Encircled manual

1. Set up the contracts
    a. Deployment:
        i. Deploying the encircled token / or the upgradable token
        ii. Deploy the Vesting smart contract [constructor args: ENCD token address]
        iii. Deploy the ICO smart contract [constructor args: ENCD token address, Vesting smart contract address, usdt address, dai address, busd address]

    b. Smart contract set up
        i. Transfer ownership of the vesting contract to the ICO contract
            1. Function transferOwnership [args: ICO smart contract address]
        ii. Exclude vesting contract from fee and reward of ENCD Token
            1. Function excludeFromFee [args: Vesting smart contract address]
            2. Function excludeFromReward [args: Vesting smart contract address]
        iii. Transfer the presale and team tokens to the vesting smart contract
            1. Function transfer [args: Vesting smart contract address, amount team + presale = 86,000,000,000,000,000,000,000,000 (amount + 18 decimals)]
        iv. Transfer the rest of the tokens to the desired address (rewards, team, etc)
            1. Function transfer [args: receiver, receiving amount (amount + 18 decimals)]

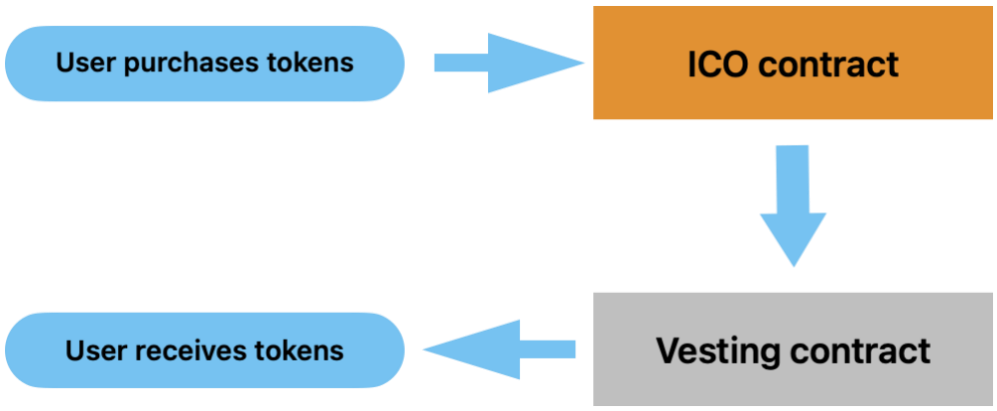
    c. Presale and vesting
        i. Set the ending time of the presale (=TGE event and beginning of the vesting period) on the ICO Contract
            1. Function startVesting [args: duration till (from the time of calling the function) the release in seconds, e.q. if it should be in one month, enter 60*60*24*30 = 2592000]
        ii. Start the presale (start seed sale) in the ICO Contract
            1. Function setStage[args: index of stage (seed = 1)]
        iii. After end of the stage advance to the next stage
            1. Function setStage[args: index of stage (private = 2)/ (public = 3]
        iv. Finish the ICO
            1. Function setStage[args: index of stage (end = 4)]

2. Buy tokens in the ICO
    a. Call the approve function with your front end of the desired stablecoin (used for purchasing your tokens)
        i. E.g. if the wants to purchase the tokens with dai call the approve function in the DAI contract:
            1. Function approve[args: ICO contract address, amount to approve (usually you just approve the maximum amount)]
    b. Call the buy function in the ICO contract:

i. Function buyToken[args: amount of tokens the user wants to buy (+ decimals) 1 token would equal to 1 000..00 (18 x 0)]

3. Withdraw the stablecoins
   a. Call the withdraw function in the ICO contract
      i. Function withdraw[args: amount of tokens you want to withdraw (+ decimals) 1 token would equal to 1 000..00 (18 x 0), index of the token you want to withdraw (1- USDT, 2-DAI, 3-BUSD]

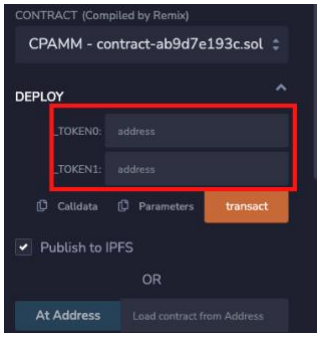
4. Releasing the tokens at TGE and during the Vesting in the Vesting contract
   a. Compute vesting schedule for uses
      i. Function computeVestingScheduleIdForAddressAndIndex[args: address of the user, id of the purchase (first purchase of the user is 0, if the same user purchases again the second one is 1,..)
   b. Checking how many tokens a user can withdraw (optional)
      i. Function computeReleasableAmount [args: return value of comuteVestingScheduleIdForAddressAndIndex)
   c. Releasing the tokens
      i. Function release[args: return value of computeVestingScheduleIdForAddressAndIndex , amount of tokens user wants to realease) = tokens will be send to the wallet of the user



Deployment see attachment.

Remarks:

If the constructor needs an argument enter it before you click on deploy:



Deployment for the Proxy Contract:

a. Deployment of the Upgradable Smart Contract

Select “Deploy with Proxy” in Remix before deploying