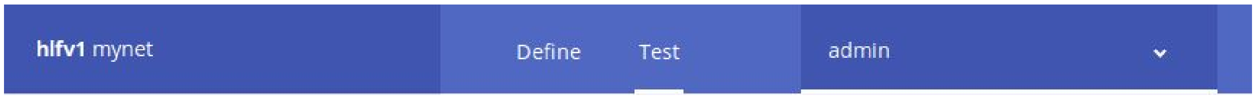
[BASE URL: /api , API VERSION: 0.0.1]

Composer Playground

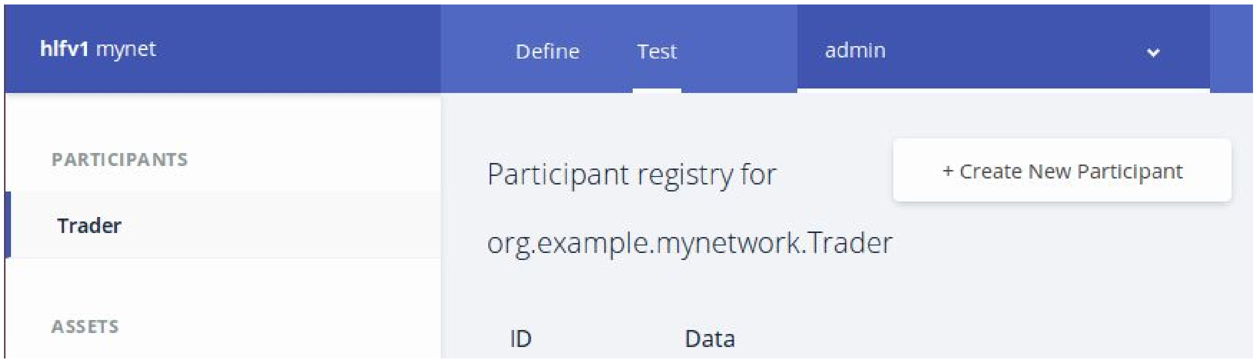
Now that we have deployed fabric on top of a business network, and started a REST server, let us take a look at the Composer Playground.
Open another terminal window and execute
composer-playground
A browser window with the address http://localhost/8080 will open.
You can now go into your project and check your model files, permission file, etc.
Playground offers a very good web interface so you can edit your code.
You can now go into admin and see all your business networks and connect to them.

Working with mynet

Let us try adding a trader now.
Connect to mynet in the web interface.
In the top, **click on "Test"**



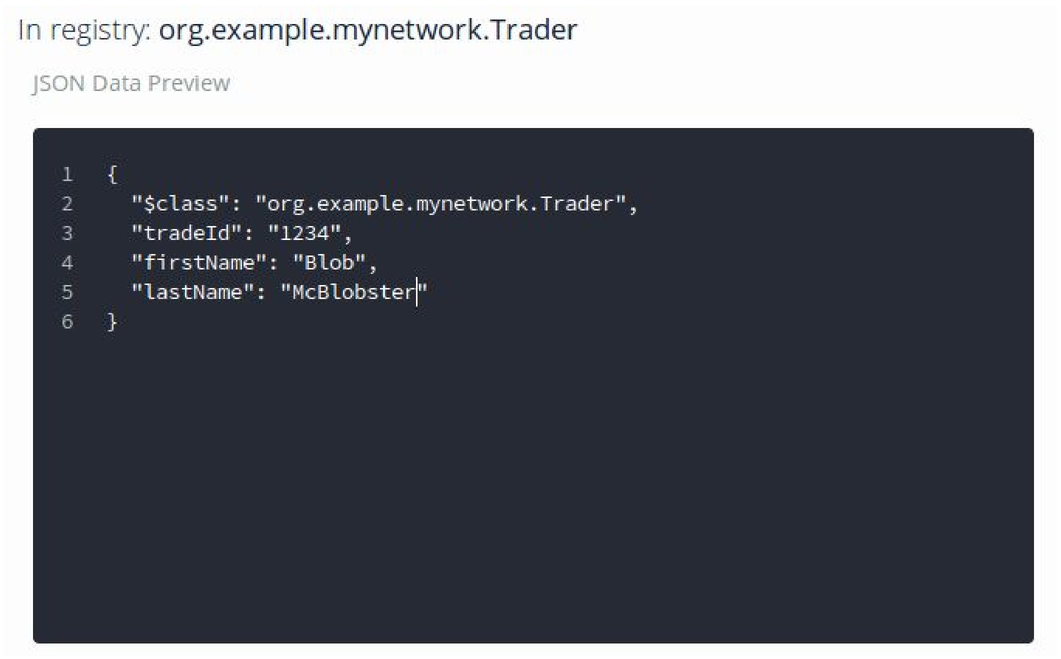
Next, in **Trader**, **click on "Create New Participant"**



Participant registry for org.example.mynetwork.Trader

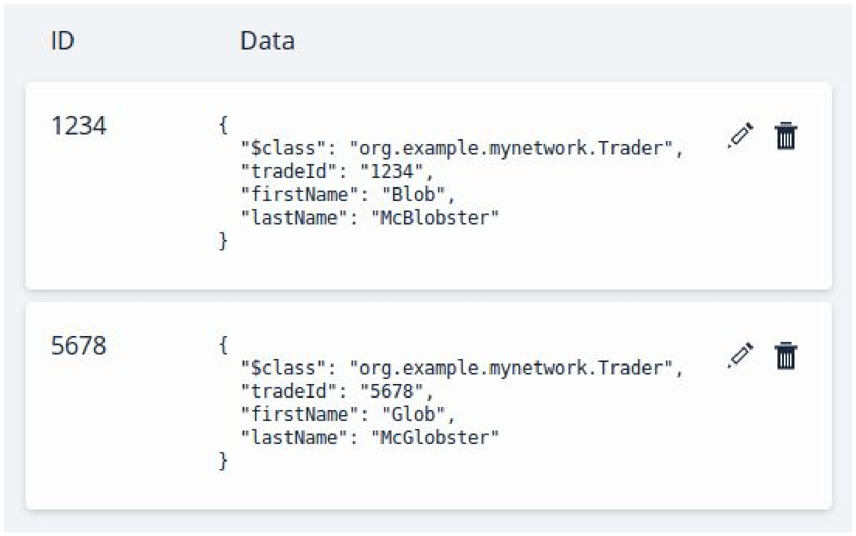
ID	Data

Create 2 traders by filling in data as follows.


In registry: org.example.mynetwork.Trader

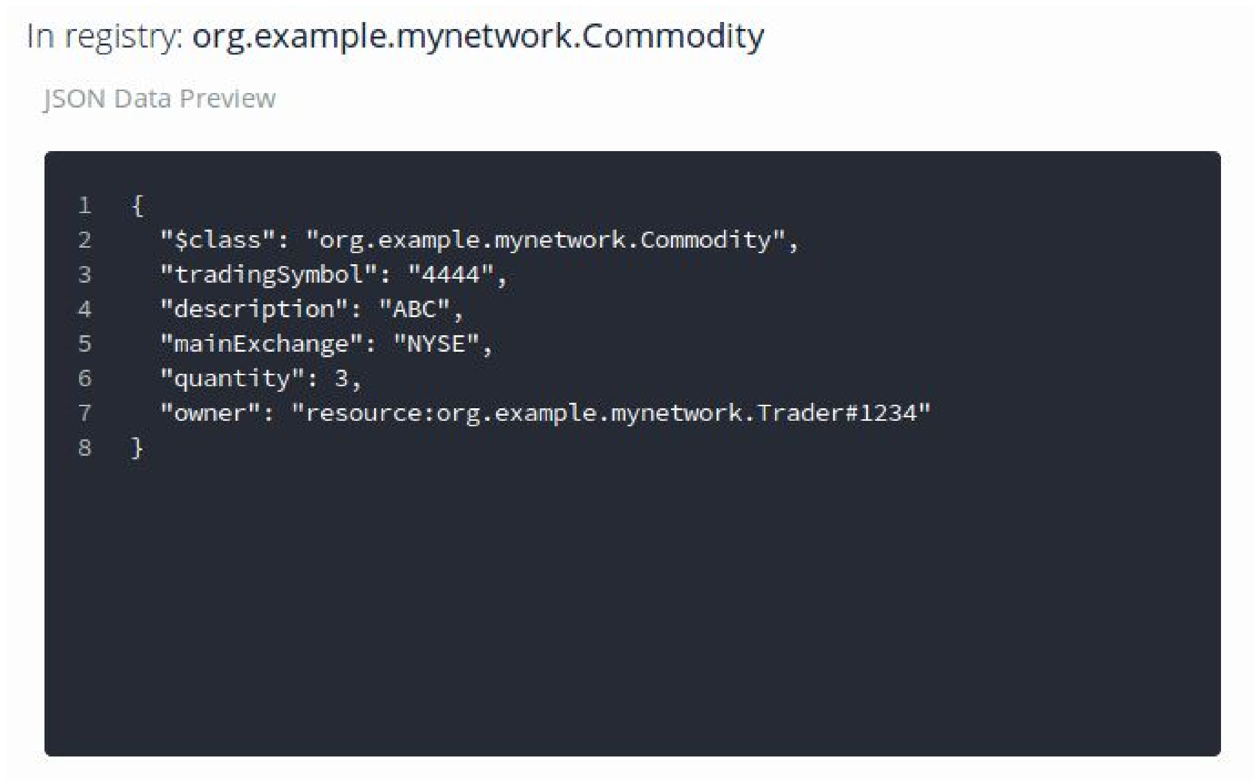
JSON Data Preview

```
{
  "$class": "org.example.mynetwork.Trader",
  "tradeId": "1234",
  "firstName": "Blob",
  "lastName": "McBlobster"
}
```



ID	Data
1234	{ "$class": "org.example.mynetwork.Trader", "tradeId": "1234", "firstName": "Blob", "lastName": "McBlobster" }
5678	{ "$class": "org.example.mynetwork.Trader", "tradeId": "5678", "firstName": "Glob", "lastName": "McGlobster" }

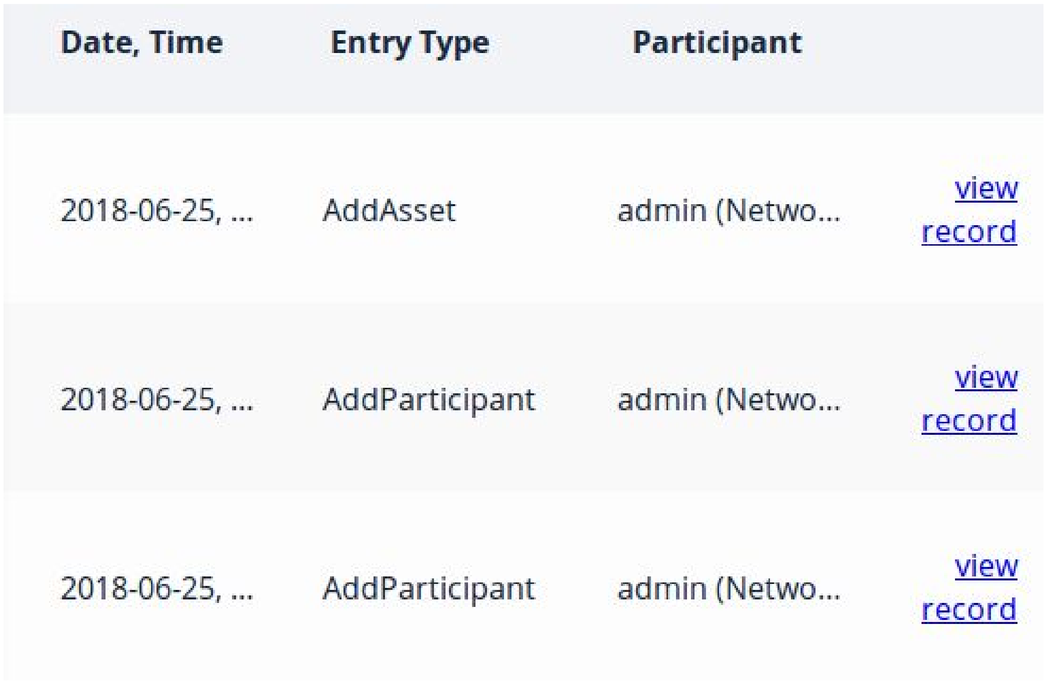
Create a new commodity by **clicking on commodity and then clicking on “Create New Asset”** as follows.



In registry: org.example.mynetwork.Commodity

```
{
  "$class": "org.example.mynetwork.Commodity",
  "tradingSymbol": "4444",
  "description": "ABC",
  "mainExchange": "NYSE",
  "quantity": 3,
  "owner": "resource:org.example.mynetwork.Trader#1234"
}
```

Click on All Transactions to see all the transactions that have happened on the network.

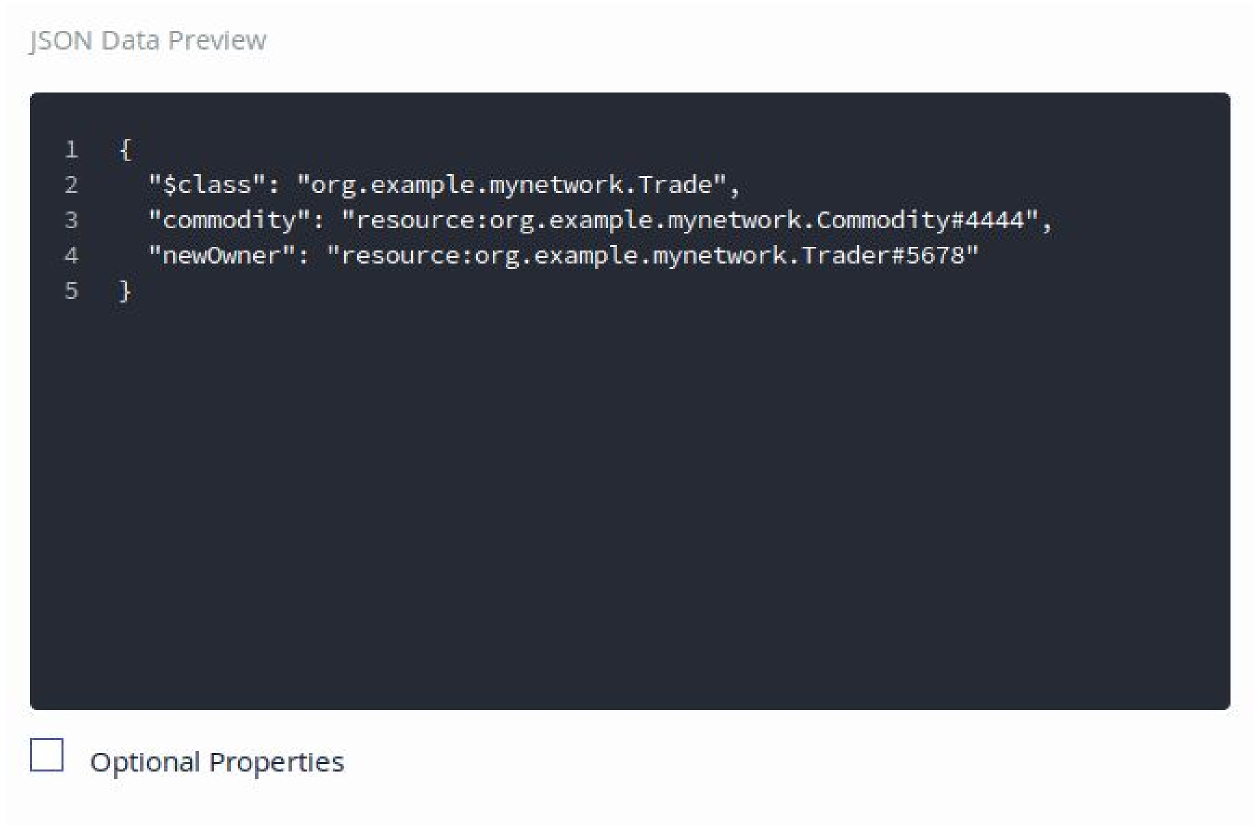


Date, Time	Entry Type	Participant	
2018-06-25, ...	AddAsset	admin (Netwo...	view record
2018-06-25, ...	AddParticipant	admin (Netwo...	view record
2018-06-25, ...	AddParticipant	admin (Netwo...	view record

Let us now transfer the ownership of the Commodity#4444 Trader#5678 (Glob McGlobster)

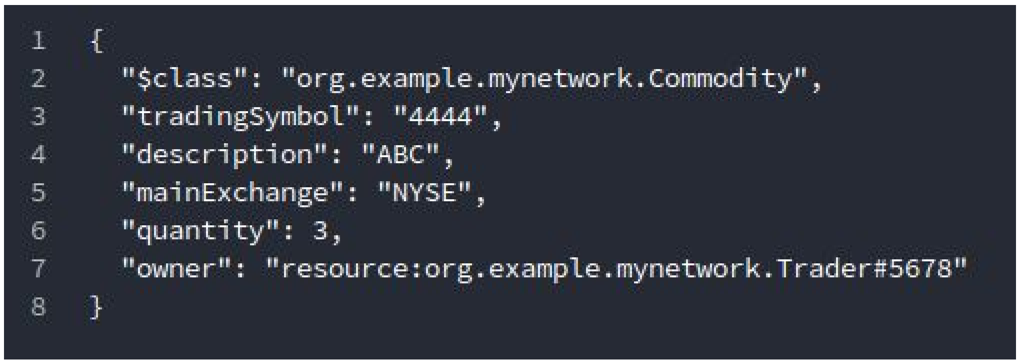
Click on Submit Transaction

Enter the Commodity's tradingSymbol and the Trader's traderId



```
{
  "$class": "org.example.mynetwork.Trade",
  "commodity": "resource:org.example.mynetwork.Commodity#4444",
  "newOwner": "resource:org.example.mynetwork.Trader#5678"
}
```

Check the details of the commodity after you submit the transaction.



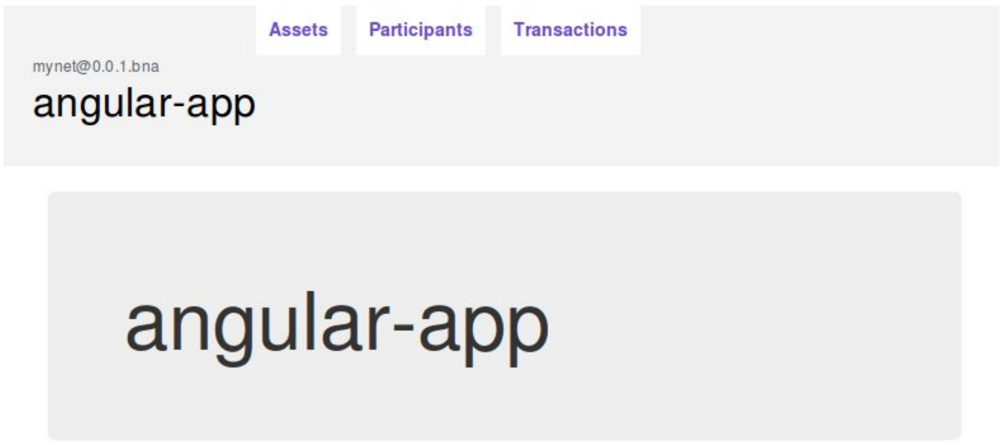
```
{
  "$class": "org.example.mynetwork.Commodity",
  "tradingSymbol": "4444",
  "description": "ABC",
  "mainExchange": "NYSE",
  "quantity": 3,
  "owner": "resource:org.example.mynetwork.Trader#5678"
}
```

Creating an Angular App to interact with the network

We are going to use the yo tool again.
In a new terminal window, execute
yo hyperledger-composer:angular
Select the following options.
Do you want to connect to a running Business Network? Y
Project name: angular-app
Description: Hyperledger Composer Angular project
Author name: <Your name>
Author email: <Your email>
License: Apache-2.0
Name of the Business Network card: admin@mynet
Do you want to generate a new REST API or connect to an existing REST API?: Connect to an existing REST API
REST server address: http://localhost
REST server port: 3000
Should namespaces be used in the generated REST API?: Namespaces are not used

Once the process is complete, go to the Angular directory by
cd /home/hyperledger/angular-app
Start the server by executing
npm start

Go to localhost:4200. You should see something like this.



You can now use your network in a manner similar to what you did in the Playground.