# ShareToken test specification (version-1)

## 1. Solidity files

The implementation is uploaded on this github: https://bitbucket.org/trung8x/smartcontract-sharetoken/src/master/

Please download or git clone into some directory.

The Solidity contract files are stored in the folder “**contracts**”.

3 main contract files:

* WhiteListManager.sol
* ShareToken.sol
* MainSale.sol

Other helper files:

* ERC20Interface.sol
* ERC20Token.sol
* Migrations.sol
* oraclizeAPI\_0.5.sol
* Owned.sol
* SafeMath.sol

## 2. Deployment procedure

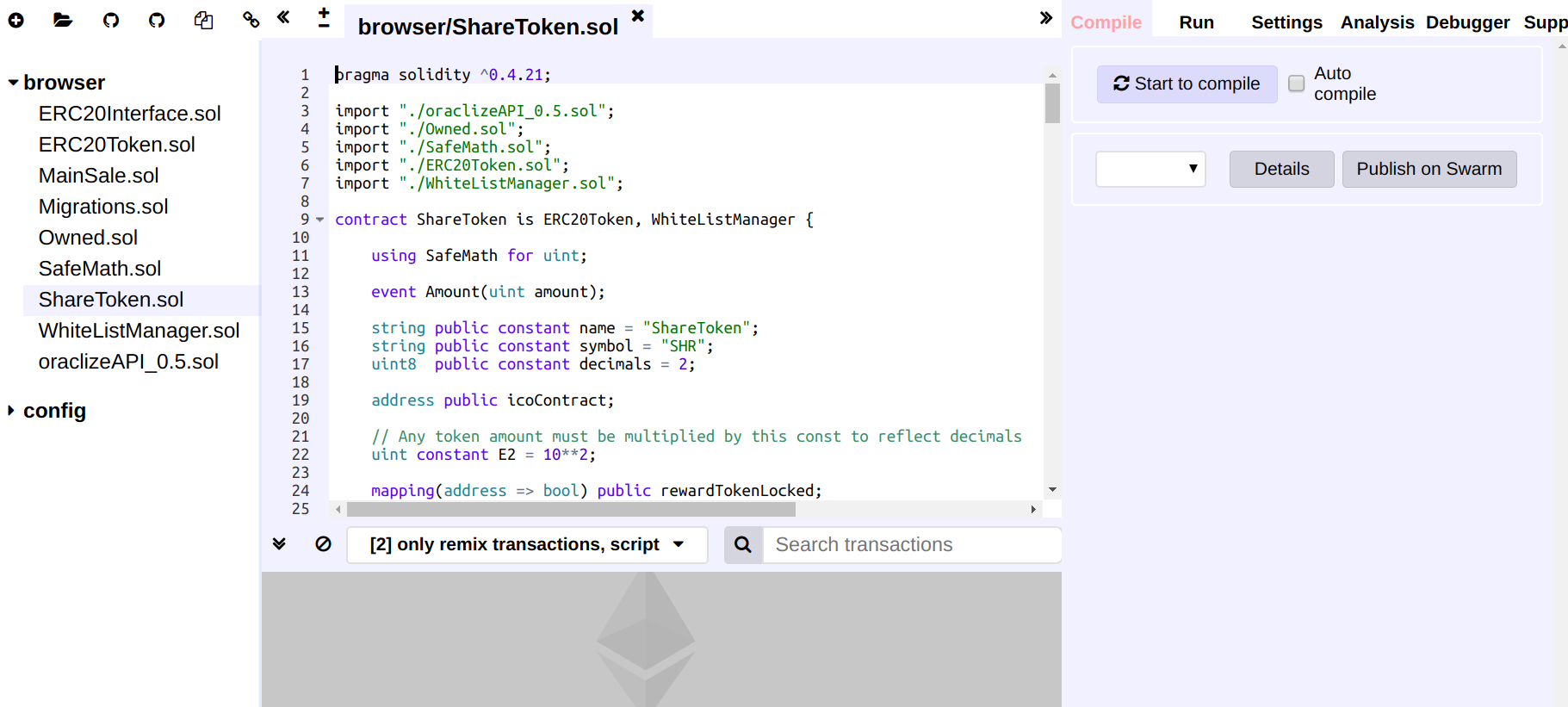
### 2.1. Tools

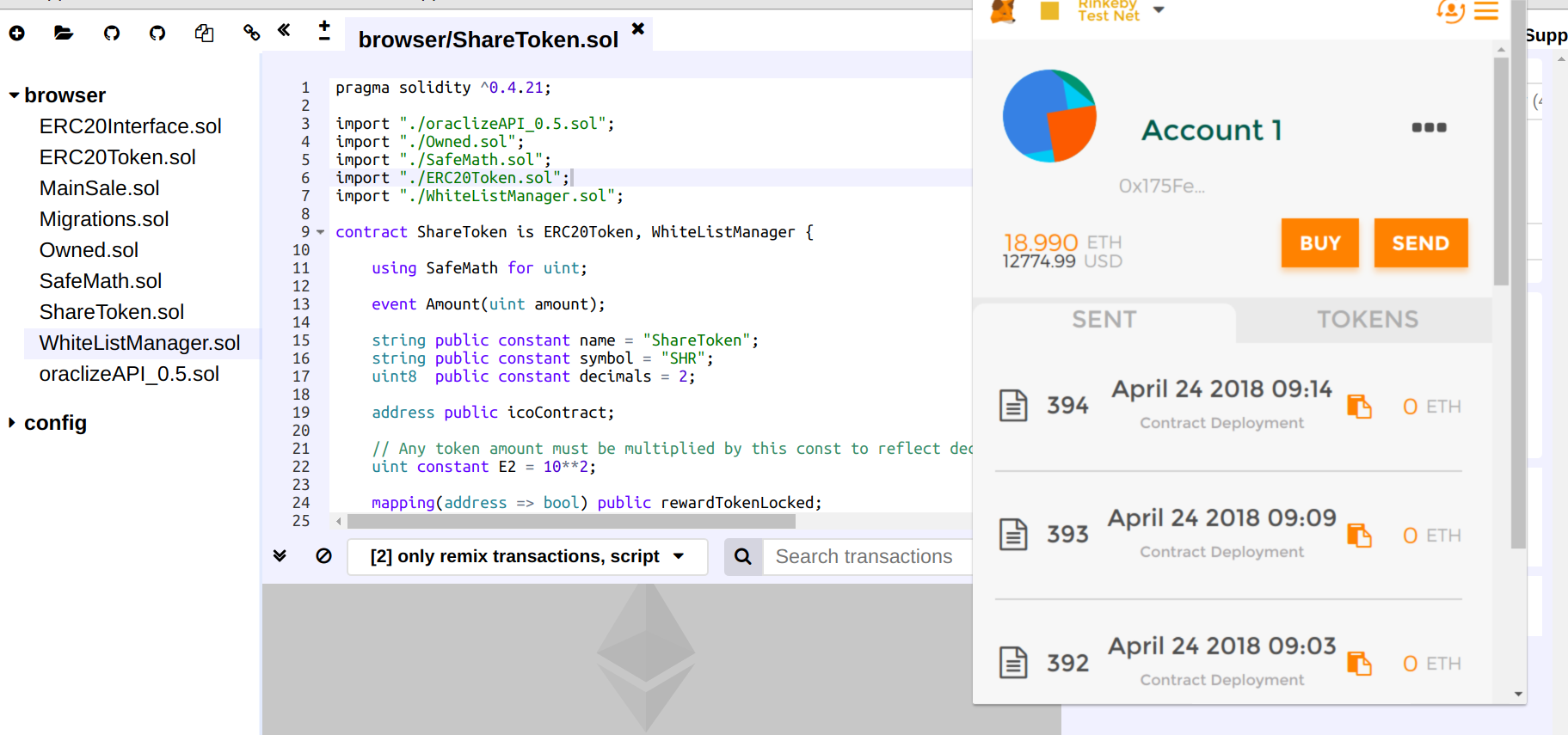
* Web browser (better with Chrome)
* Use web-based **remix** tool (<https://remix.ethereum.org/>) for compiling and deploying the contracts visually.
* Install **Metamask** as Chrome plugin for connecting to Ethereum testnet (Rinkeby) or mainnet
* Have enough ETH in the account used as owner address for deploying the contracts.

Open remix tool with some web browser (e.g. Chrome) and upload all the contract files (mentioned above) as shown in the below screenshot.

Multiple contract files can be uploaded by clicking the “Folder” button and then select as many files as needed.

Metamask plugin is also shown below.





### 2.2. Compile and deploy with Remix

**Three** contract files (ShareToken.sol, MainSale.sol and WhiteListManager.sol) need to be compiled and deployed.

The below procedure is described for ShareToken.sol and will be the same for the MainSale.sol and WhiteListManager.sol.

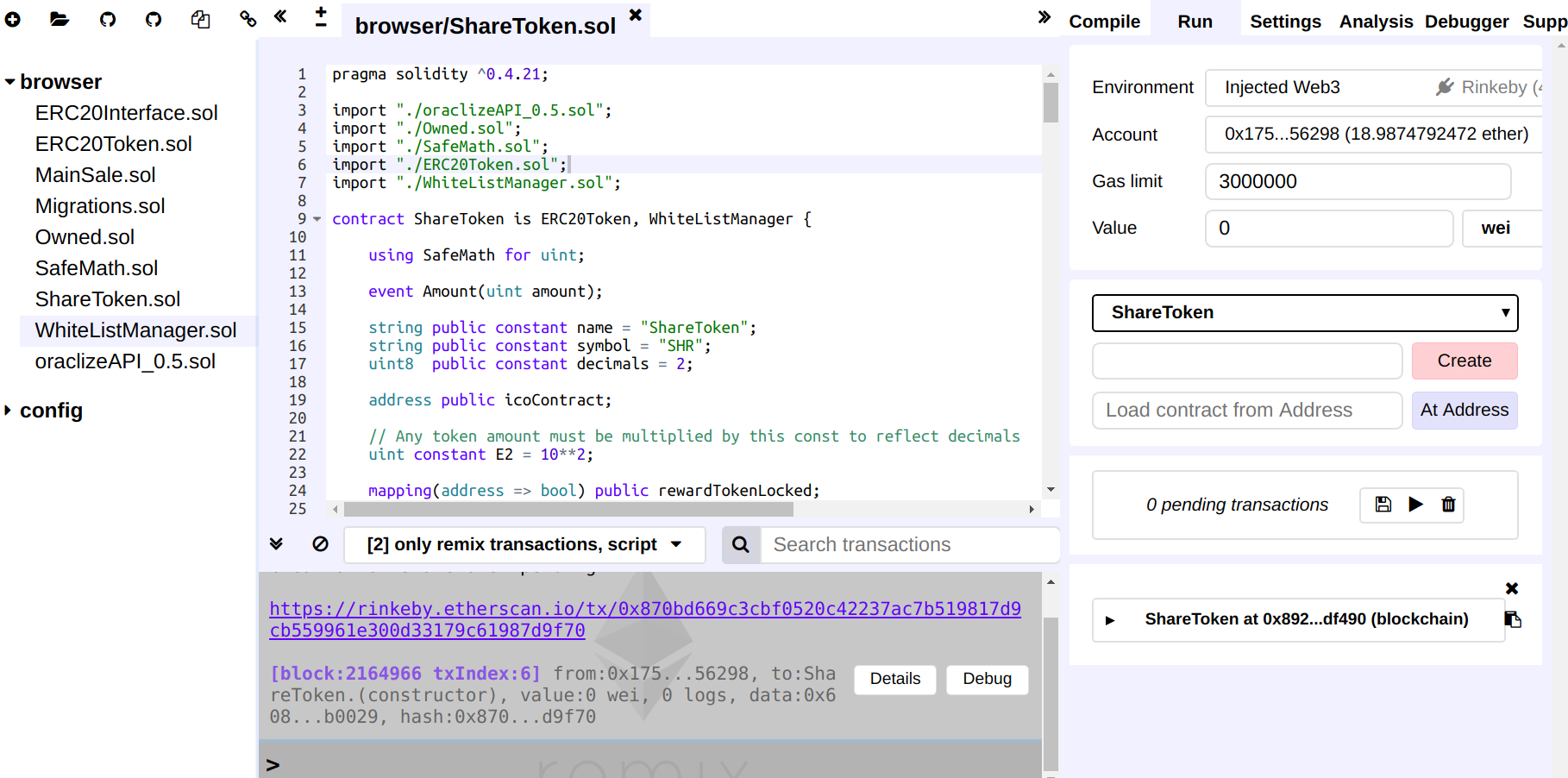
Firstly, click to open the file “ShareToken.sol”.

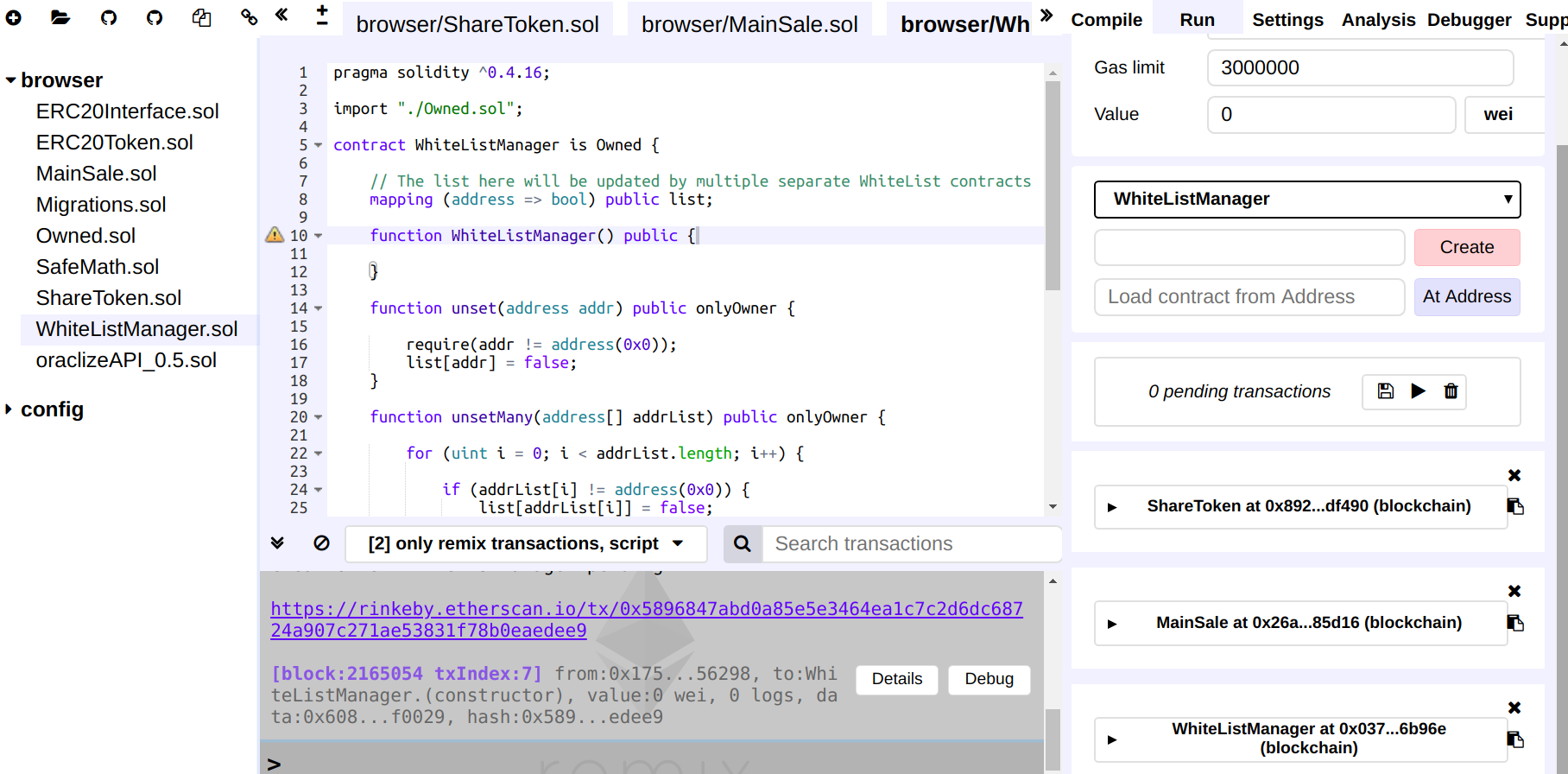
Secondly, click the tab “Compile” and then click button “Start to compile”.

(Lots of warning will come up, but no worries)

Thirdly, click the tab “Run”:

* Set the input field “ Environment” to “Injected Web3”
* After that, check if the input field “Account” shows the correct address. If not, simply change the “Environment” and then set it back to “Injected Web3”. This is because Metamask sometimes is not reflected.
* In the drop-down list, select the “ShareToken”
* Click the button “Create”.
* Be aware of the notification (sometimes poped up but sometimes not) of Metamask and click the button “Submit” for submitting the transaction of contract deployment.





### 2.3. Compile and deploy with **NodeJS scripts**

The instruction is described in the file “**README.md**”

## 3. Presale data tranfer

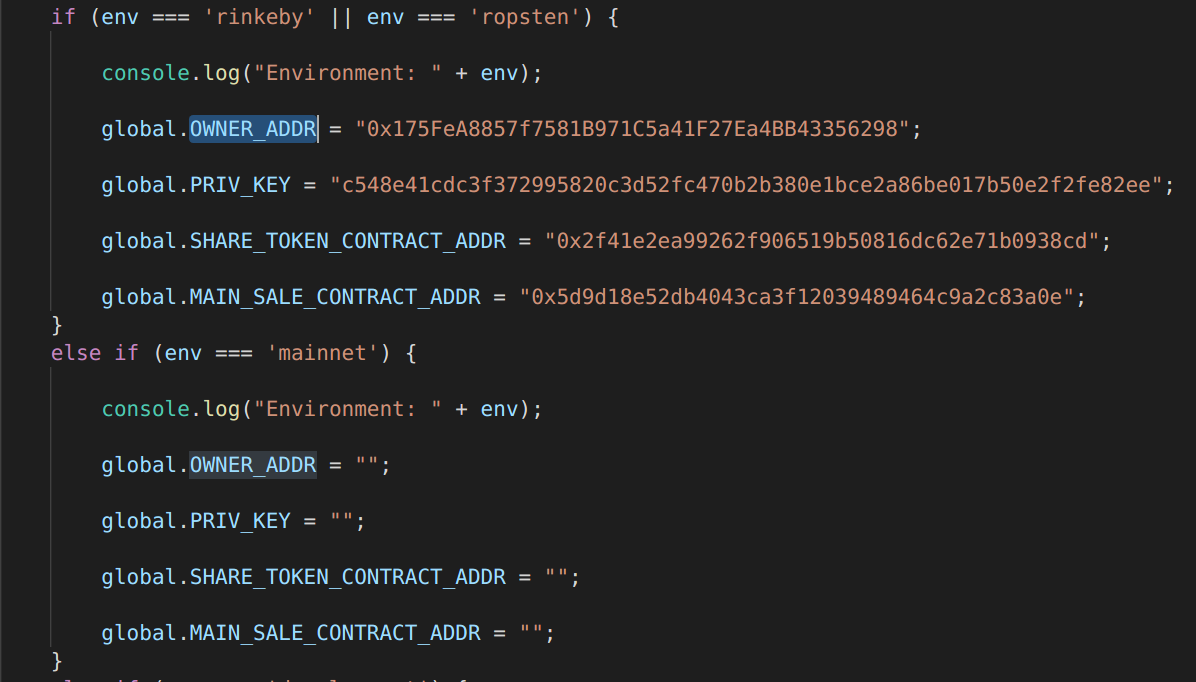
### 3.1. Configuration

After having compiled and deployed the ShareToken contract, it’s possible to transfer the presale data to the SHR token.

At first, necessary info has to be set in the configuration file “app/global.js”:

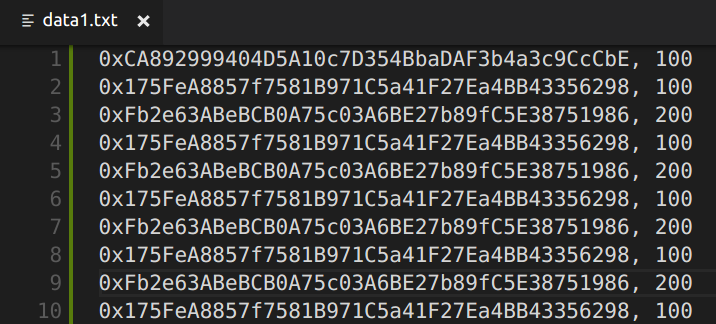
* **OWNER\_ADDR**: the address that has been used to deploy the ShareToken contract
* **PRIV\_KEY**: the private key associated with the above owner address
* **SHARE\_TOKEN\_CONTRACT\_ADDR**: the deployed address of ShareToken contract
* **MAIN\_SALE\_CONTRACT\_ADDR**: the deployed address of MainSale contract

The configuration is set for the chosen environment such as Rinkeby testnet or the mainnet. See the below screenshot for example:



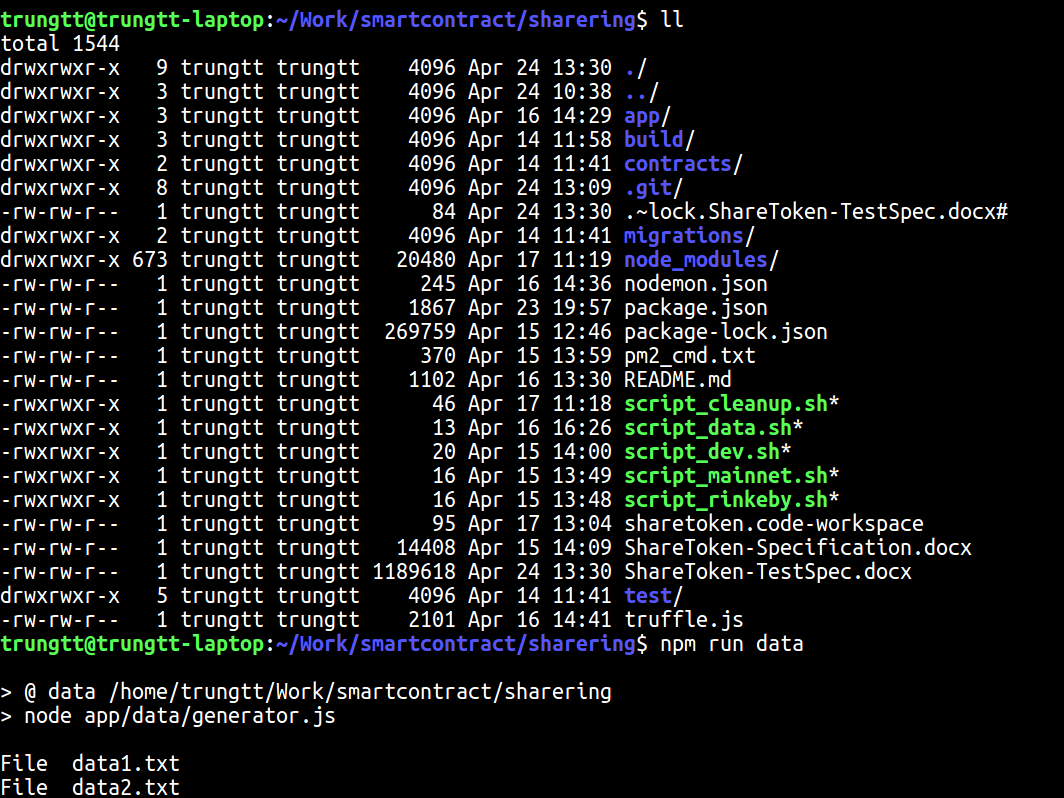
### 3.2. Generation of presale data of pre-defined format

The presale data is provided in form of raw plain text data files whose each line contain an address and the associated tokens. For example:

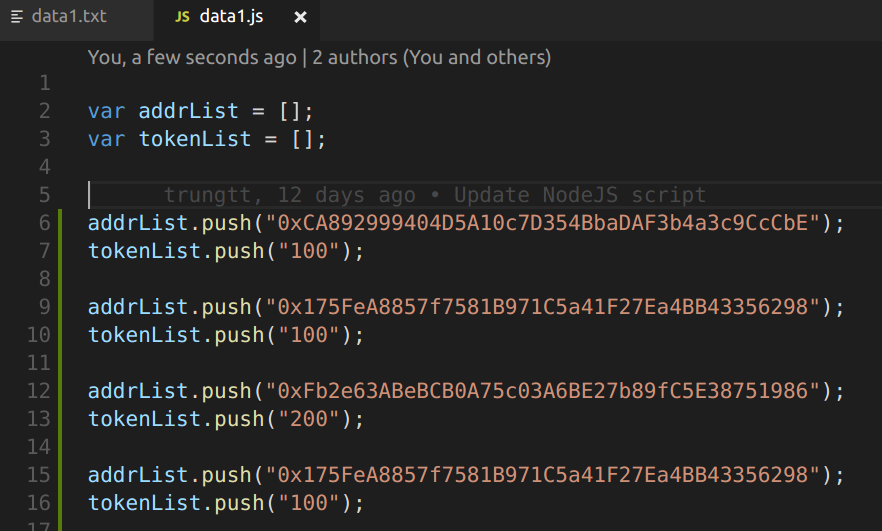


Go to the directory containing the downloaded (or git cloned) files (mentioned in section “1. Solidity files”.

Run the command “npm run data” for generating the pre-defined format of presale data from the raw data files:



The resulting files, which will be stored in the folder “app/data/output”, are actually NodeJS scripts:



## 3. Operation

### 3.1. Input presale data

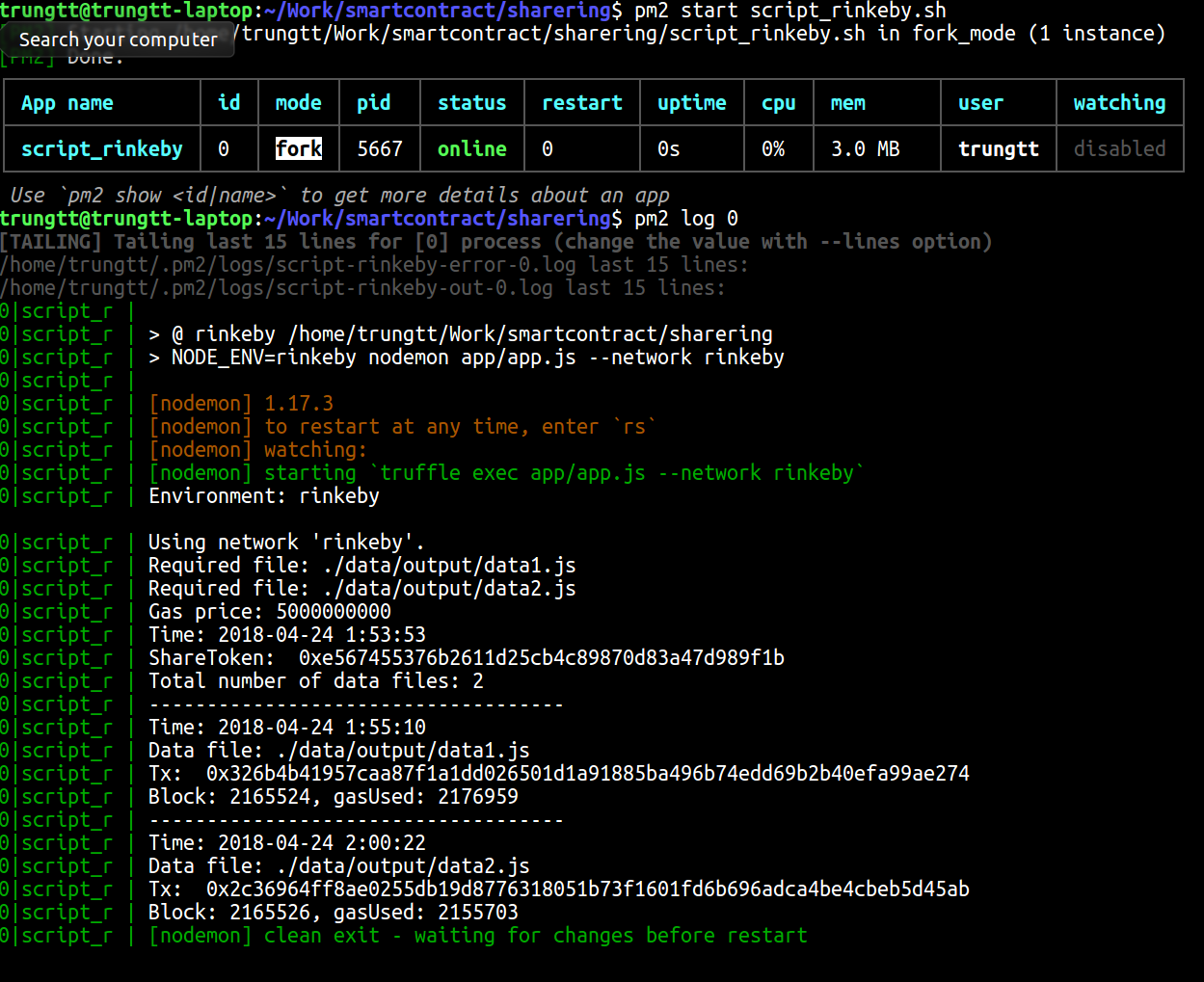
Presale data includes a huge dataset of addresses (e.g. 20K addresses). Each address is associated with some token amount.

Now, it’s ready to perform the transferring of presale data.,

Suppose presale data includes 20K addresses, then there will be totally 200 transactions each of which bundles 100 addresses to be transferred.

For the Rinkeby testnet, use the “**script\_rinkeby.sh**”. For the mainnet, use the “**script\_mainnet.sh**”

Run the command “pm2 start script\_rinkeby.sh” for starting the background process and then run this command “pm2 log <some\_id>” for watching the log output:



The log output for each transaction includes:

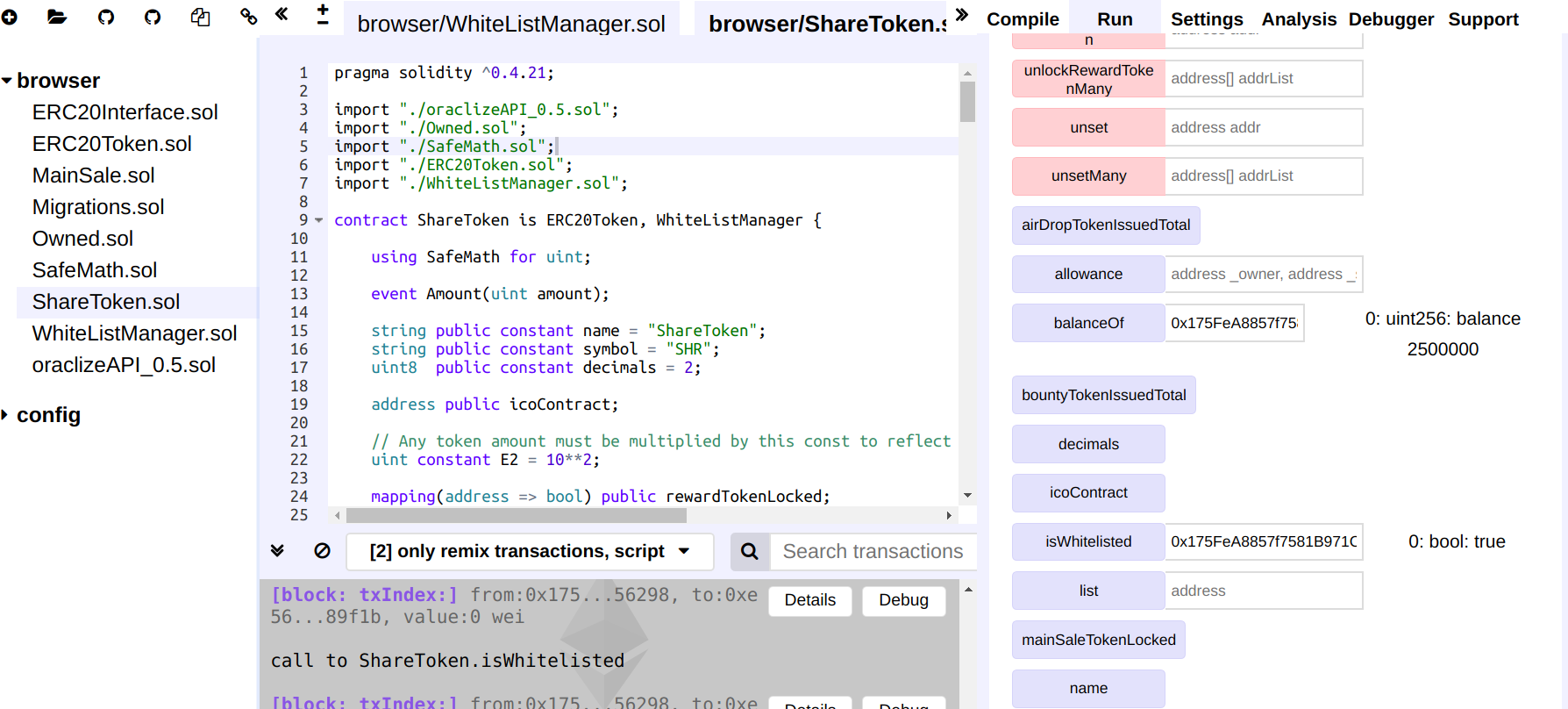
* Tx: transaction ID (than can be checked on etherscan)
* Data file: the presale data file
* Block number and gasUsed

So, if any transaction gets failed, it will be quite possible to be repeated.

## 4. Contract interaction and control

### 4.1. Presale data verification

To **manually** verify the transferred presale data, refer to the deployed ShareToken contract on the remix tool. Several buttons (e.g. balanceOf, isWhitelisted) can be clicked for checking something:



For **automatic** verification of the transferred presale data, run the command:

* “pm2 start script\_veri\_rinkeby.sh” for Rinkeby testnet
* “pm2 start script\_veri\_mainnet.sh” for the mainnet.

### 4.2. ICO start and control with Remix

Refer to the deployed MainSale contract on remix tool.

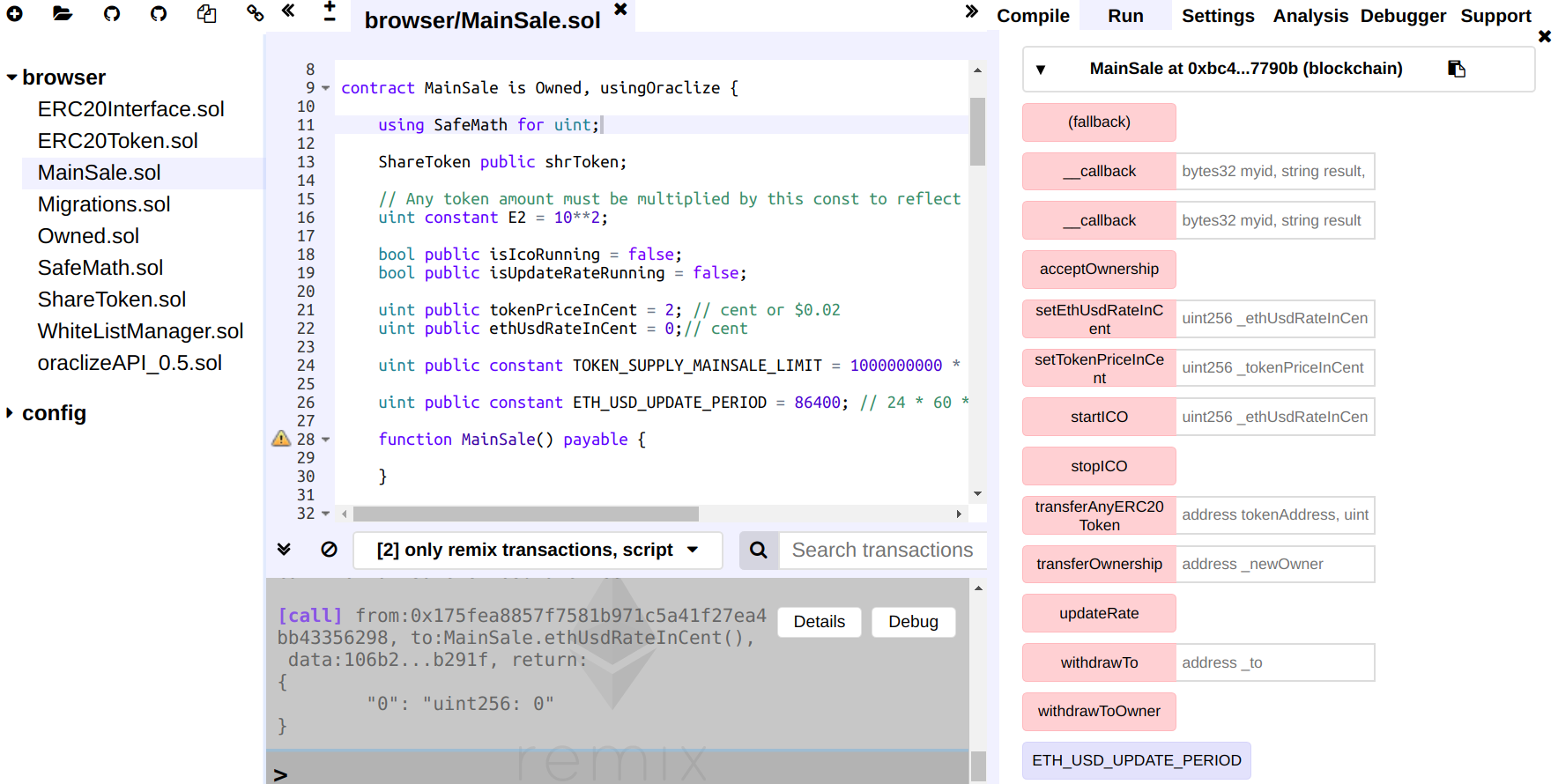
The MainSale contract owner can start the ICO by executing the method “***startICO****(uint \_ethUsdRateInCent, address \_tokenAddress)*” which needs the specification of ETH/USD rate in cent and the deployed address of ShareToken contract.

In addition, the deployed address of MainSale contract must be set by executing the method “***setIcoContract()***” of the ShareToken contract.

Once started, the whitelisted buyers can send ETH to the deployed address of the MainSale contract in exchange for SHR tokens. Unauthorised buyers will not be able to get SHR tokens and thus their transferred ETH will be refunded.

Furthermore, the ETH/USD rate is scheduled to be updated for every 24 hours (i.e. everyday) after the ICO is started.

At any time, the owner can stop the ICO by executing the method “***stopICO****()*”. The ETH/USD rate updating scheduler is also removed.



At the end of main sale, the owner can unlock the addresses by clicking the below buttons:

* unlockMainSaleToken
* unlockRewardToken

### 4.3. ICO start and control with NodeJS script

The instruction is described in the file “**README.md**”

## 3. Contribution

Tran, Thi Thuy Trang

Trinh, Tien Trung