Boid Staking Contract Usage

Jungle Testnet:

URL = https://api.jungle.alohaeos.com

OWNER = `we3i5kdsdo12`

USER = `bbeeffdd1234`

Mainnet:

URL = http://api.eosn.io

OWNER = 'boidcomtoken'

USER = 'luke12341234'

Wallet Setup:

Create Wallet:

```
cleos wallet create --name testnet_wallet
```

Open Wallet:

```
cleos wallet open --name testnet_wallet
```

Unlock Wallet:

```
cleos wallet unlock --name testnet_wallet
```

Import Account Keys

```
cleos wallet import --name testnet_wallet --private-key account_private_key
```

note: 3 accounts have already been created on the testnet website: we3i5kdsdo12, bbeeffdd1234, and luke11112222. So if you want to just use these accounts for testing, you can just copy/paste these commands:

```
cleos wallet import --name testnet_wallet --private-key \
5KEXybEm1uNLo26aoWowcvHMPqFAoem8avEk1gSWMLBeyDs4xsi
```

```
cleos wallet import --name testnet_wallet --private-key \
5JbpEn5PrARzp3KkxZ2X8upqrZa9dys66oMvZDHqsYz9SCZM2Sb
```

```
cleos wallet import --name testnet_wallet --private-key \
5JChskWpkZHj8bdg1KWQHrNLdBHjAbBhgDzcJKuNk3vg1vVCRXw
```

```
cleos wallet import --name testnet_wallet --private-key \
5J6Ymo6d83WQxWLfB4rwFanFn8D8wJcYpP1K6gQg4UivmvT2Gj8
```

```
cleos wallet import --name testnet_wallet --private-key \
5KEuiA97s9Dfh848AVyF8mZVhNzdoYvU3rqwwegTL3nk7EqNUdA
```

```
cleos wallet import --name testnet_wallet --private-key \
5KdVKZ8ED8X6ydnbCe21TcAvuvycyVLBp4i9E7E56yYUVNHhUPN
```

any other accounts you want to involve in your tests need to be created first on the testnet website and then run this Import step again for their keys also. See the following link for a detailed walk through of how to create an account on the testnet website: https://hackernoon.com/how-to-create-and-deploy-your-own-eos-token-1f4c9cc0eca1

Contract Actions and Info:

Info:

Deploy contract at specified URL to specified account:

```
cleos -u URL set contract OWNER PATH boidtoken.wasm \
boidtoken.abi -p OWNER@active
```

note: PATH must be to where ever the .wasm and .abi files are, if done from the Token-Staking-Upgrade repo directory, PATH = build/

Remove contract from specified account (un-deploy)

```
cleos -u URL set contract OWNER -c
```

note: this will not get rid of the current state of the contract if you then deploy it again under the same account.

Get the unstaked balance of a user's account

```
cleos --url URL get table OWNER USER accounts
```

Actions:

create(): Create tokens under boidtoken contract

```
cleos --url URL push action OWNER create \
'{"issuer":"OWNER", "maximum_supply":\
"1000000000.0000 BOID"}' -p OWNER
```

Arguments:

"issuer": the account that will have the ability to mint tokens and send them to other accounts. Data type must be a string of a valid EOS account name. Example: "boidcomtoken"

"maximum_supply": the maximum number of tokens the issuer can mint. Data type must be a string with a float followed by a valid EOS token symbol name. Note: whatever level of precision you give maximum_supply, this level of precision is required for several other action calls. Example: "1000000000.0000 BOID" (precision = 4 decimal places)

Permissions: Only the OWNER has permission to create a new type of token

issue(): Issue tokens under boidtoken contract

```
cleos --url URL push action OWNER issue \
'{"to":"USER", "quantity":"1000.0000 BOID", \
"memo":"memo"}' -p OWNER
```

Arguments:

"to": the account the issuer is minting and sending tokens too. Data type must be a string of a valid EOS account name. Example: "luke12341234"

"quantity": the number of tokens to mint and send to "to". Data type must be a string with a float with the same level of precision and token symbol specified in the call to the Create action. Example: "1000.0000 BOID"

"memo": a short (human readable) description of what the issue is for (or of whatever you want really). Data type must be a string. Example: "sending money to luke"

Permissions: Only the issuer (specified in Create action) has permission to issue tokens

recycle(): delete tokens under boidtoken contract

```
cleos --url URL push action OWNER recycle \
'{"quantity":"1000.0000 BOID"}' -p OWNER
```

Arguments:

"quantity": the number of tokens to delete from the issuers account balance. Data type must be a string with a float with the same level of precision and token symbol specified in the call to the Create action. Example: "1000.0000 BOID"

Permissions: Only the issuer (specified in Create action) has permission to recycle tokens

transfer(): Transfer tokens between end-user accounts

```
cleos --url URL push action OWNER transfer \
'{"from":"USER1", "to":"USER2", "quantity":"100.0000 BOID", \
"memo":"memo"}' -p USER1
```

Arguments:

"from": the account of the sender of tokens. Data type must be a string of a valid EOS account name. Example: "boidcomtoken"

"to": the account of the reciever of tokens. Data type must be a string of a valid EOS account name. Note: "to" cannot be the same as "from". Example: "luke12341234"

"quantity": the number of tokens to transfer from "from" to "to". Data type must be a string with a float with the same level of precision and token symbol specified in the call to the Create action. Example: "1000.0000 BOID"

"memo": a short (human readable) description of what the transfer is for (or of whatever you want really). Data type must be a string. Example: "sending money to luke"

Permissions: transfering tokens requires the permission of the sender

transtaked(): Transfer unstaked tokens from OWNER account to USER account and automatically stake the transfered tokens in USER's account

```
cleos --url URL push action OWNER transtaked \
'{"to":"USER", "quantity":"100000.0000 BOID", \
"memo":"memo"}' -p OWNER
```

Arguments:

"to": the account of the reciever of tokens. Data type must be a string of a valid EOS account name. Example: "luke12341234"

"quantity": the number of tokens to transfer from OWNER to "to". Data type must be a string with a float with the same level of precision and token symbol specified in the call to the Create action. Example: "100000.0000 BOID"

“memo”: a short (human readable) description of what the transtaked is for (or of whatever you want really). Data type must be a string. Example: “sending staked money to luke”

Permissions: transtaking tokens requires the permission of the OWNER

stakebreak(): Set whether users can stake or not

```
cleos --url URL push action OWNER stakebreak \
'{"on_switch":"1"}' -p OWNER
```

Arguments:

“on_switch”: the flag to toggle giving users the ability to stake and unstake. Data type must be a string of an integer. An integer of 0 turns off staking and unstaking abilities for users; any other integer turns them on. Note: a user is defined as any EOS account that owns boidtokens that is not the contract owner. Example: “1”

Permissions: the steakbreak can only be toggled by the OWNER

stake(): End-users stake BOID tokens

```
cleos --url URL push action OWNER stake \
'{"_stake_account":"USER", "_staked":\
"500.0000 BOID"}' -p USER
```

Arguments:

“_stake_account“: the account staking some or all of their tokens. Data type must be a string of a valid EOS account name. Example:”luke12341234"

“_staked“: the number of tokens to stake. Data type must be a string with a float with the same level of precision and token symbol specified in the call to the Create action. Example:”500.0000 BOID"

Permissions: only the account itself has permission to stake its own tokens

sendmessage(): record a memo on the blockchain

```
cleos --url URL push action OWNER claim \
  '{"acct":"USER", "memo":"memo"}' -p USER
```

Arguments:

“acct”: the account sending the message

“memo”: the message being sent

Permissions: any account can send a message

claim(): Auto-claim BOID token rewards CLAIM IS CURRENTLY UNAVAILABLE

```
cleos --url URL push action OWNER claim \
  '{"_stake_account":"luke12341234"}' -p USER
```

Arguments:

"_stake_account": the account with staked tokens to claim the reward for. Data type must be a string of a valid EOS account name. Example:"luke12341234"

Permissions: only the OWNER has permission to claim staking rewards for accounts that have staked tokens

unstake(): End-users unstake BOID tokens

```
cleos --url URL push action OWNER unstake \
'{"_stake_account":"USER", "quantity":"5000.0000 BOID"}' -p USER
```

Arguments:

"_stake_account": the account with staked tokens to unstake. Data type must be a string of a valid EOS account name. Example:"luke12341234"

"quantity": the number of tokens to unstake. Data type must be a string with a float with the same level of precision and token symbol specified in the call to the Create action. Example: "5000.0000 BOID"

Permissions: only the _stake_account has permission to unstake its tokens when the season is not happening. When the season is happening, only the OWNER has permission to unstake the _stake_account's tokens.

initstats(): Initialize staking tablular information

```
cleos --url URL push action OWNER initstats \
'{}' -p OWNER
```

Arguments:

None

Permissions: only the OWNER has permission to run initstats

setnewbp(): Set new boidpower for end-user

```
cleos --url URL push action OWNER setnewbp \ '{"acct":"USER",
"boidpower":"1000.00"}' -p OWNER
```

Arguments:

"acct": the account to set the boidpower for. Data type must be a string of a valid EOS account name. Example: "luke12341234"

"boidpower": the amount of boidpower to give the acct. Data type must be a string of a valid float value. Example "1000.00"

Permissions: only the OWNER has permission to set the boidpower of the acct

setroi(): Set percentage Return On Investment for staking over a 1 month period

```
cleos --url URL push action OWNER setroi \ '{"month_stake_roi":"1.3"}'
-p OWNER
```

Arguments:

"month_stake_roi": the percentage return on investment a user's account will recieve over a 1 month period as a reward for staking (assuming the user account has no boidpower). Data type must be a string of a valid float value. Example "1.3"

Permissions: only the OWNER has permission to set the month_stake_roi

setbpratio(): Set BP_BONUS_RATIO, boidpower/boidstake >= BP_BONUS_RATIO to qualify for boidpower bonus

```
cleos --url URL push action OWNER setbpratio \ '{"bp_bonus_ratio":"0.0001"}'
-p OWNER
```

Arguments:

"bp_bonus_ratio": the ratio of boidpower to tokens staked that is required for a user account to recieve the boidpower bonus reward. Data type must be a string of a valid float value. Example "0.0001"

Permissions: only the OWNER has permission to set the bp_bonus_ratio

setbpdiv(): Set BP_BONUS_DIVISOR, boidpower bonus = min((boidpower * boidstaked) / BP_BONUS_DIVISOR, BP_BONUS_MAX)

```
cleos --url URL push action OWNER setbpdiv \ '{"bp_bonus_divisor":"0.0001"}'
-p OWNER
```

Arguments:

"bp_bonus_divisor": the number the boidpower times tokens staked is divided by to calculate the boidpower bonus reward. Data type must be a string of a valid float value. Example "0.0001"

Permissions: only the OWNER has permission to set the bp_bonus_divisor

setbpmax(): Set BP_BONUS_MAX hardcap, boidpower bonus = min((boidpower * boidstaked) / BP_BONUS_DIVISOR, BP_BONUS_MAX)

```
cleos --url URL push action OWNER setbpmax \ '{"bp_bonus_max":"10000.0"}'
-p OWNER
```

Arguments:

"bp_bonus_max": the maximum value the boidpower bonus reward can have. Data type must be a string of a valid float value. Example "10000.0"

Permissions: only the OWNER has permission to set the bp_bonus_max

setminstake(): Set MIN_STAKE, the minimum number of boid tokens a user can stake

```
cleos --url URL push action OWNER setminstake \ '{"min_stake":"100000.0"}'
-p OWNER
```

Arguments:

"min_stake": the minimum number of tokens a user account is allowed to stake. Data type must be a string of a valid float value. Example "100000.0"

Permissions: only the OWNER has permission to set the min_stake