The “Contractor” Contract

```
3  ////////////////////////////////////////////////////////////////////////////////
4  // This is a Contractor payment contract.
5  //
6  // The owners of the contract are the Provider and Reciever of the work.
7  //
8  // The Provider is paid _payRate ETH per unit of time given at constructor.
9  //
10 // The Reciever funds the contract with ETH by sending ETH to the contract
11 // address.
12 //
13 // Work in progress (WIP).
14 //
15 // Burton Samograd
16 // kruhft@icloud.com
17 // Copyright 2018
18 // BusFactor1 Inc.
19 // License: AGPL
20 ////////////////////////////////////////////////////////////////////////////////
21
22 contract Contractor {
23
24     address payer;
25     address payee;
26     string firstName;
27     string lastName;
28     uint payRate;
29
30     struct TimeSpan {
31         uint start;
32         uint finish;
33         uint delta;
34     }
35
36     event Deposit(address, uint);
37     event ClockIn(address, uint);
38     event ClockOut(address, uint, uint);
39
40     mapping (address => TimeSpan[]) workRecords;
41
42     function constructor (address _payee,
43                           string _firstName, string _lastName,
44                           uint _payRate) public {
45         payer = msg.sender;
46         payee = _payee;
47         firstName = _firstName;
48         lastName = _lastName;
49         payRate = _payRate;
50     }
51
52     function () public payable {  // General contract deposit
53         // Anyone can deposit funds into this contract
54         emit Deposit(msg.sender, msg.value);
55     }
```

```
    function notPayer (address who) public view returns (bool) {
        return who != payer; }

    function notPayee (address who) public view returns (bool) {
        return who != payee; }

    function notPayerOrPayee (address who) public view returns (bool) {
        return !(who == payer || who == payee); }

    function clockIn () public {
        address them = msg.sender;
        TimeSpan[] storage workRecord = workRecords[them];
        TimeSpan storage ts = workRecord[workRecord.length - 1];

        if(ts.finish == 0) return; // already clocked in

        workRecord.length += 1;
        ts = workRecord[workRecord.length - 1];

        uint startTime = now;
        ts.start = startTime;
        ts.finish = 0;

        emit ClockIn(them, startTime);
    }

    function clockOut () public {
        address them = msg.sender;
        TimeSpan[] storage workRecord = workRecords[them];
        TimeSpan storage ts = workRecord[workRecord.length - 1];

        uint endTime = now;

        ts.finish = endTime;
        ts.delta = ts.finish - ts.start;

        uint pay = payRate * ts.delta;

        them.transfer(pay);

        emit ClockOut(them, endTime, pay);
    }

    function getBalance () public view returns (uint) {
        if(!(msg.sender == payer || msg.sender == payee)) return;

        return address(this).balance;
    }

    function getAvailableTime () public view returns (uint) {
        if(notPayerOrPayee(msg.sender)) return;

        return getBalance() / payRate;
    }

    function setPayRate (uint newRate) public {
        if(notPayer(msg.sender)) return;

        payRate = newRate;
```

```
116     }
117
118     function getPayRate () public view returns (uint) {
119         if(notPayerOrPayee(msg.sender)) return;
120
121         return payRate;
122     }
123 }
```

This is meant to be a fair method of payment where the counterparty risk of the Contractor is minimized by the staking of the future payment by the Contractee.

This contract is to be used when the Contractor is to be immediately paid out upon completion of work at clock out time.

This means that the work CANNOT BE VERIFIED by the contract. Clocking out by the Contractor guarantees payment of work done.

Question: What if the clock out time is after the amount of units available in the account?

One might think that the employee must keep track of available time and work accordingly.

TODO: Adjust contract to ensure that employee is paid (need negative balance and approval method by payee to adjust this balance)

Question: Does the above sound like a good idea to add to the contract? Why and/or why not?

This contract would work best for time based Employment based on physical completion of tasks.

—
Burton Samograd
kruhft@icloud.com