# **ElectorateTest and ElectorateLogViewer - Instructions**

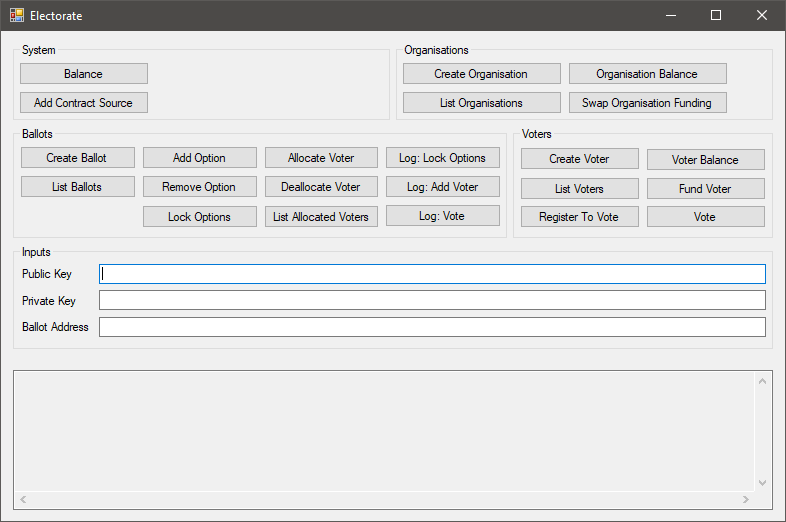
## **Configuration**

Both the test application and log viewer have hard coded database connection details, given their purpose is simply for testing and prototyping. As such if you wish to use them with your own database you'll need to alter the code to have the correct connection string.

These details can be changed in the following locations:

ElectorateTest: TestForm.cs, line 41  
ElectorateLogViewer: LogViewForm.cs, line 29

## **ElectorateTest**



### **Inputs**

Some of the functions have input requirements - the three text fields in the middle of the application serve this purpose (it's more convenient than having to re-enter the keys for every operation).

**Public Key:** Public key of the organisation to operate with.  
**Private Key:** Private key of the organisation to operate with.  
**Ballot Address:** Blockchain address of the ballot to operate with.

### **System**

**Balance:**Lists the balance (in ether) of the system's ethereum account.

**Add Contract Source:**Compiles and stores in the database the resulting bytecode of the default ballot source file (Ethereum/Ballot.sol). This is the bytecode that will be deployed to the blockchain when a ballot is created.

### **Organisation**

**Create Organisation:**Creates a new organisation, as well as a new ethereum account to represent this organisation on the blockchain.

The public key and private key will be listed in the output window, in the format: 'CREATE ORGANISATION: public\_key, private\_key, funding\_option'.

**List Organisations:**Lists all organisations which exist in the database, in the format: 'public\_key, private\_key, funding\_option'.

**Organisation Balance:***[arguments: public key, private key]*Lists the balance of the organisation.

**Swap Organisation Funding:***[arguments: public key, private key]*Swaps the funding option of the organisation between system funded and self funded:

* **System funded:** When the organisation's account lacks the funds to enact a blockchain operation, the funds will automatically be transferred from the system account.
* **Self funded:** The system never automatically funds the organisation's account - it's the responsibility of the organisation to keep it funded.

### **Ballots**

**Create Ballot***[arguments: public key, private key]*Creates a new ballot by deploying the ballot contract byte code to the blockchain, with the organisation's blockchain account as the owner of the contract. This means that only the organisation's account has the access to call functions that add or remove options, or that lock the options.

The byte code deployed is the byte code generated by the Add Contract Source step.

**List Ballots***[arguments: public key, private key]*Lists the ballots for the organisation, in the format: 'address, name, version, organisation\_address, end\_date, options\_locked'.

Also listed are the voting options of the ballot and the tallies for each.

**Add Option***[arguments: public key, private key, ballot address]*Prompts for and adds a voting option to the ballot.

**Remove Option***[arguments: public key, private key, ballot address]*Prompts for and removes a voting option from the ballot.

**Lock Options***[arguments: public key, private key, ballot address]*Locks the voting options for the ballot. This disallows adding or removing and further options, and is a required step before voting may occur.

**Allocate Voter***[arguments: public key, private key, ballot address]*Allocates a voter to the ballot, specified by the ID of the voter. Allocation means that the voter is now eligible to register to vote in the ballot - this is how the organisation can control who will be able to take part.

**Deallocate Voter***[arguments: public key, private key, ballot address]*Deallocates a voter from the ballot, specified by the ID of the voter. If the voter in question has already registered to vote then this deallocation is essentially meaningless, as registrations cannot be revoked.

**List Allocated Voters***[arguments: public key, private key, ballot address]*Lists the voters allocated to the ballot, in the format: 'voter\_id, voter\_name'.

**Log: Lock Options:***[arguments: public key, private key]*Lists the blockchain logs of all instances of locking the options of a ballot, for this organisation.

**Log: Add Voter:***[arguments: public key, private key]*Lists the blockchain logs of all instances of a voter registering to vote with a ballot, for this organisation.

**Log: Vote:***[arguments: public key, private key]*Lists the blockchain logs of all instances of a voter voting in a ballot, for this organisation.

### **Example**

* Run ElectorateTest.
* Click **System > Balance**, and check that the balance is at least 1.0 before proceeding.
* If you haven’t previously, press **Add Contract Source**.
  + If you see a database error, this step has already been run – you’re fine to proceed.
* Click **Create Organisation**. You should see output like the following:
  + C*REATE ORGANISATION: '04EE29402C88F676F1969AF0796171952C284BAE2F39458591037601B2AEEF77E1D8C3F9137C86591E2912930B1EEE678B45428929182A43ACC967837329E0D1A1', '141A12FF0C42977FB2311E3006CF9C23C173C9A020754139F56E5FBC95C337DD', 'SystemFunded'*
  + The first item is the public key, the second the private key. In this case:
    - **Private Key =** 04EE29402C88F676F1969AF0796171952C284BAE2F39458591037601B2AEEF77E1D8C3F9137C86591E2912930B1EEE678B45428929182A43ACC967837329E0D1A1
    - **Public Key =**141A12FF0C42977FB2311E3006CF9C23C173C9A020754139F56E5FBC95C337DD
* Enter the public and private keys from the above into the **Public Key** and **Private Key** fields of the **Inputs** section (yours will be different to the example keys above).
* Click **Create Ballot**. You should see output like the following:
  + CREATE BALLOT: '0x20a88395b1f71e4e0f57644032b68b25a7bf95db'
  + The text here is the ballot address: 0x20a88395b1f71e4e0f57644032b68b25a7bf95db
* Enter the ballot address from the above into the **Ballot Address** field of the **Inputs** section (yours will be different to the example above).
* Click **Add Option**, and then enter a name for the voting option.
* Click **Add Option** again, and enter a different name for the second voting option.
* To confirm that the options were added, press **List Ballots** – you should see the ballot and the options you just added to it listed, similar to the following:
  + LIST BALLOTS:   
    '0x20a88395b1f71e4e0f57644032b68b25a7bf95db', 'Test Ballot', '0.6', '0x45b16c260bb9e9d761671500771c07785bf04a60', '25/11/2018 12:36:50 PM', 'False'  
     1. Alpha: 0  
     2. Beta: 0
* Click **Lock Options**.
* Click **Create Voter**. You should see output like the following:
  + CREATE VOTER: 1017, 'Voter #697479773'
  + The first number is the voter ID – you will need this for the following functions.
* Click **Allocate Voter** and enter the voter ID.
* Click **Register To Vote** and enter the voter ID.
* Click **Vote** and enter the voter ID. Then select an option to vote for by entering the number of the option.
* Click **List Ballots** to see a list of the options and their tallies for the ballot – you should now see one vote next to the option you voted for.

## **ElectorateLogViewer**

The log viewer lists all logs (lock options, add voter, vote) for an organisation.

**Usage:** Enter the organisation's public and private keys, and then press refresh logs.