Ethereum
Blockchain Development
Week 6

1 Introduction

5–10

Getting started is covered in the video (w6-1.mpg) available from Ethereum-myLearning website. All today's exercises are to be saved in week 6 directory. Follow the instructions below:

- Open up the VM machine
- Open a terminal (CTRL-ALT-T)
- In your 'Ethereum' directory create a week 6 directory: `mkdir 6`
- In the lecture we adapted the trader example available from hyperledger. Download the file, '`trader1.bna`', to the week 6 directory. Ensure you are in the week 6 directory (`cd 6`) and download the file using the `wget` command as follows: `wget -c https://blockchain.smerf.net/6/` (alternatively open a browser and go to this address and move the file from the downloads to the week 6 directory).

- Start composer playground by typing the following in the command prompt: `composer-playground`

From the hperledger composer-playground business network page add a new network using the 'trader1.bna' file you just downloaded (please refer to the video if you are experiencing any problems) with the following details:

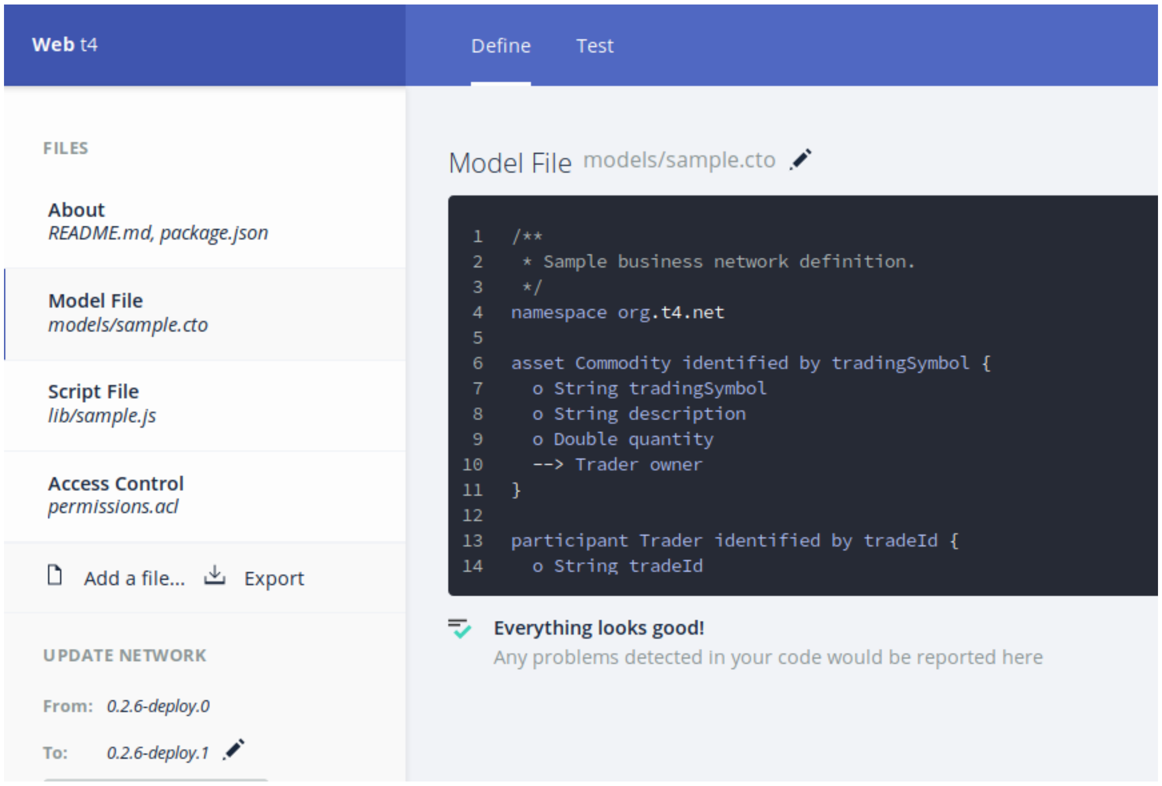
- name: t2
- namespace: org.t2.net
- Admin: admin@t2
- Deploy the network
- Connect to the admin@t2 network

2 Composer Setup

Access the business network via the composer-playground interface and you should see a similar webpage as displayed in Fig 2.1. There are 4 files; the first file contains some very basic information about the blockchain application you are building, please populate as appropriate. 5–10

Complete the following instructions:

1. Click on the Model File tab and enter the code in Fig 2.2
2. Click on the Script File tab and enter the code in Fig 2.3
3. Click on the Access Control tab and enter the code in Fig 2.4
4. Click deploy network and then Test
5. Create two participants with id '0001' and '0002', respectively
6. Create 1 commodity with id '0010' with ownership registered to trader with id, '0001'
7. Complete the transaction that transfers the ownership of the commodity, '0010' to a new trader, '0002'.



```
/**
 * Sample business network definition.
 */
namespace org.t4.net

asset Commodity identified by tradingSymbol {
  o String tradingSymbol
  o String description
  o Double quantity
  --> Trader owner
}

participant Trader identified by tradeId {
  o String tradeId
```

Everything looks good!
Any problems detected in your code would be reported here

Figure 2.1: Composer Playground development environment

```
/**
 * Sample business network definition.
 */
namespace org.t4.net

asset Commodity identified by tradingSymbol {
  o String tradingSymbol
  o String description
  o Double quantity
  --> Trader owner
}

participant Trader identified by tradeId {
  o String tradeId
  o String firstName
  o String lastName
}

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

Figure 2.2: CTO code for the business network archive

```
/**
 * transaction of a commodity from one trader to another
 * This check could be completed with ACL and is an exercise
 * @param {org.t4.net.Trade} trade - the trade to be processed
 * @transaction
 */
async function tradeCommodity(tx) {
  var ns="org.t4.net.";
    tx.commodity.owner=tx.newOwner;
    const commodityRegister = await getAssetRegistry(ns+"Commodity");
    await commodityRegister.update(tx.commodity);
}
```

Figure 2.3: JavaScript code for the business network archive

8. Complete the transaction of the same resource, ’0010’, to a participant that does not exist. Does the transaction complete?

```
/**
 * Sample access control list.
 */

rule SystemACL {
    description: "System ACL to permit all access"
    participant: "org.hyperledger.composer.system.Participant"
    operation: ALL
    resource: "org.hyperledger.composer.system.**"
    action: ALLOW
}

rule NetworkAdminUser {
    description: "Grant business network administrators full access to user resources"
    participant: "org.hyperledger.composer.system.NetworkAdmin"
    operation: ALL
    resource: "**"
    action: ALLOW
}

rule NetworkAdminSystem {
    description: "Grant business network administrators full access to system resources"
    participant: "org.hyperledger.composer.system.NetworkAdmin"
    operation: ALL
    resource: "org.hyperledger.composer.system.**"
    action: ALLOW
}
```

Figure 2.4: Access Control code for the business network archive

Code Completion

```
const ns='__________';
/**
 * transaction of a commodity from one trader to another
 * @_____ {___________.Trade} trade - the trade to be processed
 * @___________
 */
_____ _________ tradeCommodity(tx) {
  ___ traderRegistry = _____ getParticipantRegistry(ns+'.Trader');
  ___ traderExists = _____ traderRegistry.exists(tx.newOwner.getIdentifier());
  if(traderExists){
      tx.commodity._____ = tx._________;
    ___ assetRegistry = _____ getAssetRegistry(ns+'._________');
      _____ assetRegistry.______(tx._________);
  } else {
    throw new Error(" New Owner " + tx.newOwner.getIdentifier() + " does not exist");
  }
}
```

Figure 3.1: Incomplete code of transaction with Exist function.

3 Exists

25–30

Write the code using the method exist, prevent the above transaction from occuring and ensure that only existing participants can be the new owner. Complete the code in Fig. 3.1.

4 Add Method

35–40

In this section we are going to Add a new member of staff. To complete this, changes in the CTO, ACL and JS files are required and explained in the following subsections, respectively.

Code Completion

```
namespace org.t2.net
enum Role{
  o Manager
  o Trader
  o Consultant
  o Clerk
}

⋮

participant Trader identified by tradeId {
  o String tradeId
  o Role role
  o String firstName
  o String lastName
}

transaction AddStaff{
  o Trader newStaff
}
```

Figure 4.1: Incomplete CTO code for adding Status, adding Staff and roles to business network archive

4.1 CTO

Complete the CTO code in Fig. 4.1.

Code Completion

```
rule addNewStaff{
  description: "Manager adds new staff"
  participant: "org.t2.net.______"
  operation: ALL
  resource: "org.t2.net.Trader"
  action:ALLOW
}

rule addNewStafftx{
  description: "Manager to access tx"
  participant: "org.t2.net.______"
  operation: ALL
  resource: "org.t2.net.________"
  action:ALLOW
}
```

Figure 4.2: Access Control code for allowing create privileges to add staff business network archive

4.2 ACL

Complete the ACL code in Fig. 4.2.

Code Completion

```
/**
* transaction to add a new member of staff - manager access only
* @_____ {___________.AddStaff} tx - one param
* @___________
*/
_____ ________ addNewStaff(tx) {
  ___ me=getCurrentParticipant();
  if (me.____==='Manager')
  {
    ___ traderReg= _____ getParticipantRegistry(ns+'.______');
    _____ traderReg.___(tx.________);
  }
  else
  {
    throw new Error("You have insufficient privileges");
  }
}
```

Figure 4.3: Javascript code for Adding a new member of staff business network archive

4.3 JS

Complete the Javascript code in Fig. 4.3.

4.4 Deploy

Complete the following:

- Create three users, with clerk, consultant and manager status
- Issue new ID and wallets for these three users, call them clerk, consultant and manager
- Enter the system as a manager and test the addStaff function. Does it work? Look for console.log output.
- Enter the system as a clerk and test the addStaff function. Does it work? Look for the error.
- Test and evaluate your addStaff function, are there any restrictions and can they be overcome?

Code Completion

```
transaction removeStaff{
  -->Trader removedStaff
}
```

Figure 5.1: Incomplete CTO code for removing Staff from the business network.

5 Remove Staff

In this section we are going to Remove a member of staff. Only a Trader 25–30
with the role, `Manager`, can remove staff. To complete this, changes in the CTO, ACL and JS files are required and explained in the following subsections, respectively.

5.1 CTO

Complete the CTO code in Fig. 5.1.

Code Completion

```
rule removeStaff{
  description: "Manager remove staff"
  participant(p): "org.t2.net.______"
  operation: ALL
  resource(r): "org.t2.net.______"
  condition: (p.role==="_______")
  action: ALLOW
}

rule removeStafftx{
  description: "Manager remove staff tx"
  participant(p): "org.t2.net.______"
  operation: ALL
  resource(r): "org.t2.net.___________"
  condition: (p.role==="_______")
  action: ALLOW
}
```

Figure 5.2: Access Control code for allowing create privileges to add staff business network archive

5.2 ACL

Complete the ACL code in Fig. 5.2.

Code Completion

```
/**
* transaction to remove a new member of staff - manager access only
* @_____ {__________.removeStaff} tx - one param
* @___________
*/
_____ ________ removeStaff(tx) {
    ___ traderReg= _____ getParticipantRegistry(ns+'.______');
    _____ traderReg.remove(tx.____________);
}
```

Figure 5.3: Javascript code for Adding a new member of staff business network archive

5.3 JS

Complete the Javascript code in Fig. 5.3.

5.4 Deploy

Complete the following:

- Create three users, with clerk, consultant and manager status
- Issue new ID and wallets for these three users, call them clerk, consultant and manager
- Enter the system as a manager and test the removeStaff function. Does it work? Look for console.log output.
- Enter the system as a clerk and test the removeStaff function. Does it work? Look for the error.
- Test and evaluate your remove function, are there any restrictions and can they be overcome?

References