CSE 598 – Engineering Blockchain Applications – Project 1

Title: Hyperledger Fabric Private Blockchain and Smart Contracts

Student Name: Abhishek Patil

ASU ID: 1216126522

Abstract:

This document introduces the Hyperledger Fabric Network for understanding blockchain networks and experimenting with it. It then describes the important keywords related to the project in a systematic way. After that, it lists the phases of the project in goal description and mentions the important functions to code. Then, it describes how the project solution was handled and gives an example to run a function in Git Bash and check Docker for output. Moreover, it discusses the issues faced and the workarounds or solutions to it. It ends with a conclusion highlighting the importance of project learnings and blockchain in general.

Keywords:

Hyperledger Fabric, Peers, Network, Asset, Ledger, Chaincode, Git Bash, Docker, CouchDB, Selector Query, Index

Introduction:

In the recent times we have seen an increasing amount of use of blockchain technologies especially in the form of cryptocurrency, especially Bitcoin. The core concept of making important transactions that are safe, and immutable makes this technology very popular. It keeps a history of all transactions as a chain of blocks which can be used to check the validity of transactions taking place. Thus, it is important to understand its working and for that we have this project which uses Hyperledger Fabric software.

Terminology:

<u>Hyperledger Fabric</u>:

Hyperledger Fabric is an open-sourced project from the Linux foundation and is a de facto standard for enterprise blockchain deployments. It uses a plug-and-play model to accompany so many applications [1]. In this project, we use this software to create a Fabric Network of organizations to undergo supply chain management of a home appliance company.

<u>Fabric Network</u>:

A Fabric Network is a technical infrastructure where multiple organizations come together as a consortium to take advantage of the ledger services provided. It is a permissioned blockchain network where the permissions are determined by a set of policies that have been agreed to by the consortium [2].

<u>Consortium</u>:

A consortium is a collection of non-orderer organizations on blockchain network. These organizations form channels and own peers [3]. These peers run chaincode containers to perform read/write operations to a ledger [3].

<u>Asset</u>:

Assets can be any item that is tangible or intangible. These assets can be modified as needed by a valid transaction. They are represented as key-value pairs and their states are stored in a ledger for that specific network [4].

<u>Chaincode</u>:

Chaincode is software program that is run on peers of organizations to read/write changes of asset states to the ledger [5]. In this project, we are implementing ‘smart contracts’, and these are basically built by programming functionality for these chaincodes.

<u>Ledger</u>:

Ledger is a sequenced, tamper-resistant record of state transitions in the fabric system. Chaincodes are used to create/update/delete values for assets and such transactions are committed to this ledger [4].

Goal Description:

Phase I:

Implement some functions to read and update records. Additionally, handle functions that are not known.

1. Define *getProductByKey* function to show the record given a product ID and name.
2. Define *setQuantity* and *getQuantity* methods (getter and setter) for the quantity field of an asset.
3. Define *updateQuantity* function to change the quantity field of an asset.
4. Define *unknownTransaction* function to return an error for wrong function input.

Phase II:

Use indexing (CouchDB) to query by *productType* (product types) and *mfg_date* (manufacture dates).

1. Create mfg_dateIndex.json file to produce an index for faster querying.
2. Define *queryByProductType* function to test the *productType* column indexing using CouchDB selector query.
3. Define *querybyMfgdate* function to test the *mfg_date* column indexing using CouchDB selector query.
4. Define *querybyProduct_Type_Dualfunction* function with an advanced CouchDB selector query to show results of dual product types.

Description of proposed solution:

The first step of the solution was to setup Hyperledger fabric system. This required installation of pre-requisites like: Git with Bash, cURL, Docker and Docker Compose, and Node.js (with npm). Then I used the cURL command to download fabric-samples folder. After that, I downloaded the project codebase and put it in fabric_samples folder. A zip file will be created in the test_network folder for the downloaded codebase.

Before explaining the script commands to run the code, the steps to write the code itself is explained for each of the two phases below:

<u>Phase I</u>: (tasks 1,2,3,7)

For phase 1 of the project, I first went through the 3 files provided to us. Each file a class along with some fields or methods attached to them. These are: Productlist, ProductRecord and Productcategory. For the first task, I had to get product by key. It receives ctx (transaction context), product id and product name. The product id and name are converted to a product record key using makeKey function of the ProductRecord class. I used this key in the getPRecord method for productList object of the ctx. As this getPRecord is async function, I used it with await so that the next line which returns the record is only executed once the record is received. <u>To test this</u> first I used the 'createProductRecord' and then ran 'getProductByKey' on it to get the information.

The second task was to write setQuantity and getQuantity methods for ProductRecord. In setQuantity, I use the 'this' pointer (current object) and assign the quantity field with value received to the function. This is then returned. The getQuantity just returns the quantity field of the 'this' pointer.

The third task was to update the quantity value in a record. This receives ctx, product id, name, mfg_date, and quantity. Like the first task, it creates the product record key using the makeKey function of ProductRecord class. Also, it gets the product record for this product record key as described before.

Then, it uses this product record and calls the setQuantity method to update the quantity given as arguments. Next, it calls the updatePRecord method for productList object of the ctx and passes the updated product record to it. This is also called with the await keyword to register the value to buffer. <u>To test this and the second task</u> I used the 'updateQuantity' function.

The seventh task was to just return an error message for unknown functions. I just added the string 'Function name missing' as parameter to the Error object initialization.

<u>Phase II</u>: (tasks 4,5,6)

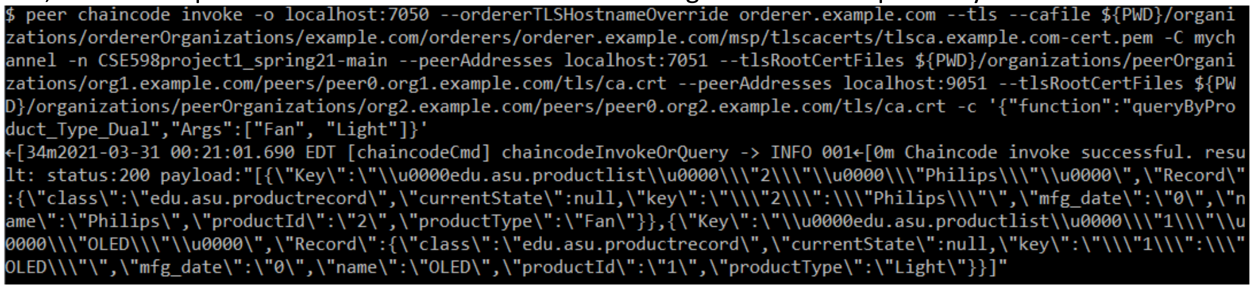
For task 4, I created a copy of the ProductTypeIndex.json and edited it to mfg_dateIndex.json file. The 'fields' value is 'mfg_date', 'ddoc' is 'mfg_dateIndexDoc', 'name' is 'Mfg_dateIndex', and 'type' is 'json'. In addition to that, I defined the 'queryByProductType' by creating a CouchDB selector query (as dictionary object) which I then used in JSON.stringify function. The query has 'selector' and 'use_index' arguments. The 'selector' argument gets a dictionary of key 'productType' and value 'productType'. Then 'use_index' gets an array of document name and index name which are 'productTypeIndexDoc' and 'ProductTypeIndex' respectively. This is then passed to 'queryWithQueryString' along with the ctx transaction which returns the list of respective data.

Then for task 5, it is like defining function in task 4. The query has 'selector' and 'use_index'. The 'selector' argument gets a dictionary of key 'mfg_date and value 'mfg_date. Then 'use_index' gets an array of document name and index name which are 'mfg_dateIndexDoc and 'Mfg_dateIndex' respectively. This is also JSON stringified and then passed to 'queryWithQueryString' just like task 4.

Finally, for task 6, it is again like tasks 4 and 5 where we define the function 'queryByProduct_Type_Dual' and create the selector query and pass it to 'queryWithQueryString' function. The difference is the selector query definition which additionally uses the '$or' to look for both product types in the records.

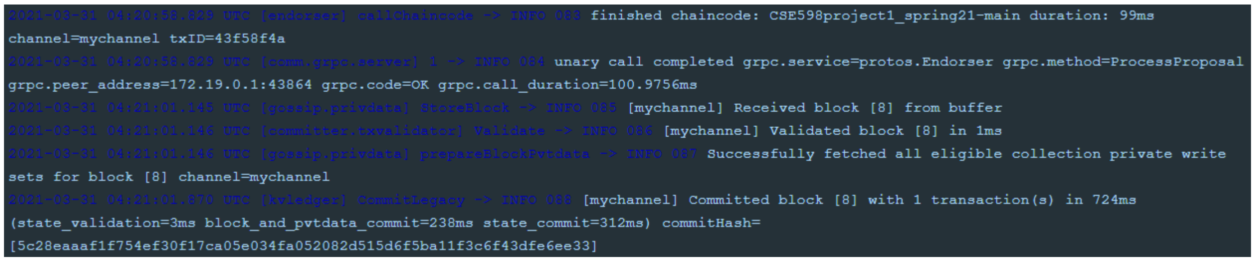
<u>To test the phase II</u>, I first created new records for multiple values of manufactured date and product type. Then, I ran each of the 3 functions defined in tasks 4, 5, and 6.

Due to space limitations, I have provided a picture of how one function ('queryByProduct_Type_Dual') is run, with its output in Git Bash and in Docker as seen in figure 1 and 2 respectively.



```
$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile ${PWD}/organi
zations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mych
annel -n CSE598project1_spring21-main --peerAddresses localhost:7051 --tlsRootCertFiles ${PWD}/organizations/peerOrgani
zations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles ${PW
D}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function":"queryByPro
duct_Type_Dual","Args":["Fan", "Light"]}'
←[34m2021-03-31 00:21:01.690 EDT [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001←[0m Chaincode invoke successful. resu
lt: status:200 payload:"[{\"Key\":\"\\u0000edu.asu.productlist\\u0000\\\"2\\\"\\u0000\\\"Philips\\\"\\u0000\",\"Record\"
:{\"class\":\"edu.asu.productrecord\",\"currentState\":null,\"key\":\"\\\"2\\\":\\\"Philips\\\"\",\"mfg_date\":\"0\",\"n
ame\":\"Philips\",\"productId\":\"2\",\"productType\":\"Fan\"}},{\"Key\":\"\\u0000edu.asu.productlist\\u0000\\\"1\\\"\\u
0000\\\"OLED\\\"\\u0000\",\"Record\":{\"class\":\"edu.asu.productrecord\",\"currentState\":null,\"key\":\"\\\"1\\\":\\\"
OLED\\\"\",\"mfg_date\":\"0\",\"name\":\"OLED\",\"productId\":\"1\",\"productType\":\"Light\"}}]"
```

Figure 1. Git Bash output for querying by dual product types 'Light' and 'Fan'



```
2021-03-31 04:20:58.829 UTC [endorser] callChaincode -> INFO 083 finished chaincode: CSE598project1_spring21-main duration: 99ms
channel=mychannel txID=43f58f4a
2021-03-31 04:20:58.829 UTC [comm.grpc.server] 1 -> INFO 084 unary call completed grpc.service=protos.Endorser grpc.method=ProcessProposal
grpc.peer_address=172.19.0.1:43864 grpc.code=OK grpc.call_duration=100.9756ms
2021-03-31 04:21:01.145 UTC [gossip.privdata] StoreBlock -> INFO 085 [mychannel] Received block [8] from buffer
2021-03-31 04:21:01.146 UTC [committer.txvalidator] Validate -> INFO 086 [mychannel] Validated block [8] in 1ms
2021-03-31 04:21:01.146 UTC [gossip.privdata] prepareBlockPvtdata -> INFO 087 Successfully fetched all eligible collection private write
sets for block [8] channel=mychannel
2021-03-31 04:21:01.870 UTC [kvledger] CommitLegacy -> INFO 088 [mychannel] Committed block [8] with 1 transaction(s) in 724ms
(state_validation=3ms block_and_pvtdata_commit=238ms state_commit=312ms) commitHash=
[5c28eaaaf1f754ef30f17ca05e034fa052082d515d6f5ba11f3c6f43dfe6ee33]
```

Figure 2. Docker output for querying by dual product types ‘Light’ and ‘Fan’

Issues Faced and Methods used to resolve them:

The main issue faced was in understanding the commands needed to run and execute the chaincode in a very limited timeframe. It can get very confusing. Thus, I kept trying different VMs to see if it works in different environments. I tried multiple methods like using a Linux VM and a Windows VM. For the Linux VM it gave some authorization errors when installing docker. For the Windows VM (latest version) it gave Docker installation failures. This might have to do with some incompatibility issues with latest Windows VM release. So, I decided not to use VM at all and re-started all downloads and installed with proper versions specified in the course document. This time everything worked properly, and I was able to execute functions for testing.

One other issue was that some variables turned out to be returning empty objects. I was able to resolve this issue by using the ‘await’ keyword for the ‘async’ functions.

Related Works (if any):

[1]. “What is Hyperledger Fabric?,” IBM. [Online]. Available: https://www.ibm.com/topics/hyperledger. [Accessed: 25-Mar-2021].

[2]. “Hyperledger Fabric Network” hyperledger. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.2/network/network.html. [Accessed: 25-Mar-2021].

[3]. “Glossary,” hyperledger. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.2/glossary.html. [Accessed: 25-Mar-2021].

[4]. “Glossary,” hyperledger. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.2/glossary.html. [Accessed: 25-Mar-2021].

[5]. “Glossary,” hyperledger. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.2/glossary.html. [Accessed: 27-Mar-2021].

Conclusions:

This project has been a huge learning experience in terms of understanding how the Hyperledger implementation of blockchain works. Obviously, it gives me an upper hand in knowing the inner working of peers and organizations to work together and come up with best blockchain. This technology is recent and having it as another tool in my resume is a great achievement.

Bibliography:
- no separate articles or textbooks used

DIRECTIONS I TOOK TO RUN:

NOTE: I ran the code line-by-line. It is shown as a different format/color.

1. First place the codebase file in the test-network folder under fabric-samples. File name given to us "CSE598project1_spring21-main"
2. cd to the test-network folder in Git Bash
3. Run
   ```
   npm install
   ```
4. Then use the network.sh file to setup network:
   ```
   ./network.sh down

   ./network.sh up -s couchdb

   ./network.sh createChannel
   ```
5. Setup organization 1 using exports:
   ```
   SEQ_NO=1

   export PATH=${PWD}/../bin:${PWD}:$PATH

   export FABRIC_CFG_PATH=$PWD/../config/

   export CORE_PEER_TLS_ENABLED=true

   export CORE_PEER_LOCALMSPID="Org1MSP"

   export
   CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.c
   om/peers/peer0.org1.example.com/tls/ca.crt

   export
   CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.co
   m/users/Admin@org1.example.com/msp

   export CORE_PEER_ADDRESS=localhost:7051
   ```
6. Package and install project for this organization:
   ```
   peer lifecycle chaincode package CSE598project1_spring21-main.tar.gz --path
   CSE598project1_spring21-main --lang node --label CSE598project1_spring21-main

   peer lifecycle chaincode install CSE598project1_spring21-main.tar.gz

   export PATH=$PWD/../bin:$PATH

   export FABRIC_CFG_PATH=$PWD/../config/
   ```
7. Run queryinstalled and then approve and check commit:
   ```
   peer lifecycle chaincode queryinstalled
   ```
   The output of the above gives a Package ID. Export it as follows:
   ```
   export CC_PACKAGE_ID= <insert Package ID here>
   ```

```
peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name CSE598project1_spring21-main --version 1.0 --package-id $CC_PACKAGE_ID --sequence $SEQ_NO --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name CSE598project1_spring21-main --version 1.0 --sequence $SEQ_NO --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --output json
```

8. Setup organization 2 using exports:

```
export CORE_PEER_LOCALMSPID="Org2MSP"

export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt

export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt

export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp

export CORE_PEER_ADDRESS=localhost:9051
```

9. Install project for this organization:

```
peer lifecycle chaincode install CSE598project1_spring21-main.tar.gz
```

10. Run queryinstalled and then approve and check commit:

```
peer lifecycle chaincode queryinstalled
```

The output of the above gives a Package ID. Export it as follows:

```
export CC_PACKAGE_ID= <insert Package ID here>

peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name CSE598project1_spring21-main --version 1.0 --package-id $CC_PACKAGE_ID --sequence $SEQ_NO --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name CSE598project1_spring21-main --version 1.0 --sequence $SEQ_NO --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --output json
```

11. Run commit and then querycommitted:

```
peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name CSE598project1_spring21-main --version 1.0 --sequence $SEQ_NO --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --peerAddresses localhost:7051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt

peer lifecycle chaincode querycommitted --channelID mychannel --name CSE598project1_spring21-main --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

12. Before testing, run ‘createProductRecord’ function to insert some records:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n CSE598project1_spring21-main --peerAddresses localhost:7051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function":"createProductRecord","Args":["1", "OLXD", "06-17-2018", "Light"]}'

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n CSE598project1_spring21-main --peerAddresses localhost:7051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function":"createProductRecord","Args":["2", "Philips", "03-24-2019", "Fan"]}'

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n CSE598project1_spring21-main --peerAddresses localhost:7051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function":"createProductRecord","Args":["3", "GMA", "11-24-2017", "Light"]}'
```

13. Task 1:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/
msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n CSE598project1_spring21-main
--peerAddresses localhost:7051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.co
m/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.co
m/tls/ca.crt -c '{"function":"getProductByKey","Args":["2", "Philips"]}'
```

14. Tasks 2 and 3 (first ‘updateQuantity’ then ‘getProductByKey’ to check quantity update):

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/
msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n CSE598project1_spring21-main
--peerAddresses localhost:7051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.co
m/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.co
m/tls/ca.crt -c '{"function":"updateQuantity","Args":["1", "OLXD", "06-17-2018", "4"]}'

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/
msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n CSE598project1_spring21-main
--peerAddresses localhost:7051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.co
m/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.co
m/tls/ca.crt -c '{"function":"getProductByKey","Args":["1", "OLXD"]}'
```

15. Task 4:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/
msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n CSE598project1_spring21-main
--peerAddresses localhost:7051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.co
m/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.co
m/tls/ca.crt -c '{"function":"queryByProductType","Args":["Light"]}'
```

16. Task 5:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/
msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n CSE598project1_spring21-main
--peerAddresses localhost:7051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.co
m/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles
```

```
${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.co
m/tls/ca.crt -c '{"function":"queryByMfgdate","Args":["03-24-2019"]}'
```

17. Task 6:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/
msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n CSE598project1_spring21-main
--peerAddresses localhost:7051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.co
m/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.co
m/tls/ca.crt -c '{"function":"queryByProduct_Type_Dual","Args":["Fan", "Light"]}'
```

18. Task 7:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/
msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n CSE598project1_spring21-main
--peerAddresses localhost:7051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.co
m/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles
${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.co
m/tls/ca.crt -c '{"function":"randomFunctionName","Args":["1"]}'
```