**###Transaction Contract**

'use strict';

const { Contract } = require('fabric-contract-api');

class TransactionContract extends Contract {

async initLedger(ctx) {

console.info('Instantiating the ledger with some transactions');

const transactions = [

{

DEALERID: 'D1001',

MSISDN: '1234567890',

MPIN: '1111',

BALANCE: 1000,

STATUS: 'Active',

TRANSAMOUNT: 500,

TRANSTYPE: 'Debit',

REMARKS: 'Initial transaction'

},

{

DEALERID: 'D1002',

MSISDN: '0987654321',

MPIN: '2222',

BALANCE: 2000,

STATUS: 'Active',

TRANSAMOUNT: 1000,

TRANSTYPE: 'Credit',

REMARKS: 'Deposit transaction'

}

];

for (const transaction of transactions) {

transaction.docType = 'transaction';

await ctx.stub.putState(transaction.DEALERID, Buffer.from(JSON.stringify(transaction)));

console.info(`Transaction ${transaction.DEALERID} initialized`);

}

}

async queryTransaction(ctx, dealerId) {

const transactionAsBytes = await ctx.stub.getState(dealerId);

if (!transactionAsBytes || transactionAsBytes.length === 0) {

throw new Error(`${dealerId} does not exist`);

}

console.log(transactionAsBytes.toString());

return transactionAsBytes.toString();

}

async createTransaction(ctx, dealerId, msisdn, mpin, balance, status, transAmount, transType, remarks) {

console.info('Creating new transaction...');

const transaction = {

DEALERID: dealerId,

MSISDN: msisdn,

MPIN: mpin,

BALANCE: parseFloat(balance),

STATUS: status,

TRANSAMOUNT: parseFloat(transAmount),

TRANSTYPE: transType,

REMARKS: remarks

};

await ctx.stub.putState(dealerId, Buffer.from(JSON.stringify(transaction)));

console.info(`Transaction ${dealerId} created`);

}

async updateTransaction(ctx, dealerId, status, balance) {

console.info('Updating transaction...');

const transactionAsBytes = await ctx.stub.getState(dealerId);

if (!transactionAsBytes || transactionAsBytes.length === 0) {

throw new Error(`${dealerId} does not exist`);

}

const transaction = JSON.parse(transactionAsBytes.toString());

transaction.STATUS = status;

transaction.BALANCE = parseFloat(balance);

await ctx.stub.putState(dealerId, Buffer.from(JSON.stringify(transaction)));

console.info(`Transaction ${dealerId} updated`);

}

async deleteTransaction(ctx, dealerId) {

const exists = await ctx.stub.getState(dealerId);

if (!exists || exists.length === 0) {

throw new Error(`${dealerId} does not exist`);

}

await ctx.stub.deleteState(dealerId);

console.info(`Transaction ${dealerId} deleted`);

}

}

module.exports = TransactionContract;

**###Testing The Smart Contract**

'use strict';

const { Gateway, Wallets } = require('fabric-network');

const path = require('path');

const fs = require('fs');

async function main() {

try {

const ccpPath = path.resolve(\_\_dirname, 'connection.json');

const ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'));

const walletPath = path.join(process.cwd(), 'wallet');

const wallet = await Wallets.newFileSystemWallet(walletPath);

const gateway = new Gateway();

await gateway.connect(ccp, {

wallet,

identity: 'user1',

discovery: { enabled: true, asLocalhost: true }

});

const network = await gateway.getNetwork('mychannel');

const contract = network.getContract('transactionContract');

// Query a transaction

const result = await contract.evaluateTransaction('queryTransaction', 'D1001');

console.log(`Transaction: ${result.toString()}`);

// Create a new transaction

await contract.submitTransaction('createTransaction', 'D1003', '1122334455', '3333', '500', 'Active', '200', 'Credit', 'Test Transaction');

console.log('Transaction created.');

// Update a transaction

await contract.submitTransaction('updateTransaction', 'D1003', 'Suspended', '300');

console.log('Transaction updated.');

// Delete a transaction

await contract.submitTransaction('deleteTransaction', 'D1003');

console.log('Transaction deleted.');

await gateway.disconnect();

} catch (error) {

console.error(`Failed to submit transaction: ${error}`);

}

}

main();