Variables and Mappings

Notation

Stored quantities

Variables

- Σ - Total amount staked (max 1e12 * 1e18)
- S - Total number of shares in circulation (1e30)
- R - Total amount of unstaked rewards on the validator (1e30)
- Δ - Total Claimed Rewards (rewards are automatically claimed on deposit or unbond at the validator). (1e18 x log_10(theta_R))

Constants

- ϕ - fee amount. Set this at 10% initially. (precision set to 1e4)
- θ_R - threshold amount for restaking. Set to 10,000 MATIC.
- θ_Δ - threshold amount for staking claimed rewards. Set initially to 10,000 MATIC. Should be higher than θ_R if the gas cost to stake is higher than restake.

User balances

User balances are stored in a mapping. Users fall into several types dependent on their role.

- **Owner:** deployer account, can call special setter functions
- **Reserve:** this account is used so that users can always withdraw money. Balance should be greater than or equal to $\theta_\Delta + \theta_R$ at inception. This reserve can actually be the Treasury.
- **Treasury**: the account into which fees are passed (multisig).
- Ordinary users' share holdings: User_1 : s_1, … , User_k : s_k (1e18)

Allocations

Allocations essentially are a way of keeping track of allocated amounts and the share prices at which they were allocated. The distributor for a given allocation is under no obligation to actually distribute the rewards and can delete the allocation at their leisure.

Note: We used to have 2 different settings for allocations: strict and loose. Only loose remains but for upgradability and backward compatibility, we have had to keep a boolean (always set to false) in a few mapping objects, namely `recipients`, `distributors` and `allocations`.

- **Allocations**: this stores all the amounts allocated and the TruMATIC / MATIC exchange rate at which they were allocated.

distributor (address)	recipient (address)	amount (MATIC)	price (MATIC per TruMATIC)
user_1	user_10	10000	1.02
user_12	user_7	100	1.02778

  - $\alpha_d(r)$ (1e18) the amount in MATIC that distributor d has allocated to recipient r.
  - $\pi_d(r)$ (numerator 1e36, denominator 1e18) the average share price at which the allocation from d to r occurred.

- **Total Allocated:** For each user it's important to store the total amount allocated. This will be needed when users want to withdraw etc (plus will need to be shown on the front end).
  - $\alpha_d := \sum_r \alpha_d(r)$ **total allocated** by user d across all recipients. This is a MATIC amount (1e30).
  - $\frac{1}{\pi_d} := \frac{1}{\alpha_d} \sum_r \alpha_d(r) \cdot \frac{1}{\pi_d(r)}$ **average price** at which user d allocated. Note, this isn't always correct due to rounding errors after `distribute` call. ***It should never be used to calculate money to be moved, only when performing checks.***

- **Distribution Fee:** ψ the proportion of each distribution that's taken as a fee.

💾 *Data Structures for Allocations*

1. `allocations = mapping(address=>(address=>struct(amount, price))`
   maps distributor⇒recipient⇒[amount,price]
2. `recipients = mapping(distributor=>[recipient array])`
   maps each distributor to an array of non-zero recipients.
3. `distributors = mapping(recipient=>[distributor array])`
   maps each recipient to an array of non-zero distributors.
4. `totalAllocated = mapping(distributor=>(bool=>[totalAllocated,priceAllocated]))`

   maps each distributor to the total MATIC they have allocated and the net price at which they have allocated it.

Calculated Quantities

These aren't stored anywhere, rather calculated on the fly as needed.

- `Share price` $P = \frac{\Sigma+\Delta+(1-\phi)R}{S}$ (stored as numerator $\overline{P}$ (1e52) and denominator $\underline{P}$ (1e34))
- `Dust` $D = \phi R$. Note this is also equal to $(\Sigma + \Delta + R) - PS$, i.e. The failure of the total amount staked plus rewards to equal the number of outstanding shares times the share price.