How to use the MultiSig wallet?

MultiSig wallet can be used for two purposes:

1. Modify master contracts for yRace
2. Deposit tokens into and withdraw tokens from the contract

Modify master contracts for yRace

For the owner to be able to modify the state of contract (like adding a new pool or modifying existing pool), he must submit a transaction to MultiSigWallet contract that queues the required transaction(which will add/set a pool) and then other owners can confirm that transaction. When no. of confirmations reaches the ‘required’ value, the queueTransaction is executed and the required transaction is queued.
Similarly, a transaction must be submitted to multisig for executing the required transaction.
But in order to submit a transaction, the owner needs the destination address(which will be the address of the timelock contract) and call data of queueTransaction or executeTransaction, which can be obtained using the “TLscript.js” file.

In order to obtain callData, follow the steps:

1. Open the file "TLsciript.js"
2. Move to the bottom of the file and modify the last line => obj.callDataforModify("","","",[],"")
3. Remove the “//” from the start of line 65
4. Within the first double quotes ("") of the callDataforModify, first specify the name of operation to be performed using timelock contract :
    - Use “queue” to queue transaction
    - Use “execute” to execute transaction
5. Within the second double quotes (""), specify the target which is actually the contract on which pool is to be added/modified :
    - Use “Seed” to add pool in seed master contract
    - Use “LP” to add pool in LP master contract
6. Within the third double quotes (""), specify the function name which specifies if pool is to be added or modified :
    - Use “add” to add pool
    - Use “set” to modify pool

7. Within the square brackets ([]) of the getData, pass the list of arguments with each argument within double quotes, separated with a comma.
   Example : ["100", "0xEC5dCb5Dbf4B114C9d0F65BcCAb49EC54F6A0867", "200", "true"]);
8. Within the fourth double quotes (""), specify the eta which is the time in epoch form and it specifies when the transaction shall be allowed to execute via timelock. Epoch time can be obtained from https://www.epochconverter.com/
9. Save the file, and open the terminal in the file location.
10. Run "node TLscript.js"
11. The output will show callData needed which can be copied and pasted on bscscan.com
12. Put back the "//" in front of the line 65 to avoid wrong output

Withdraw tokens/native currency from MultiSigWallet

For the owner to be able to withdraw tokens/native currency from MultiSigWallet, he must submit a transaction to MultiSigWallet contract that then other owners can confirm that transaction. When no. of confirmations reaches the 'required' value, the transaction is executed, and tokens/native currency is transferred to the owner account.

But in order to submit a transaction, the owner needs the destination address(which will be the address of the multisig wallet contract) and callData of withdrawNative or withdrawToken, which can be obtained using the "TLscript.js" file.

In order to obtain callData, follow the steps:

1. Open the file "TLsciript.js"
2. Move to the bottom of the file and modify the last line => obj.callDataForWithdraw("","","","")
3. Remove the "//" from the start of line 66
4. Within the first double quotes ("") of the callDataforModify, first specify the name of function that specifies if native currency tokens are to be withdrawn:
   - Use "native" for native currency withdraw
   - Use "token" for token withdraw
5. Within the second double quotes (""), specify the token contract address:
   - Use "0x0" if native currency withdraw
   - Use "token_contract_address" if tokens withdraw

6. Within the third double quotes (""), specify your user address.
7. Within the fourth double quotes (""), specify the amount.
8. Save the file, and open the terminal in the file location.
9. Run "node TLscript.js"
10. The output will show callData needed which can be copied and pasted on bscscan.com
11. Put back the “//” in front of the line 66 to avoid wrong output