```
#!/bin/bash

echo
echo " _____   _____   _   _____  _____ "
echo "/  ___| |_   _| / \ |  _  \|_   _|"
echo "\___ \   | |  / _ \ | |_) |  | |  "
echo " ___) |  | | / ___ \|  _ <   | |  "
echo "|____/   |_|/_/   \_\_| \_\  |_|  "
echo
echo "////// Scenario = 'Purchase Data' ///////"
echo
CHANNEL_NAME_BASE="$1"
DELAY="$2"
: ${CHANNEL_NAME_BASE:="channel"}
: ${DELAY:=3}
: ${TIMEOUT:="2"}
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganiza
tions/zak.codes/orderers/orderer.zak.codes/msp/tlscacerts/tlsca.zak.codes-cert.pem

echo "Channel name : "$CHANNEL_NAME_BASE"1"
echo "Channel name : "$CHANNEL_NAME_BASE"2"
echo "Channel name : "$CHANNEL_NAME_BASE"3"

# verify the result
verifyResult () {
  if [ $1 -ne 0 ] ; then
    echo "!!!!!!!!!!!!!!! "$2" !!!!!!!!!!!!!!!"
    echo "========= ERROR !!! FAILED to execute End-2-End Scenario ==========="
    echo
      exit 1
  fi
}

setGlobals () {

  if [ $1 -eq 0 -o $1 -eq 1 ] ; then
    CORE_PEER_LOCALMSPID="City1MSP"
    CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/c
rypto/peerOrganizations/city1.zak.codes/peers/peer0.city1.zak.codes/tls/ca.crt
    CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypt
o/peerOrganizations/city1.zak.codes/users/Admin@city1.zak.codes/msp
    if [ $1 -eq 0 ]; then
      CORE_PEER_ADDRESS=peer0.city1.zak.codes:7051
    else
      CORE_PEER_ADDRESS=peer1.city1.zak.codes:7051
      CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/cry
pto/peerOrganizations/city1.zak.codes/users/Admin@city1.zak.codes/msp
    fi
  else
    CORE_PEER_LOCALMSPID="City2MSP"
    CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/c
rypto/peerOrganizations/city2.zak.codes/peers/peer0.city2.zak.codes/tls/ca.crt
    CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypt
o/peerOrganizations/city2.zak.codes/users/Admin@city2.zak.codes/msp
    if [ $1 -eq 2 ]; then
      CORE_PEER_ADDRESS=peer0.city2.zak.codes:7051
    else
      CORE_PEER_ADDRESS=peer1.city2.zak.codes:7051
    fi
  fi

  env |grep CORE
}

chaincodeQueryCheck () {
  PEER=$1
  CC_NAME=$2
  EXPECTED_VALUE=$3
  echo "===================== Querying on PEER$PEER on channel '$CHANNEL_NAME' and c
haincode '$CC_NAME' ... ===================== "
  echo "--> Expected value: ${EXPECTED_VALUE}"
  setGlobals $PEER
  local rc=1
```

```
  local starttime=$(date +%s)

  # continue to poll
  # we either get a successful response, or reach TIMEOUT
  while test "$(($(date +%s)-starttime))" -lt "$TIMEOUT" -a $rc -ne 0
  do
     sleep $DELAY
     echo "Attempting to Query PEER$PEER ...$(($(date +%s)-starttime)) secs"
     peer chaincode query -C $CHANNEL_NAME -n $CC_NAME -c "${PAYLOAD}" >&log.txt
     test $? -eq 0 && VALUE=$(cat log.txt | awk '/Query Result/ {print $NF}')
     test "$VALUE" = "$EXPECTED_VALUE" && let rc=0
  done
  echo
  cat log.txt
  if test $rc -eq 0 ; then
  echo "===================== Query on PEER$PEER on channel '$CHANNEL_NAME' and chai
ncode '$CC_NAME' is successful ===================== "
  else
  echo "!!!!!!!!!!!!!!! Query result on PEER$PEER is INVALID !!!!!!!!!!!!!!!!"
        echo "================== ERROR !!! FAILED to execute Purchase Data Scenario
=================="
  echo
  exit 1
  fi
}

chaincodeQuery () {
  PEER=$1
  CC_NAME=$2
  echo "===================== Querying on PEER$PEER on channel '$CHANNEL_NAME' and c
haincode '$CC_NAME' ... ===================== "
  setGlobals $PEER
  local rc=1
  local starttime=$(date +%s)

  # continue to poll
  # we either get a successful response, or reach TIMEOUT
  while test "$(($(date +%s)-starttime))" -lt "$TIMEOUT" -a $rc -ne 0
  do
     sleep $DELAY
     echo "Attempting to Query PEER$PEER ...$(($(date +%s)-starttime)) secs"
     peer chaincode query -C $CHANNEL_NAME -n $CC_NAME -c "${PAYLOAD}" >&log.txt
     test $? -eq 0 && RETURNED_QUERY=$(cat log.txt | awk '/Query Result/ {print $NF}
') && let rc=0
  done
  echo
  cat log.txt
  if test $rc -eq 0 ; then
  echo "===================== Query on PEER$PEER on channel '$CHANNEL_NAME' and chai
ncode '$CC_NAME' is successful ===================== "
  else
  echo "!!!!!!!!!!!!!!! Query result on PEER$PEER is INVALID !!!!!!!!!!!!!!!!"
        echo "================== ERROR !!! FAILED to execute Purchase Data Scenario
=================="
  echo
  exit 1
  fi
}

chaincodeInvoke () {
  PEER=$1
  CC_NAME=$2
  setGlobals $PEER
  # while 'peer chaincode' command can get the orderer endpoint from the peer (if jo
in was successful),
  # lets supply it directly as we know it using the "-o" option
  if [ -z "$CORE_PEER_TLS_ENABLED" -o "$CORE_PEER_TLS_ENABLED" = "false" ]; then
    peer chaincode invoke -o orderer.zak.codes:7050 -C $CHANNEL_NAME -n $CC_NAME -c
"${PAYLOAD}" >&log.txt
  else
    peer chaincode invoke -o orderer.zak.codes:7050  --tls $CORE_PEER_TLS_ENABLED --
cafile $ORDERER_CA -C $CHANNEL_NAME -n $CC_NAME -c "${PAYLOAD}" >&log.txt
  fi
```

```
  res=$?
  cat log.txt
  verifyResult $res "Invoke execution on PEER$PEER failed "
  # Extract the returned payload without quotes
  RETURNED_PAYLOAD=$(cat log.txt | awk -F"payload:" '{print $2}')
    RETURNED_PAYLOAD=$(echo $RETURNED_PAYLOAD | awk -F">" '{print $1}')
  echo "===================== Invoke transaction on PEER$PEER on channel '$CHANNEL_N
AME' and chaincode '$CC_NAME' is successful ===================== "
  echo
}

# Create global variable RETURNED_PAYLOAD that is used in chaincodeInvoke function
RETURNED_PAYLOAD=""

# Invoke on chaincode_data on Peer0/City1
echo "--> Sending invoke second transaction createDate on City1/peer0 on chaincode_d
ata ..."
echo "--> Creating data on blockchain 1 <--"
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}1"
CREATION_TIME=$(date +%s)
PAYLOAD='{"Args":["createData", "2", "test data", "100", "Celsius", "'
PAYLOAD=$PAYLOAD$CREATION_TIME
ENDING='", "marcel"]}'
PAYLOAD=$PAYLOAD$ENDING

chaincodeInvoke 0 chaincode_data

# Invoke on chaincode_ad on Peer0/City1
# Paid data 10
echo "--> Sending invoke second transaction createDataEntryAd on Peer0/City1 on chai
ncode_ad ..."
echo "--> Creating data advertisement on blockchain 2 <--"
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}2"
PAYLOAD='{"Args":["createDataEntryAd", "2", "test data", "???", "Celsius", "'
PAYLOAD=$PAYLOAD$CREATION_TIME
ENDING='", "marcel", "10", "1"]}'
PAYLOAD=$PAYLOAD$ENDING
chaincodeInvoke 0 chaincode_ad

# Get the current amount of tokens on account 1
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["getAccountTokens", "1"]}'
chaincodeQuery 0 chaincode_tokens
ACC1_TOKENS=$RETURNED_QUERY

# Get the current amount of tokens on account 2
PAYLOAD='{"Args":["getAccountTokens", "2"]}'
chaincodeQuery 0 chaincode_tokens
ACC2_TOKENS=$RETURNED_QUERY

# Invoke on chaincode_tokens on Peer2/City2
echo "--> Sending invoke transaction sendTokensSafe on Peer2/City2 on chaincode_toke
ns"
echo "--> Sending 10 tokens for data purchase from account 2 to account 1 on blockch
ain 3 <--"
echo
sleep $DELAY
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["sendTokensSafe", "2", "1", "10", "true"]}'
chaincodeInvoke 2 chaincode_tokens

# Query on chaincode_tokens on Peer0/City1
echo "--> Sending invoke transaction getAccountTokens on Peer0/City1 on chaincode_to
kens"
echo "--> Even though tokens were sent to account 1, they are not available until da
ta retrieved <--"
echo
sleep $DELAY # required sleep to wait for previous data to commit and being availabl
e
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["getAccountTokens", "1"]}'
```

```
chaincodeQueryCheck 0 chaincode_tokens $ACC1_TOKENS

# Query on chaincode_tokens on Peer2/City2
echo "--> Sending invoke transaction changePendingTx on Peer2/City2 on chaincode_tok
ens"
echo "--> Check if tokens were subtracted from account 2 <--"
echo
sleep $DELAY # required sleep to wait for previous data to commit and being availabl
e
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["getAccountTokens", "2"]}'
chaincodeQueryCheck 2 chaincode_tokens $(( $ACC2_TOKENS - 10 ))

# Invoke on chaincode_ad on Peer0/City1
echo "--> Sending invoke transaction revealPaidData on Peer0/City1 on chaincode_ad"
echo "--> Using TxID from tokens transfer as a ticket to reveal paid data <--"
echo $RETURNED_PAYLOAD
echo
TXID=$RETURNED_PAYLOAD
sleep $DELAY  # required sleep to wait for previous data to commit and being availab
le
CHANNEL_NAME="${CHANNEL_NAME_BASE}2"
PAYLOAD='{"Args":["revealPaidData", "channel1", "chaincode_data", "2", "'
PAYLOAD=$PAYLOAD$CREATION_TIME
MIDDLE='", "channel3", "chaincode_tokens", '
PAYLOAD=$PAYLOAD$MIDDLE$RETURNED_PAYLOAD
PAYLOAD="${PAYLOAD}]}"
chaincodeInvoke 0 chaincode_ad

# Invoke on chaincode_tokens on Peer2/City2
echo "--> Sending invoke transaction changePendingTx on Peer2/City2 on chaincode_tok
ens"
echo "--> Now data entry has revealed value and Blocked tokens can be changed to ava
ilable <--"
echo
sleep $DELAY # required sleep to wait for previous data to commit and being availabl
e
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["changePendingTx", "channel2", "chaincode_ad", '
PAYLOAD=$PAYLOAD$TXID
PAYLOAD="${PAYLOAD}]}"
chaincodeInvoke 2 chaincode_tokens

# Query on chaincode_tokens on Peer0/City1
echo "--> Sending invoke transaction getAccountTokens on Peer0/City1 on chaincode_to
kens"
echo "--> Check if tokens are now available on account 2 <--"
echo
sleep $DELAY # required sleep to wait for previous data to commit and being availabl
e
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["getAccountTokens", "1"]}'
chaincodeQueryCheck 0 chaincode_tokens $(( $ACC1_TOKENS + 10 ))

# peer chaincode invoke --tls true --cafile /opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/ordererOrganizations/zak.codes/orderers/orderer.zak.codes/msp/tlsc
acerts/tlsca.zak.codes-cert.pem -n chaincode_ad -c '{"Args":["revealPaidData", "chan
nel1", "chaincode_data", "2", "20180321160000", "channel3", "chaincode_tokens", "txI
D"]}' -C channel2
# peer chaincode invoke --tls true --cafile /opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/ordererOrganizations/zak.codes/orderers/orderer.zak.codes/msp/tlsc
acerts/tlsca.zak.codes-cert.pem -n chaincode_tokens -c '{"Args":["getAccountTokens",
 "1"]}' -C channel3
echo
echo
echo " _____   _   _   ____   "
echo "| ____| | \ | | |  _ \  "
echo "|  _|   |  \| | | | | | "
echo "| |___  | |\  | | |_| | "
echo "|_____| |_| \_| |____/  "
echo
echo "////// Scenario = 'Purchase Data' executed successfully ///////"
echo
```

```
exit 0
```