IBM Blockchain

Lab 1: Set up Hyperledger Composer Playground

Overview

This lab is divided into three parts:

- Set up the playground by importing the car auction code
- Transfer assets in a blockchain
- Explore the editor views and archive data

This lab takes place entirely in the web browser using Hyperledger *Composer Playground*.
Playground simulates the entire blockchain network within the browser by providing a sandpit environment to define, test, and explore business networks defined by using Composer. It is possible to connect to a live blockchain Hyperledger Fabric instance or install the Composer Playground on a local machine for more developer friendly tools.
Hyperledger Composer Playground is one method to use Hyperledger Composer. Other methods are also available at https://hyperledger.github.io/composer/installing/installing-index.html.

Hyperledger Composer

Hyperledger Composer (https://hyperledger.github.io/composer) is an open-source set of tools designed to make building blockchain applications easier.

It allows users to model the business networks, assets, and transactions that are required for blockchain applications, and to implement those transactions by using simple JavaScript functions. The blockchain applications run on instances of Linux Foundation Hyperledger Fabric (www.hyperledger.org).

The purpose of this lab is to introduce you to the concepts of a blockchain by showing you how a blockchain transfers assets between participants in a business network. We will use the implementation of a simple blind car auction as the scenario for the lab.

The car auction business network has a set of known participants (buyers and sellers), assets (cars and car listings), and transactions (placing bids and closing auctions). We will model these by using Hyperledger Composer and test the business logic that makes the auction work.

Crucially, a blockchain could be used to bring together the buyers and sellers of these assets without needing any trusted third party. However, an auctioneer could be used to provide visibility and governance of the network if required.

Prerequisites

Skill requirements:

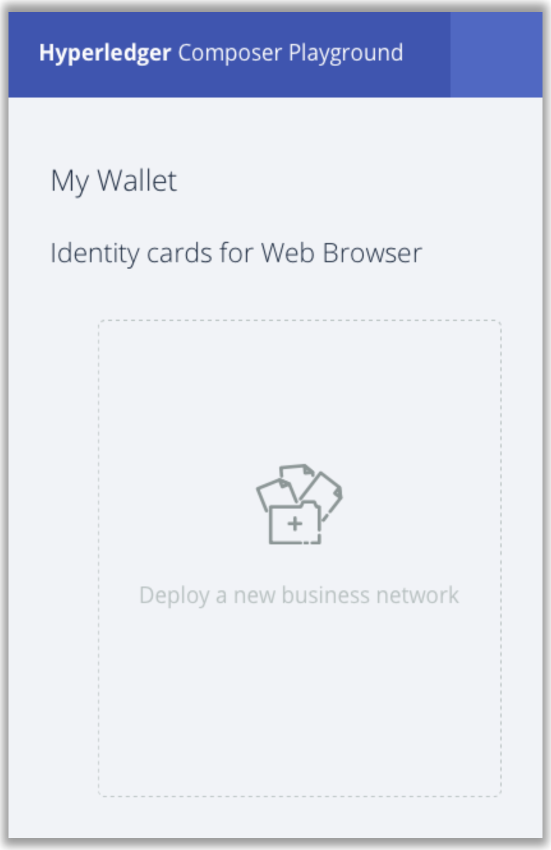
- There are no skill prerequisites to completing the first two parts.
- It is desirable but not essential to have some background knowledge of JavaScript for the last part called "Explore the editor views and export data."

Technical prerequisites:

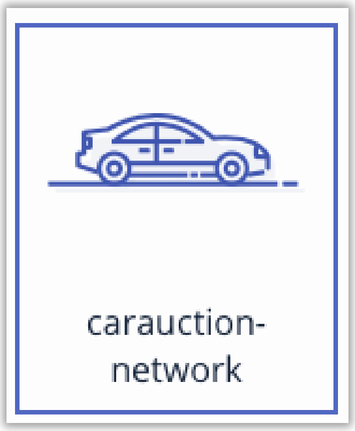
- Internet connection
- Web browser

Step 1. Import the car auction sample

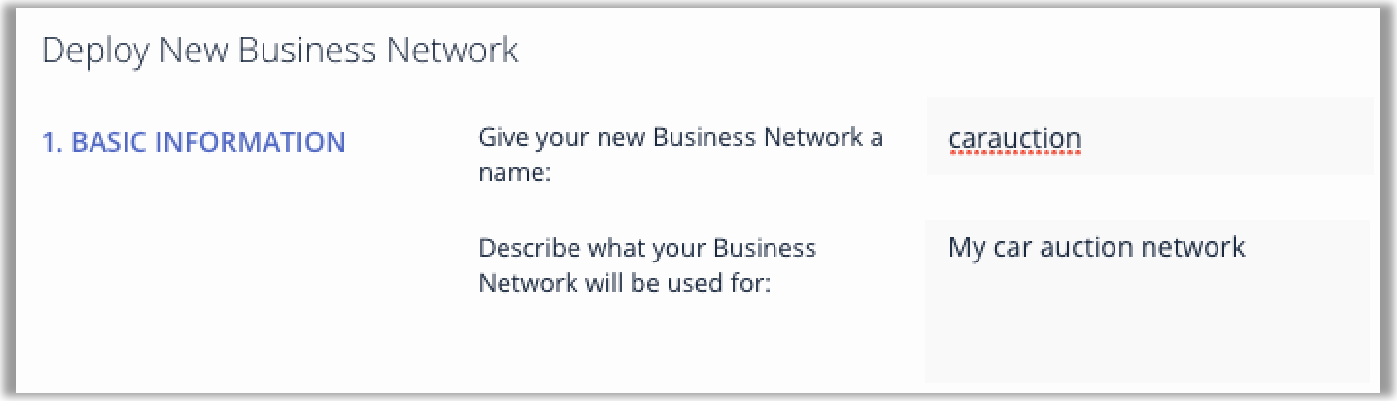
__1. Open a web browser and go to http://composer-playground.mybluemix.net. Dismiss the welcome screen to show the playground wallet screen which is used to connect and deploy new business networks:


Hyperledger Composer Playground
My Wallet
Identity cards for Web Browser
Deploy a new business network

__2. Click the "Deploy a business network" box. Then scroll down and select the carauction-network:



__3. Next give the business network a name and description:



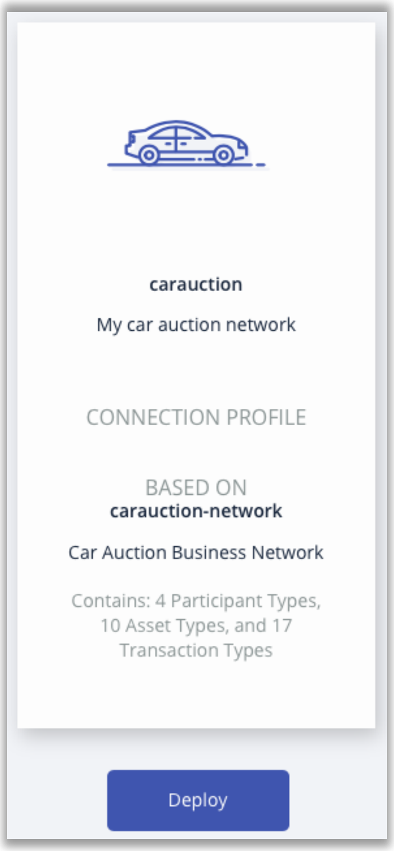
Deploy New Business Network

1. BASIC INFORMATION

Give your new Business Network a name: caraucution

Describe what your Business Network will be used for: My car auction network

__4. Click the Deploy button to deploy the new car auction business network:



caraucution

My car auction network

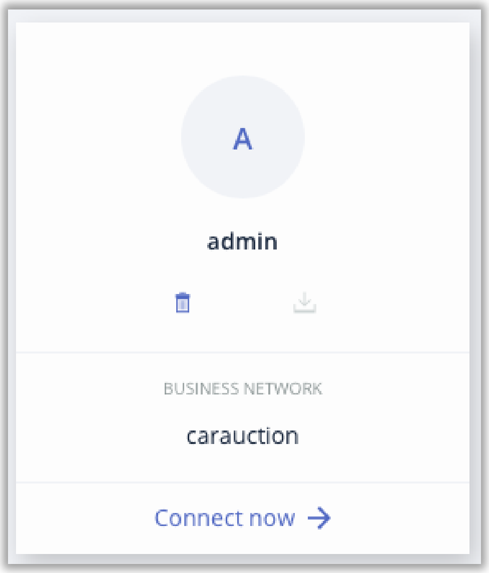
CONNECTION PROFILE

BASED ON
caraucution-network

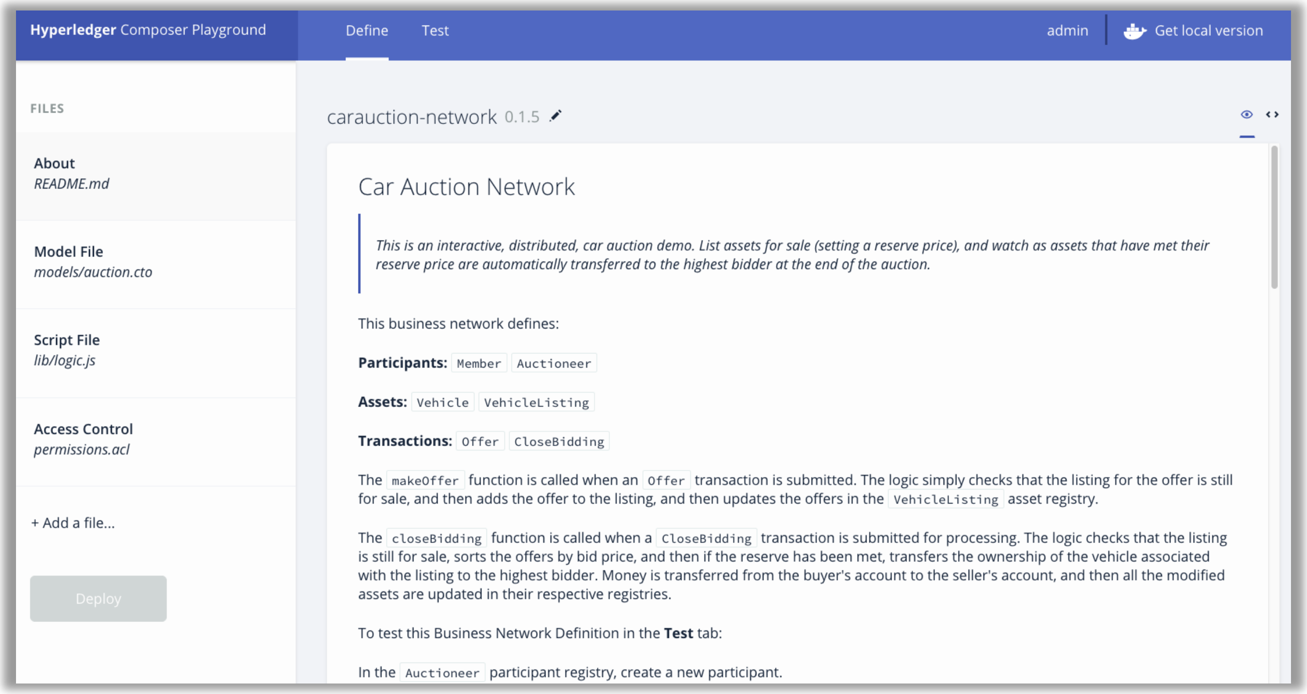
Car Auction Business Network

Contains: 4 Participant Types, 10 Asset Types, and 17 Transaction Types

__5. Click “Connect now” in the new identity card for the carauction network:



__6. Take a minute to read through the description of the car auction sample, to help understand the participants, assets and transactions associated with this particular network.



Car Auction Network

This is an interactive, distributed, car auction demo. List assets for sale (setting a reserve price), and watch as assets that have met their reserve price are automatically transferred to the highest bidder at the end of the auction.

This business network defines:

Participants: `Member` `Auctioneer`

Assets: `Vehicle` `VehicleListing`

Transactions: `Offer` `CloseBidding`

The `makeOffer` function is called when an `Offer` transaction is submitted. The logic simply checks that the listing for the offer is still for sale, and then adds the offer to the listing, and then updates the offers in the `VehicleListing` asset registry.

The `closeBidding` function is called when a `CloseBidding` transaction is submitted for processing. The logic checks that the listing is still for sale, sorts the offers by bid price, and then if the reserve has been met, transfers the ownership of the vehicle associated with the listing to the highest bidder. Money is transferred from the buyer's account to the seller's account, and then all the modified assets are updated in their respective registries.

To test this Business Network Definition in the **Test** tab:

In the `Auctioneer` participant registry, create a new participant.

1.1.1. Add Three Participants

In the next section we will now work with the deployed car auction blockchain network.

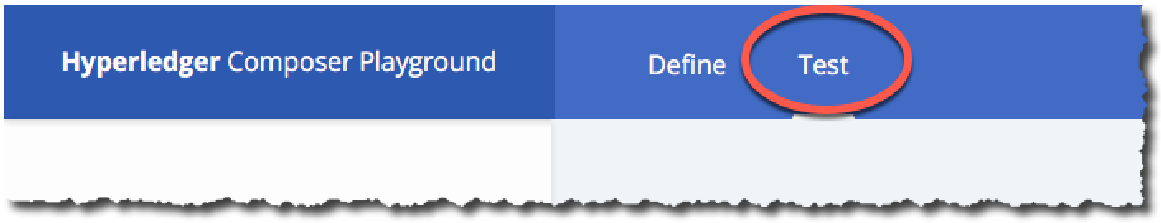
We will first instantiate three *Member* participants of the car auction business network:

- Alice Smith (alice@email.com), who will make a bid on a car,
- Bob Jones (bob@email.com), who will also make a bid on a car, and

- Charlie Brown (charlie@email.com), who currently owns a car.

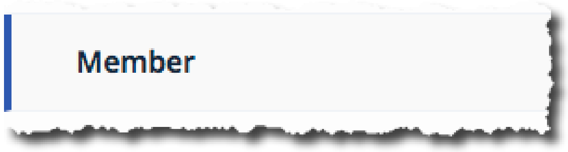
We will not instantiate an Auctioneer in this demo; this could be used in order to provide oversight of the network, although is not necessary.

__7. Click the **Test** tab and then click on the *Member* participant registry:

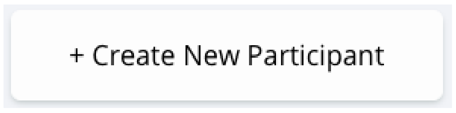


The registry is empty as no members have currently been defined.

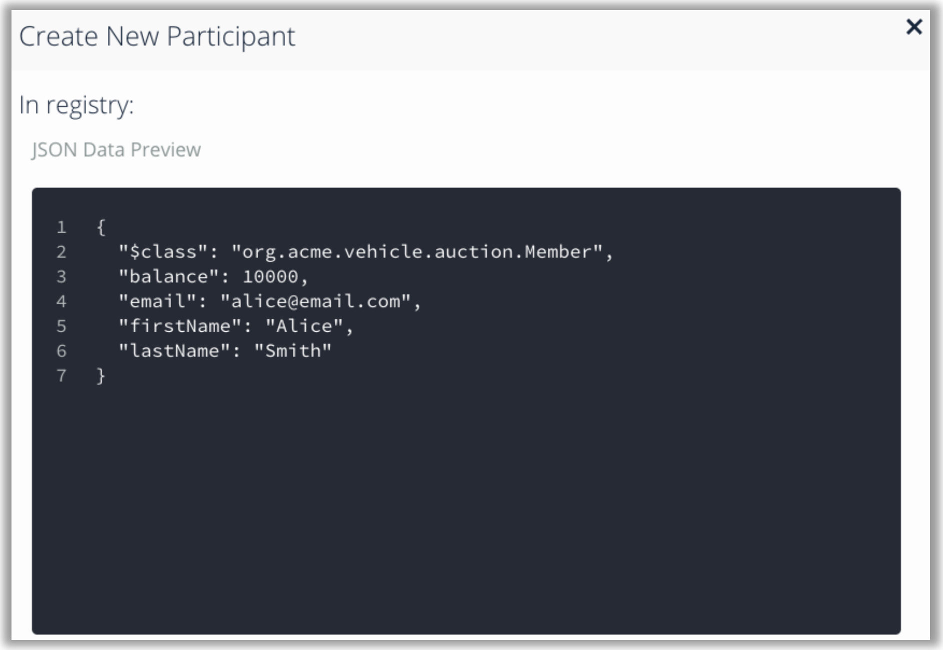
__8. Click on **Member** to view there are no current members in the environement



__9. Click **Create New Participant** to add a new Member.



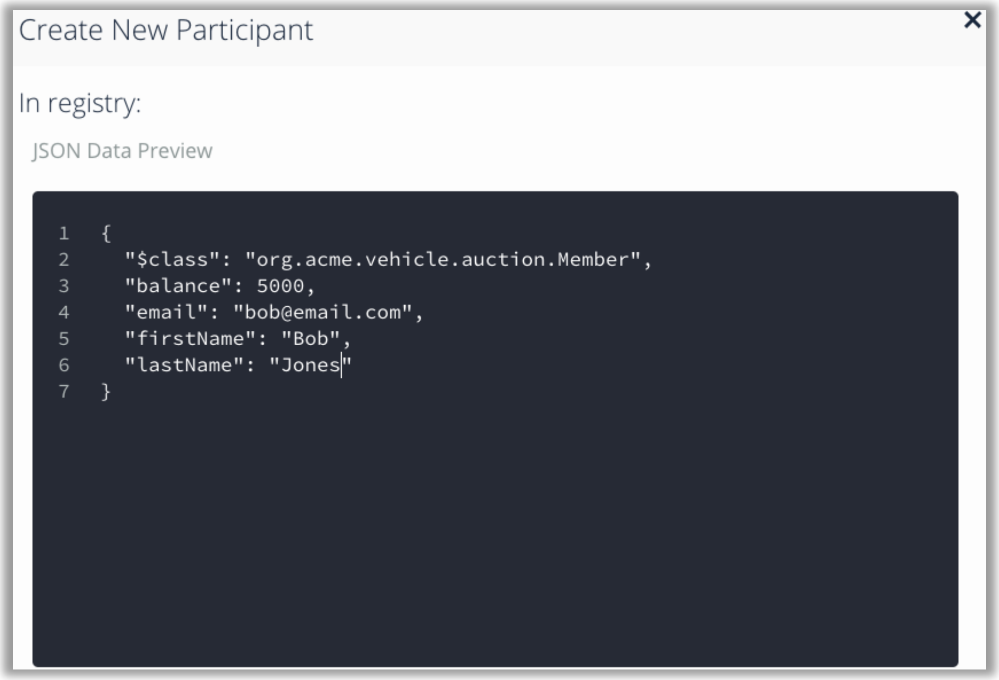
__10. Type the correct values into the JSON data structure to add Alice to the business network. Let's give her a starting balance of 10000.



```
{
  "$class": "org.acme.vehicle.auction.Member",
  "balance": 10000,
  "email": "alice@email.com",
  "firstName": "Alice",
  "lastName": "Smith"
}
```

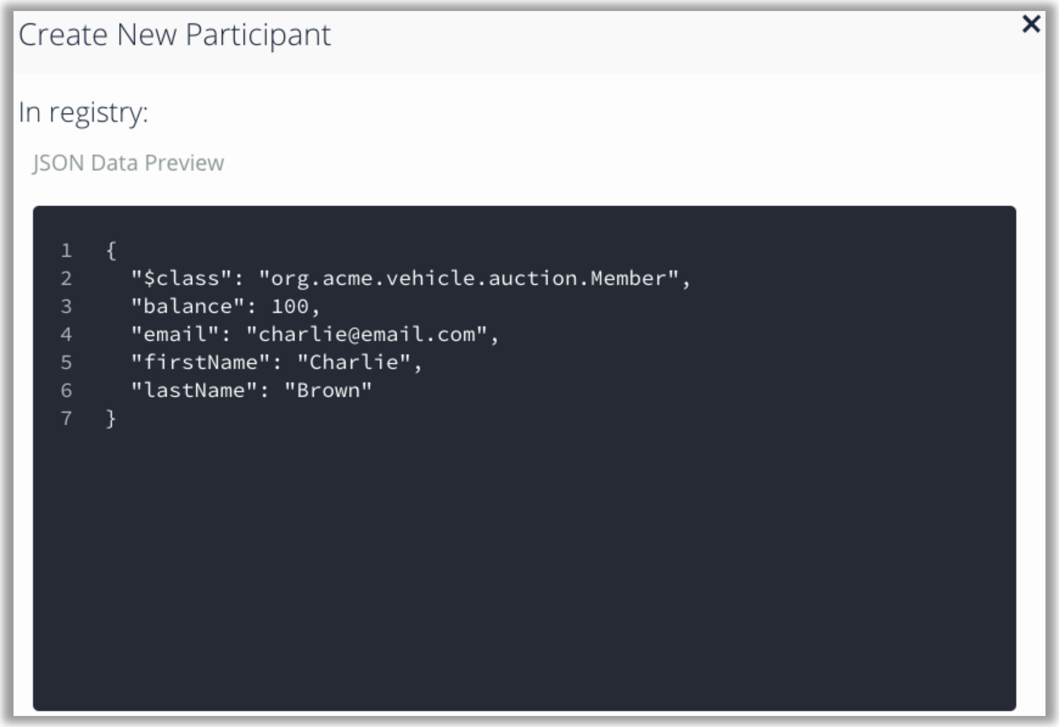
__11. Click **Create New** to add Alice to the registry.

__12. Do the same for Bob. Let’s give him a starting balance of 5000.



```
{
  "$class": "org.acme.vehicle.auction.Member",
  "balance": 5000,
  "email": "bob@email.com",
  "firstName": "Bob",
  "lastName": "Jones"
}
```

__13. Finally do the same for Charlie. He hasn't got so much money (he's selling his car, after all) so let's give him a starting balance of 100.



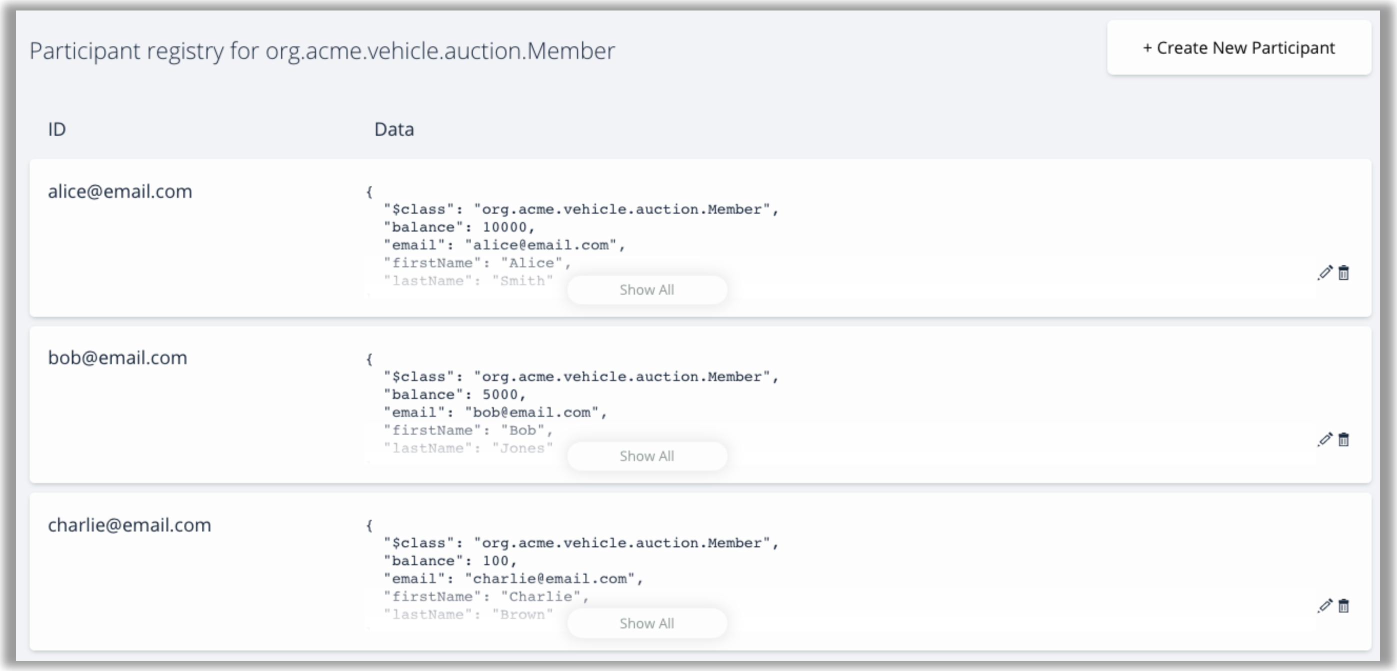
Create New Participant

In registry:

JSON Data Preview

```
{
  "$class": "org.acme.vehicle.auction.Member",
  "balance": 100,
  "email": "charlie@email.com",
  "firstName": "Charlie",
  "lastName": "Brown"
}
```

__14. Verify that all participants in the business network have been correctly defined. Use the appropriate Edit button (✎) to make any changes.



Participant registry for org.acme.vehicle.auction.Member

+ Create New Participant

ID	Data
alice@email.com	{ "$class": "org.acme.vehicle.auction.Member", "balance": 10000, "email": "alice@email.com", "firstName": "Alice", "lastName": "Smith"
bob@email.com	{ "$class": "org.acme.vehicle.auction.Member", "balance": 5000, "email": "bob@email.com", "firstName": "Bob", "lastName": "Jones"
charlie@email.com	{ "$class": "org.acme.vehicle.auction.Member", "balance": 100, "email": "charlie@email.com", "firstName": "Charlie", "lastName": "Brown"