**DFX Project Details**

To implement the requirements that is,

A tax on the native (L1) Token’s movement in case of transfer, buy and sell, I researched and preferred the following:

1. Go-Ethereum ( Go language implementation of the Ethereum Protocol)

Execution Layer + Networking Layer

1. Prysm

Consensus (Beacon Client) Layer

1. Foundary

Cast is used to create or use pre funded wallets in genesis.json to test transactions on the POS Testnet.

# **What was changed:**

The files that were changed from the vanilla(normal) Ethereum fork from Github, are as following:

1. core/vm/evm.go (function NewEVM was modified to tax buy,sell and transfer)
2. core/vm/instructions.go (2 functions namely “Opselfdestruct and Opselfdestruct6780” were rewritten to implement taxes on token burning)
3. core/vm/tax\_transfer.go (New file was created to centralize tax logic)
4. params/config.go (type ChainConfig now includes tax fields to be disabled later at will on mainnet without a hard fork)
5. genesis.json ( Now includes initializing the tax fields like enabling, percentage and beneficiary (treasury) address)

Note: If you decide to modify the tax logic make sure you’re not taxing block validators(miners) or double taxing or taxing funds being moved to the DAO(treasury) itself.

# **TESTING ON TESTNET**

1. Using the other guide, you can successfully the run the test net and only after that use cast(foundary) to test tax logic using:

### Step: Create Accounts Before Genesis

1. **Generate accounts using cast**

cast wallet new

This will give you:

* An **address** (0x...)
* A **private key**

Repeat this for however many accounts you want to pre-fund.

👉 Save those addresses. You don’t need the private keys inside genesis.json (those are only for the user to hold and sign transactions).

1. **Add them into genesis.json under alloc**  
   For example:

{

"config": {

"chainId": 12345,

…..

"alloc": {

"123463a4b065722e99115d6c222f267d9cabb524": {

"balance": "0x43c33c1937564800000"

},

"182a90db75b3263e8904df8d99d139e98ed6605b": {

"balance": "0x21e19e0c9bab2400000" }

}

…….

}

Each address you generated goes in here with a starting balance (in wei).

1. Keep the console open, in another console, run “./testnet.sh” that wil initialize the updated genesis.json and pickup the new pre-funded accounts in this blockchain session.
2. Come back to this console, and run the following:
   1. Precheck the balance (replace 0x….abcd with the address)
      1. cast balance 0x1234...abcd -r http://127.0.0.1:8545 –ether
      2. similary pre check balances of sender, receiver and treasury
   2. Send a transaction (replace variables of treas and rpc and pk (private key))
      1. cast send $TREAS --value 1ether -r $RPC --private-key $PK\_FROM >/tmp/tx.txt
   3. Check balance of sender and receiver
      1. cast balance <SENDER\_ADDRESS> -r://localhost:8000 –ether
      2. cast balance <RECEIVER\_ADDRESS> -r http://localhost:8000 –ether
   4. Check if treasury received +5% of the amount
      1. cast balance -r http://localhost:8000 <Treasury\_Address)