Jungle Testnet:

URL = https://jungle.eosio.cr:443

contract_owner_account = `we3i5kdsdo12`

user_account = `bbeeffdd1234`

Mainnet:

URL = http://api.eosn.io

contract_owner_account = boidcomtoken

user_account = 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: 2 accounts have already been created on the testnet website: we3i5kdsdo12 and bbeeffdd1234. 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
```

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 contract_owner_account PATH boidtoken.wasm \
boidtoken.abi -p contract_owner_account@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 contract_owner_account -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 contract_owner_account user_account
accounts
```

Actions:

create(): Create tokens under boidtoken contract

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

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 contract_owner_account has permission to create a new type of token

issue(): Issue tokens under boidtoken contract

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

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 is be a string. Example: "sending money to luke"

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

transfer(): Transfer tokens between end-user accounts

```
cleos --url URL push action contract_owner_account transfer \
'{"from":"boidcomtoken", "to":"luke12341234", "quantity":"100.0000 BOID", \
"memo":"memo"}' -p from_account
```

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 issue is for (or of whatever you want really). Data type is be a string. Example: "sending money to luke"

Permissions: transfering tokens requires the permission of the sender

running(): Set whether staking is active or no

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

Arguments:

"on_switch": the flag to toggle staking abilities of the contract: claim, unstake, and stake. Data type must be a string of an integer. An integer of 0 turns off staking abilities; any other integer turns them on. Example: "1"

Permissions: running can only be toggled by the contract_owner_account

stakebreak(): Set whether users can stake or not

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

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 contract_owner_account

stake(): End-users stake BOID tokens

```
cleos --url URL push action contract_owner_account stake \
'{"_stake_account":"luke12341234", "_stake_period":"1", \
"_staked":"500.0000 BOID"}' -p user_account
```

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"

“_stake_period“: the amount of time a user can stake token for. Data type must be a string of an integer. Valid values are:”1" for a 1 month stake, or “2” for a 1 quarter stake.

“_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

claim(): Auto-claim BOID token rewards

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

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 contract_owner_account has permission to claim staking rewards for accounts that have staked tokens

unstake(): End-users unstake BOID tokens

```
cleos --url URL push action contract_owner_account unstake \
'{"_stake_account":"luke12341234"}' -p acct
```

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 _stake_account has permission to unstake its tokens when the stakebreak is happening. When the stakebreak is not happening, only the contract_owner_account has permission to unstake the _stake_account’s tokens.

initstats(): Initialize staking tablular information

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

Arguments:

None

Permissions: only the contract_account_owner has permission to run initstats

setnewbp(): Set new boidpower for end-user

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

Arguments:

"user_account": 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 user_account. Data type must be a string of a valid float value. Example "1000.00"

Permissions: only the contract_owner_account has permission to set the boidpower of the user_account

setmonth(): Set percentage Return On Investment for a 1 month stake over a 1 month period

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

Arguments:

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

Permissions: only the contract_owner_account has permission to set the month_stake_roi

setquarter(): Set percentage Return On Investment for a 1 quarter stake over a 1 month period

```
cleos --url URL push action contract_owner_account setquarter \
'{"quarter_stake_roi":"1.5"}' -p contract_owner_account
```

Arguments:

"quarter_stake_roi": the percentage return on investment a user's account will recieve over a 1 month period as a reward for a 1 quarter stake (assuming the user account has no boidpower). Data type must be a string of a valid float value. Example "1.5"

Permissions: only the contract_owner_account has permission to set the quarter_stake_roi

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

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

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 contract_owner_account 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 contract_owner_account setbpdiv \
'{"bp_bonus_divisor":"0.0001"}' -p contract_owner_account
```

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 contract_owner_account 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 contract_owner_account setbpmax \
'{"bp_bonus_max":"10000.0"}' -p contract_owner_account
```

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 contract_owner_account 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 contract_owner_account setminstake \
'{"min_stake":"100000.0"}' -p contract_owner_account
```

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 contract_owner_account has permission to set the min_stake