Zero To Blockchain Chapter04 for hyplerledger composer.

Table of Contents

This chapter focuses on the following files:

```
Chapter 04
  ↳ network
   permissions.acl
   queries.qry
    ↳lib
      sample.js
    ↳models
      base.cto
      events.cto
      sample.cto
    ↳test
      sample.js
```

Key files:

- **models/sample.cto**
  - This is the 'model' file, which defines the types of members and assets in a business network and also establishes the speicif transactions which can be executed in a network.
- **lib/sample.js**
  - The sample.js file in the lib folder contains all of the code to implement each transaction defined in the sample.cto file.
- **test/sample.js**
  - This file provides the unit tests for the network.

Installing the code:

- run this command from a terminal window: `npm install`
  - when you run `npm install` command, you will get the following error, which you can ignore. The error is caused by the installation process for the SDK tooling, which searches for all instances of files ending in `.cto` . Because we have those files in the answers folder, as well as in the (correctly placed) network/models folder, npm install gets confused. It's ok, the process will still work correctly.

```
Looking for package.json of Business Network
Definition
Input directory:
/Users/robertdill/Documents/GitHub/Z2B_Master/Ch
apter04
 Error: namespace already exists
     at ModelManager.addModelFiles
(/usr/local/lib/node_modules/composer-
cli/node_modules/composer-
common/lib/modelmanager.js:241:31)
     at Function._processModelFiles
(/usr/local/lib/node_modules/composer-
cli/node_modules/composer-
common/lib/businessnetworkdefinition.js:457:43)
     at Promise.resolve.then
(/usr/local/lib/node_modules/composer-
cli/node_modules/composer-
common/lib/businessnetworkdefinition.js:620:18)
     at process._tickCallback
(internal/process/next_tick.js:109:7)
     at Module.runMain (module.js:606:11)
     at run (bootstrap_node.js:383:7)
     at startup (bootstrap_node.js:149:9)
     at bootstrap_node.js:496:3
 Error: namespace already exists
```

```
        at ModelManager.addModelFiles
  (/usr/local/lib/node_modules/composer-
  cli/node_modules/composer-
  common/lib/modelmanager.js:241:31)
        at Function._processModelFiles
  (/usr/local/lib/node_modules/composer-
  cli/node_modules/composer-
  common/lib/businessnetworkdefinition.js:457:43)
        at Promise.resolve.then
  (/usr/local/lib/node_modules/composer-
  cli/node_modules/composer-
  common/lib/businessnetworkdefinition.js:620:18)
        at process._tickCallback
  (internal/process/next_tick.js:109:7)
        at Module.runMain (module.js:606:11)
        at run (bootstrap_node.js:383:7)
        at startup (bootstrap_node.js:149:9)
        at bootstrap_node.js:496:3 ```
```

- then run buildAndDeploy (yes, case is important)
- then run npm run test
- this should deliver the following result:

```
Finance Network
   #createOrder
     ✓ should be able to create an order (134ms)
   #issueBuyRequest
     ✓ should be able to issue a buy request (76ms)
   #issueOrderFromSupplier
     ✓ should be able to issue a supplier order
(64ms)
   #issueRequestShipment
     ✓ should be able to issue a request to ship
product (51ms)
   #issueDelivery
     ✓ should be able to record a product delivery
```

```
   #issueRequestPayment
     ✓ should be able to issue a request to request
payment for a product (58ms)
   #issuePayment
     ✓ should be able to record a product payment
   #issueDispute
     ✓ should be able to record a product dispute
   #issueResolution
     ✓ should be able to record a dispute resolution
(41ms)
   #issueBackorder
     ✓ should be able to record a product backorder


  10 passing (2s)
```

The business network definitions are contained in the **sample.cto** file, which includes the *base.cto* and the *events.cto* file.

Participant

- Buyer
- Seller
- Provider
- Shipper
- FinanceCo

Asset

- Order
- CreateOrder
- Buy
- OrderFromSupplier

- RequestShipping
- Deliver
- RequestPayment
- Pay
- Dispute
- Resolve
- Backorder

Event

- (none yet)

Orders are created by Buyers and executed by Sellers, who may work with a 3rd part (Provider) to fulfill the order. Either Sellers or Providers can RequestShipment, which is fulfilled by a Shipper who executes a Deliver transaction when complete. Orders can be Disputed. Disputed Orders can be resolved. Payments are made against either Delivered or Resolved Orders.

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

Create a Order asset:

```
asset Order identified by orderNumber {
    o String orderNumber
    o String status
    o Integer amount
    o String created
    o String bought
    o String ordered
    o String dateBackordered
    o String requestShipment
    o String delivered
    o String disputeOpened
    o String disputeResolved
    o String paymentRequested
    o String orderRefunded
    o String paid
    o String[] vendors
    o String dispute
    o String resolve
    o String backorder
    o String refund
    --> Buyer buyer
    --> Seller seller
```

Create a participant:

```
participant Buyer identified by buyerID {
    o String buyerID
    o String companyName
}
participant Seller identified by sellerID {
    o String sellerID
    o String companyName
}

asset Order identified by orderNumber {
    o String orderNumber
    o String status
    o Integer amount
    o String created
    o String bought
    o String ordered
    o String dateBackordered
    o String requestShipment
    o String delivered
    o String disputeOpened
    o String disputeResolved
    o String paymentRequested
    o String orderRefunded
    o String paid
    o String[] vendors
    o String dispute
    o String resolve
    o String backorder
    o String refund
    --> Buyer buyer
    --> Seller seller

}
participant Shipper identified by shipperID {
    o String shipperID
    o String companyName
}
participant Provider identified by providerID {
    o String providerID
    o String companyName
}
participant FinanceCo identified by financeCoID {
    o String financeCoID
    o String companyName
}
```

Submit a transaction:

```
  transaction CreateOrder {
    o Integer amount
    --> Order order
    --> Buyer buyer
    --> Seller seller
}
  transaction Buy {
    --> Order order
    --> Buyer buyer
    --> Seller seller
```

```
}
  transaction OrderFromSupplier {
    --> Order order
    --> Provider provider
}
  transaction RequestShipping {
    --> Order order
    --> Shipper shipper
}
  transaction Deliver {
    --> Order order
    --> Shipper shipper
}
  transaction BackOrder {
    o String backorder
    --> Order order
    --> Provider provider
}
  transaction Dispute {
    o String dispute
    --> Order order
    --> Buyer buyer
    --> Seller seller
    --> FinanceCo financeCo
}
  transaction Resolve {
    o String resolve
    --> Order order
    --> Buyer buyer
    --> Seller seller
    --> FinanceCo financeCo
}
  transaction RequestPayment {
    --> Order order
    --> Buyer buyer
    --> Seller seller
    --> FinanceCo financeCo
}
  transaction Pay {
    --> Order order
    --> Seller seller
    --> FinanceCo financeCo
}
  transaction Refund {
    o String refund
    --> Order order
    --> Buyer buyer
    --> Seller seller
    --> FinanceCo financeCo
}
```