Configuration Manual

MSc Research Project
Data Analytics

Guillaume Van Aelst
x17140552

School of Computing
National College of Ireland

Supervisor: Sean McNally

National College of Ireland
Project Submission Sheet – 2017/2018
School of Computing

Student Name:	Guillaume Van Aelst
Student ID:	x17140552
Programme:	Data Analytics
Year:	2018
Module:	MSc Research Project
Lecturer:	Sean McNally
Submission Due Date:	20/12/2018
Project Title:	Configuration Manual
Word Count:	2097 Words

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	18th December 2018

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Guillaume Van Aelst

x17140552

MSc Research Project in Data Analytics

19th December 2018

1. System Requirements/Hardware specifications:

All works regarding this project were performed on a Lenovo Thinkpad L380 Yoga:

- Windows 10 Enterprise Build 17134,
- Intel Core i5-8250U, 4 Cores – 8 Threads, 6MB Cache, 1.60 GHz base frequency - 3.40 GHz max turbo-frequency,
- Intel® UHD Graphics 620, 4GB Total Memory, 128MB VRAM
- 8GB RAM,
- 256GB SSD.

2. Languages/Frameworks/Packages/Libraries

Blockchain implementation:
Step by step source taken from: https://medium.freecodecamp.org/how-to-sync-an-ethereum-node-using-geth-and-ethereum-wallet-81423d42a583.
Installed programs:

- Geth 1.8.17 for Windows: command prompt window used to synchronise with the Ethereum Blockchain
- Ethereum Wallet 0.11.1 for Windows: User interface to interact with contracts and accounts.

Due to synchronisation issues and impossibility to get ETH with faucet on the Ropsten testnet, it was decided to use the Rinkeby testnet. Downloading the full node took around 6 hours for over 26GB of space stored under: C:\Users\gvanaelst\AppData\Roaming\Ethereum\testnet. This increased to 30GB by the end of this project on the 19/12/2018.

In order to synchronize the full node on the used computer, the following PowerShell command was used:

```
geth -testnet -rpc -rpcapi eth,web3,net,personal
```

3. How to deploy the application:

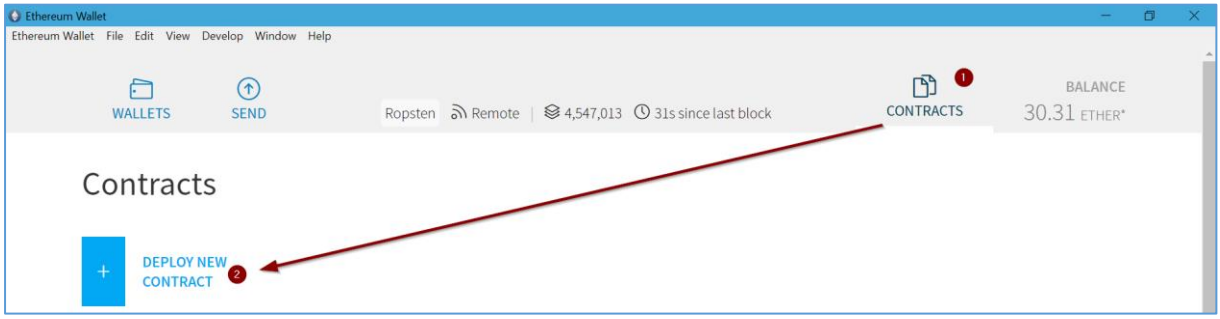
Two different contracts need to be deployed on the Ethereum Wallet prior to the voters being able to interact with the system, the Token contract and the Democracy contract.

TOKEN CONTRACT:

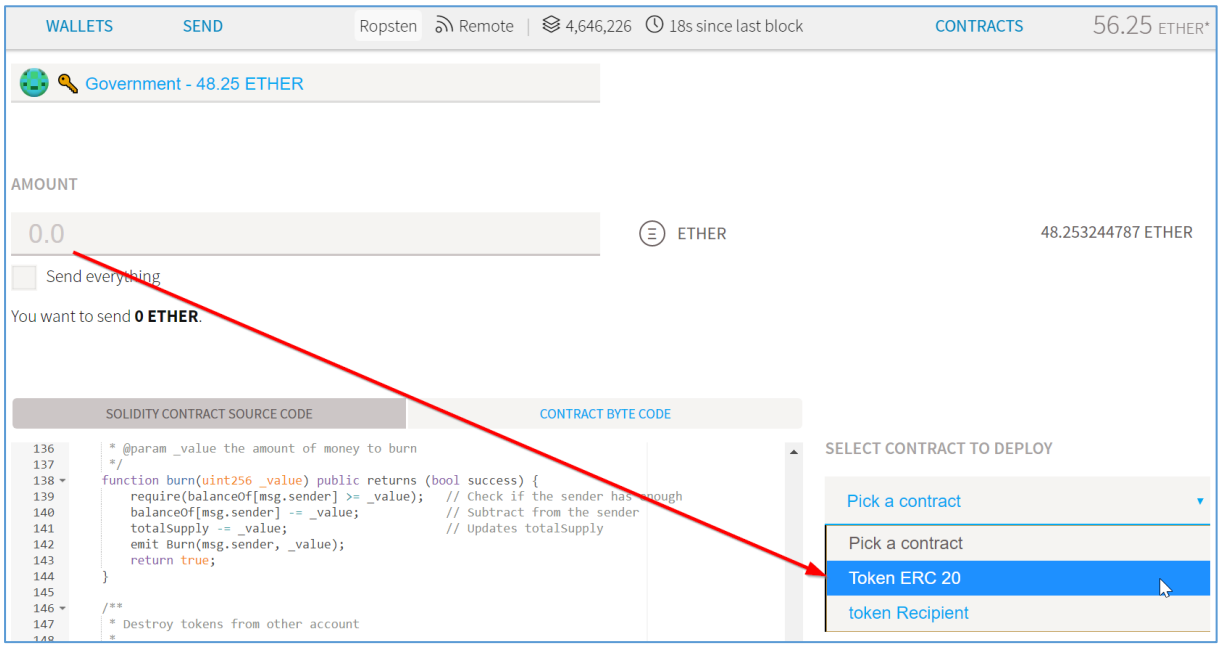
The Token Contract currently enables the users who own it to be allowed to interact with the Democracy Contract, hence being able to vote and make proposals. This allows the government to control who can interact or not with the Democracy Contract as only its owner can transfer it to other members.

A future works will be to enable these tokens to be directly used to vote on the Democracy Contract, replacing the "weight" number by an actual number of Tokens used/burned for the process of casting one's vote.

1) In the Ethereum Wallet, go to CONTRACTS > DEPLOY NEW CONTRACT:



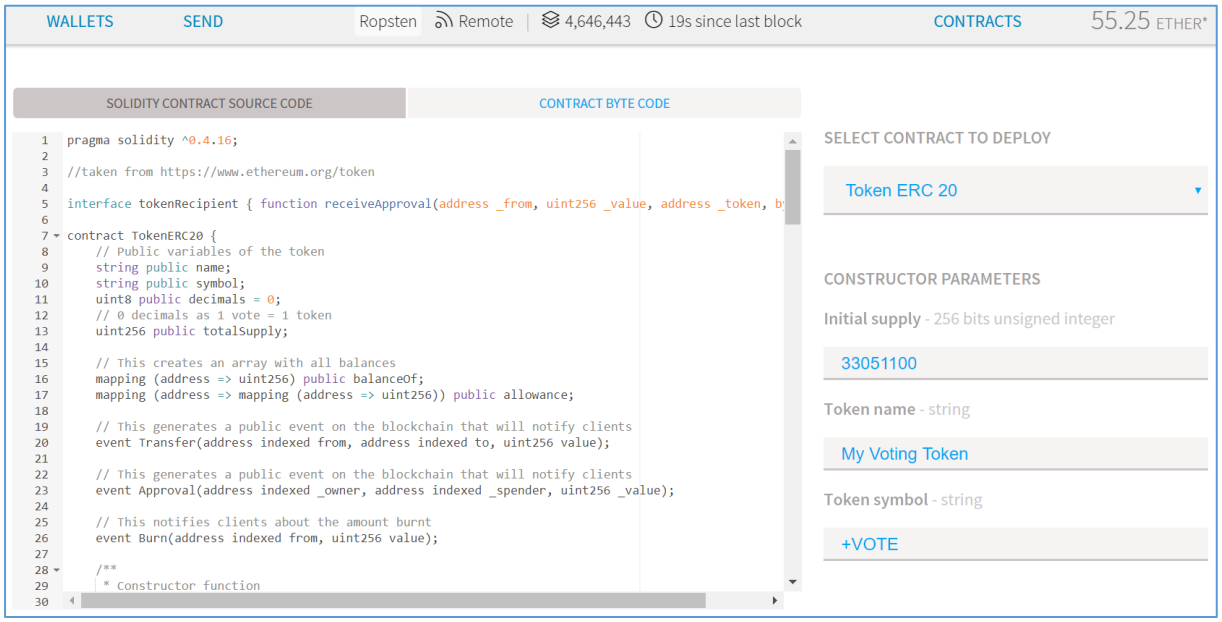
2) Copy the Solidity Code from the file: "1-Token.txt", and paste it into the "SOLIDITY CONTRACT SOURCE CODE", with no ETH amount to transfer. Then pick the "Token ERC 20" contract to deploy:



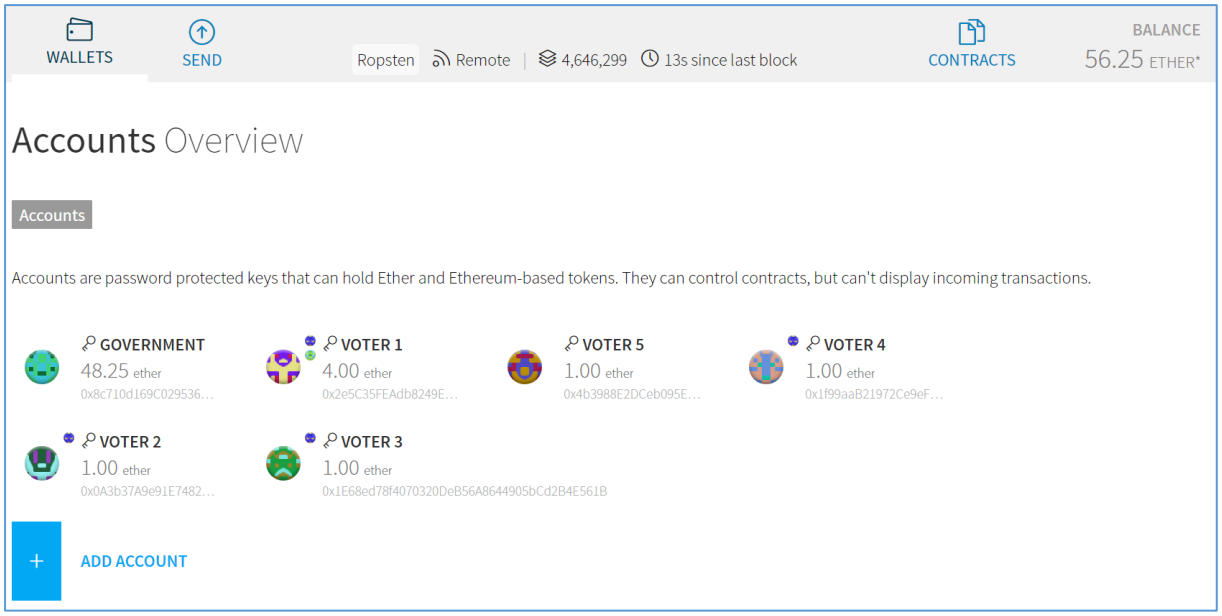
3) The CONSTRUCTOR PARAMETERS to be used in this contact before deploying it are:

a. Initial Supply: Any amount, but ideally the number of voting citizens, multiplied by 10 (as 10 token each to cast their votes per issue)
b. Token Name: Any value such as “My Voting Token”
c. Token Symbol: Any value such as “+VOTE”

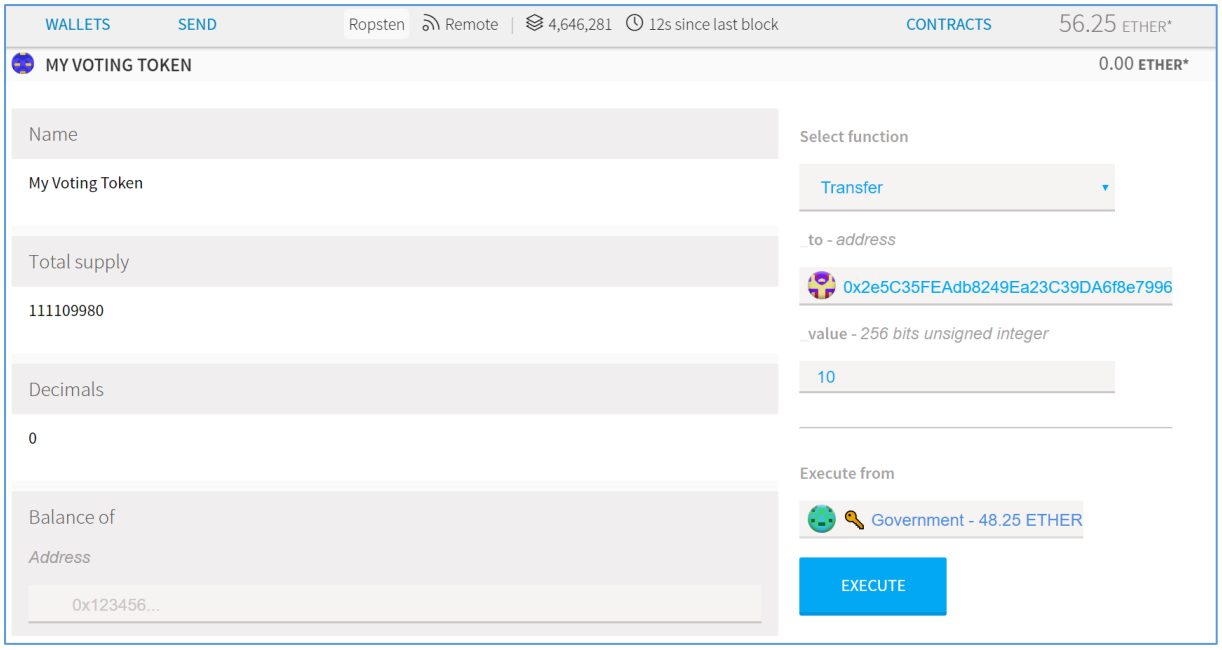
(By default, there are no decimal on this token as one vote equals one token.)



4) In order to carry some tests, it is advised to create multiple accounts as for example:



5) On the Token ADMIN PAGE, transfer the 10 tokens to each eligible voter:

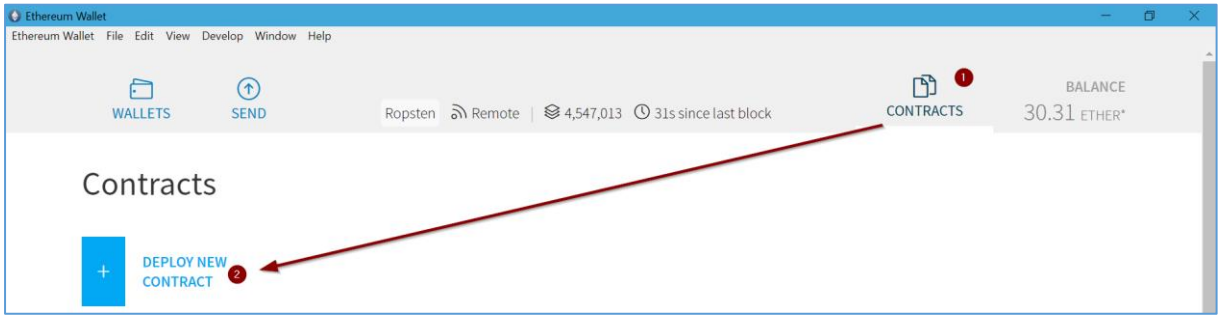


6) Those accounts owning this token will now be allowed to interact with the Democracy Contract described below.

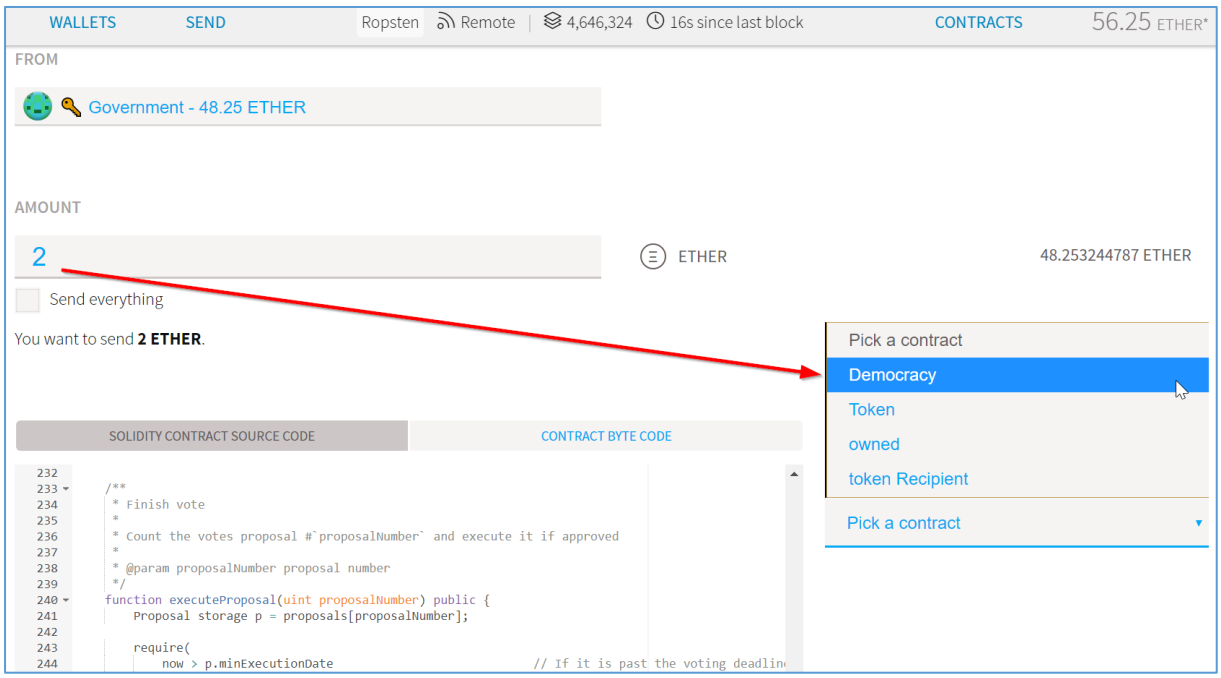
DEMOCRACY CONTRACT:

This contract contains all the rules that allow a user to make a proposal, vote and delegate their votes to a trusted individual. Initially, the contract is the be deployed by the "Government" in order to gather ideas from its citizens on how to tackle issues. In the future, the citizens themselves could be allowed to deploy a Democracy contract funded by the Government in order to gather ideas on how to tackle their own issues.

1) In the Ethereum Wallet, go to CONTRACTS > DEPLOY NEW CONTRACT:

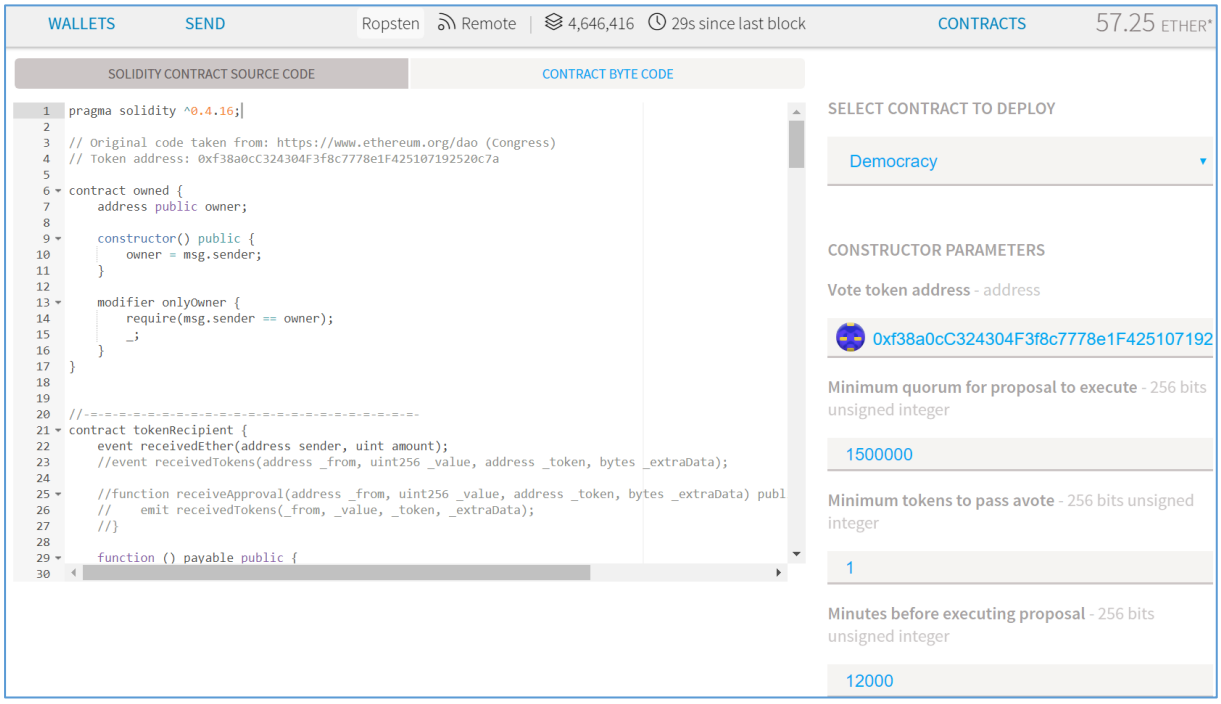


2) Copy the Solidity Code from the file: "2-DemocracyContract.txt", and paste it into the "SOLIDITY CONTRACT SOURCE CODE", with as any ETH amount to transfer on. (you can later on top-up the contract if needed). Then, pick the "Democracy" contract to deploy:
Note: the account FROM which you deploy the contract will be the Owner of the contract.



3) The CONSTRUCTOR PARAMETERS to be used in this contact before deploying it are:
   a. **Vote token address**: This is the address of the previously deployed Voting Token
   b. **Minimum quorum for proposal to execute**: This is the minimum number of voters before being able to execute the proposal. This could be set to at least half of the voting population in order to be sufficiently representative.
   c. **Minimum tokens to pass a vote**: Should be set to at least 1 and not over 10 to enable those with the tokens to be allowed to cast their votes.

d. **Minutes before executing proposal**: This is the number of minutes that need to pass before being allowed to execute the proposal. This allows for a proposal to be sufficiently discussed before being executed.



```
pragma solidity ^0.4.16;

// Original code taken from: https://www.ethereum.org/dao (Congress)
// Token address: 0xf38a0cC324304F3f8c7778e1F425107192520c7a

contract owned {
    address public owner;

    constructor() public {
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }
}


//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
contract tokenRecipient {
    event receivedEther(address sender, uint amount);
    //event receivedTokens(address _from, uint256 _value, address _token, bytes _extraData);

    //function receiveApproval(address _from, uint256 _value, address _token, bytes _extraData) publ
    //    emit receivedTokens(_from, _value, _token, _extraData);
    //}

    function () payable public {
```

SELECT CONTRACT TO DEPLOY

Democracy

CONSTRUCTOR PARAMETERS

Vote token address - address

0xf38a0cC324304F3f8c7778e1F425107192

Minimum quorum for proposal to execute - 256 bits unsigned integer

1500000

Minimum tokens to pass avote - 256 bits unsigned integer

1

Minutes before executing proposal - 256 bits unsigned integer

12000

4) Once the contract is deployed, it can be interacted with by the voters.

4. How to use the application:

The following represents the functional use of application directly on the Ethereum Wallet. As future works, a web interface will be made available to the voters with proper login and seamless interaction with the blockchain and its contracts.

On a pre-deployed contract (called "Tackle Homelessness in Dublin" for example), the voters have 4 functions to interact with on the Democracy Contract, namely:

1. New Proposal: 2 variables are required:
   a. **Eth amount**: (integer) amount suggested required to implement the proposal
   b. **Job description**: (string) free text that explains what the proposal is all about.

   If the proposal is passed after execution, the account from which the proposal was executed will receive aforementioned "Eth amount" in order to fund the proposal.

2. Vote: 4 variables are required:
   a. **Proposal number**: (integer) the ID number of the proposal, as multiple proposals can be made per deployed contract (first Proposal = 0)
   b. **Supports proposal:** (boolean) the tick box set to false by default (in Ethereum Wallet) that represents the support in the proposal or not.
   c. **Vote weight:** (integer) represents the importance (sentiment) that the voter gives to the proposal and can range from 0 to 10.
   d. **Justification text:** (string) free text allowing the voter to explain why he/she voted for or against the proposal and/or propose amendments to the proposal.

3. Delegate Vote: 4 variables are required:
   a. **Proposal number**: (integer) the ID number of the proposal, as multiple proposals can be made per deployed contract (first Proposal = 0)
   b. **Delegate address:** (address) This is the public key address of the (trusted) person whom the liquid votes will be assigned to.
   c. **Vote weight:** (integer) represents the importance (sentiment) that the voter gives to the proposal and can range from 0 to 10.
   d. **Justification text:** (string) free text allowing the voter to explain why he/she voted for or against the proposal and/or propose amendments to the proposal.

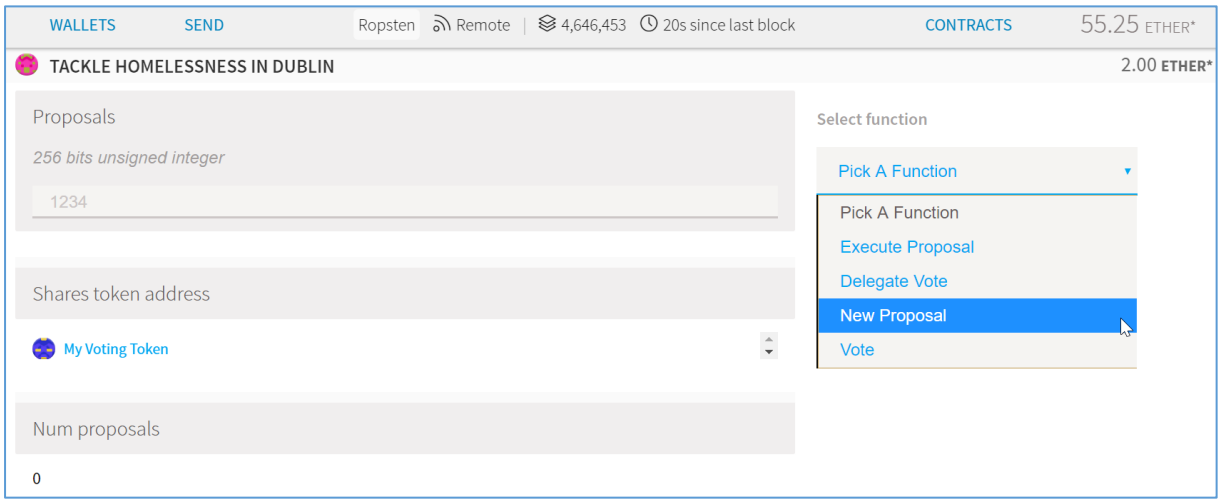
4. Execute proposal: 1 variable is required:
   a. **Proposal number:** (integer) the ID number of the proposal, as multiple proposals can be made per deployed contract (first Proposal = 0).

   This function can only be executed if the quorum and time conditions are met. Anyone can execute this function, owning Voting Tokens or not. Once executed, the proposal will be either passed or not, depending on the conditions that: the weight (sentiment) is positive and that the number of "Yea" votes is strictly higher than the "Nay". (Solidity Code: "p.currentResult >= 0 && p.currentCountYea > p.currentCountNay"). A future works will be to automatically execute a proposal once the predefined conditions are met.

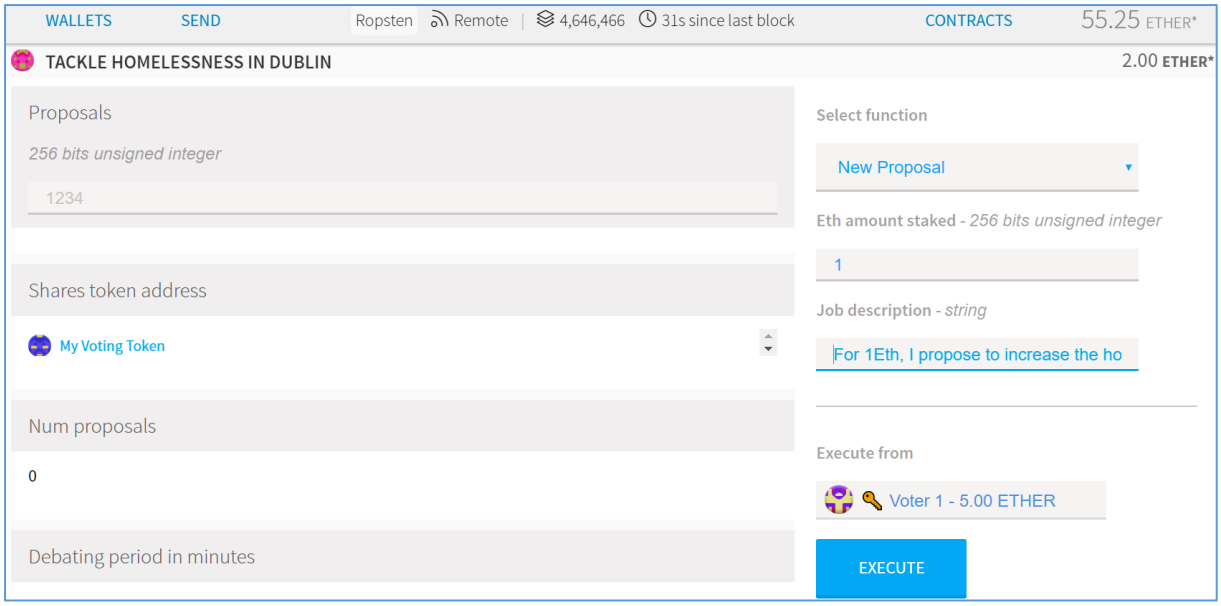
Illustrations of a user's interaction with the Democracy Contract:

New Proposal:

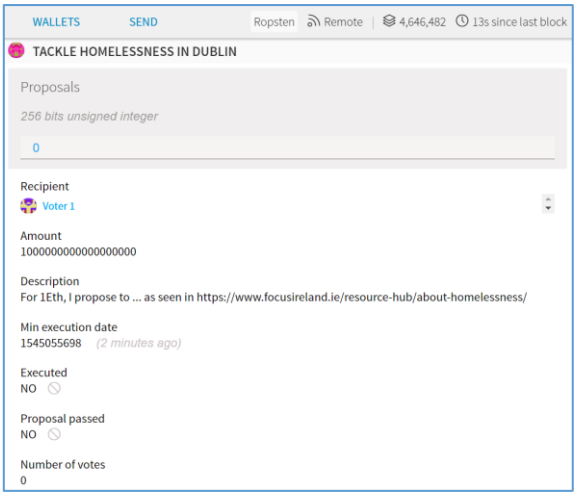
1) As explained above, once inside of Democracy Contract, the voter is given 4 functions to interact with the contract. If the contract has no proposal on it yet, the "Execute Proposal", "Delegate Vote" or "Vote" functions, logically, will not be able to execute.

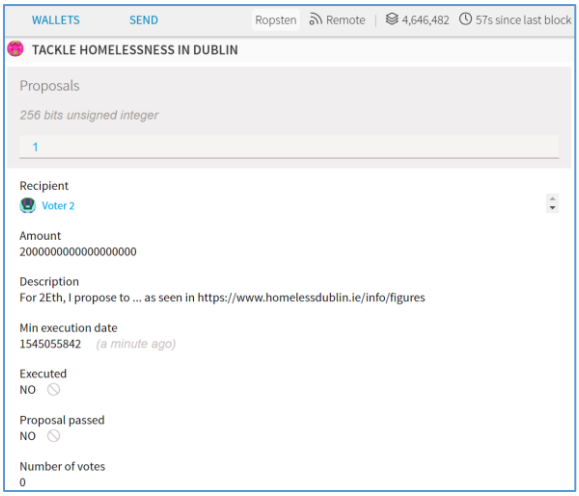


2) Illustrated below is the "Voter 1" making a proposal to Tackle Homelessness in Ireland. "Voter 1" needs 1 Eth in order to help fund the proposal and explains how in the Job description.



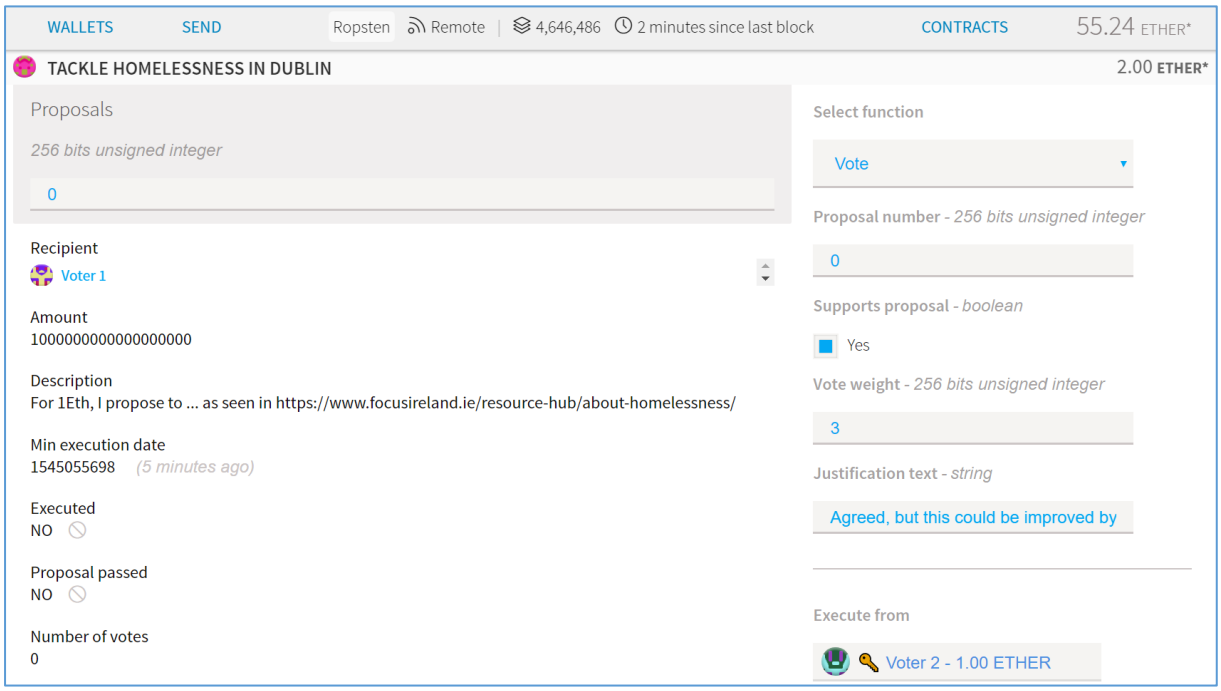
Below is the comparision of two different proposals made on the same contract by 2 different voters, having different Eth amount needed to fund their proposal:



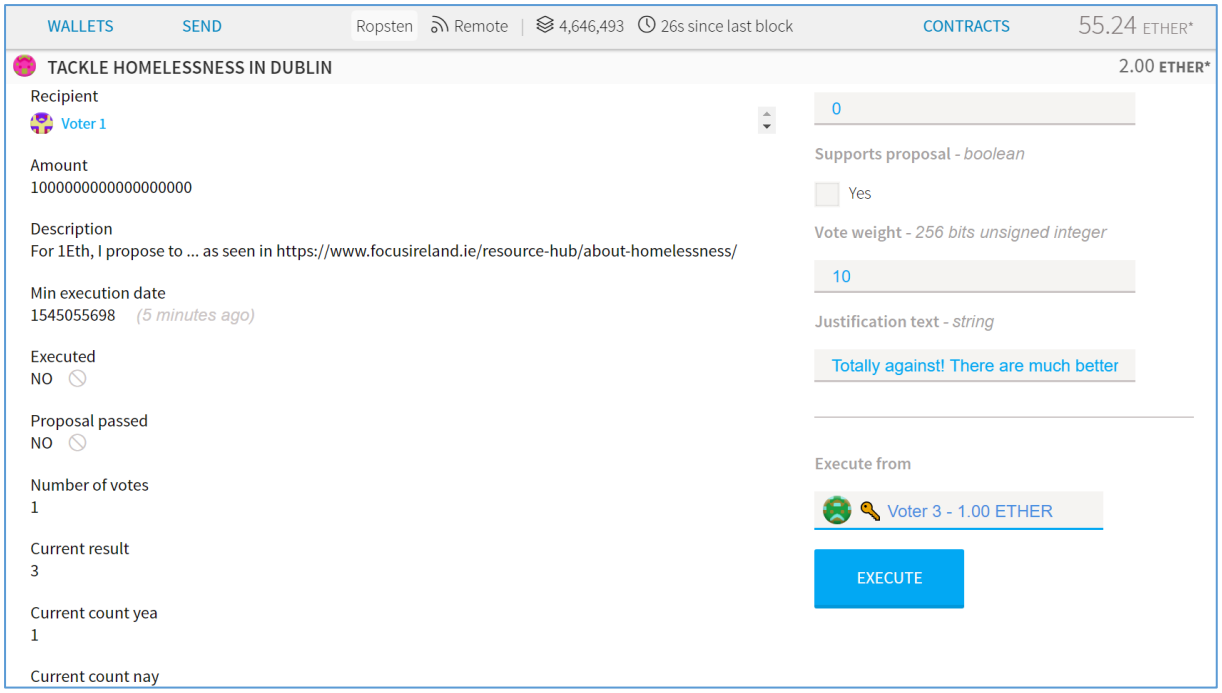


Vote:

3) Once a proposal is made on the Democracy Contract, a voter has the possibility of voting on it, as illusstrated below. The "Voter 2" is voting on the proposal 0 made by "Voter 1". "Voter 2" agrees with the proposal, but not strongly as having a weight of 3/10 and proposes some improvements in the Justification text.

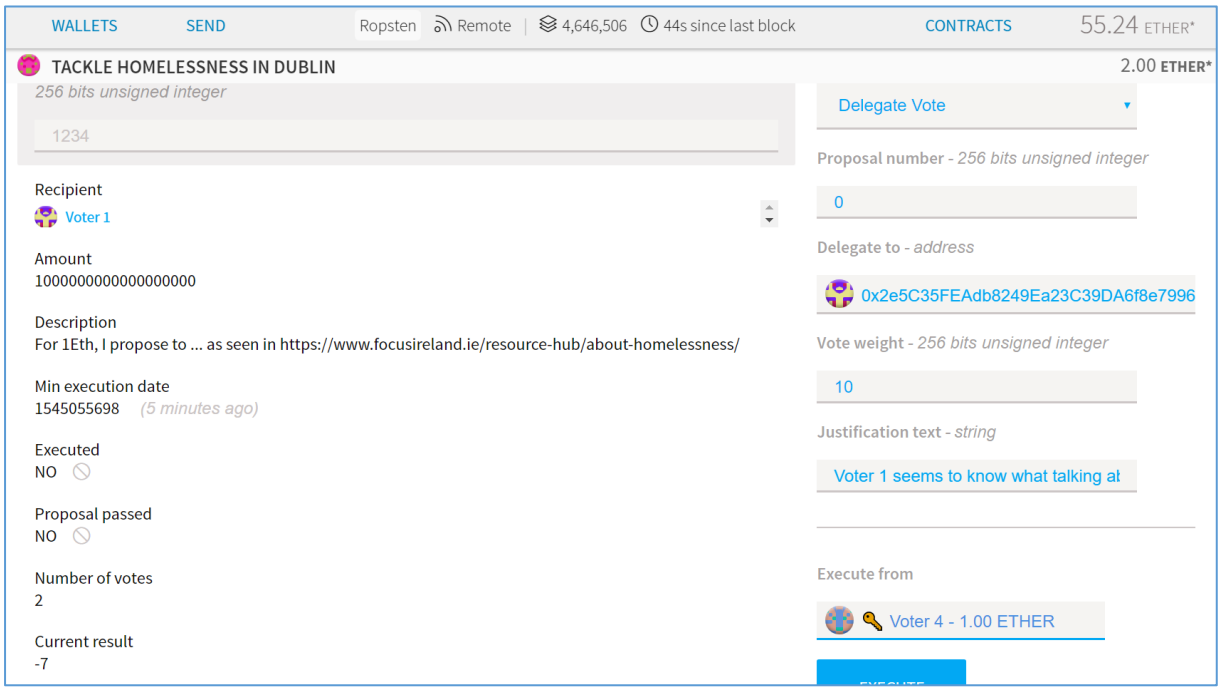


Below is another illustration of "Voter 3" voting against the Proposal 0 made by "Voter 1". The vote is strongly against as the weight is of 10/10. On the left side of the illustration, the vote of the "Voter 2" are being reflected on the contract:

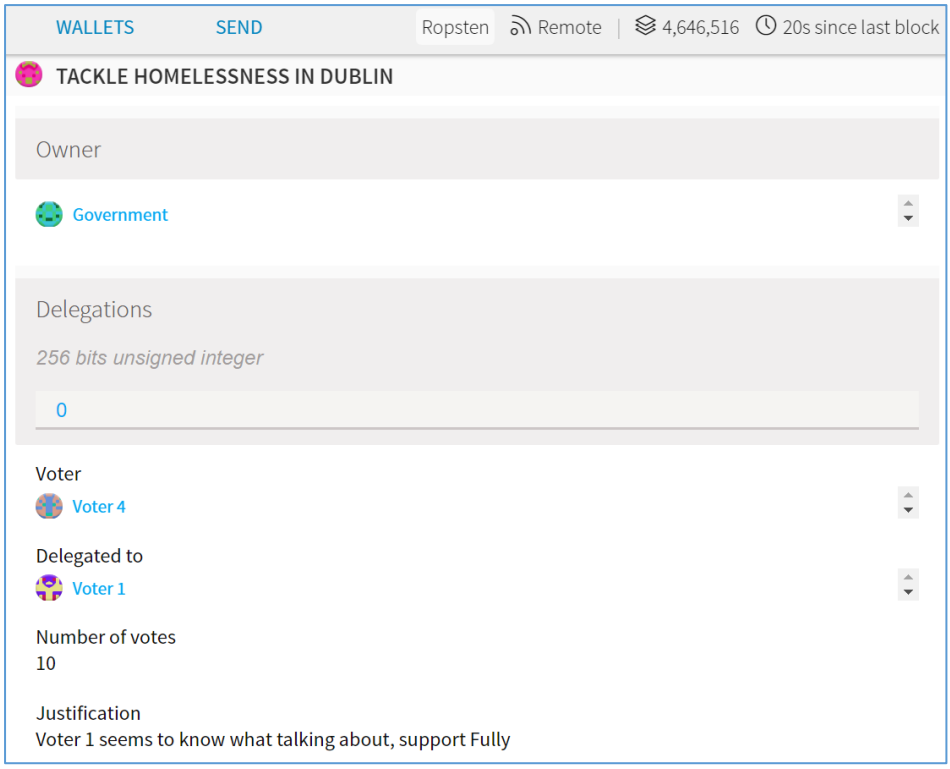


Delegate Vote:

4) The voters have the option of delegating their vote on issues which they do not feel strongly about for example. Illustrated below is the "Voter 4" delegating his vote to "Voter 1" by transferring his vote and his 10 vote weight. A voter can delegate his/her vote only once per proposal.



Below is the illustration of the aforementioned delegation recorded on the blockchain.



Following this delegation, if "Voter 1" has not already voted, when he does so, should he vote in favour of that proposal with a weight of 5 for example, his vote will automatically count for 2 votes in favour and adding a total weight of 15.

If "Voter 1" had already voted prior to the delegation, the delegated vote will automatically follow the decision of "Voter 1" and automatically increase the count of Yea or Nay as well as the weight.

Currently, this functionality is not fully functioning due to technical difficulties as, even though recorded on the blockchain, the delegate ("Voter 1" in this example) cannot vote with the delegated votes.

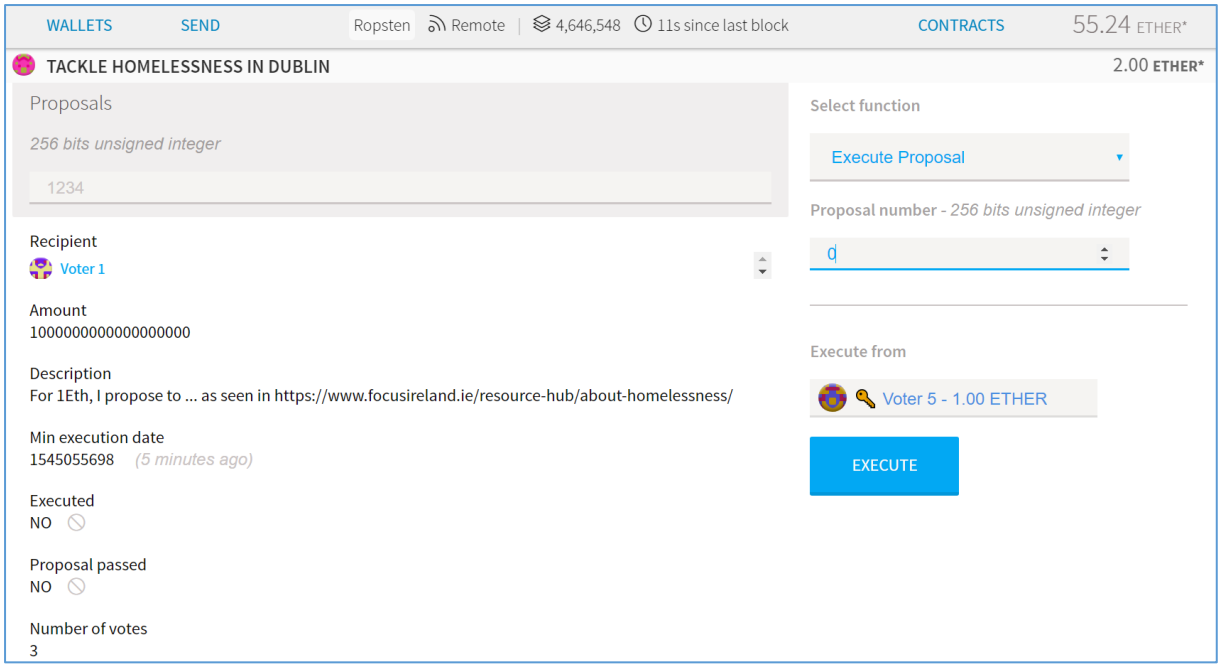
In the future, the web-interface will enable a voter to see how many delegated votes he/she owns before voting with a notification system alerting them when delegated votes are received.

<u>Execute Proposal:</u>

5) The function of Executing the Proposal checks that the 2 conditions of the minimum quorum and time before executing the proposal are met. If these conditions are not met, it will not be possible to execute the proposal. If these conditions are met, the proposal will be marked as Executed. In order for a Proposal to be "passed", it will need to fulfil both the following conditions:
   - Having strictly more votes in favour of the proposal (Yea) than not (Nay).
   - Having a positive sentiment (current result).

   For example, if more voters voted in favour of a proposal, but the sentiment of it is negative, the proposal will not pass. The same goes for a proposal that has more votes not in favour of it, even if the sentiment is positive, the proposal will not be passed.

Furthermore, as the proposal isn't currently set to be automatically executed as it would involve extensive work on the Ethereum Alarm Clock, the "Execute Proposal" function is programed to be executed by anyone even not owning any "Voting Tokens" as shown below ("Voter 5" not being an eligible voter):



Upon execution, this specific example can be Executed, but at the time of its execution, would not be set to Passed since the "sentiment" (Current Result) is negative (-7). This is a failed proposal and the "Recipient: Voter 1" will not receive the proposed 1 Eth.

WALLETS SEND Ropsten Remote | 4,646,559 24s since last block

TACKLE HOMELESSNESS IN DUBLIN

Recipient
Voter 1

Amount
1000000000000000000

Description
For 1Eth, I propose to ... as seen in https://www.focusireland.ie/resource-hub/about-homelessness/

Min execution date
1545055698 *(26 minutes ago)*

Executed
YES

Proposal passed
NO

Number of votes
3

Current result
-7

Current count yea
1

Current count nay

The following illustration is the second proposal that has received the right number of positive votes (2 Yea, 0 Nay) and has a positive sentiment (17 Current result).

WALLETS SEND Ropsten Remote | 4,646,563 42s since last block CONTRACTS 55.24 ETHER*

TACKLE HOMELESSNESS IN DUBLIN 2.00 ETHER*

Proposals
256 bits unsigned integer
1

Recipient
Voter 2

Amount
2000000000000000000

Description
For 2Eth, I propose to ... as seen in https://www.homelessdublin.ie/info/figures

Min execution date
1545055842 *(26 minutes ago)*

Executed
NO

Proposal passed
NO

Number of votes
0

Select function
Vote

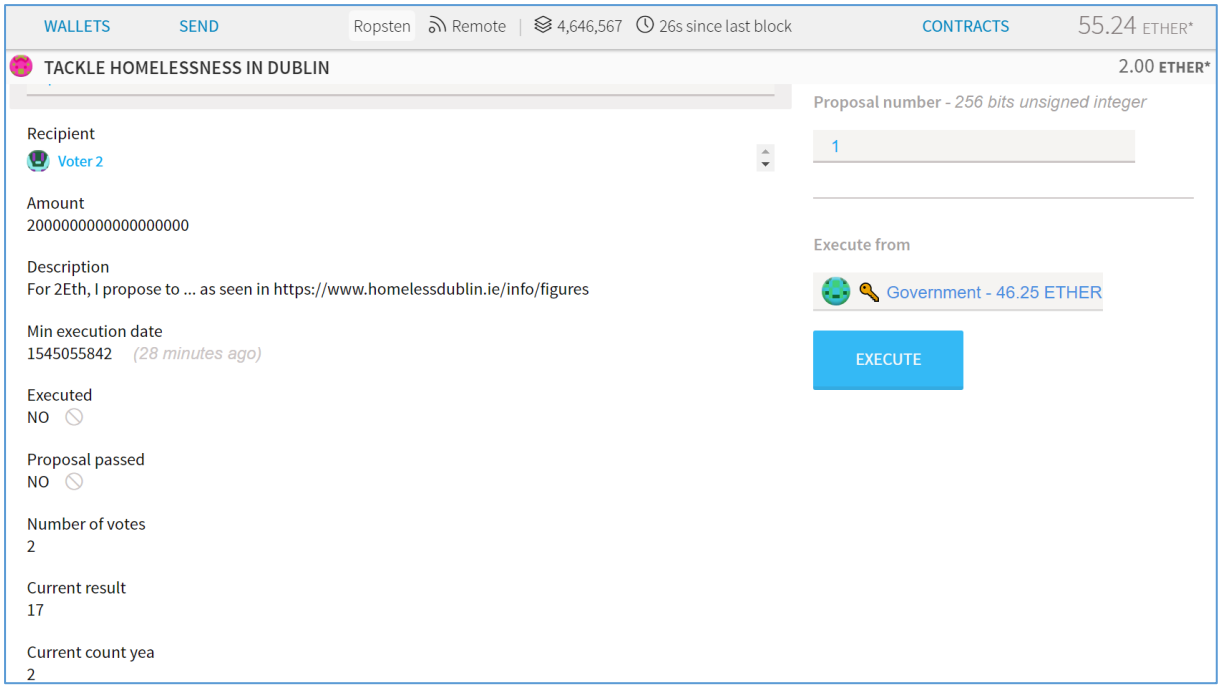
Proposal number - *256 bits unsigned integer*
1

Supports proposal - *boolean*
Yes

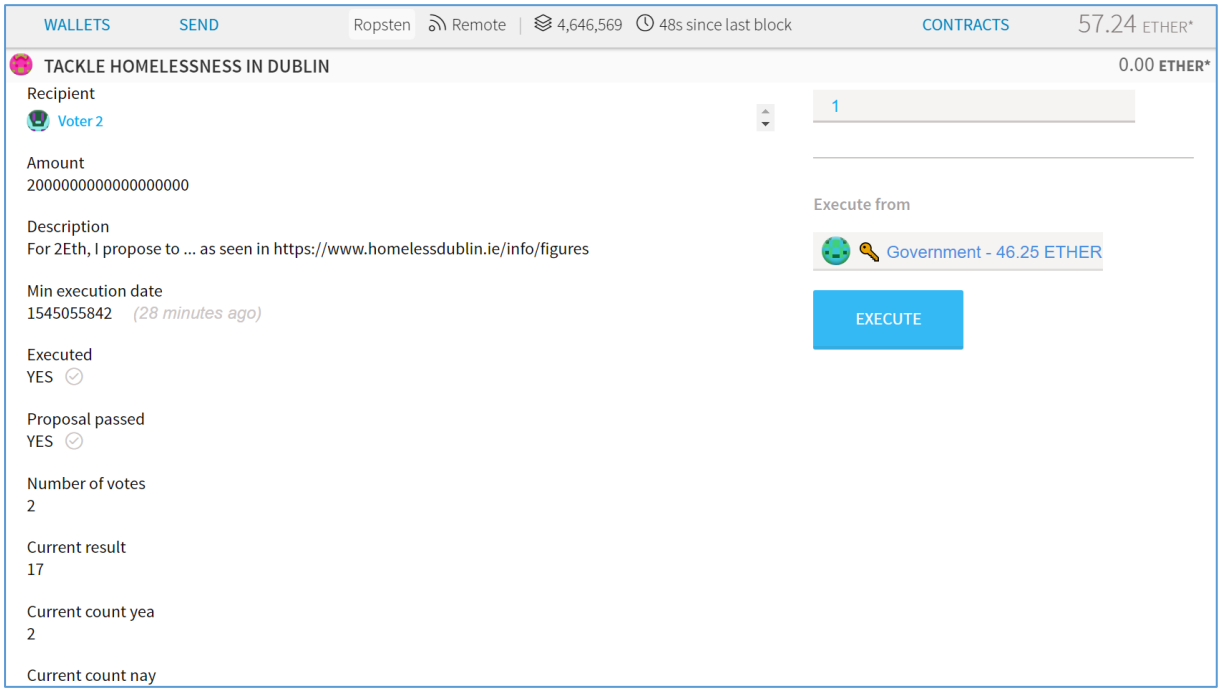
Vote weight - *256 bits unsigned integer*
10

Justification text - *string*
Great idea!

Execute from
Voter 1 - 5.00 ETHER



As illustrated below, when executed, the proposal is passed and the contract automatically transfers the Eth amount of 2 to the Recipient "Voter 2".



Even though the Democracy Contract is now empty, it can be topped-up ideally by the Government should the issue of Tackling Homelessness in Dublin not be sufficiently fixed by the passed proposal.

Illustrated below is the account of the "Voter 2" who is now 2 Eth "richer", having been funded that amount after a successful proposal. To also note that voter was entitled to vote as he owns 11 My Voting Token.

WALLETS SEND Ropsten Remote | 4,646,574 49s since last block CONTRACTS 57.24 ETHER*

Voter 2

0x0A3b37A9e91E7482442d83E830050d86a7a18Ed6

2.99904227 ETHER*

Transfer Ether & Tokens

Copy address

Show QR-Code

My Voting Token	11 +VOTE

NOTE

Accounts can't display incoming transactions, but can receive, hold and send Ether. To see incoming transactions create a wallet contract to store ether.

If your balance doesn't seem updated, make sure that you are in sync with the network.

Latest Transactions

5. Structure of code + Discuss some aspects:

The final Solidity code was inspired by the following sources:

- https://www.ethereum.org/dao (Congress)
- https://www.ethereum.org/token
- https://www.ethereum.org/dao (Shareholder Association)
- https://github.com/DemocracyEarth/ (DemocracyEarth)
- https://www.ethereum.org/dao (Liquid Democracy)

Due to the numerous contract deployments (over 50 of them), a regular "purge" had to be done in order not to get confused about which contract to interact with.
The command used on the Ethereum Wallet was the following:

```
//In Ethereum Wallet, type CTRL+ALT+i then on the Console tab, type:
CustomContracts.find().fetch()

//Then check the _id of the contract you want to remove
CustomContracts.remove('_id')
```