图示

描述已自动生成

**Pool**

池合约是存入 StakeWise 池的切入点。该合约存储从用户那里收集的ETH，然后再发送到ETH2验证人注册合约。

**StakedEthToken**

StakedEthToken是ERC-20合约。它反映了质押者以sETH2代币的形式进行的存款。代币以 1 比 1 映射到 ETH。sETH2 的总供应量是所有 StakeWise 池验证者有效余额的总和，加上等待纳入新验证器的额外金额（32 ETH - 1 Wei）。

**RewardEthToken**

RewardEthToken是ERC-20合约。它反映了质押者以 rETH2 代币形式积累的奖励。代币以 1 比 1 映射到 ETH。rETH2的总供应量是高于为StakeWise池注册的所有验证者的有效余额的金额。

**Oracle**

预言机合约存储负责根据链下数据提交或更新值的账户。

研究一下这个reth怎么产生怎么更新的， 自己做一个这种流动性小额质押挖矿，需要做些什么，部署哪些合约，然后，这个小额挖矿的背后， 肯定是有真实的validator的，怎么对应起来的

Reth: 根据seth2的份额，分配reward,余额监控合约检查验证者合约的余额，余额增加，reth增加，余额减少，reth减少，当reth减少到0，就开始扣seth

通过pool合约收集eth,当eth达到32，注册验证者节点，加入注册序列，存储验证者公钥并且监控其余额

# pool

The [Pool](https://github.com/stakewise/contracts/blob/master/contracts/collectors/Pool.sol) contract is an entry point for deposits into the StakeWise Pool. This contract stores ETH collected from the users before it is sent to the [ETH2 Validator Registration Contract](https://github.com/ethereum/eth2.0-specs/tree/dev/solidity_deposit_contract).

## Add Deposit

通过在池合约中调用 [addDeposit](https://github.com/stakewise/contracts/blob/master/contracts/collectors/Pool.sol#L70) 函数 ，用户将以太币添加到池中。合约将以 存入的金额以 1 比 1 铸造 stETH 代币。例如，如果您存入 10 个以太币，您将获得 10 stETH，代表您在 StakeWise 池中质押的 ETH 数量。

图示

描述已自动生成

## Register Validator

当池合约中收集了 32 个或更多 ETH 时，具有操作员角色的用户可以注册新的验证者。如果收集了足够的以太币，则可以在一个批次中创建多个验证器。为了注册新的验证器，操作员必须为每个验证器生成[存款数据](https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/deposit-contract.md#deposit-function)，并通过调用[registerValidator](https://github.com/stakewise/contracts/blob/master/contracts/collectors/Pool.sol#L85)函数提交。请注意，StakeWise 池将使用自己的[提款](https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/deposit-contract.md" \l "withdrawal-credentials)凭据，对于每个池的验证器，这些凭据保持不变。提款凭据存储在[“设置](https://github.com/stakewise/contracts/blob/master/contracts/Settings.sol#L74)”合约中，私钥在受信任的实体之间分发并存储在冷存储中。一旦支持撤回到 ETH1 地址，这种托管方法将被不同的机制取代。

运营商为每个验证人生成存款数据并启动验证人注册

操作员在群集中运行并侦听池的事件。一旦收集到足够的ETH，运营商就会启动验证者节点，并通过池合约注册他们的存款数据 。验证人合约跟踪池合约和独奏合约注册的所有验证人 。它拒绝使用已使用的公钥注册验证器，因为每个公钥应该只有一个验证器注册调用。[ETH2验证人](https://github.com/ethereum/eth2.0-specs/tree/dev/solidity_deposit_contract)注册合同是注册新验证人的正式合同。

# Solos

StakeWise Solos smart contract details.

[Solos](https://github.com/stakewise/contracts/blob/master/contracts/collectors/Solos.sol)合约是StakeWise Solo存款的切入点。该合约存储从用户那里收集的ETH，然后再发送到[ETH2验证人注册合约](https://github.com/ethereum/eth2.0-specs/tree/dev/solidity_deposit_contract)。

该合约 **不可升级** ，以防止 StakeWise 通过升级机制单方面更改提款凭证。

## Add Deposit

要创建独立验证器，用户的余额必须至少有 32 个以太币。除了存款金额外，用户还必须提供可以按照我们的[指南](https://docs.stakewise.io/how-to/generate-credentials)生成的提款凭据。通过在 Solos 合约中调用 [addDeposit](https://github.com/stakewise/contracts/blob/master/contracts/collectors/Solos.sol#L60) 函数 ，用户将被置于验证人注册队列中。对于用户存入的每 32 个 ETH，将创建一个新的验证器。

## Cancel Deposit

用户可以通过调用[cancelDeposit](https://github.com/stakewise/contracts/blob/master/contracts/collectors/Solos.sol" \l "L87)函数取消存款，如果存入的以太币在[提款LockDuration](https://github.com/stakewise/contracts/blob/master/contracts/Settings.sol#L98)变量指定的时间段内没有转移到[ETH2验证人注册合约](https://github.com/ethereum/eth2.0-specs/tree/dev/solidity_deposit_contract)（目前设置为1天），则将其提取回去。

操作员需要时间锁定，以便在用户不中断注册过程的情况下注册验证器。例如，恶意用户可能会发送取消存款的交易，并且运营商会通过尝试注册取消的存款而失去气体。

## Register validators

当 Solos 合约中有存款时，具有操作员角色的用户可以注册新的验证人。如果有多个存款，则可以在一个批次中创建多个验证器。为了注册新的验证人，运营商必须 为每个验证人生成[存款数据](https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/deposit-contract.md" \l "deposit-function)，并通过[registerValidator](https://github.com/stakewise/contracts/blob/master/contracts/collectors/Solos.sol#L115)函数提交。运营商不能使用与用户提交的不同的提款凭证，因为[ETH2验证人注册合同](https://github.com/ethereum/eth2.0-specs/tree/dev/solidity_deposit_contract)将拒绝交易。

图示

描述已自动生成

Operator generates deposit data for every validator and initiates validator registration

操作员在集群中运行并侦听 Solos 的事件。一旦收集到足够的ETH，运营商就会启动验证者节点，并通过Solos合约注册他们的存款数据。验证人合约跟踪池合约和独奏合约注册的所有验证人 。它拒绝使用已使用的公钥注册验证器，因为每个公钥应该只有一个验证器注册调用。[ETH2验证人](https://github.com/ethereum/eth2.0-specs/tree/dev/solidity_deposit_contract)注册合同（VRC）是注册新验证人的正式合同。

# StakedEthToken

StakeWise StakedEthToken smart contract details.

[StakedEthToken](https://github.com/stakewise/contracts/blob/master/contracts/tokens/StakedEthToken.sol)是ERC-20合约。它反映了质押者以stETH代币的形式进行的存款。代币以 1 比 1 映射到 ETH。

stETH 的总供应量是所有 StakeWise 池验证者有效余额的总和，加上等待纳入新验证器的额外金额（32 ETH - 1 Wei）。 ​

## Emission

stETH 代币在每次[池存款](https://github.com/stakewise/contracts/blob/master/contracts/collectors/Pool.sol#L70)上以 1 比 1 铸造到存入的 ETH 金额。 质押者持有的 stETH 数量将决定它们累积多少 rwETH。如果质押者的 rwETH 余额低于 0，则会从 stETH 余额中扣除。一旦 rwETH 余额恢复到 0，质押者的 stETH 余额将恢复到其初始值。

### An example of tokens' emission:

1. 1.

5 ETH 的存款被添加到 StakeWise 池中。StakedEthToken合约的总供应量现在是5 stETH。

1. 2.

28 ETH 的存款被添加到 StakeWise 池中。StakedEthToken合约的总供应量现在为32 stETH。新的验证程序现在正在为池运行。

1. 3.

验证者的余额已从 32 ETH 更改为 33 ETH。[余额报告员](https://github.com/stakewise/contracts/blob/master/contracts/access/BalanceReporters.sol)合约向RewardEthToken合约提交1 ETH的总奖励 。质押EthToken的总供应量保持32 stETH，RewardEthToken的总供应量为1 rwETH。

1. 4.

验证者的余额从 33 ETH 下降到 31 ETH。[BalanceReporters](https://github.com/stakewise/contracts/blob/master/contracts/access/BalanceReporters.sol) 合约向 RewardEthToken 合约提交 -2 ETH 的总奖励 。质押乙醚令牌的总供应量变为31 stETH，奖励币的总供应量变为 0 rwETH。

1. 5.

验证者余额从 31 ETH 增加到 35 ETH。[余额报告员](https://github.com/stakewise/contracts/blob/master/contracts/access/BalanceReporters.sol)合约向RewardEthToken合约提交3 ETH的总奖励 。质押乙醚代币的总供应量变为32 stETH，奖励币的总供应量变为3 rwETH。

# RewardEthToken

StakeWise RewardEthToken smart contract details.

[RewardEthToken](https://github.com/stakewise/contracts/blob/master/contracts/tokens/RewardEthToken.sol)是ERC-20合约。它反映了质押者以 rwETH 代币形式积累的奖励。代币以 1 比 1 映射到 ETH。

rwETH 的总供应量是高于注册 StakeWise 池的所有验证者的有效[余额](https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase0/beacon-chain.md" \l "validator)的金额。 ​ ​

## Checkpoints

检查点用于跟踪质押者先前累积的奖励并计算当前奖励。在以下情况下更新检查点：

1. stETH 数量增加了。质押者开始获得更大比例的奖励，因为他现在持有更多的 stETH。
2. stETH数量有所下降。质押者开始获得较小比例的奖励，因为他现在持有的 stETH 较少。
3. 质押者已将 rwETH 代币转移到不同的地址。质押者将继续像往常一样赚取奖励，但他的 rwETH 余额将减少转移的 rwETH 金额。

Every checkpoint stores:

1. rewardPerToken - 每个 stETH 代币赚取的 rwETH 代币数量
2. 奖励 - 质押者根据奖励每代币计算的 rwETH 总余额

## Emission

矿池的奖励由[余额报告者](https://github.com/stakewise/contracts/blob/master/contracts/access/BalanceReporters.sol)合约使用更新[总奖励](https://github.com/stakewise/contracts/blob/master/contracts/tokens/RewardEthToken.sol#L123)功能定期更新。当总奖励更新时， 将计算[奖励每代币](https://github.com/stakewise/contracts/blob/master/contracts/tokens/RewardEthToken.sol" \l "L34)的新值。

如果奖励令牌小于前一个值，则意味着矿池的验证者受到了惩罚。每个质押者的当前奖励都会按比例减少，以达到他们在 stETH 总供应中的份额。如果质押者的 rwETH 金额降至 0 以下，则负值将从其 stETH 余额中扣除。一旦质押者的 rwETH 金额增长到 0 以上，stETH 余额就会恢复到其初始值（请参阅[此处](https://docs.stakewise.io/smart-contracts/stakedethtoken)的示例）。

如果奖励令牌大于前一个值，则意味着池的奖励已增加。 然后，每个 stETH 持有者的 rwETH 代币数量将根据他们在 stETH 总供应中的份额增加。

调用 [rewardOf](https://github.com/stakewise/contracts/blob/master/contracts/tokens/RewardEthToken.sol#L84) 函数时，每个质押者的 rwETH 代币总量根据以下公式计算：

文本

描述已自动生成

where:

文本, 信件

描述已自动生成

# StakedTokens

StakeWise StakedEthToken smart contract details.

[质](https://github.com/stakewise/contracts/blob/master/contracts/tokens/StakedTokens.sol)押代币合约可用于锁定支持的 ERC-20 代币，以允许质押者在将 stETH 转移到外部 ERC-20 合约时继续积累 rwETH。

## Toggling tokens support

具有管理员角色的地址可以使用切换令牌合约功能启用/禁用对 ERC-20 [合约](https://github.com/stakewise/contracts/blob/master/contracts/tokens/StakedTokens.sol#L65)的支持。启用代币支持后，质押者可以在质押代币合约中锁定支持的 ERC-20 代币 。如果代币支持被禁用，质押者可以提取他们锁定的代币和奖励，但 rwETH 停止为禁用的 ERC-20 合约累积。为禁用的ERC-20合约质押新代币将被禁止。添加禁用功能是为了以防受支持的合同开始行为异常。

## Staking tokens

假设质押者已将 stETH 和 ETH 存入 stETH/ETH Uniswap 流动性池，并收到 Uniswap LP 代币作为回报。为了继续 从存入的 stETH 中赚取 rwETH，质押者需要通过调用质[押代币](https://github.com/stakewise/contracts/blob/master/contracts/tokens/StakedTokens.sol#L83)函数将其 LP 代币锁定在质押代币合约中 。当 LP 代币被锁定在质押代币合约中时，质押者能够从中提取他的那部分 rwETH。他可以提取的金额是根据他在质押代币合约中锁定的 LP 代币总量中的份额计算的 。

## Withdrawing tokens

质押者可以随时通过调用 [withdrawTokens](https://github.com/stakewise/contracts/blob/master/contracts/tokens/StakedTokens.sol#L110) 函数来提取其锁定的代币。累积的 rwETH 将被转移到质押者的地址。

## Withdrawing rewards

每次质押者锁定或解锁代币时，累积的 rwETH 都会自动提取。也可以 通过调用[提款](https://github.com/stakewise/contracts/blob/master/contracts/tokens/StakedTokens.sol#L137)奖励函数来提取累积的 rwETH。

图示

描述已自动生成

图示

描述已自动生成