**Background**

The aim of this project was to harness the abilities of Ethereum-based smart contracts and cryptocurrency to aid in the execution, tracking and settlement of transactions within an Insurance contract. These transactions are based on a savings contract between an individual and a Life Assurer.

The client is looking to invest in a diverse basket of assets, each offering a different amount of investment growth, with the aim of accumulating funds for his/her child’s College education. Once a sufficient amount has been saved, the client will have the option to withdraw all or part of their funds upon request.

It is assumed, for benefit of our exercise, that the client is happy to both fund and accept withdrawals into their personal wallet on the Ethereum Network. It is further assumed that all transactions occur in either ETH (Ethereum) or the tokens issued by the Insurer itself, albeit some of the activity refer to ongoing ‘real-world’ transactions, as highlight below.

**Rationale and Requirements**

A client has established a Regular Savings contract with a Life Assurer. It will save a fixed amount on an ongoing basis. The savings will be split equally between 3 sub-Wallets, each applying a different rate of growth/uplift depending on market returns. The savings come from a separate personal account, which is unrelated, receiving no Yield and has no influence on the contract.

The client can save a fixed amount of ETH per month using this Regular Savings Contract with the hope of reaching a given amount of ETH that’ll pay for College Fee’s at some point in the future. The monthly contributions are split equally into 3 different asset types, each contained in its own sub-Wallet of the Insurer Reserves:

**Sub-Wallet Type:** Equity Wallet; Property Wallet; Bonds Wallet

**Annualised Growth Rate:** A company Employee will check the performance of the three assets once a month and apply an adjustment to the Tokens held by the wallets to reflect this. We didn’t use an Oracle to track real asset performances as it’s expensive on ‘Gas’. Each time the contract reaches out to check a performance it costs Gas. This would add up quickly, so we have chosen a monthly manual process instead.

The employee can also apply a reduction in tokens from the fund if the asset made a loss.

At some point in the future the client will decide to crystallise his/her assets. The assets are transferred back to the Client Savings whereby the process ceases.

The idea here being to lock-in/secure some of the value of the savings.

**The Smart Contract’s Intention**

The aim of the Smart Contract is to execute a regular action, moving assets from one wallet to another, and apply a calculation to the amount. Monies are moving from Client Savings into 3 Wallets:

* Equity Wallet
* Property Wallet
* Bonds Wallet

These 3 Wallets are sub-wallets of the Life Assurance Company’s assets, named Insurer Reserve. So, as these increase in value, so too does the Insurer Reserve. Upon instruction from the client, an ‘Employee’ of the Life Assurer manually triggers the movement of the assets from the three wallets back to Client Savings, where the process comes to a halt.

**Design and requirements analysis process, and final design**

The objective is to create a Smart Contract whereby the following is taking place:

* Client adds an amount to his Client Savings (note that this is a recurring event).
* This amount is taken and put through the contract.
* This amount is split in 3 and appended to its own unique wallet – categorised by asset type.
* The assets from each Asset Wallet is sent to the ICO Contract to purchase CSV’s.
* A calculation is run monthly on each wallet to see should a positive/negative number of tokens be applied to the client’s savings.

**Implementation**  
3 contracts are required:

1. Fund Splitter Contract  
   - **Purpose**  
   Identify when the Client Savings have received monies.   
   Take these monies and divide the amount received in 3 equal amounts.   
   Transfer to each Wallet (Equity, Property, and Bonds).
2. ICO Contract  
   - **Purpose**  
   Convert ETH to new tokens – called ‘CSV Tokens’. These can be used to buy back ETH at a later date when a withdrawal is requested.  
   Looks at how much ETH it receives and sends 100 tokens back to the fund account for every 1 ETH received
3. Token Profit/Loss Contract

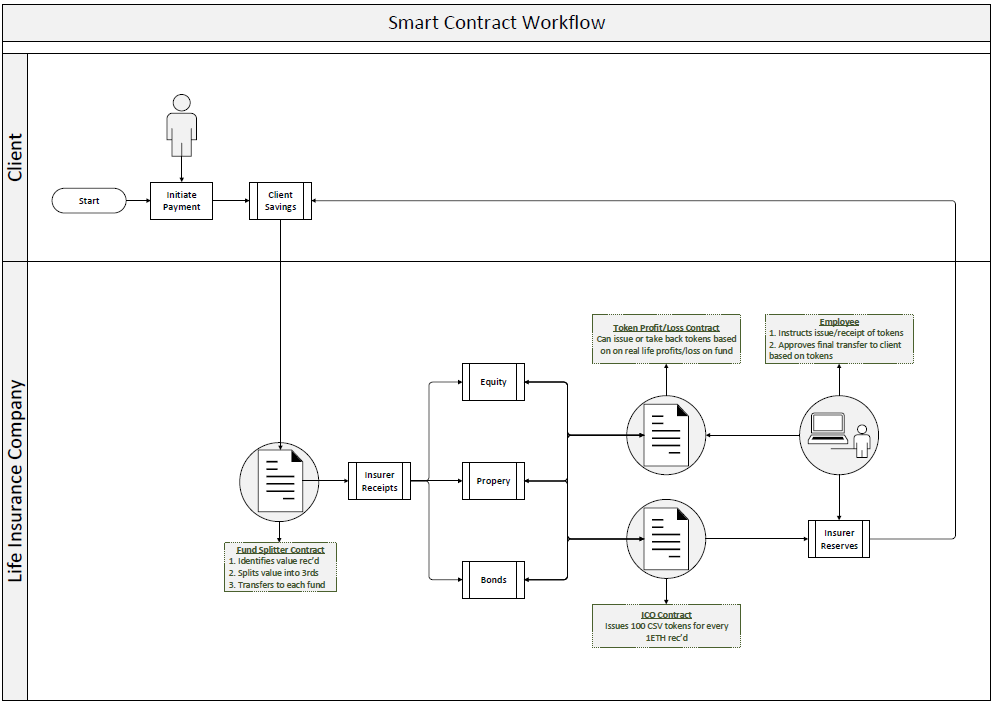
- **Purpose**  
To add or remove tokens based on profit/loss of the three asset types  
Evaluate the growth/loss applicable to each asset type

The process then repeats itself each time the client adds to his Client Savings.

When a withdrawal request is received an entity called ‘Employee’ manually instructs the contract to subtract the equivalent number of CSV Tokens. These are used to buy ETH which is sent back to the Client Savings.

**Appendix:**

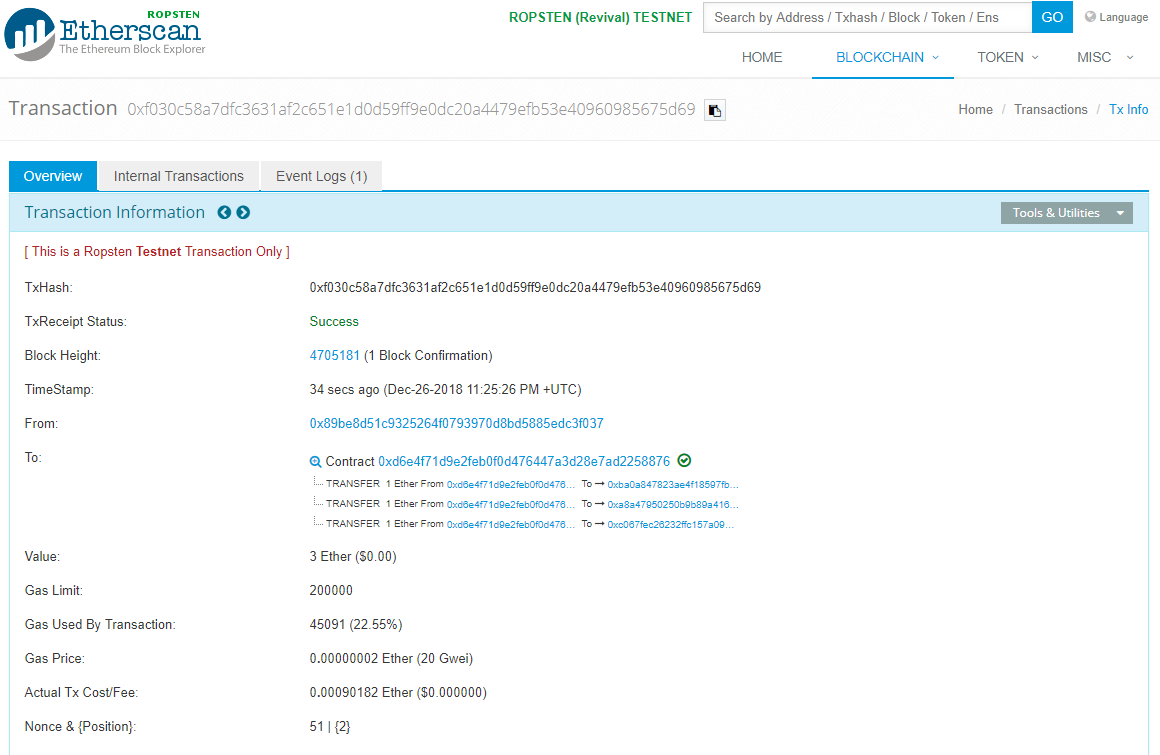
1. **Workflow:**



*Fig 1. Lifecyle of client savings*

1. **Screenshots of Live Process**
2. Client initiates transfer of 3 ETH from their personal wallet (‘Client Savings’) to a ‘Fund Splitter’ Contract, provided to them at inception. This smart contract automatically splits the incoming funds into three equal parts and transfers these amounts onwards to three fund accounts, pre-agreed with the client. Each client would need to be setup with their individual ‘Fund Splitter’ contract and Client Fund accounts at inception.

* **Client Savings:** 0x89be8d51c9325264f0793970D8bd5885eDc3f037
* **Fund Splitter Contract:** 0xD6e4F71d9E2FEB0f0D476447a3D28e7ad2258876
* **Insurer Receipts:** 0x20ef5163841644962ED3e2424d659E21Cb96Fd54
* **Equity Account:** 0xba0A847823ae4F18597Fb894dEb0B5e9e9325A13
* **Property Account:** 0xa8A47950250b9B89A416066952E09FB10baa9878
* **Bonds Account:** 0xC067FEC26232fFC157A09be5212290389cd04d7c

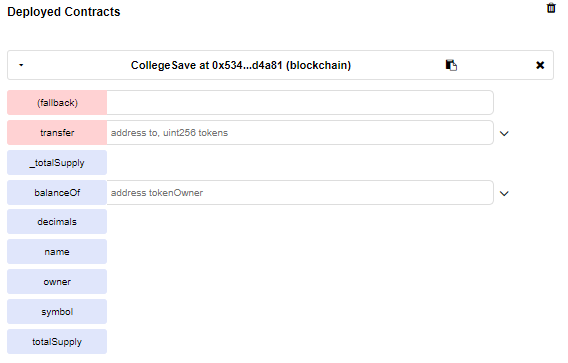


*Fig 2. Fund Splitter contract sending ETH received*

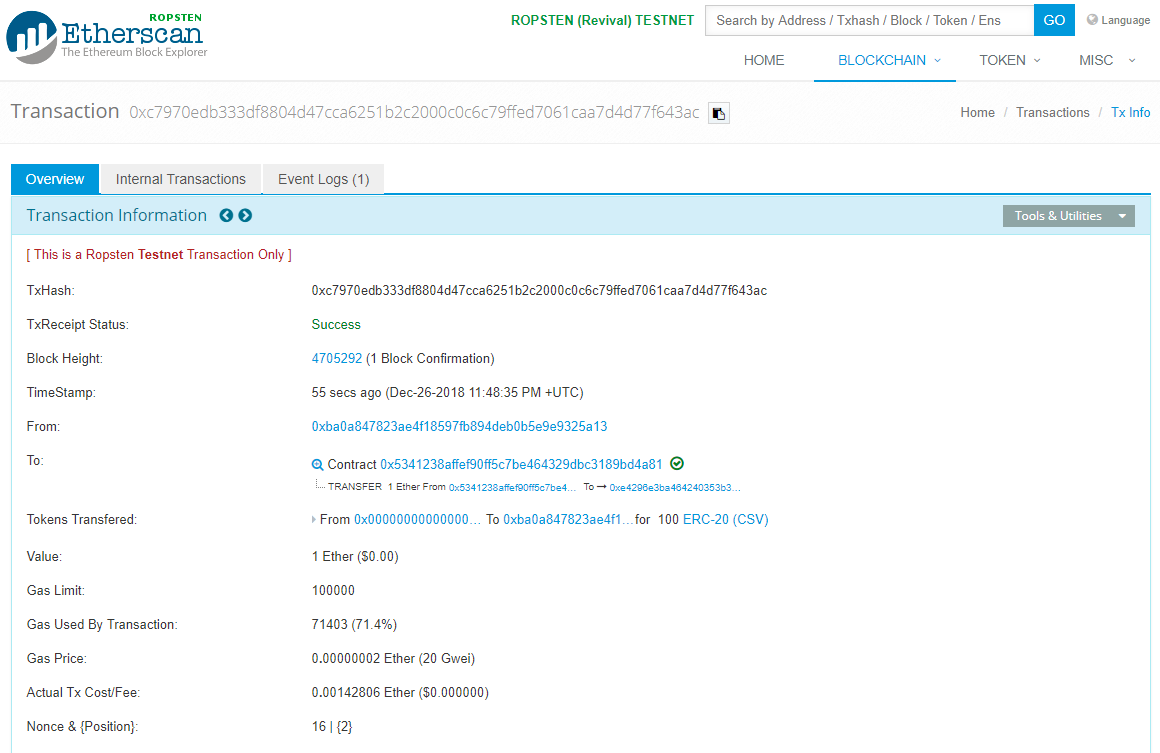
*to three client fund accounts*

An employee of the insurer is responsible for manually transferring any ETH received to the client fund account to the ‘Token Issuance’ contract. The contract will automatically issue 100 **College Save** tokens or ‘CSV’ in return for any ETH received. This operates like an Initial Coin Offering (ICO) and has an unlimited supply. The contract can issue CSV to any account it receives ETH and it can track the individual token balance of any address on the network using the ‘balanceOf’ function.

* 1. **CSV Token ICO Contract:** 0x5341238affef90ff5c7be464329dbc3189bd4a81
  2. **Insurer Reserves:** 0xe4296e3BA464240353b39ff520C5584Ed5925b9D



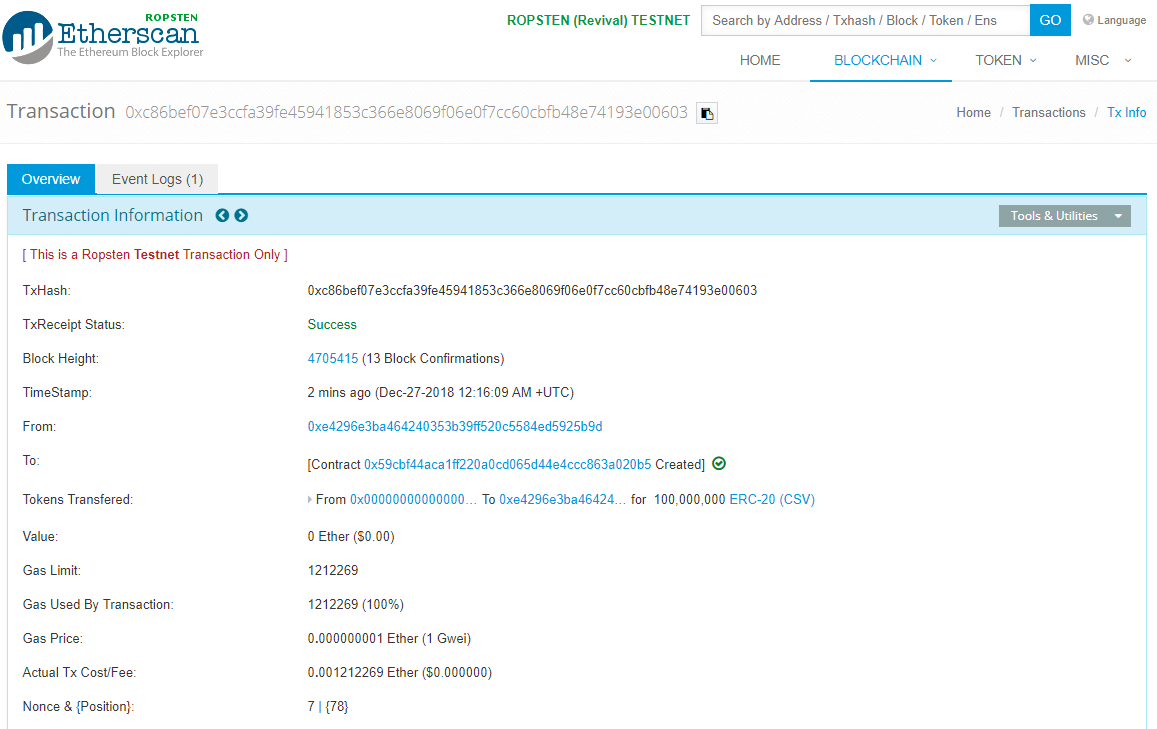
*Fig 3. College Save ICO contract deployed in Remix*



*Fig 2. College Save Contract issuing 100 tokens*

*in return for 1 ETH received from Equity Fund Account*

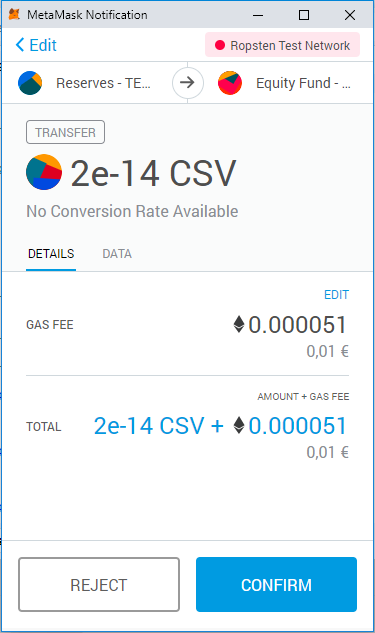
1. An employee tracks the ‘real life’ performance of each fund in the market. Each month, the employee sends each fund further CSV or calls for funds to be returned depending on whether the fund made a profit or loss in each month. This contract is much like the ICO contract, except this contract does not accept ETH and has been granted a limited supply.
2. **CSV Token P&L Contract:** 0x59cbf44aca1ff220a0cd065d44e4ccc863a020b5

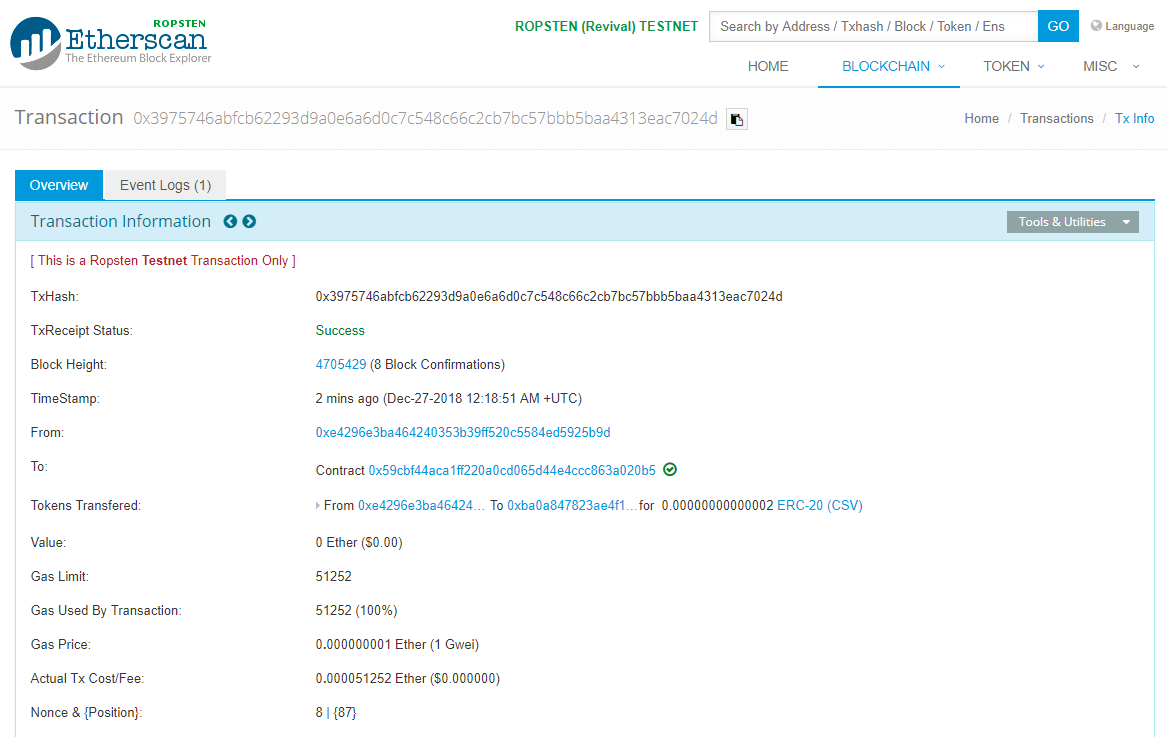


*Fig 4. CSV Token P&L Contract deployed to*

*Insurer Reserves Account*



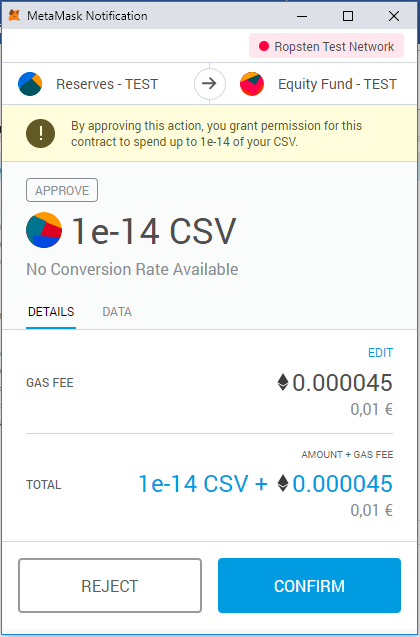




*Fig 5. Transferring 20,000 CSV tokens to the*

*Equity Fund account via the Token P&L Contract*



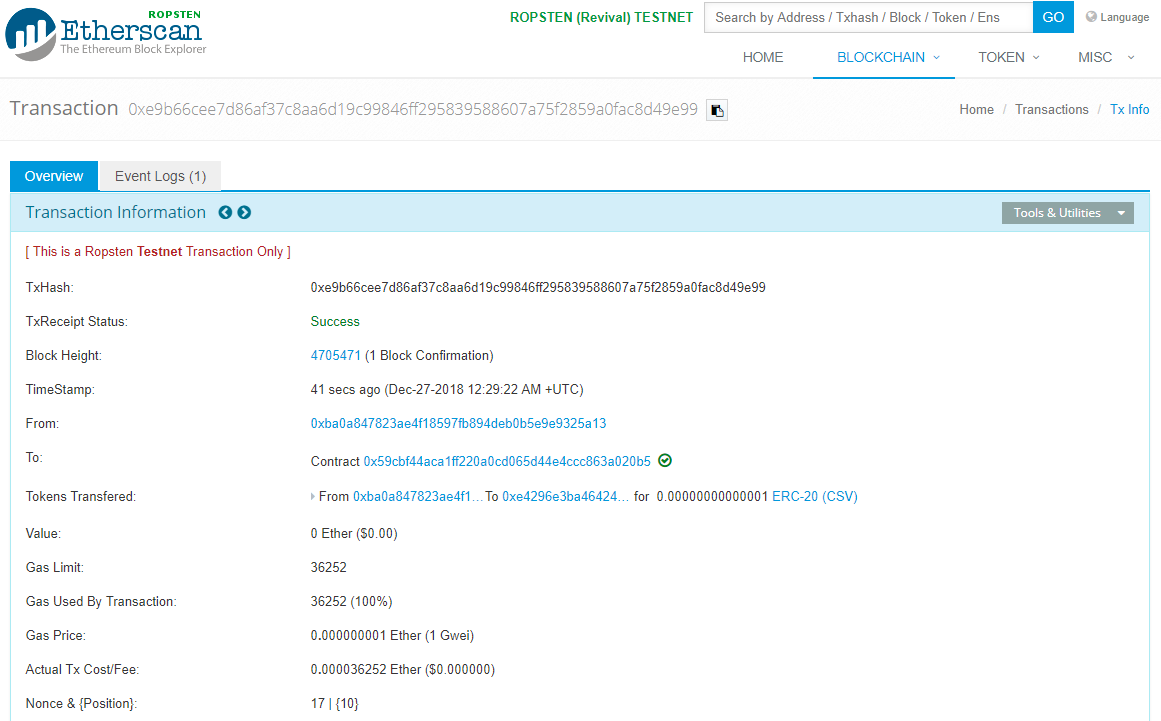












*Fig 7. Ten thousand CSV Tokens are approved for withdrawal*

*and subsequently withdrawn from the Client Equity account*



*Fig 8. Checking balance of CSV Tokens in Equity Account*

*Via smart contract in Remix*

**References:**

1. CSV Token ICO Contract Base Code:

<https://github.com/bitfwdcommunity/ICO-tutorial/blob/master/ico-contract.sol>

1. CSV Token P&L Contract Base Code:

<https://github.com/bitfwdcommunity/Issue-your-own-ERC20-token/blob/master/contracts/erc20_tutorial.sol>