# ShareToken specification (version-2)

## 1. Implementation files

The implementation is uploaded on this github: <https://github.com/trungtt1981/smartcontract>

### 1.1. Test files

The test files are stored in the folder “**test**”. Test instruction is described in the file README.md

### 1.2. Solidity files

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

The main contracts below are ownable to ensure that critical methods can only be done by the contract owner:

* **WhiteListManager.sol**
  + Owned contract to ensure that critical methods can only be done by the contract owner.
  + Contain a list of white-listed addresses
  + Provide several methods for updating the whitelist by the owner
  + Provide a common method for checking if a given address is whitelist
* **ShareToken.sol**
  + Conform to the standardized ERC20 interface
  + Inherited from the **WhiteListManager** above
  + Provide the properties and methods of the ShareToken token according to the requirement specification.
  + Some important methods:
    - *sell(address buyer, uint tokens)*: transfer the main sale tokens to the buyer. The total amount of main sale tokens issued is stored in the variable “totalTokenIssued”. The token buyer is locked. The contract owner will unlock all token buyers manually.
    - *rewardAirdrop(address \_to, uint \_amount)*: transfer the reward tokens of Airdrop to the receiver. The receiver is locked and will be unlocked by the contract owner.
    - *handlePresaleTokenMany(address[] addrList, uint[] amountList)*: add the presale addresses into whitelist and also transfer the presale tokens to the correspondent addresses. The presale tokens are registered in the balance of the SHR token contract.
  + Some important properties:
    - *address icoContract*: hold the address of the deployed MainSale contract for being used in the *sell()* method
    - *bool mainSaleTokenLocked*: default to true and will be changed manually (e.g. 7 days after main sale)
    - *rewardTokenLocked*: hold a list of reward token receivers for unlocking separately from the presale whitelist.
* **MainSale.sol**
  + Inherited from Oraclize for using its API that allows to query ETH/USD rate. The query is scheduled for 24 hours (i.e. every day). The actual cost (that has been observed) is about 0.013 ETH per updating. So if the main sale runs for 4 weeks, then the total fee will be 4 \* 7 \* 0.013 = 0.364 ETH. This fee is automatically deducted from the MainSale contract balance in ETH.
  + Contain reference to the ShareToken contract used to check if a given address is whitelist or to sell tokens to the buyer according to the received ETH.
  + Implement the autonomous procedure of issuing tokens to the whitelisted buyers upon the received ETH. If the hard limit of main sale tokens (1 billion) is exceeded, the remaining ETH will be refunded and ICO is stopped.
  + Provide 2 methods for withdrawing all of the MainSale contract balance (in ETH):
    - *withdrawToOwner()*: used to withdraw to the owner of MainSale contract
    - *withdrawTo(address \_to)*: used to withdraw to a given address
  + Provide methods for starting/stopping the ICO:
    - *startICO(uint \_ethUsdRateInCent, address \_tokenAddress)*: need the specification of ETH/USD rate in cent and the deployed address of ShareToken contract.
    - *stopICO()*

### 1.3. Script files

The script files are stored in the folder “**app**” which contains the following:

* Folder “**data**” includes a “*generator.js*” script that reads plain text files (stored under sub-folder “**input**”) and produces the data files. The plain text files hold the presale data where each presale address is associated with the tokens. The produced data files (in form of NodeJS script) will be used as input to the script “*app.js*” described below. Each plain text file corresponds to a data file.
* Script “*app.js*” performs interaction with the deployed ShareToken contract to add the presale addresses into the whitelist and also to transfer the presale tokens to the correspondent addresses.

## 2. Deployment procedure

* Firstly, deploy the ShareToken contract.
* Secondly, deploy the MainSale contract.

## 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.

It is required to upload the presale data onto blockchain. This is done by executing the provided script “*app.js*” which splits the huge dataset of presale into multiple transactions to be sent to the network. Each transaction includes an acceptable number of addresses.

Before running the script “*app.js*”, one should prepare the presale data. The raw presale data in plain text files are stored in sub-folder “**input**”. The script “**generator.js**” is executed to produce the prepared presale data files.

### 3.2. Whitelist

The whitelist is mostly input from the presale data. It is always possible to update the whitelist by the owner.

### 3.3. ICO start

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.

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.

## 4. Notice

### 4.1

If the hard limit of main sale tokens (1 billion) is exceeded, the remaining ETH will be refunded and ICO is stopped to prevent further token buying.

### 4.2

Both of the seed and presale tokens are transferred to the correspondent addresses without the care if seed tokens or presale tokens.

### 4.3

The total number of tokens issued is the sum of main sale tokens sold, presale tokens transferred and reward tokens for Airdrop/Bounty

### 4.4

During main sale, all token buyers will be locked. The owner will have to manually unlock (for example, 7 days after main sale end) by executing the method “*unlockMainSaleToken()*”. This method will not unlock the receivers of reward tokens. Another method can be used (e.g. *unlockRewardToken(address addr*).

### 4.5

Checking if a given address is locked is done as follows:

* As long as ICO main sale is running, any address is considered locked and thus impossible to pass on tokens
* If ICO is already stopped and the owner also already unlocks:
  + Any address in the whitelist is considered unlocked
  + Any address in the reward list is checked if locked or not
  + Any other address that is not in the whitelist nor in the reward list is considered unlocked.

## 3. Contribution

Tran, Thi Thuy Trang

Trinh, Tien Trung