Lattice Reward Staking Contract

Overview

The Lattice Gateway is introducing a new feature providing staking pools for users to deposit tokens and gather rewards. This functionality also supports projects seeking to leverage the Lattice platform to incentivize users to stake tokens or offer an additional yield to those who have already contributed liquidity to their projects.

For each project interested in this service, the Lattice team will deploy an updated staking smart contract with suitable pool parameters. In collaboration with the respective project teams, the Lattice team will establish the staking pools. Pool parameters will be customized to each project's unique requirements, ensuring a versatile and adaptable solution.

The Lattice Fixed Reward Staking Contract has been developed as a Solidity smart contract that allows users to stake tokens and accumulate rewards over time. This contract is inspired by SushiSwap's Staking Contract, and this document offers a technical overview of the smart contract's intended features.

Roles

The Lattice Fixed Reward Staking Contract incorporates roles to ensure proper functionality and control over the contract. The two roles in the contract are the ConfigurationRole and the StewardRole.

Configuration Role

The Configuration Role is assigned to the contract owner or deployer, who has the responsibility of setting up the initial parameters for the staking contract.

Responsibilities of the Configuration Role include:

- Initializing the staking program's parameters, such as staking and reward tokens, minimum staking and reward amounts, program timeline, and tax ratios, through the initializeProgram() function.
- Modifying the staking and reward tokens, minimum staking and reward amounts, and the subsequent reward amount change using the configureProgramCondition() function.
- Adjusting the tax ratio through the configureProgramTax() function.

Steward Role

Project teams that utilize the staking contract to incentivize their users are granted the StewardRole. All stewards, being part of the same team, share the responsibility of coordinating their actions to manage the staking pool effectively. They hold the authority to control reward deposits, withdraw undistributed rewards, modify program start and end dates, and claim stuck ERC20 tokens. Additionally, it is up to the stewards to decide when and how to interact with users while performing any actions related to the staking pool.

Responsibilities of the Steward Role include:

- Modify the program's start and end dates, enabling project teams to adjust the duration of the staking program as per the project's requirements.
- Depositing additional rewards into the contract serves to maintain the pool's appeal and encourages continued user participation.
- Withdraw undistributed rewards from the staking pool. This action may be performed at any time, and users should be aware that it could lead to potential balance fluctuations.
- Claim any ERC20 tokens that may become stuck in the contract. This action helps to ensure that no tokens are lost due to unforeseen circumstances.

Key Features

- Staking of ERC20 tokens
- Reward distribution in ERC20 tokens
- Configurable staking program conditions
- Configurable program timelines
- Tax on rewards
- Withdrawal and claiming of rewards
- Role-based access control

Contract Components

- **Staking Token:** The ERC20 token that users stake in the contract, configurable by the contract owner.
- **Minimum Staking Amount:** The minimum amount of staking tokens that a user must stake in the contract.
- **Reward Token:** The ERC20 token is distributed as a reward to users who stake tokens in the contract, configurable by the contract owner.
- **Minimum Reward Amount:** The minimum amount of reward tokens that can be claimed by a user.

- **Program Start and End Dates:** The timeline for the staking program is configurable by the contract owner.
- **Tax Ratio:** The ratio of taxes deducted

Functions

- **constructor():** Initializes the contract with staking and reward token addresses, minimum amounts, program start and end dates, tax ratio, and manager addresses.
- **configureProgramConditions():** Allows the contract owner to modify the staking and reward tokens, minimum staking and reward amounts, and the next reward amount change.
- **configureProgramTimeline():** The contract owner can modify the program start and end dates.
- **configureProgramTax():** Allows the contract owner to modify the tax ratio.
- **stake():** Allows users to stake tokens into the contract and optionally claim existing rewards.
- **withdraw():** Allows users to withdraw their staked tokens from the contract and optionally claim existing rewards.
- **claimRewards():** Allows users to claim their earned rewards from the contract.
- **availableRewards():** Calculates the available rewards and taxes for a user based on their staked amount and current program reward per liquidity