Execution

Use Test Networks

OpenIP

Putty로 접속 203.254.143.178 port :7777
Use our key.
-현재 mycc
Couch db: http://203.254.143.178:5984/_utils/
203.254.143.178:8080

4nodes: Fullgear

Node1: http://203.254.143.183:60007
Node2: http://203.254.143.183:60005
Node3: http://203.254.143.183:60010
Node3: http://203.254.143.183:60003

Application API

Installation
cd fabric/fabric-sample/fabcar/apiserver
api

queryAllSubjects

GET: /api/queryallsubjects
Response : JSON(list of all subjects)

ex) http://203.254.143.167:8080/api/queryallsubjects


```
{
   "response": [
       {
           "Key": "Subject0",
           "Record": {
               "classification": "동의",
```

```
                "consent_ver": "2.1",
                "dosage_dt": "",
                "investigator": "김민걸",
                "iv_sign_dt": "201001311107",
                "site": "site1",
                "study": "Jpx3415-0010-t",
                "subject": "김명화",
                "subject_sign": "201001311100"
            }
        },
        {
            "Key": "Subject1",
            "Record": {
                "classification": "투약",
                "consent_ver": "2.1",
                "dosage_dt": "201002011512",
                "investigator": "김민걸",
                "iv_sign_dt": "201002011513",
                "site": "site1",
                "study": "Jpx3415-0010-t",
                "subject": "김명화",
                "subject_sign": "201002011513"
            }
        },
        ...
 ]
}
```


queryConsent

GET: /api/queryconsent/:consentId

Response : JSON(data of a consent)

ex) http://203.254.143.167:8080/api/queryconsent/Consent0


```
{
   "response": {
        "consent_ver": "2.1",
        "contents": "alskdjfoiquwelirj",
        "site": "site1",
        "study": "Jpx3415-0010-t"
   }
}
```


querySubject

GET: /api/querysubject/:subjectId

Response : JSON(data of a subject)

ex) http://203.254.143.167:8080/api/querysubject/Subject0


```
{
```

```
    "response": {
        "classification": "동의",
        "consent_ver": "2.1",
        "dosage_dt": "",
        "investigator": "김민걸",
        "iv_sign_dt": "201001311107",
        "site": "site1",
        "study": "Jpx3415-0010-t",
        "subject": "김명화",
        "subject_sign": "201001311100"
    }
}
```


createSubject

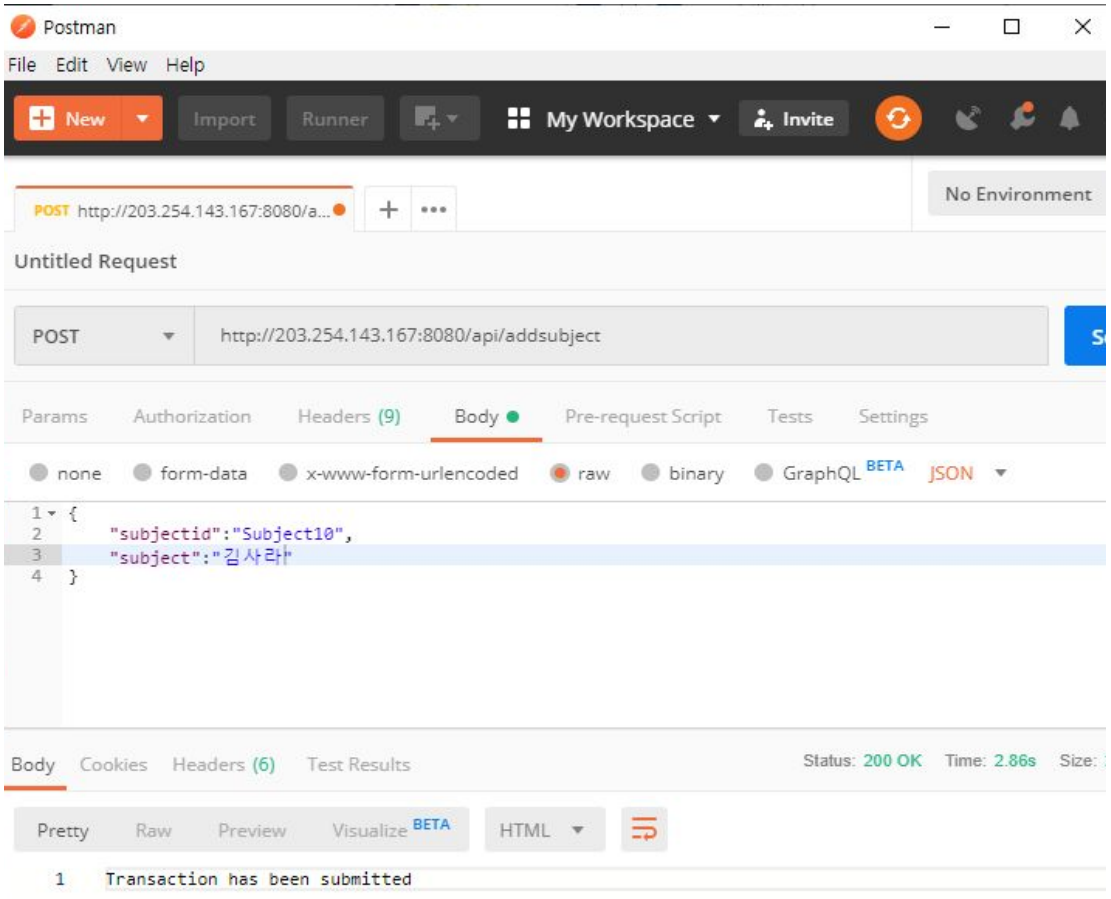
POST: /api/createsubject

`Input: {"subjectid": "{subject id}", "subject" : "{name}"}`

Response : 'transaction has been submitted'

ex) http://203.254.143.167:8080/api/createsubject

```
Input:  {
        "subjectid":"Subject10",
        "subject":"김사라"
}
```



createConsent

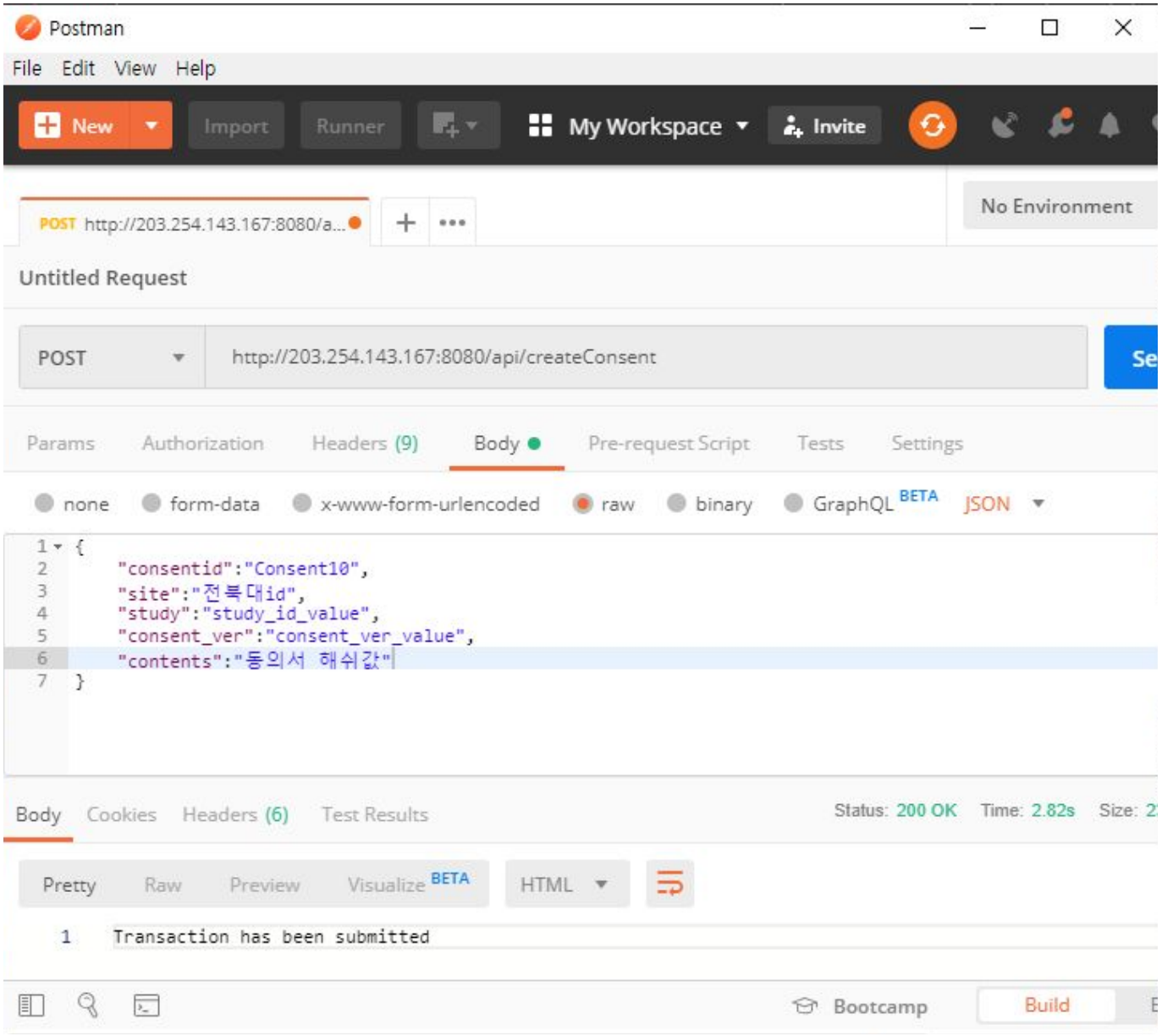
POST: /api/createconsent

```
Input: {“consentid”: “{consent id}”,
         “site”: “{site id}”,
         “study”: “{study id}”,
         “consent_ver” : “{version}”,
         “contents”: “{hashed(consent)}”}
```

Response : ‘transaction has been submitted’

ex) http://203.254.143.167:8080/api/createconsent

```
Input:  {
      "consentid":"Consent10",
      "site":"전북대id",
      "study":"study_id_value",
      "consent_ver":"consent_ver_value",
      "contents":"동의서 해쉬값"
}
{    "consentid":"Consent001",     "site":"JBCP001",
"study":"study_id_value",     "consent_ver":"consent_ver_value",
"contents":"0493571b39a82aceb05570bd0c1f242c6f49a56242cf2c9e09d674c40ab4
9501"}
```



agreeConsent

PUT: /api/agreeconsent/:subjectId

```
Input: {"site": "{site id}",
        "study": "{study id}",
        "consent_ver" : "{consent_version}",
        "subject_sign" : "{subject_sign}",
        "investigator" : "{investigator}",
        "iv_sign_dt" : "{iv_sign_dt}",
        "classification": "동의"}
```

Response : "Transaction has been submitted"

ex) http://203.254.143.167:8080/api/agreeconsent/Subject10

Input

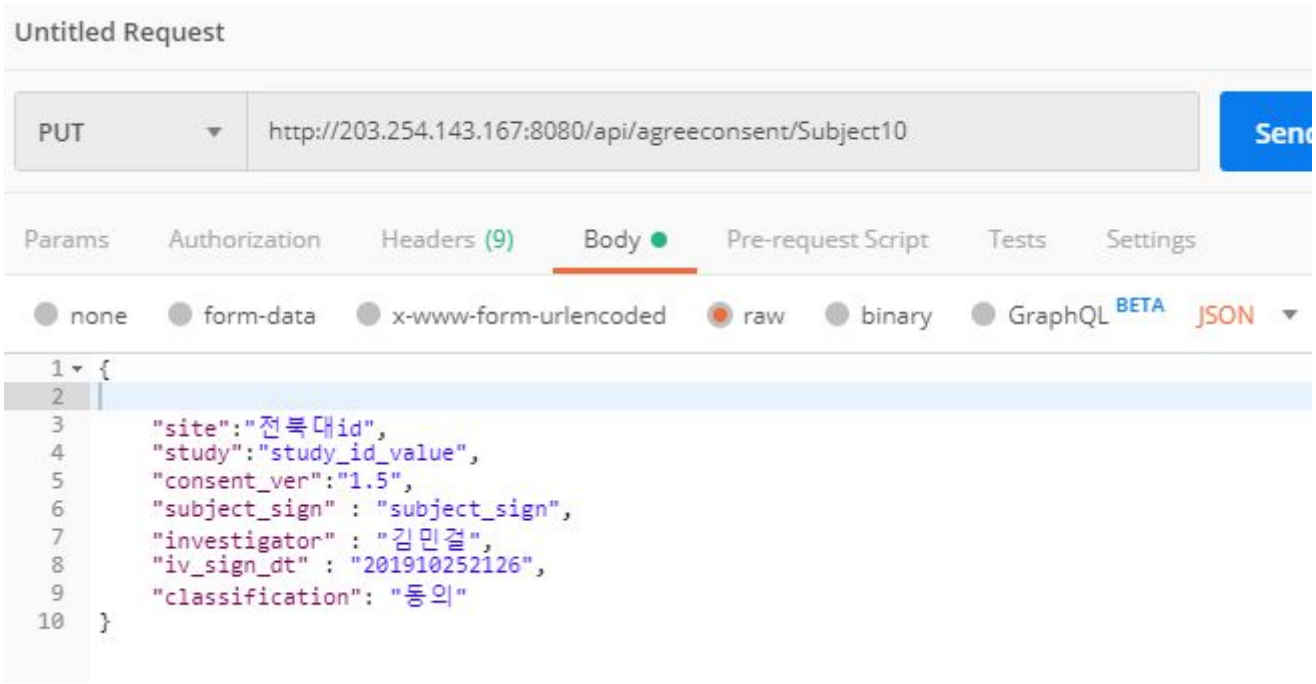
```
{
	"site":"전북대id",
	"study":"study_id_value",
	"consent_ver":"1.5",
	"subject_sign" : "subject_sign",
	"investigator" : "김민걸",
	"iv_sign_dt" : "2018/05/10/08:30",
	"classification": "동의"
}
```



```
{
  "_id": "Subject10",
  "_rev": "1-e8dc6b7ee98c0d3bec0b2eca
  "classification": "",
  "consent_ver": "",
  "dosage_dt": "",
  "investigator": "",
  "iv_sign_dt": "",
  "site": "",
  "study": "",
  "subject": "김사라",
  "subject_sign": "",
  "~version": "\u0000CgMBBQA="
}
```



```
{
  "_id": "Subject10",
  "_rev": "4-518c1d520aca0667ec05c6212547fe3d
  "classification": "동의",
  "consent_ver": "1.5",
  "dosage_dt": "",
  "investigator": "김민걸",
  "iv_sign_dt": "201910252126",
  "site": "전북대id",
  "study": "study_id_value",
  "subject": "김사라",
  "subject_sign": "subject_sign",
  "~version": "\u0000CgMBCgA="
}
```

withdrawConsent

PUT: /api/withdrawconsent/:subjectId

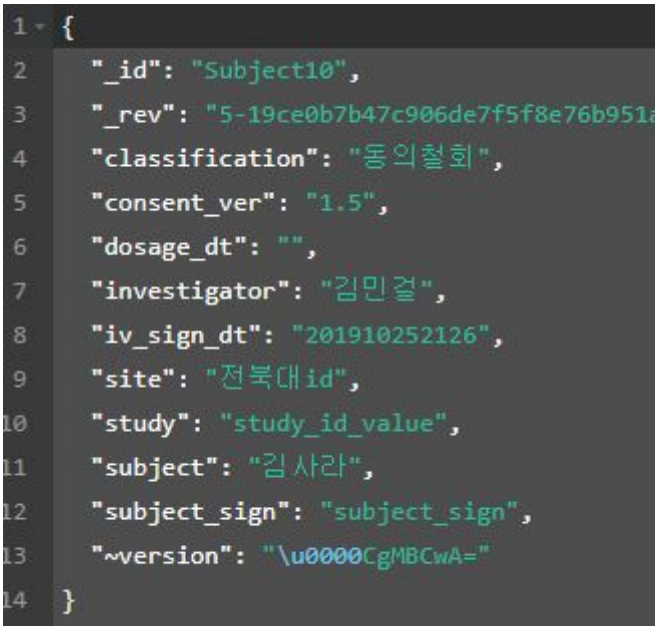
Input: {"site": "{site id}",
"study": "{study id}",
"consent_ver" : "{consent_version}",

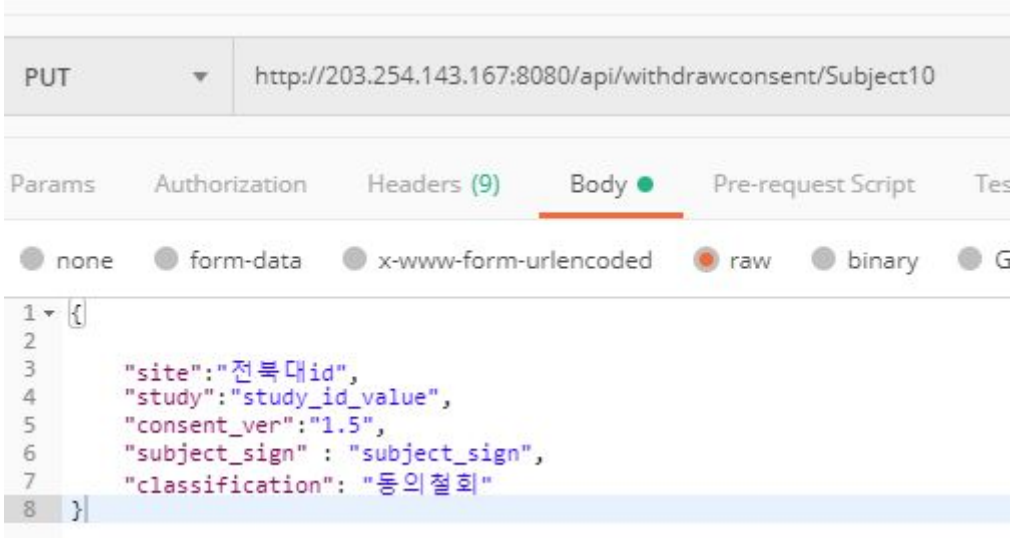
```
"subject_sign" : "{subject_sign}",
"classification": "동의철회"}
```

Response : “Transaction has been submitted”

ex) http://203.254.143.167:8080/api/withdrawconsent/Subject10

```
Input
{
        "site":"전북대id",
        "study":"study_id_value",
        "consent_ver":"1.5",
        "subject_sign" : "subject_sign",
        "classification": "동의철회"
}
```





Chain Code

Installation Chaincode

cd fabric-samples/basic-network

./teardown.sh
./generate.sh & ./init.sh (한번만)
Docker-compose.sh에서 ca 에 peer key 값 수정
./start.sh

docker-compose up -d cli

```
ubuntu@as:~/fabric/fabric-samples/chaincode/ctcc$ ll
total 24
drwxrwxr-x  2 ubuntu ubuntu  4096 Nov 24 15:03 ./
drwxrwxr-x 10 ubuntu ubuntu  4096 Nov 24 13:51 ../
-rw-rw-r--  1 ubuntu ubuntu 15664 Nov 24 13:48 test.go
```

docker exec cli peer chaincode install -n mycc -p github.com/ctcc -v 0.1
(go 파일이 있는 폴더이름까지만 입력)

docker exec cli peer chaincode instantiate -o orderer.example.com:7050 -C mychannel -n mycc github.com/ctcc -v 0.1 -c '{"Args": []}' -P "OR('Org1MSP.member')"

docker exec cli peer chaincode invoke -C mychannel -n mycc -c '{"Args":["initLedger"]}'

Upgrade chaincode

docker exec cli peer chaincode install -n mycc -p github.com/test -v 1.0

docker exec cli peer chaincode upgrade -n mycc -v 1.0   -p github.com/test -c '{"Args": []}' -C mychannel -o orderer.example.com:7050 -P "OR('Org1MSP.member')"

----------basic network 안됨.----------
Run on firstnetowrk
Go to fabric-samples/fabcar/
./startFabric.sh
Or firstnetwork /byfn.sh up

--Peer0.org1에 체인코드 삽입
+ echo 'Installing smart contract on peer0.org1.example.com'
Installing smart contract on peer0.org1.example.com

+ docker exec -e CORE_PEER_LOCALMSPID=Org1MSP -e CORE_PEER_ADDRESS=peer0.org1.example.com:7051 -e CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt cli peer chaincode install -n mycc -v 1.0 -p github.com/chaincode/ctcc -l golang
2019-11-25 01:20:12.288 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-11-25 01:20:12.288 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-11-25 01:20:12.492 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >

+ echo 'Installing smart contract on peer1.org1.example.com'
Installing smart contract on peer1.org1.example.com
+ docker exec -e CORE_PEER_LOCALMSPID=Org1MSP -e CORE_PEER_ADDRESS=peer1.org1.example.com:8051 -e CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt cli peer chaincode install -n mycc -v 1.0 -p github.com/chaincode/ctcc -l golang
2019-11-25 01:20:12.955 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc

2019-11-25 01:20:12.955 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-11-25 01:20:13.166 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >

+ echo 'Installing smart contract on peer0.org2.example.com'
Installing smart contract on peer0.org2.example.com

+ docker exec -e CORE_PEER_LOCALMSPID=Org2MSP -e CORE_PEER_ADDRESS=peer0.org2.example.com:9051 -e CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp -e CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt cli peer chaincode install -n mycc -v 1.0 -p github.com/chaincode/ctcc -l golang
2019-11-25 01:20:13.666 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-11-25 01:20:13.666 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-11-25 01:20:13.855 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >

+ echo 'Installing smart contract on peer1.org2.example.com'
Installing smart contract on peer1.org2.example.com
+ docker exec -e CORE_PEER_LOCALMSPID=Org2MSP -e CORE_PEER_ADDRESS=peer1.org2.example.com:10051 -e CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp -e CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt cli peer chaincode install -n mycc -v 1.0 -p github.com/chaincode/ctcc -l golang
2019-11-25 01:20:14.344 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-11-25 01:20:14.345 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-11-25 01:20:14.564 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >

--인스턴스화

docker exec -e CORE_PEER_LOCALMSPID=Org1MSP -e CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp cli peer chaincode instantiate -o orderer.example.com:7050 -C mychannel -n mycc -l golang -v 1.0 -c '{"Args":[]}' -P 'AND(\"Org1MSP.member\",\"Org2MSP.member\")' --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt

DB 초기화

echo 'Submitting initLedger transaction to smart contract on mychannel'

Submitting initLedger transaction to smart contract on mychannel

+ echo 'The transaction is sent to all of the peers so that chaincode is built before receiving the following requests'

The transaction is sent to all of the peers so that chaincode is built before receiving the following requests

+ docker exec -e CORE_PEER_LOCALMSPID=Org1MSP -e CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp cli peer chaincode invoke -o orderer.example.com:7050 -C mychannel -n mycc -c '{"function":"initLedger","Args":[]}' --waitForEvent --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --peerAddresses peer0.org1.example.com:7051 --peerAddresses peer1.org1.example.com:8051 --peerAddresses peer0.org2.example.com:9051 --peerAddresses peer1.org2.example.com:10051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt --tlsRootCertFiles

/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt

Methods

Log 확인

Docker ps
Docker logs {chaincode names}

queryAllSubjects

docker exec cli peer chaincode invoke -C mychannel -n mycc -c '{"Args":["queryAllSubjects"]}'

querySubject

docker exec cli peer chaincode invoke -C mychannel -n mycc -c '{"Args":["querySubject","Subject3"]}'

queryConsent

docker exec cli peer chaincode invoke -C mychannel -n mycc -c '{"Args":["queryConsent","Consent0"]}'

queryAllSubjectsWithPagination

```
docker exec cli peer chaincode invoke -C mychannel -n mycc -c
'{"Args":["queryAllSubjectsWithPagination","3","Subject0"]}'
```

createSubject

```
docker exec cli peer chaincode invoke -C mychannel -n mycc -c
'{"Args":["createSubject","Subject3","ls
"]}'

docker exec cli peer chaincode invoke -C mychannel -n mycc -c
'{"Args":["createSubject","Subject4","윤지박","site2", "S001-tesert",
"1.1", "", "201003050905", "이승환","201003050910", "동의"]}'
```

doseSubject

```
docker exec cli peer chaincode invoke -C mychannel -n mycc -c
'{"Args":["doseSubject","Subject2","site2", "S001-tesert",
"201005010900", "박치복사인", "이승환","201005010902", "투약"]}'
```

changeSubject

createConsent

```
docker exec cli peer chaincode invoke -C mychannel -n mycc -c
'{"Args":["createConsent","Consent2","site2","newStudy","0.1","해쉬값"]}'
```

agreeConsent

```
docker exec cli peer chaincode invoke -C mychannel -n fabric -c
'{"Args":["agreeConsent","Subject10","site0","Study0","0.1","싸인시간","담
당연구자id", "2018/05/10/08:30","동의"]}'
```

withdrawConsent

```
docker exec cli peer chaincode invoke -C mychannel -n mycc -c
'{"Args":["withdrawConsent","Subject3","site0","Study0","0.1","싸인시간"
,"동의철회"]}'
```

```
{
  "_id": "Subject3",
  "_rev": "1-215d5fb3c0b680d2955b763ec83df315",
  "classification": "",
  "consent_ver": "",
  "dosage_dt": "",
  "investigator": "",
  "iv_sign_dt": "",
  "site": "",
  "study": "",
  "subject": "임수정",
  "subject_sign_dt": "",
  "~version": "\u0000CgMBBQA="
}
```

```
{
  "_id": "Subject3",
  "_rev": "5-a3f6cd2169f0b207d04db9701f74ca89",
  "classification": "동의철회",
  "consent_ver": "0.1",
  "dosage_dt": "",
  "investigator": "담당연구자id",
  "iv_sign_dt": "2018/05/10/08:30",
  "site": "site0",
  "study": "Study0",
  "subject": "임수정",
  "subject_sign": "싸인시간",
  "~version": "\u0000CgMBEgA="
}
```


Structure

```
type Consent struct {
        //ObjectType string `json:"docType"`
        //Classcification string `json:"classcification"`

        Site          string `json:"site"`
        Study         string `json:"study"`
        Version       string `json:"version"`
        Contents      string `json:"contents"`
}
```

The attribute `docType` is a pattern used in the chaincode to differentiate different data types that may need to be queried separately. When using CouchDB, it recommended to include this `docType` attribute to distinguish each type of document in the chaincode namespace. (Each chaincode is represented as its own CouchDB database, that is, each chaincode has its own namespace for keys.)

```
type Subject struct{
        Subject       string `json:"subject"`
```

```
        Site          string `json:"site"`
        Study         string `json:"study"`

              ConsentVer      string `json:"consent_ver"`
        DosageDT          string `json:"dosage_dt"`

        SubjectSignDT     string `json:"subject_sign_dt"`
        Investigator            string `json:"investigator"`
        IvSignDT          string `json:"iv_sign_dt"`
        Classification string `json:"classification"`

}
```

Chaincode General

Chaincode Key APIs

An important interface that you can use when writing your chaincode is defined by Hyperledger Fabric —

- ChaincodeStub
- ChaincodeStubInterface

The ChaincodeStub provides functions that allow you to interact with the underlying ledger to query, update, and delete assets.

state

The ledger's current state data represents the latest values for all keys ever included in the chain transaction log. Since current state represents all latest key values known to the channel, it is sometimes referred to as World State.
Chaincode invocations execute transactions against the current state data. To make these chaincode interactions extremely efficient, the latest values of all keys are stored in a state database.

Smart contracts primarily put, get and delete states in the world state, and can also query the state change history. Chaincode “shim” APIs implements ChaincodeStubInterface which contain methods for access and modify the ledger, and to make invocations between chaincodes. Main methods are:
* GetState(key string) ([]byte, error) performs a query to retrieve information about the current state of a object
* PutState(key string, value []byte) error creates a new object or modifies an existing one in the ledger world state
* DelState(key string) error removes an object from the current state of the ledger, but not its history
* GetStateByPartialCompositeKey(objectType string, keys []string) (StateQueryIteratorInterface, error)queries the state in the ledger based on given partial composite key
* GetHistoryForKey(key string) (HistoryQueryIteratorInterface, error)returns a history of key values across time.

https://medium.com/coinmonks/hyperledger-fabric-smart-contract-data-model-protobuf-to-chaincode-state-mapping-191cdcfa0b78