INTEROPERABLE TRANSACTIONS

BY:

SAI & KAMAL

INDEX

The Incredible Technology: BLOCKCHAIN

1. **Ledger**
A blockchain is an open, distributed ledger.

2. **Network**
The data is distributed across all the devices in the network

3. **Decentralized**
The whole blockchain is decentralized.

4. **Secure**
The data inside the blockchain is very secure

5. **Logging**
We get the whole history of activities.

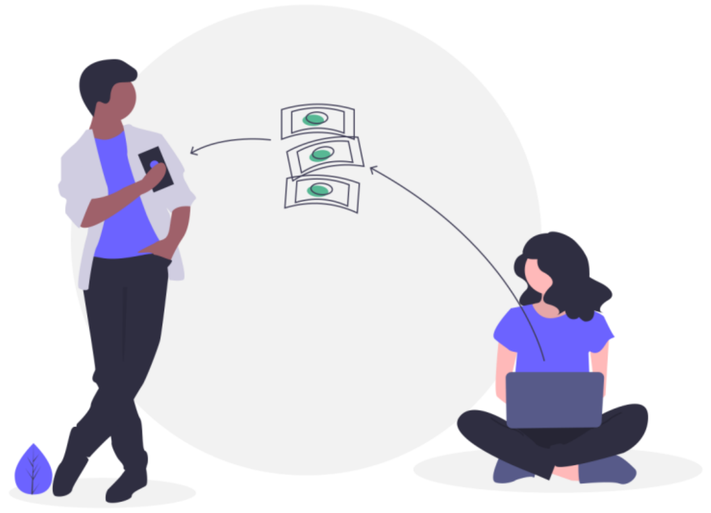
INTEROPERABILITY AND INTEROPERABLE PAYMENTS

INTEROPERABILITY

Interoperability is seen as a means for people worldwide to make electronic payments in a convenient, affordable, fast, seamless and secure way through a transaction account.

INTEROPERABLE PAYMENTS

When payment systems are interoperable, they allow two or more proprietary platforms or even different products to interact seamlessly. Interoperability can promote competition, reduce fixed costs and enable economies of scale that help ensure the financial viability of the service and make payment services more convenient.



BUSINESS CASE REQUIREMENT

- *To create a platform in which peers of different banks of different organizations can transfer asserts between each other without any external intermediate involvement.*

- *The transfer of asserts between banks is performed by using cryptocurrency as a medium as exchange.*

- *The whole system should be flexible, secure and trustworthy.*

TECH STACK

Hyperledger-Fabric
The blockchain technology for creating the private distributed ledger

Node SDK
Used to communicate with the Frontend

Angular
Frontend Framework

ERC20Token
For Tokenization of Asserts

HYPERLEDGER FABRIC.

Permissioned membership

Hyperledger Fabric is a framework for permissioned networks, where all participants have known identities. When considering a permissioned network, you should think about whether your blockchain use case needs to comply with data protection regulations.

Protection of digital keys

HSM (Hardware Security Module) support is vital for safeguarding and managing digital keys for strong authentication. Hyperledger Fabric provides modified and unmodified PKCS11 for key generation, which supports cases like identity management that need more protection

Data on a need-to-know basis

Hyperledger Fabric is a framework for permissioned networks, Businesses, due to competitiveness, protection laws, and regulation on confidentiality of personal data dictate the need for privacy of certain data elements, which can be achieved through data partitioning on the blockchain. Channels, supported in Hyperledger Fabric, allow for data to go to only the parties that need to know all participants have known identities. When considering a permissioned network, you should think about whether your blockchain use case needs to comply with data protection regulations.

RIPPLE XRP

- *Ripple XRP is an open source, peer to peer, payment network – a simple way for anyone in the world to send money at practically no cost*
- *The XRP Ledger is an online system for payments, powered by a community without a central leader. Anyone can connect their computer to the peer-to-peer network that manages the ledger. The XRP Ledger is the home of XRP, a digital asset designed to bridge the world's many currencies. The XRP Ledger is one part of the developing Internet of Value: a world in which money moves the way information does today.*

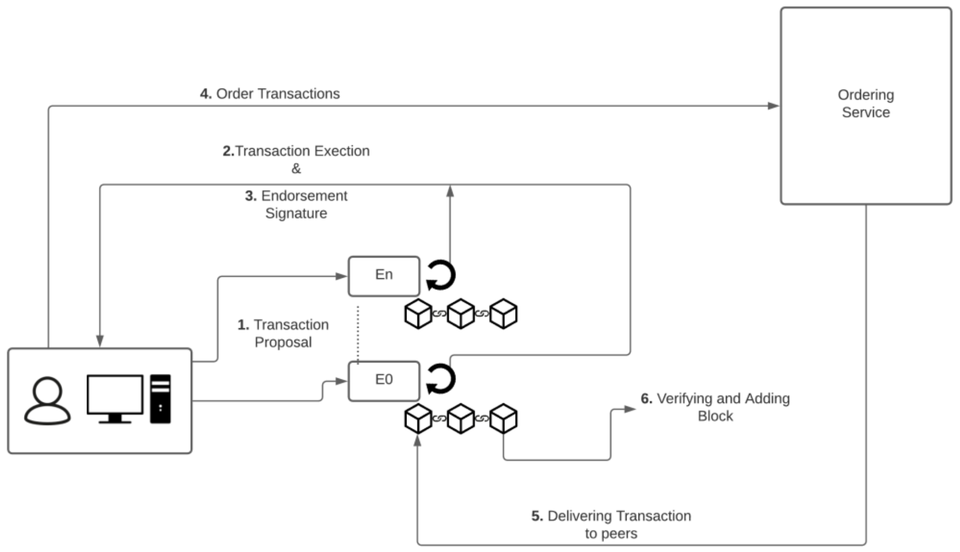
- "ripple-lib" API's are used to manage the transactions
- It uses XRP as the currency
- Key attributes of RIPPLE XRP:
    - The ripple API Server
    - The Ripple API ServerNet credentials
        - Address
        - Secret
- The address is the unique address of the sender or receiver.
- The secret is basically the password of the address.
- The address acts as the sender address when we send XRP from that address.
- The same address acts as the receiver when XRP is received.
- The user needs to provide the secret when they act as the sender in the transaction.
- The address can be shared but the secret should never be shared with others.
- When the XRP is transferred the new owner of the asset is updated in the ledger.

TRANSACTION FLOW

- **Step1: Propose Transaction**
  Client application will propose a transaction by sending the transaction details to all the endorsing peers available on the network. All the endorsing peers on the network will receive .

- ***Step2: Executing Proposed Transaction***
  All the endorsing peers in the network will execute the proposed transaction. Each execution will capture the set of Read and Write data, called RW Sets. Transactions can be signed and encrypted.

- **Step3: Proposal Response**
  The Read Write sets are asynchronously returned to the application. The Read Write sets are signed by each endorser and this will be later checked in the consensus process.

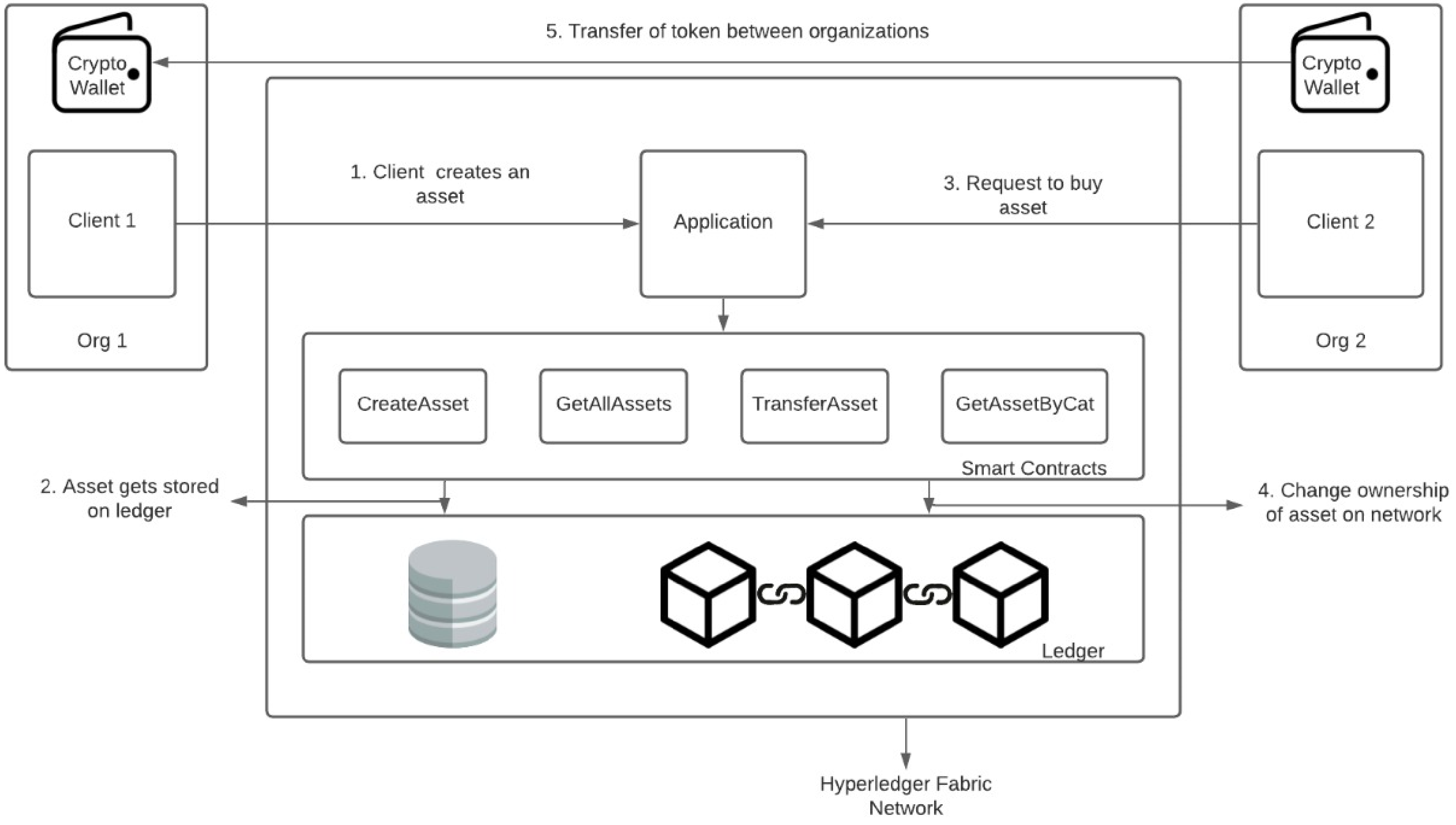


- **Step4: Order Transaction**
  Application submits responses as a transaction to be ordered. Ordering happens across the fabric in parallel with transactions submitted by other applications

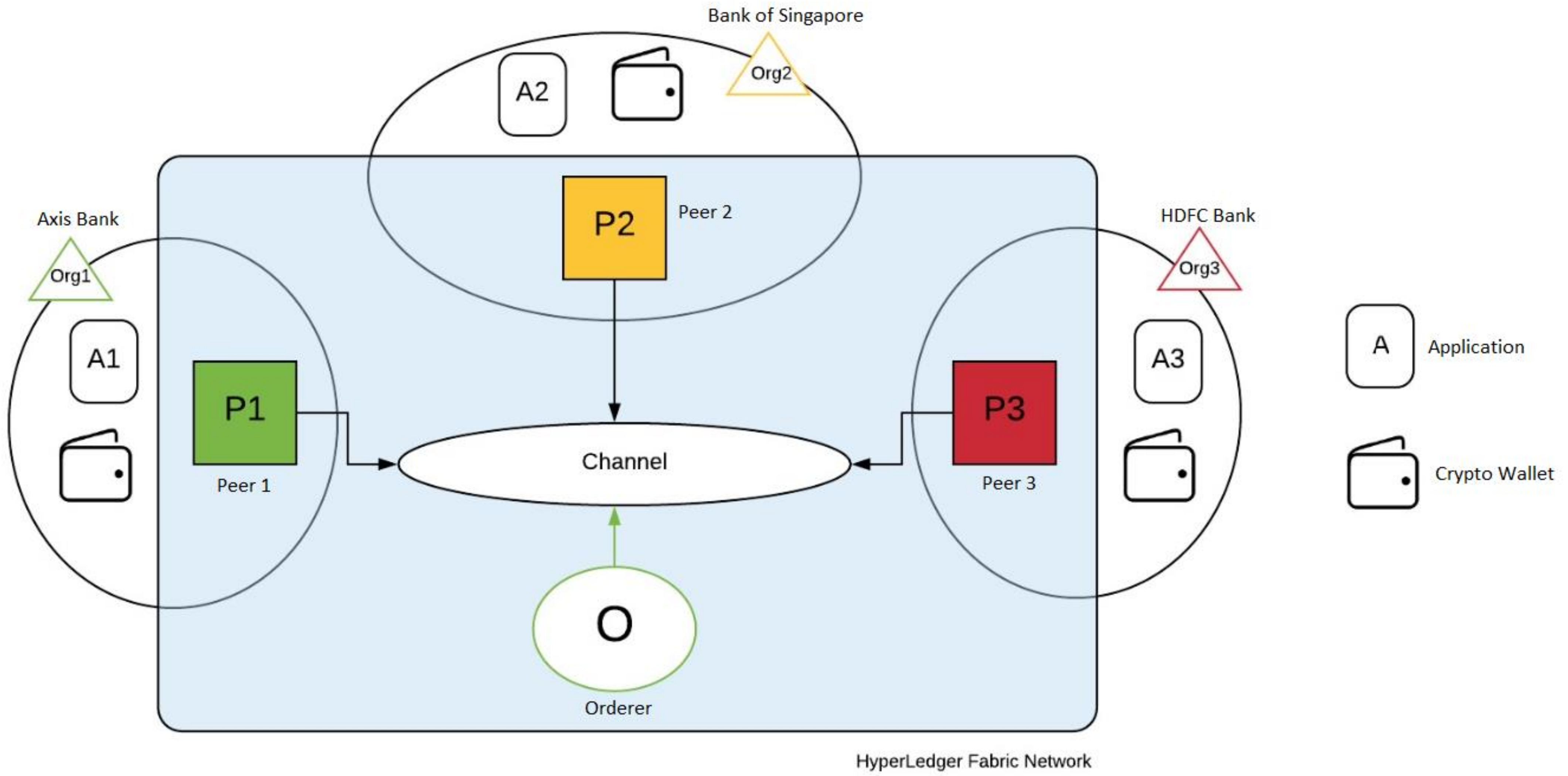
- **Step5: Deliver Transaction**
  Ordering service collects transactions into proposed blocks for distribution to committing peers. The ordering service forms a block of transactions by using ordering services like Kafka, solo (here we are using solo).

- **Step6: Validate Transaction**
  Every committing peer validates against the endorsement policy. Validated transactions are applied to the world state and retained on the ledger. Invalid transactions are also retained on the ledger, but it will not update the world state

HIGH LEVEL ARCHITECTURAL FLOW


5. Transfer of token between organizations
Crypto Wallet
Crypto Wallet
1. Client creates an asset
3. Request to buy asset
Client 1
Application
Client 2
Org 1
Org 2
CreateAsset
GetAllAssets
TransferAsset
GetAssetByCat
Smart Contracts
2. Asset gets stored on ledger
4. Change ownership of asset on network
Ledger
Hyperledger Fabric Network

NETWORK OVERVIEW



OUR SOLUTION!

Synopsis

- *Using Hyperledger Fabric to setup a permissioned blockchain network and adding all the banks (of different organizations)*
- *We use chaincodes (smart contracts) to implement the business logic.*
- *Every organization has its own ledger*
- *We use membership providers to add peers into the network*
- *Every member in the network is a trusted member by the organization*
- *We can hide the internal details of a transaction.*
- *Every transactions is recorded in the ledger.*
- *We use ERC20Token to tokenize the asserts.*

Data Structure

We defined a data structure which contains the details of the transaction

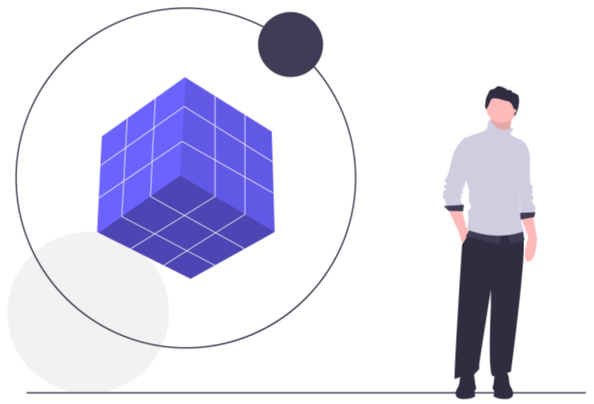
```
{
    UNQID:
    VERSION:
    ISSUERID:
    ISSUERNAME:
    OWNER:
    PRODUCTNAME:
    ISSUEDATE:
    VALUE:
    STATE:
    CATEGORY:
    GASFEE:
}
```

Category

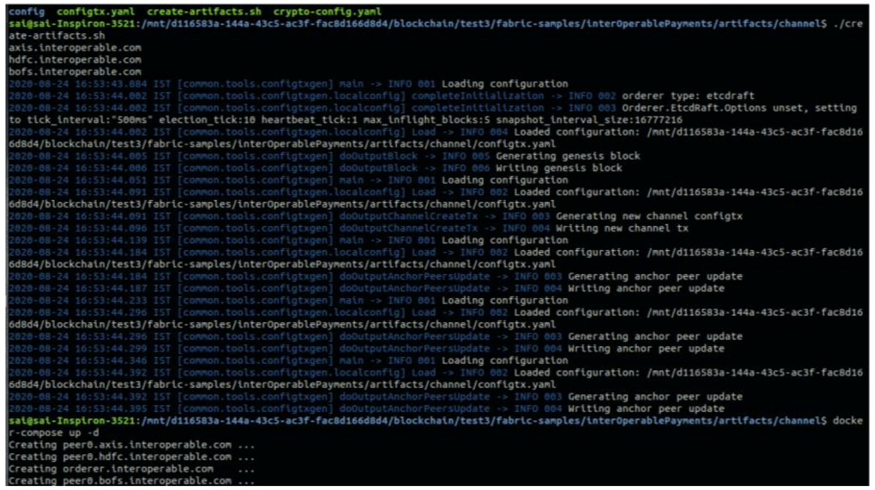
Every assert is categorized into one of the below category:
VEHICLES, MACHINERY, COMPUTERS, STOCKS AND BONDS

Working

- *We categorize every bank based on its organization.*
- *We create a channel and add all the organizations in the channel.*
- *The channel acts as a medium of exchange.*
- *We tokenize every assert so that it can be used in the assert transfer*
- *We can issue or assign or query an assert*

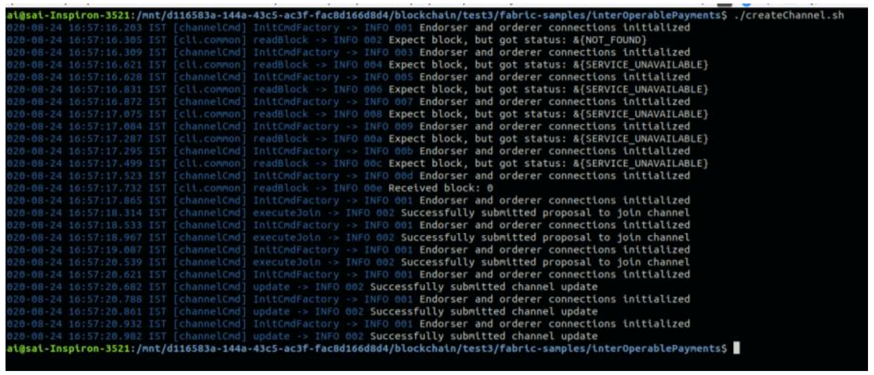


OUTPUT SCREENS



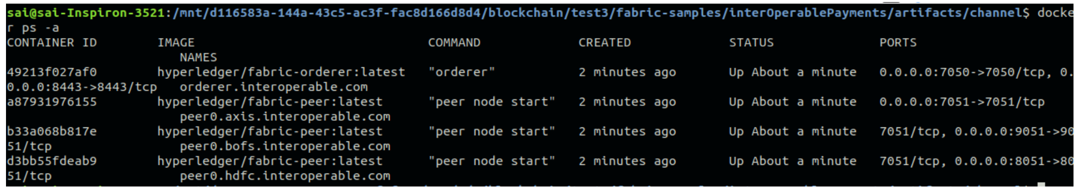
```
config  configtx.yaml  create-artifacts.sh  crypto-config.yaml
sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments/artifacts/channel$ ./cre
ate-artifacts.sh
axis.interoperable.com
hdfc.interoperable.com
bofs.interoperable.com
2020-08-24 16:53:43.884 IST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-08-24 16:53:44.002 IST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 002 orderer type: etcdraft
2020-08-24 16:53:44.002 IST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 Orderer.EtcdRaft.Options unset, setting
to tick_interval:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_blocks:5 snapshot_interval_size:16777216
2020-08-24 16:53:44.002 IST [common.tools.configtxgen.localconfig] Load -> INFO 004 Loaded configuration: /mnt/d116583a-144a-43c5-ac3f-fac8d16
6d8d4/blockchain/test3/fabric-samples/interOperablePayments/artifacts/channel/configtx.yaml
2020-08-24 16:53:44.005 IST [common.tools.configtxgen] doOutputBlock -> INFO 005 Generating genesis block
2020-08-24 16:53:44.006 IST [common.tools.configtxgen] doOutputBlock -> INFO 006 Writing genesis block
2020-08-24 16:53:44.051 IST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-08-24 16:53:44.091 IST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /mnt/d116583a-144a-43c5-ac3f-fac8d16
6d8d4/blockchain/test3/fabric-samples/interOperablePayments/artifacts/channel/configtx.yaml
2020-08-24 16:53:44.091 IST [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 003 Generating new channel configtx
2020-08-24 16:53:44.096 IST [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 004 Writing new channel tx
2020-08-24 16:53:44.139 IST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-08-24 16:53:44.184 IST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /mnt/d116583a-144a-43c5-ac3f-fac8d16
6d8d4/blockchain/test3/fabric-samples/interOperablePayments/artifacts/channel/configtx.yaml
2020-08-24 16:53:44.184 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2020-08-24 16:53:44.187 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
2020-08-24 16:53:44.233 IST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-08-24 16:53:44.296 IST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /mnt/d116583a-144a-43c5-ac3f-fac8d16
6d8d4/blockchain/test3/fabric-samples/interOperablePayments/artifacts/channel/configtx.yaml
2020-08-24 16:53:44.296 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2020-08-24 16:53:44.299 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
2020-08-24 16:53:44.346 IST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-08-24 16:53:44.392 IST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /mnt/d116583a-144a-43c5-ac3f-fac8d16
6d8d4/blockchain/test3/fabric-samples/interOperablePayments/artifacts/channel/configtx.yaml
2020-08-24 16:53:44.392 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2020-08-24 16:53:44.395 IST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments/artifacts/channel$ docke
r-compose up -d
Creating peer0.axis.interoperable.com ...
Creating peer0.hdfc.interoperable.com ...
Creating orderer.interoperable.com    ...
Creating peer0.bofs.interoperable.com ...
```

Creating Certificates and Starting the containers



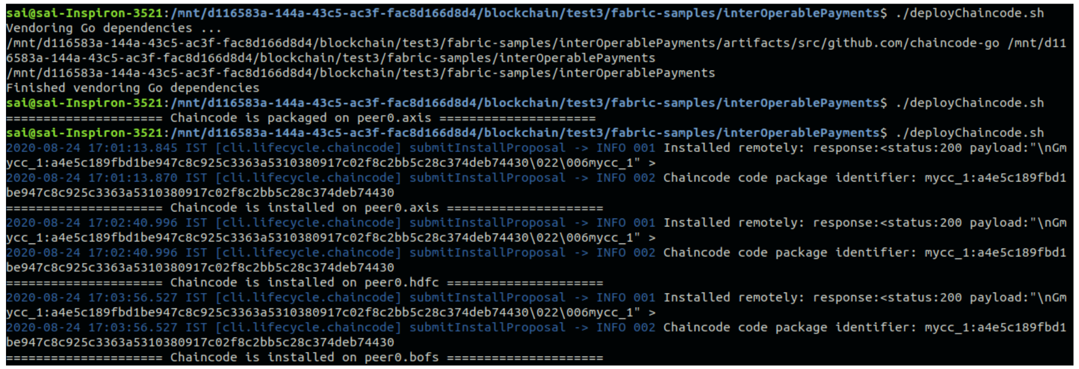
```
ai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./createChannel.sh
020-08-24 16:57:16.203 IST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
020-08-24 16:57:16.305 IST [cli.common] readBlock -> INFO 002 Expect block, but got status: &{NOT_FOUND}
020-08-24 16:57:16.309 IST [channelCmd] InitCmdFactory -> INFO 003 Endorser and orderer connections initialized
020-08-24 16:57:16.621 IST [cli.common] readBlock -> INFO 004 Expect block, but got status: &{SERVICE_UNAVAILABLE}
020-08-24 16:57:16.628 IST [channelCmd] InitCmdFactory -> INFO 005 Endorser and orderer connections initialized
020-08-24 16:57:16.831 IST [cli.common] readBlock -> INFO 006 Expect block, but got status: &{SERVICE_UNAVAILABLE}
020-08-24 16:57:16.872 IST [channelCmd] InitCmdFactory -> INFO 007 Endorser and orderer connections initialized
020-08-24 16:57:17.075 IST [cli.common] readBlock -> INFO 008 Expect block, but got status: &{SERVICE_UNAVAILABLE}
020-08-24 16:57:17.084 IST [channelCmd] InitCmdFactory -> INFO 009 Endorser and orderer connections initialized
020-08-24 16:57:17.287 IST [cli.common] readBlock -> INFO 00a Expect block, but got status: &{SERVICE_UNAVAILABLE}
020-08-24 16:57:17.295 IST [channelCmd] InitCmdFactory -> INFO 00b Endorser and orderer connections initialized
020-08-24 16:57:17.499 IST [cli.common] readBlock -> INFO 00c Expect block, but got status: &{SERVICE_UNAVAILABLE}
020-08-24 16:57:17.523 IST [channelCmd] InitCmdFactory -> INFO 00d Endorser and orderer connections initialized
020-08-24 16:57:17.732 IST [cli.common] readBlock -> INFO 00e Received block: 0
020-08-24 16:57:17.865 IST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
020-08-24 16:57:18.314 IST [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
020-08-24 16:57:18.533 IST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
020-08-24 16:57:18.967 IST [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
020-08-24 16:57:19.087 IST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
020-08-24 16:57:20.539 IST [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
020-08-24 16:57:20.621 IST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
020-08-24 16:57:20.682 IST [channelCmd] update -> INFO 002 Successfully submitted channel update
020-08-24 16:57:20.788 IST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
020-08-24 16:57:20.861 IST [channelCmd] update -> INFO 002 Successfully submitted channel update
020-08-24 16:57:20.932 IST [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
020-08-24 16:57:20.982 IST [channelCmd] update -> INFO 002 Successfully submitted channel update
ai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$
```

Creating Channel



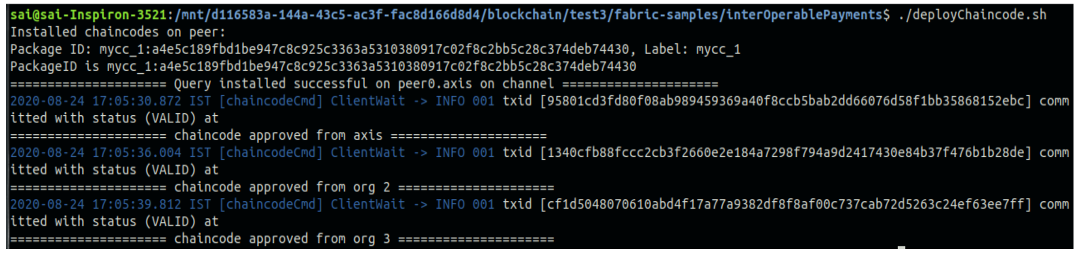
```
sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments/artifacts/channel$ docke
r ps -a
CONTAINER ID        IMAGE                               COMMAND             CREATED             STATUS              PORTS
                      NAMES
49213f027af0        hyperledger/fabric-orderer:latest   "orderer"           2 minutes ago       Up About a minute   0.0.0.0:7050->7050/tcp, 0.
0.0.0:8443->8443/tcp    orderer.interoperable.com
a87931976155        hyperledger/fabric-peer:latest      "peer node start"   2 minutes ago       Up About a minute   0.0.0.0:7051->7051/tcp
                      peer0.axis.interoperable.com
b33a068b817e        hyperledger/fabric-peer:latest      "peer node start"   2 minutes ago       Up About a minute   7051/tcp, 0.0.0.0:9051->90
51/tcp                peer0.bofs.interoperable.com
d3bb55fdeab9        hyperledger/fabric-peer:latest      "peer node start"   2 minutes ago       Up About a minute   7051/tcp, 0.0.0.0:8051->80
51/tcp                peer0.hdfc.interoperable.com
```

Containers Created



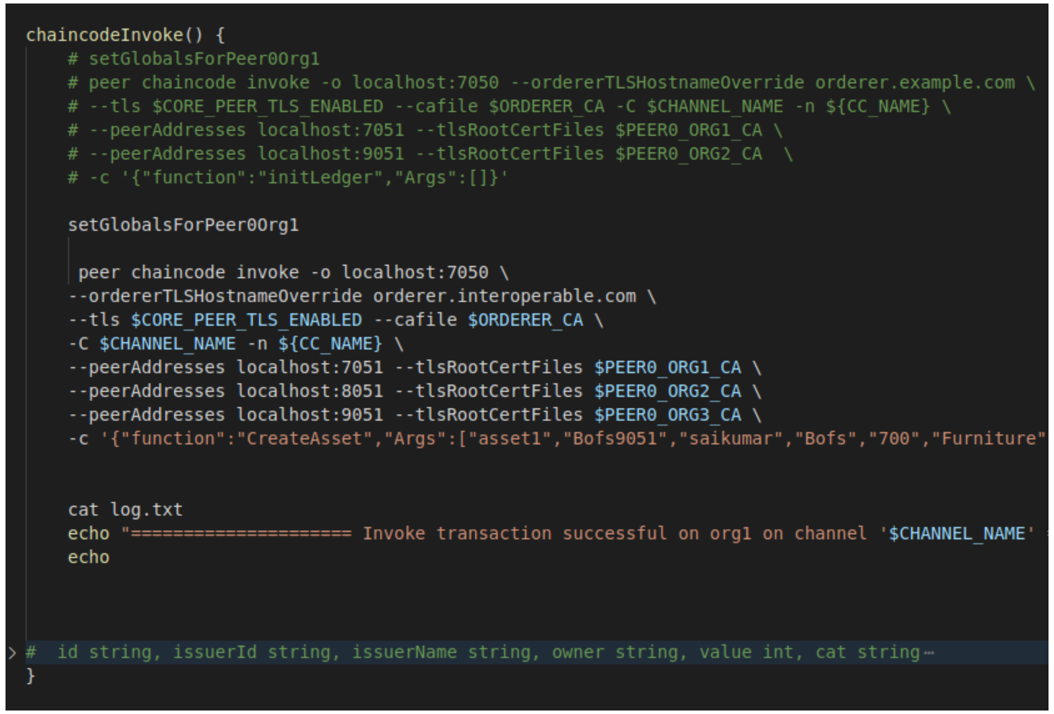
```
sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh
Vendoring Go dependencies ...
/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments/artifacts/src/github.com/chaincode-go /mnt/d11
6583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments
/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments
Finished vendoring Go dependencies
sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh
===================== Chaincode is packaged on peer0.axis =====================
sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh
2020-08-24 17:01:13.845 IST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:<status:200 payload:"\nGm
ycc_1:a4e5c189fbd1be947c8c925c3363a5310380917c02f8c2bb5c28c374deb74430\022\006mycc_1" >
2020-08-24 17:01:13.870 IST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code package identifier: mycc_1:a4e5c189fbd1
be947c8c925c3363a5310380917c02f8c2bb5c28c374deb74430
===================== Chaincode is installed on peer0.axis =====================
2020-08-24 17:02:40.996 IST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:<status:200 payload:"\nGm
ycc_1:a4e5c189fbd1be947c8c925c3363a5310380917c02f8c2bb5c28c374deb74430\022\006mycc_1" >
2020-08-24 17:02:40.996 IST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code package identifier: mycc_1:a4e5c189fbd1
be947c8c925c3363a5310380917c02f8c2bb5c28c374deb74430
===================== Chaincode is installed on peer0.hdfc =====================
2020-08-24 17:03:56.527 IST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:<status:200 payload:"\nGm
ycc_1:a4e5c189fbd1be947c8c925c3363a5310380917c02f8c2bb5c28c374deb74430\022\006mycc_1" >
2020-08-24 17:03:56.527 IST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code package identifier: mycc_1:a4e5c189fbd1
be947c8c925c3363a5310380917c02f8c2bb5c28c374deb74430
===================== Chaincode is installed on peer0.bofs =====================
```

Chaincode Installed



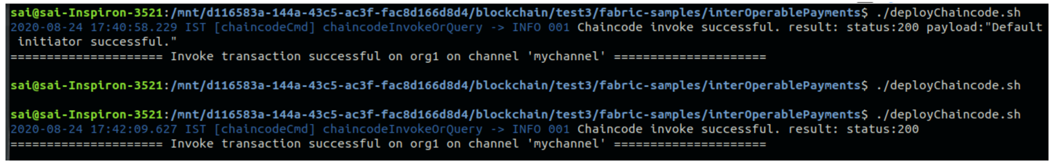
```
sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh
Installed chaincodes on peer:
Package ID: mycc_1:a4e5c189fbd1be947c8c925c3363a5310380917c02f8c2bb5c28c374deb74430, Label: mycc_1
PackageID is mycc_1:a4e5c189fbd1be947c8c925c3363a5310380917c02f8c2bb5c28c374deb74430
===================== Query installed successful on peer0.axis on channel =====================
2020-08-24 17:05:30.872 IST [chaincodeCmd] ClientWait -> INFO 001 txid [95801cd3fd80f08ab989459369a40f8ccb5bab2dd66076d58f1bb35868152ebc] comm
itted with status (VALID) at
===================== chaincode approved from axis =====================
2020-08-24 17:05:36.004 IST [chaincodeCmd] ClientWait -> INFO 001 txid [1340cfb88fccc2cb3f2660e2e184a7298f794a9d2417430e84b37f476b1b28de] comm
itted with status (VALID) at
===================== chaincode approved from org 2 =====================
2020-08-24 17:05:39.812 IST [chaincodeCmd] ClientWait -> INFO 001 txid [cf1d5048070610abd4f17a77a9382df8f8af00c737cab72d5263c24ef63ee7ff] comm
itted with status (VALID) at
===================== chaincode approved from org 3 =====================
```

Chaincode Approved



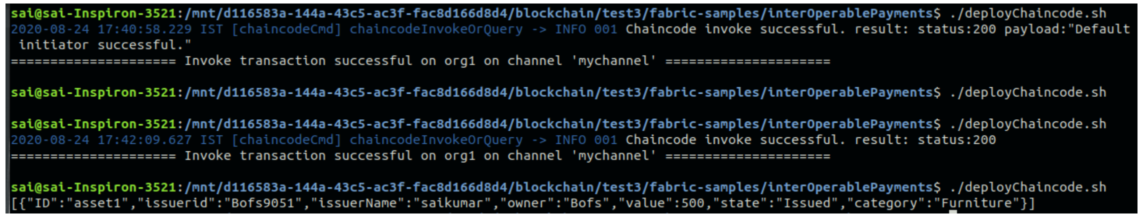
```
chaincodeInvoke() {
    # setGlobalsForPeer0Org1
    # peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com \
    # --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA -C $CHANNEL_NAME -n ${CC_NAME} \
    # --peerAddresses localhost:7051 --tlsRootCertFiles $PEER0_ORG1_CA \
    # --peerAddresses localhost:9051 --tlsRootCertFiles $PEER0_ORG2_CA  \
    # -c '{"function":"initLedger","Args":[]}'

    setGlobalsForPeer0Org1

     peer chaincode invoke -o localhost:7050 \
    --ordererTLSHostnameOverride orderer.interoperable.com \
    --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA \
    -C $CHANNEL_NAME -n ${CC_NAME} \
    --peerAddresses localhost:7051 --tlsRootCertFiles $PEER0_ORG1_CA \
    --peerAddresses localhost:8051 --tlsRootCertFiles $PEER0_ORG2_CA \
    --peerAddresses localhost:9051 --tlsRootCertFiles $PEER0_ORG3_CA \
    -c '{"function":"CreateAsset","Args":["asset1","Bofs9051","saikumar","Bofs","700","Furniture"


    cat log.txt
    echo "==================== Invoke transaction successful on org1 on channel '$CHANNEL_NAME'
    echo



> #  id string, issuerId string, issuerName string, owner string, value int, cat string…
}
```

Create Assert Function



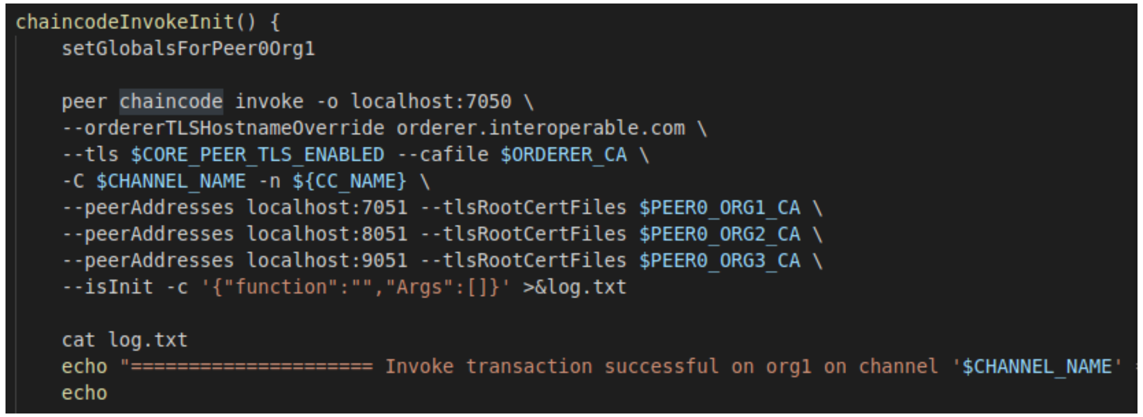
```
sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh
2020-08-24 17:40:58.229 IST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"Default
 initiator successful."
==================== Invoke transaction successful on org1 on channel 'mychannel' ====================

sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh

sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh
2020-08-24 17:42:09.627 IST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
==================== Invoke transaction successful on org1 on channel 'mychannel' ====================
```

Invoke create assert



```
sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh
2020-08-24 17:40:58.229 IST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"Default
 initiator successful."
==================== Invoke transaction successful on org1 on channel 'mychannel' ====================

sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh

sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh
2020-08-24 17:42:09.627 IST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
==================== Invoke transaction successful on org1 on channel 'mychannel' ====================

sai@sai-Inspiron-3521:/mnt/d116583a-144a-43c5-ac3f-fac8d166d8d4/blockchain/test3/fabric-samples/interOperablePayments$ ./deployChaincode.sh
[{"ID":"asset1","issuerid":"Bofs9051","issuerName":"saikumar","owner":"Bofs","value":500,"state":"Issued","category":"Furniture"}]
```

Query after creating
assert



```
chaincodeInvokeInit() {
    setGlobalsForPeer0Org1

    peer chaincode invoke -o localhost:7050 \
    --ordererTLSHostnameOverride orderer.interoperable.com \
    --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA \
    -C $CHANNEL_NAME -n ${CC_NAME} \
    --peerAddresses localhost:7051 --tlsRootCertFiles $PEER0_ORG1_CA \
    --peerAddresses localhost:8051 --tlsRootCertFiles $PEER0_ORG2_CA \
    --peerAddresses localhost:9051 --tlsRootCertFiles $PEER0_ORG3_CA \
    --isInit -c '{"function":"","Args":[]}' >&log.txt

    cat log.txt
    echo "==================== Invoke transaction successful on org1 on channel '$CHANNEL_NAME'
    echo
```

Chaincode Invoke Init
Function

Thank You!