$SAGE Token Staking Mechanic

December 2023

1 Asset Income

The $SAGE staking contract sources assets used for providing yield from two primary sources.

- ### 1.1 SAGE token buy and sell tax

  Whenever SAGE tokens are bought our sold on the open market a 5% tax is charged. Tax collected from token acquisitions is swapped for ETH while the remaining tax stays as SAGE.

  From here tokens are split into

  - Protocol owned liquidity [20%]
  - Protocol treasury [60%]
  - Staking contract [20%]

1.2 Telesages share trading fee

Trading shares on the upcoming Telesages contract generated a trading fee income for the protocol in ETH. This income is added to the SAGE staking contract in full.

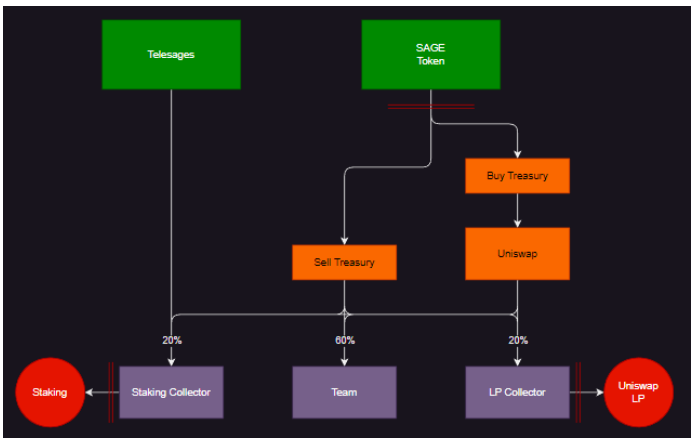

Figure 1: SAGE ecosystem revenue flow

2 Staking Model

2.1 The "StakePool"

Assets that are being sent to the Staking contract are distributed to the StakePool instantly or streamed slowly over a given duration. Each staker receives the as much of a share of the rewards as he hold shares in the StakePool.

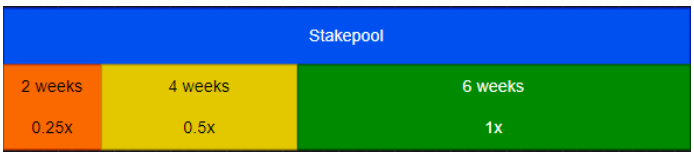

Figure 2: Stakepool composition example

Not all tokens added to the StakePool are weighted equally. Depending on the lockup duration chosen by the staker a certain factor is multiplied to the amount of SAGE staked.

The specific durations and factors available are dynamic and can be changed by the protocol at any time. Changing these values does not however have any effect on historical yield and only applies for all rewards added to the staking contract past the change.

2.1.1 Example Calculation

As an example lets consider a StakePool with the lockup-durations shown in 2. There are 3 stakers in the pool.

- UserA: 1000 SAGE staked for 2 weeks
- UserB: 1000 SAGE staked for 4 weeks
- UserC: 250 SAGE staked for 6 weeks

Now 0.5 ETH of yield is added to the pool without a streaming period (instantly). Based on the duration factors and staked tokens the users will receive:

- UserA: 1000 x 0.25 x 0.5 = 0.125 ETH
- UserB: 1000 x 0.5 x 0.5 = 0.25 ETH
- UserC: 250 x 1 x 0.5 = 0.125 ETH

2.2 Staking

Staking your SAGE tokens works by sending a transaction calling the respective stake method on the staking contract.

```
function stake(uint256 amount, uint256 duration) external;
```

This function will transfer the amount of tokens specified in the call from your account to an escrow contract specifically created for each staker. This escrow is used to delegate voting power of the staked tokens back to the staker.

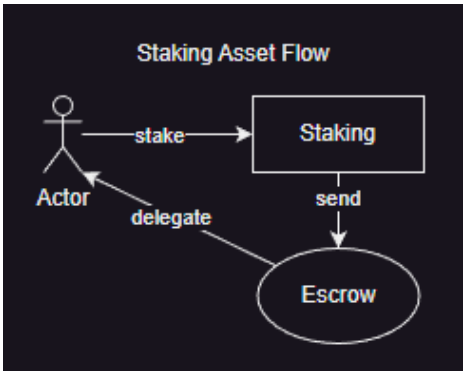

Figure 3: Staking transaction flow

Once the tokens are in the escrow the only way to get them back out is by unstaking and withdrawing them. There is no expiry of the stake and no renewal is required.

While the tokens are staked and not pending for unstaking they will be eligible for rewards using the distribution discussed in 2.1. Rewards will accrue automatically and do not have an expiry time.

2.3 Unstaking & Withdrawing

If a staker decides to exit the staking contract it's necessary to unstake and withdraw the tokens. This is the only way for funds to exit the contract.

To do so the staker needs to send an unstake transaction by calling the unstake method on the staking contract.

```
function unstake(uint256 duration) external;
```

This method will

- Claim all outstanding unclaimed rewards
- Mark the tokens of the specific staker in the specific lockup-duration as ineligible for rewards
- Start a countdown as long as the lockup-duration for withdrawal

At this point the staker will have to wait for the full lockup-duration. It is not possible to cancel that unstaking process or create a new stake in the same duration that is currently unstaking. Once the wait time is complete the withdraw method can be called on the staking contract.

```
function withdraw(uint256 duration) external;
```

This method will transfer all the funds from the escrow contract back to the users wallet and unblocks the staker from staking new funds in that given lockup-duration.

3 Claiming / Compounding Rewards

Rewards in the staking contract cannot expire and do not have any vesting period. Stakers can decide to claim their outstanding rewards at any time by calling the claim method on the staking contract.

```
function claim(uint256 duration) external;
```

This will claim rewards for all the reward assets (SAGE, ETH) distributed to the staker in the given lockup-duration.

4 Adding rewards / Streaming rewards

The staking contract supports all ERC20 tokens as well as ETH as staking rewards. In order to prevent users from adding potential spam or scam reward tokens the reward addition requires the rewarder to be explicitly whitelisted. This whitelist can be changed at any time by the protocol, see 5.

4.1 Reward streaming

When adding a reward to the contract rewarders can specify a duration over which the reward should be streamed to users. If the duration specified is 0, all rewards will be immediately claimeable by stakers.

However in order to make the yield more predicteable and avoid having to constantly add rewards it can be beneficial to specify a longer streaming duration instead.

These streamed rewards will be linearily unlocked. That means that after adding the reward at first no rewards can be claimed. After half of the specified stream durations half of the rewards are claimeable and so on.

Whenever the StakePool (see 2.1) changes in size or rewards are claimed, all streaming rewards need to be settled. This requires going over all running reward streams on-chain. In order to prevent high gas cost for users it should be avoided to run more than 3 simultaneous reward streams.

5 Administrative access & Permissions

The staking contract comes with 3 roles / permissions that can be held by other addresses

- **5.0.1 Admin**

  Permission to grant roles to other addresses

- **5.0.2 Rewarder**

  Permission to add reward tokens / ETH to the contract

- **5.0.3 Lockup Manager**

  Permission to change the available lockup-durations as well as their factors

6 Technical Details

6.1 Gas efficient reward distribution

Operations on-chain cost gas and can quickly exceed an affordeable range. In order to be able to distribute rewards to any amount of stakers without having to iterate over all of them an accounting trick is used.

The contract does not store outstanding rewards for each staker but their valuePerShare instead. Whenever rewards are added to the staking contract, the global valulePerShare is increased.

Whenever a staker claims rewards the outstanding rewards are (simplified):

```
(globalValuePerShare[asset] - lock.valuePerShare[asset]) * tokens * factor
```

After claiming the lock's valuePerShare will be increased to the global value.

6.2 Dynamic factor changes

The mechanism shown in 6.1 breaks down if the factor can dynamically change. In order to allow changing the factor without having to iterate over all stakers, factor changes create a snapshot of the globalValuePerShare.

When trying to claim reward for a lock that is using a previous factor, the claim will only use the difference till the next factor snapshot. It will continue going from snapshot to snapshot till the latest is reached again.

This allows for accrued rewards to stay untouched even if a factor is changed.