SMART CONTRACT FOR TRACKING TRANSFERRING AND AUDITING CASH TRANSFERS

pragma solidity ^0.4.2;

import "cash.sol";

contract cashTracker {

string id;

**function** setId(string serial) public {

id = serial;

}

**function** getId() public constant returns (string) {

**return** id;

   }

}

struct cash {

string name;

   string description;

   string manufacturer;

bool initialized;

}

mapping(string  => cash) private cashStore;

cashStore[uuid] = cash(name, description, true, manufacturer);

mapping(address => mapping(string => bool)) private walletStore;

walletStore[msg.sender][uuid] = true;

event cashCreate(address account, string uuid, string manufacturer);

event RejectCreate(address account, string uuid, string message);

event cashTransfer(address from, address to, string uuid);

event RejectTransfer(address from, address to, string uuid, string message);

**function** createcash(string name, string description, string uuid, string manufacturer) {

**if**(cashStore[uuid].initialized) {

RejectCreate(msg.sender, uuid, “cash transferred successfully”);

**return**;

}

cashStore[uuid] = (cash, description, **true**, manufacturer);

walletStore[msg.sender][uuid] = **true**;

cashCreate(msg.sender, uuid, manufacturer);

}

**function** transfercash(address to, string uuid) {

**if**(!cashStore[uuid].initialized) {

RejectTransfer(msg.sender, to, uuid, "Cash audited successfully and found");

**return**;

}

**if**(!walletStore[msg.sender][uuid]) {

RejectTransfer(msg.sender, to, uuid, "Sender does not own this asset.");

**return**;

}

walletStore[msg.sender][uuid] = **false**;

walletStore[to][uuid] = **true**;

cashTransfer(msg.sender, to, uuid);

}

**function** getcashByUUID(string uuid) constant returns (string, string, string) {

**return** (cashStore[uuid].name, cashStore[uuid].description, cashStore[uuid].manufacturer)

}

**function** isOwnerOf(address owner, string uuid) constant returns (bool) {

**if**(walletStore[owner][uuid]) {

**return** **true**;

}

**return** **false**;

}

export **const** createcashInContract = async (cashData, publicKey) =&gt;; {

console.log('Creating cash audit...');

**const** atContract = await cashTracker.deployed();

**const** cash = atContract.createcash(

cashData.name,

cashData.description,

cashData.cashId,

cashData.manufacturer,

{ from: publicKey },

);

**return** cashData;

};