Voting Escrow

- Votes have a weight depending on time, so that users are committed to the future of (whatever they are voting for).
- Vote weight decays linearly over time. Lock time cannot be more than MAXTIME (4 years)
    - Voting escrow to have time weighted votes
    - Votes have a weight depending on time, so that users are committed to the future of (whatever they are voting for).
    - The weight in this implementation is linear, and lock cannot be more than maxtime.

Structs

1. Point struct
- bias
- slope
- ts (timestamp)
- blk (block)

We cannot really do block numbers per se because slope is per time, not per block and per block could be fairly bad because Ethereum changes blocktimes. What we can do is extrapolate at functions

2. LockedBalance struct:
- amount
- end

Interfaces:

1. ERC20:
- decimals()
- name()
- symbol()
- transfer()
- transferFrom()

2. Smart wallet checker

Interface for checking whether address belongs to a whitelisted type of a smart wallet. When new types are added – the while contract is changed.
The check() method is modifying to be able to use caching for individual wallet addresses

check(address) → bool

Constants

- DEPOSIT_FOR_TYPE
- CREATE_LOCK_TYPE
- INCREASE_LOCK_AMOUNT
- INCREASE_UNLOCK_TIME

WEEK
MAXTIME (4 years)
MULTIPLIER

Events

1. Commit Ownership
- admin: Address

2. Apply Ownership
- admin: Address

3. Deposit
- provider: address
- value: int
- locktime
- type
- ts (timestamp)

4. Withdraw
- provider: address
- value: int
- ts(timestamp)

5. Supply
- prevSupply
- supply

State Variables

- token (address)
- supply(uint256)
- locked(HashMap[address, LockedBalance])

- epoch(uint256)
- point_history(Point[1000000000000000000000000000000])
- user_point_history(HashMap[address, Point[1000000000])
- user_point_epoch(HashMap[address, uint256)
- slope changes(HashMap[uint256, int128))

Aragon compatibility vars
- controller: address
-transfersEnabled: bool

- name
- symbol
- version
- decimals

Checker for whitelisted (smart contract) wallets which are allowed to deposit. The goal is to prevent tokenizing the escrow.

- future_smart_wallet_checker: address
- smart_wallet_checker: address

Methods

1. **constructor**

- Params
    - token_addr: Address: ERC20CRV token address
    - _name: String[64]: Token name
    - _symbol: String[32]: Token Symbol
    - _version: String[32]: Contract version - required for Aragon compatibility

  Does initial setup of the contract.

2. **commit_transfer_ownership**
   Transfer ownership of voting escrow contract to addr
- Params
    - addr: Address: Address to have ownership transferred to

3. **apply_transfer_ownership**
   Apply ownership transfer

4. **commit_smart_wallet_checker**
   Set an external contract to check for approved smart contract wallets
- Params
    - addr: Address: Address of smart contract checker

5. **apply_smart_wallet_checker**
   Apply setting external contract to check approved smart contract wallets

6. **assert_not_contract (internal)**
   Check if the call is from a whitelisted smart contract, revert if not
- Params
    - addr: Address: Address to be checked

7. **get_last_user_slope (readonly)**
   Get the most recently recorded rate of voting power decrease for `addr`
- Params
    - addr: Address: Address of the user wallet
- Returns
    - int128, Value of the slope

8. **user_point_history__ts (readonly)**
   Get the timestamp for checkpoint _idx for _addr
- Params
    - _addr: Address: User wallet address
    - _idx: uint256: User epoch number
- Returns
    - uint256: Epoch time of the checkpoint

9. **locked__end (readonly)**
   Get timestamp when _addr's lock finishes
- Params
    - _addr: Address: User wallet
- Returns
    - uint256: Epoch time of the lock end

10. **_checkpoint (internal)**
    Record global and per-user data to checkpoint
    The entire contract's main logic is here.
- Params
    - addr: Address: User's wallet address. No user checkpoint if 0x0
    - old_locked: LockedBalance: Previous locked amount / end lock time for the user

- new_locked: LockedBalance: New locked amount / end lock time for the user

11. _deposit_for (internal)

Deposit and lock tokens for a user

- Params
  - _addr: Address: User's wallet address
  - _value: uint256: Amount to deposit
  - unlock_time: uint256: New time when to unlock the tokens, or 0 if unchanged
  - locked_balance: LockedBalance: Previous locked amount / timestamp

12. checkpoint

Record global data to the checkpoint. Calls internal checkpoint method

13. deposit_for

Deposit `_value` tokens for `_addr` and add to the lock. Anyone (even a smart contract) can deposit for someone else, but cannot extend their locktime and deposit for a brand new user.

- Params
  - _addr: Address: User's wallet address
  - _value: uint256: Amount to add to user's lock

14. create_lock

Deposit `_value` tokens for `msg.sender` and lock until `_unlock_time`

- Params
  - _value: uint256: Amount to deposit
  - _unlock_time: uint256: Epoch time when tokens unlock, rounded down to whole weeks

15. increase_amount

Deposit `_value` additional tokens for `msg.sender` without modifying the unlock time

- Params
  - _value: uint256: Amount of tokens to deposit and add to the lock

16. increase_unlock_time

Extend the unlock time for `msg.sender` to `_unlock_time`

- Params
  - _unlock_time: uint256: New epoch time for unlocking

17. withdraw

Withdraw all tokens for `msg.sender`. Only possible if the lock has expired.

18. find_block_epoch (readonly)

Binary search to estimate timestamp for block number

- Params

    - _block: uint256: Block to find
    - max_epoch: uint256: Don't go beyond this epoch
- Returns
    - uint256: Approximate timestamp for block

19. balanceOf (readonly)

Get the current voting power for `msg.sender`. Adheres to the ERC20 `balanceOf` interface for Aragon compatibility.

- Params
    - addr: Address: User wallet address
    - _t: uint256: defaults to block.timestamp: Epoch time to return voting power at
- Returns
    - uint256: User voting power

20. balanceOfAt (readonly)

Measure voting power of `addr` at block height `_block`. Adheres to MiniMe `balanceOfAt` interface: https://github.com/Giveth/minime

- Params
    - addr: Address: User's wallet address
    - _block: uint256: Block to calculate the voting power at
- Returns
    - uint256: Voting power

21. supply_at (readonly)

Calculate total voting power at some point in the past

- Params
    - point: Point: The point (bias/slope) to start search from
    - t: uint256: Time to calculate the total voting power at
- Returns
    - uint256: Total voting power at that time

22. totalSupply (readonly)

Calculate total voting power. Adheres to the ERC20 `totalSupply` interface for Aragon compatibility

- Params
    - t: uint256: Defaults to block.timestamp
- Returns
    - uint256: Total voting power

23. totalSupplyAt (readonly)

Calculate total voting power at some point in the past

- Params
    - _block: uint256: Block to calculate the total voting power at
- Returns

- uint256: Total voting power at `_block`

24. changeController

Dummy method required for Aragon compatibility