VeraLux / v0.1 - 24/04/2025

# Introduction

This document serves as a comprehensive guide to the VeraLux smart contract, a Solana-based decentralized application that manages the lifecycle, governance, and security of the VeraLux token.

This document is tailored for developers, auditors, and users seeking an in-depth understanding of the contract’s functionality, tokenomics, and operational mechanisms.

Its purpose is to offer both a high-level perspective and detailed technical insights into the VeraLux ecosystem.

In this document, you will find:

* A detailed breakdown of the smart contract’s state, including key data structures and their roles.
* An overview of the accounts used to manage state, user data, and governance processes.
* A thorough explanation of the contract’s instructions (functions), including their parameters, behavior, and constraints.
* Insights into the tokenomics, covering supply, distribution, and economic mechanisms.
* A comprehensive look at the governance system, detailing how stakers influence contract parameters.
* An in-depth analysis of security measures implemented to safeguard the contract and its users.
* A glossary of key terms and appendices with constants, calculation examples, and error codes for reference.

This document is structured to ensure accessibility, providing a broad overview for newcomers while offering the technical depth required for advanced readers.

# Smart Contract Overview

The VeraLux smart contract is a Solana-based program that governs the VeraLux token, a utility token engineered to enable decentralized governance, staking, and liquidity provision within the VeraLux ecosystem. Built with an emphasis on security, transparency, and community empowerment, the contract allows token holders to actively participate in shaping the project’s direction.

## State

The smart contract’s state is the foundation of its functionality, storing all critical data that defines its behaviour and interactions. This includes configuration settings, user balances, transaction limits, and governance details. Understanding the state is essential for developers, auditors, and users to comprehend how the contract operates.

#### Pool Type Enum

* Purpose: Categorizes treasury pools for token allocation.
* Values:
  + Staking – Tokens allocated for staking rewards.
  + Airdrop – Tokens for airdrop distributions.
  + Governance – Tokens for governance-related expenses.
  + Marketing – Tokens for marketing initiatives.
  + Emergency – Tokens reserved for unforeseen needs.
  + LiquidityIncentive – Tokens for liquidity provider incentives.
  + Team – Tokens allocated to the project team.
* Attributes: Implements AnchorSerialize, AnchorDeserialize, Clone, Copy, PartialEq, and Eq for serialization and comparison functionality.

#### Global Configuration Structures

###### ContractState Struct

* **Purpose:** Central storage for the contract’s global configuration and operational settings.
* **Fields:**
  + admin: Pubkey – Multisig account with administrative privileges.
  + paused: bool – Indicates whether the contract is paused.
  + pause\_reason: String – Reason for pausing (max length: 100 bytes).
  + treasury: Pubkey – Address of the treasury wallet.
  + charity\_wallet: Pubkey – Wallet for charity donations.
  + team\_wallet: Pubkey – Wallet for team allocations.
  + liquidity\_pool: Pubkey – Liquidity pool wallet address.
  + proposal\_count: u64 – Total number of governance proposals.
  + total\_voting\_power: u64 – Sum of all stakers’ voting power.
  + launch\_timestamp: i64 – Start time for presale vesting (Unix timestamp).
  + tax\_rate: u64 – Base transaction tax rate (in basis points).
  + staking\_tiers: [u64; 4] – Token thresholds for staking tiers.
  + Tax Allocations (basis points):
    - burn\_alloc: u64 – Allocation for token burning.
    - treasury\_alloc: u64 – Allocation to the treasury.
    - liquidity\_pool\_alloc: u64 – Allocation to the liquidity pool.
    - lp\_incentive\_alloc: u64 – Allocation for LP incentives.
    - charity\_alloc: u64 – Allocation for charity.
    - team\_alloc: u64 – Allocation for the team.
  + reduction\_thresholds: [u64; 3] – Thresholds for pool depletion triggering reward reductions.
  + reduction\_factors: [u64; 4] – Multipliers for reward reductions.
  + dex\_programs: Vec<Pubkey> – List of DEX program IDs for sell detection (max: 10).
  + whitelisted\_contracts: Vec<(Pubkey, [u8; 32])> – Whitelisted contracts with version hashes (max: 20).
  + allowed\_destinations: Vec<Pubkey> – Allowed transfer destinations (max: 10).
  + Whitelisted Contract Versioning:
    - - Stored as `(Pubkey, [u8; 32])` pairs (contract ID, version hash).
    - - Checked in `whitelisted\_transfer` against the caller’s program ID.
    - - Fails with `ErrorCode::VersionMismatch` if hashes don’t match.
    - - Ensures only verified versions of external contracts can perform privileged operations.
  + Transaction Limits:
    - max\_sell\_txn\_limit: u64 – Maximum tokens per sell transaction.
    - daily\_sell\_limit: u64 – Daily sell limit per user.
    - max\_transfer\_limit: u64 – Maximum tokens per non-sell transfer.
    - daily\_transfer\_limit: u64 – Daily transfer limit per user.
  + progressive\_tax\_threshold: u64 – Threshold for applying higher tax rates.
  + staking\_rewards: [u64; 4] – Weekly rewards per staking tier.
  + presale\_usdt\_receiver: Pubkey – USDT receiver address for presale.
  + presale\_active: bool – Indicates if the presale is active.
  + total\_presale\_sold: u64 – Total tokens sold in the presale.
  + last\_processed\_day: i64 – Last day rewards were processed (Unix timestamp).
  + is\_processing: bool – Flag to prevent reentrancy during reward processing.
  + last\_processed\_index: u64 – Index for batch reward processing.
* **Constants:**
  + MAX\_DEXES: usize = 10 – Maximum number of DEX programs.
  + MAX\_WHITELISTED: usize = 20 – Maximum number of whitelisted contracts.
  + MAX\_ALLOWED\_DESTINATIONS: usize = 10 – Maximum number of allowed destinations.
  + MAX\_PAUSE\_REASON\_LEN: usize = 100 – Maximum length of pause reason.
  + LEN: usize – Total size for on-chain storage (calculated based on fields).

Token Precision and Decimals

* Purpose: Ensures accurate representation and calculation of token amounts for all operations like transfers, staking, and rewards.
* **Parameters:** 
  + Defined as TOKEN\_DECIMALS = 9 in Constants.rs.
  + Token amounts are scaled by 10^9 (e.g., 1 token = 1\_000\_000\_000 units).
* **Validation:** 
  + All token-related inputs and outputs must account for this 9-decimal precision.
  + No explicit validation in code; it’s a foundational constant assumed correct in calculations.
* **Impact:** 
  + Prevents rounding errors in fractional token amounts, ensuring fairness in tax distributions, staking rewards, and presale purchases.
  + Users must scale inputs (e.g., 20,000 tokens = 20\_000 \* 10^9) when interacting with the contract.

###### Treasury Struct

* Purpose: Tracks token allocations across different treasury pools.
* Fields:
  + staking\_pool: u64 – Pool for staking rewards.
  + airdrop\_pool: u64 – Pool for airdrop distributions.
  + governance\_reserve: u64 – Pool for governance expenses.
  + marketing\_fund: u64 – Pool for marketing initiatives.
  + emergency\_fund: u64 – Pool for emergency needs.
  + liquidity\_incentive: u64 – Pool for liquidity provider incentives.
  + team\_pool: u64 – Pool for team allocations.
* Constants:
  + LEN: usize = 8 + 8 \* 7 – Total size (64 bytes).

###### Multisig Struct

* Purpose: Configures the multisig mechanism for administrative actions.
* Fields:
  + owners: Vec<Pubkey> – List of multisig owners (max: 5).
  + threshold: u8 – Minimum number of signatures required.
* Constants:
  + MAX\_OWNERS: usize = 5 – Maximum number of owners.
  + LEN: usize = 8 + 4 + 32 \* MAX\_OWNERS + 1 – Total size (173 bytes).

###### MigrationState Struct

* Purpose: Manages the global state of token migration.
* Fields:
  + total\_locked: u64 – Total tokens locked for migration.
  + migration\_active: bool – Indicates if migration is active.
  + migration\_toggle\_timestamp: i64 – Timestamp of the last migration toggle.
* Constants:
  + LEN: usize = 8 + 8 + 1 + 8 – Total size (25 bytes).

#### 3. User Data Structures

###### 3.1 Staker Struct

**Purpose:** Tracks staking data for individual users.

**Fields:**

* + tier: u8 – Staking tier (0-3).
  + amount: u64 – Amount of tokens staked.
  + start\_time: i64 – Time staking began (Unix timestamp).
  + last\_claim: i64 – Time of last reward claim.

**Constants:**

* + LEN: usize = 8 + 1 + 8 + 8 + 8 – Total size (33 bytes).

###### 3.2 LPStaker Struct

**Purpose:** Manages data for liquidity provider staking.

**Fields:**

* + amount: u64 – Amount of LP tokens staked.
  + last\_action\_time: i64 – Time of last staking action.
  + unclaimed\_rewards: u64 – Accumulated unclaimed rewards.

**Constants:**

* + LEN: usize = 8 + 8 + 8 + 8 – Total size (32 bytes).

###### TransactionRecord Struct

**Purpose:** Enforces transaction limits (sell and transfer) for users.

**Fields:**

* + last\_txn\_time: i64 – Time of the last transaction.
  + sell\_buckets: [u64; 24] – Hourly sell amounts for the past 24 hours.
  + transfer\_buckets: [u64; 24] – Hourly transfer amounts for the past 24 hours.
  + current\_bucket\_index: u8 – Current hour index (0-23).
  + bucket\_start\_time: i64 – Start time of the current bucket.
  + sell\_cooldown\_start: i64 – Start time of sell cooldown.
  + transfer\_cooldown\_start: i64 – Start time of transfer cooldown.

**Constants:**

LEN: usize = 8 + 8 + 8\*24 + 8\*24 + 1 + 8 + 8 + 8 – Total size (457 bytes).

###### Transaction Bucket System

**Purpose:** Enforces daily limits on sell and non-sell transfers to prevent market manipulation or excessive token movement.

**Parameters:**

- Defined in `TransactionRecord` with `sell\_buckets` and `transfer\_buckets` (24 hourly slots each).

- Updated via `advance\_buckets` function based on elapsed time.

- Limits: `state.daily\_sell\_limit` and `state.daily\_transfer\_limit` (initially 0.5% of total supply).

**Validation:**

- Ensures total hourly bucket sums don’t exceed daily limits (e.g., `daily\_sell <= state.daily\_sell\_limit`).

- Fails with `ErrorCode::DailySellLimitExceeded` or `ErrorCode::DailyTransferLimitExceeded` if violated.

**Impact:**

- Limits large token dumps or transfers, stabilizing price and liquidity.

- Provides granular hourly tracking within a 24-hour window, balancing flexibility and control.

###### PresalePurchase Struct

* Purpose: Tracks presale purchases per wallet.
* Fields:
  + wallet: Pubkey – Buyer’s public key.
  + total\_purchased: u64 – Total tokens purchased in the presale.
  + kyc\_verified: bool – Indicates if KYC is verified.
* Constants:
  + LEN: usize = 8 + 32 + 8 + 1 – Total size (49 bytes).

###### 3.5 PresaleVesting Struct

* Purpose: Manages vesting schedules for presale participants.
* Fields:
  + total\_amount: u64 – Total tokens vested.
  + claimed\_amount: u64 – Tokens claimed so far.
* Constants:
  + LEN: usize = 8 + 8 \* 2 – Total size (24 bytes).

###### 3.6 VoteRecord Struct

* Purpose: Records a staker’s vote on a governance proposal.
* Fields:
  + staker: Pubkey – Staker’s public key.
  + proposal\_id: u64 – ID of the proposal voted on.
  + voted: bool – Indicates if the staker has voted.
  + in\_favor: bool – Indicates if the vote was in favor.
* Constants:
  + LEN: usize = 8 + 32 + 8 + 1 + 1 – Total size (50 bytes).

###### 3.7 MigrationRecord Struct

* Purpose: Tracks individual user migration status.
* Fields:
  + user: Pubkey – User’s public key.
  + locked\_amount: u64 – Tokens locked for migration.
  + migrated: bool – Indicates if migration is complete.
  + migration\_confirmed: bool – Indicates if migration is confirmed.
* Constants:
  + LEN: usize = 8 + 32 + 8 + 1 + 1 – Total size (50 bytes).

#### 4. Vesting and Rewards Structures

###### 4.1 TeamVesting Struct

* Purpose: Manages vesting schedules for team members.
* Fields:
  + team\_member: Pubkey – Team member’s public key.
  + total\_amount: u64 – Total tokens vested.
  + claimed\_amount: u64 – Tokens claimed so far.
  + start\_time: i64 – Vesting start time (Unix timestamp).
  + canceled: bool – Indicates if vesting is canceled.
* Constants:
  + LEN: usize = 8 + 32 + 8 + 8 + 8 + 1 – Total size (65 bytes).

###### 4.2 FreelancerVesting Struct

* Purpose: Tracks vesting schedules for freelancers.
* Fields:
  + freelancer: Pubkey – Freelancer’s public key.
  + total\_amount: u64 – Total tokens vested.
  + released\_amount: u64 – Tokens released for claiming.
  + claimed\_amount: u64 – Tokens claimed so far.
  + start\_time: i64 – Vesting start time (Unix timestamp).
  + last\_claim\_time: i64 – Time of last claim.
* Constants:
  + LEN: usize = 8 + 32 + 8 + 8 + 8 + 8 + 8 – Total size (80 bytes).

#### 5. Governance Structures

###### 5.1 Proposal Struct

* Purpose: Represents governance proposals submitted by stakers.
* Fields:
  + id: u64 – Unique proposal identifier.
  + description: String – Description of the proposal (max length: 200 bytes).
  + votes\_for: u64 – Total votes in favor.
  + votes\_against: u64 – Total votes against.
  + status: u8 – Proposal status (e.g., 0 = pending).
  + start\_time: i64 – Voting start time (Unix timestamp).
  + end\_time: i64 – Voting end time (Unix timestamp).
  + execution\_time: i64 – Time of proposal execution.
  + proposal\_type: u8 – Type of proposal.
  + proposal\_values: Vec<u64> – Values associated with the proposal (max: 7).
* Constants:
  + MAX\_DESCRIPTION\_LEN: usize = 200 – Maximum description length.
  + MAX\_PROPOSAL\_VALUES: usize = 7 – Maximum number of proposal values.
  + LEN: usize – Calculated size based on fields.

#### 6. Pending Actions Structures

###### 6.1 PendingWithdrawal Struct

* Purpose: Tracks pending treasury withdrawals with a delay.
* Fields:
  + amount: u64 – Amount to withdraw.
  + initiation\_slot: u64 – Slot when withdrawal was initiated.
  + delay\_slots: u64 – Number of slots to delay completion.
* Constants:
  + LEN: usize = 8 + 8 \* 3 – Total size (32 bytes).

###### 6.2 PendingWhitelistChange Struct

* Purpose: Manages pending changes to the whitelist with a time-lock.
* Fields:
  + contract: Pubkey – Contract to add or remove.
  + add: bool – True to add, false to remove.
  + initiation\_time: i64 – Timestamp of initiation.
* Constants:
  + LEN: usize = 8 + 32 + 1 + 8 – Total size (49 bytes).

###### 6.3 PendingMultisigChange Struct

* Purpose: Tracks pending changes to multisig configuration.
* Fields:
  + new\_owners: Vec<Pubkey> – Proposed new owners (max: 5).
  + new\_threshold: u8 – Proposed new threshold.
  + initiation\_time: i64 – Timestamp of initiation.
* Constants:
  + MAX\_OWNERS: usize = 5 – Maximum number of owners.
  + LEN: usize = 8 + 4 + 32 \* MAX\_OWNERS + 1 + 8 – Total size (181 bytes).

###### 6.4 PendingPause Struct

* Purpose: Manages pending pause actions with a time-lock.
* Fields:
  + reason: String – Reason for pausing (max length: 100 bytes).
  + initiation\_time: i64 – Timestamp of initiation.
* Constants:
  + MAX\_REASON\_LEN: usize = 100 – Maximum reason length.
  + LEN: usize = 8 + 4 + MAX\_REASON\_LEN + 8 – Total size (120 bytes).

###### 6.5 PendingResume Struct

* Purpose: Tracks pending resume actions with a time-lock.
* Fields:
  + initiation\_time: i64 – Timestamp of initiation.
* Constants:
  + LEN: usize = 8 + 8 – Total size (16 bytes).

## Accounts

Accounts in this smart contract are used to store and manage state, track user interactions, and enforce governance, staking, vesting, and security mechanisms. This section provides a comprehensive breakdown of all accounts defined in the contract, including their fields, purposes, and how they are utilized across instructions.

### Global State Accounts

These accounts hold the contract’s configuration, operational settings, and treasury allocations.

Program Derived Addresses (PDAs)

* Purpose: Secures sensitive operations by delegating authority to program-controlled accounts rather than external wallets.
* Parameters:
  + Examples include:
    - treasury\_pda: Seed b"treasury\_authority", used for treasury token transfers.
    - staking\_pda: Seed b"staking\_authority", used for staking operations.
    - lp\_staking\_pda: Seed b"lp\_staking\_authority", used for LP staking.
  + Each PDA has a unique bump seed for derivation.
* Validation:
  + PDAs are verified via seeds and bumps in account structs (e.g., seeds = [b"treasury\_authority"], bump).
  + Only the program can sign transactions using these PDAs.
* Impact:
  + Enhances security by ensuring only the contract’s logic can authorize critical actions like reward payouts or token burns.
  + Reduces reliance on external signers, minimizing risks of unauthorized access.

ContractState

* Purpose: Stores the global configuration and operational data for the contract, including multisig admin settings, tax rates, staking tiers, and transaction limits.
* Fields:
  + admin: Pubkey – Public key of the multisig account for administrative actions.
  + paused: bool – Flag indicating if the contract is paused.
  + pause\_reason: String – Reason for pausing the contract (max 100 bytes).
  + treasury: Pubkey – Treasury token account address.
  + charity\_wallet: Pubkey – Charity wallet address.
  + team\_wallet: Pubkey – Team wallet address.
  + liquidity\_pool: Pubkey – Liquidity pool wallet address.
  + proposal\_count: u64 – Total number of governance proposals.
  + total\_voting\_power: u64 – Sum of voting power from all stakers.
  + launch\_timestamp: i64 – Unix timestamp for presale vesting start (default: May 1, 2025).
  + tax\_rate: u64 – Base tax rate in basis points (default: 500 = 5%).
  + staking\_tiers: [u64; 4] – Token thresholds for staking tiers (default: [20K, 100K, 500K, 5M] tokens).
  + burn\_alloc: u64 – Tax allocation for burning tokens (default: 2000 bps).
  + treasury\_alloc: u64 – Tax allocation to treasury (default: 2000 bps).
  + liquidity\_pool\_alloc: u64 – Tax allocation to liquidity pool (default: 2400 bps).
  + lp\_incentive\_alloc: u64 – Tax allocation for LP incentives (default: 600 bps).
  + charity\_alloc: u64 – Tax allocation to charity (default: 2000 bps).
  + team\_alloc: u64 – Tax allocation to team (default: 1000 bps).
  + reduction\_thresholds: [u64; 3] – Thresholds for reward reduction (default: [250, 500, 750]).
  + reduction\_factors: [u64; 4] – Multipliers for reward reduction (default: [512, 640, 800, 1000]).
  + dex\_programs: Vec<Pubkey> – List of DEX program IDs for sell detection (max 10).
  + whitelisted\_contracts: Vec<(Pubkey, [u8; 32])> – Whitelisted contracts with version hashes (max 20).
  + allowed\_destinations: Vec<Pubkey> – Allowed transfer destinations (max 10).
  + max\_sell\_txn\_limit: u64 – Maximum tokens per sell transaction (default: 0.5% of total supply).
  + daily\_sell\_limit: u64 – Daily sell limit per user (default: 0.5% of total supply).
  + max\_transfer\_limit: u64 – Maximum tokens per non-sell transfer (default: 0.5% of total supply).
  + daily\_transfer\_limit: u64 – Daily transfer limit per user (default: 0.5% of total supply).
  + progressive\_tax\_threshold: u64 – Threshold for applying tripled tax (default: 0.5% of total supply).
  + staking\_rewards: [u64; 4] – Weekly rewards per staking tier (default: [500, 2500, 12500, 125000] tokens).
  + presale\_usdt\_receiver: Pubkey – USDT receiver for presale payments.
  + presale\_active: bool – Flag indicating if presale is active.
  + total\_presale\_sold: u64 – Total tokens sold in presale.
  + last\_processed\_day: i64 – Last day rewards were processed (Unix timestamp).
  + is\_processing: bool – Flag to prevent reentrancy during reward processing.
  + last\_processed\_index: u64 – Index for batch processing of rewards.
* Constants:
  + MAX\_DEXES: usize = 10 – Maximum number of DEX programs.
  + MAX\_WHITELISTED: usize = 20 – Maximum number of whitelisted contracts.
  + MAX\_ALLOWED\_DESTINATIONS: usize = 10 – Maximum number of allowed destinations.
  + MAX\_PAUSE\_REASON\_LEN: usize = 100 – Maximum length for pause reason.
  + LEN: usize = 1130 + 4 + MAX\_PAUSE\_REASON\_LEN + 4 + (32 + 32) \* MAX\_WHITELISTED + 4 + 32 \* MAX\_ALLOWED\_DESTINATIONS – Total size in bytes.
* Usage: Initialized in initialize with default values, updated via governance proposals (e.g., execute\_proposal), and referenced in most instructions to enforce limits and configurations.

Treasury

* Purpose: Tracks token allocations across various pools for staking, airdrops, governance, marketing, emergencies, LP incentives, and team allocations.
* Fields:
  + staking\_pool: u64 – Tokens for staking rewards (default: 30% of treasury reserve).
  + airdrop\_pool: u64 – Tokens for airdrop distributions (default: 8% of treasury reserve).
  + governance\_reserve: u64 – Tokens for governance expenses (default: 16% of treasury reserve).
  + marketing\_fund: u64 – Tokens for marketing initiatives (default: 18% of treasury reserve).
  + emergency\_fund: u64 – Tokens for emergencies (default: 5% of treasury reserve).
  + liquidity\_incentive: u64 – Tokens for liquidity provider incentives (default: 0, grows via tax).
  + team\_pool: u64 – Tokens for the project team (default: 18% of treasury reserve).
* Constants:
  + LEN: usize = 8 + 8 \* 7 = 64 – Total size in bytes.
* Usage: Initialized in initialize, adjusted in transfer, unstake, airdrop, distribute\_lp\_incentives, and transfer\_between\_pools to manage treasury funds.

Multisig

* Purpose: Configures the multisig mechanism for administrative actions, defining owners and the approval threshold.
* Fields:
  + owners: Vec<Pubkey> – List of multisig owners (max 5).
  + threshold: u8 – Minimum number of signatures required (min 2).
* Constants:
  + MAX\_OWNERS: usize = 5 – Maximum number of owners.
  + LEN: usize = 8 + 4 + 32 \* MAX\_OWNERS + 1 = 173 – Total size in bytes.
* Usage: Initialized in initialize, updated via confirm\_set\_multisig, and used to validate multisig approvals in administrative instructions (e.g., pause, airdrop).

MigrationState

* Purpose: Manages the global state for token migration, tracking locked tokens and migration activity.
* Fields:
  + total\_locked: u64 – Total tokens locked for migration.
  + migration\_active: bool – Flag indicating if migration is active.
  + migration\_toggle\_timestamp: i64 – Timestamp of the last migration toggle.
* Constants:
  + LEN: usize = 8 + 8 + 1 + 8 = 25 – Total size in bytes.
* Usage: Initialized in initialize, updated in toggle\_migration\_active, and checked in migration instructions like lock\_for\_migration and unlock\_for\_migration.

### User Data Accounts

These accounts store data specific to individual users, such as staking, LP staking, transaction records, and migration status.

Staker

* Purpose: Tracks staking data for individual users, including tier, amount staked, and reward claim timestamps.
* Fields:
  + tier: u8 – Current staking tier (0-3, 255 for ineligible).
  + amount: u64 – Amount of tokens staked.
  + start\_time: i64 – Unix timestamp when staking began.
  + last\_claim: i64 – Unix timestamp of the last reward claim.
* Constants:
  + LEN: usize = 8 + 1 + 8 + 8 + 8 = 33 – Total size in bytes.
* Usage: Created or updated in stake, used in unstake, claim\_rewards, and vote to manage staking and governance participation.

LP Staker

* Purpose: Manages data for users staking liquidity provider (LP) tokens, tracking staked amounts and rewards.
* Fields:
  + amount: u64 – Amount of LP tokens staked.
  + last\_action\_time: i64 – Unix timestamp of the last staking action (stake or unstake).
  + unclaimed\_rewards: u64 – Accumulated unclaimed rewards.
* Constants:
  + LEN: usize = 8 + 8 + 8 + 8 = 32 – Total size in bytes.
* Usage: Created or updated in stake\_lp, used in unstake\_lp, claim\_lp\_rewards, and process\_daily\_rewards to manage LP staking and rewards.

Transaction Record

* Purpose: Enforces transaction limits and cooldowns for users, tracking sell and transfer activities.
* Fields:
  + last\_txn\_time: i64 – Unix timestamp of the last transaction.
  + sell\_buckets: [u64; 24] – Hourly sell amounts for the past 24 hours.
  + transfer\_buckets: [u64; 24] – Hourly transfer amounts for the past 24 hours.
  + current\_bucket\_index: u8 – Current hour index (0-23).
  + bucket\_start\_time: i64 – Unix timestamp of the current bucket’s start.
  + sell\_cooldown\_start: i64 – Start time of sell cooldown.
  + transfer\_cooldown\_start: i64 – Start time of transfer cooldown.
* Constants:
  + LEN: usize = 8 + 8 + 8 \* 24 + 8 \* 24 + 1 + 8 + 8 + 8 = 457 – Total size in bytes.
* Usage: Created or updated in transfer to enforce limits and cooldowns on sell and transfer transactions.

Presale Purchase

* Purpose: Tracks presale token purchases per wallet, including KYC status.
* Fields:
  + wallet: Pubkey – Buyer’s public key.
  + total\_purchased: u64 – Total tokens purchased in the presale.
  + kyc\_verified: bool – Flag indicating if KYC is verified for large purchases (required for ≥ $1000 USDT).
* Constants:
  + LEN: usize = 8 + 32 + 8 + 1 = 49 – Total size in bytes.
* Usage: Created or updated in buy\_presale to record purchases and enforce per-wallet limits (max 2M tokens).

Presale Vesting

* Purpose: Manages vesting schedules for presale participants, tracking total and claimed tokens.
* Fields:
  + total\_amount: u64 – Total tokens vested from presale purchases.
  + claimed\_amount: u64 – Tokens claimed so far.
* Constants:
  + LEN: usize = 8 + 8 \* 2 = 24 – Total size in bytes.
* Usage: Created or updated in buy\_presale, used in claim\_presale\_tokens to manage vesting and claims.

Vote Record

* Purpose: Records a staker’s vote on a specific governance proposal.
* Fields:
  + staker: Pubkey – Staker’s public key.
  + proposal\_id: u64 – ID of the proposal voted on.
  + voted: bool – Flag indicating if the staker has voted.
  + in\_favor: bool – Indicates if the vote was in favor.
* Constants:
  + LEN: usize = 8 + 32 + 8 + 1 + 1 = 50 – Total size in bytes.
* Usage: Created or updated in vote to track voting participation and prevent double-voting.

Migration Record

* Purpose: Tracks individual user migration status, including locked tokens and migration completion.
* Fields:
  + user: Pubkey – User’s public key.
  + locked\_amount: u64 – Tokens locked for migration.
  + migrated: bool – Flag indicating if migration is complete.
  + migration\_confirmed: bool – Flag indicating if migration is confirmed.
* Constants:
  + LEN: usize = 8 + 32 + 8 + 1 + 1 = 50 – Total size in bytes.
* Usage: Created or updated in lock\_for\_migration, used in unlock\_for\_migration and burn\_locked\_tokens to manage migration processes.

### Vesting and Rewards Accounts

These accounts manage vesting schedules for team members and freelancers.

Team Vesting

* Purpose: Tracks vesting schedules for team members, including total and claimed amounts.
* Fields:
  + team\_member: Pubkey – Team member’s public key.
  + total\_amount: u64 – Total tokens allocated for vesting.
  + claimed\_amount: u64 – Tokens claimed so far.
  + start\_time: i64 – Unix timestamp when vesting begins.
  + canceled: bool – Flag indicating if vesting is canceled.
* Constants:
  + LEN: usize = 8 + 32 + 8 + 8 + 8 + 1 = 65 – Total size in bytes.
* Usage: Created or updated in update\_team\_vesting, used in claim\_team\_vesting and cancel\_team\_vesting to manage team vesting.

Freelancer Vesting

* Purpose: Manages vesting schedules for freelancers, tracking released and claimed tokens.
* Fields:
  + freelancer: Pubkey – Freelancer’s public key.
  + total\_amount: u64 – Total tokens allocated for vesting.
  + released\_amount: u64 – Tokens released for claiming.
  + claimed\_amount: u64 – Tokens claimed so far.
  + start\_time: i64 – Unix timestamp when vesting begins.
  + last\_claim\_time: i64 – Unix timestamp of the last claim.
* Constants:
  + LEN: usize = 8 + 32 + 8 + 8 + 8 + 8 + 8 = 80 – Total size in bytes.
* Usage: Created or updated in update\_freelancer\_vesting, used in claim\_freelancer\_vesting and release\_freelancer\_milestone to manage freelancer vesting.

### Governance Accounts

These accounts support the governance system, including proposals and pending actions.

Proposal

* Purpose: Represents a governance proposal, including its description, voting status, and execution details.
* Fields:
  + id: u64 – Unique proposal identifier.
  + description: String – Proposal description (max 200 bytes).
  + votes\_for: u64 – Total votes in favor.
  + votes\_against: u64 – Total votes against.
  + status: u8 – Proposal status (0 = pending, 1 = approved, 2 = rejected).
  + start\_time: i64 – Unix timestamp when voting starts.
  + end\_time: i64 – Unix timestamp when voting ends (14 days after start).
  + execution\_time: i64 – Unix timestamp when the proposal can be executed (3 days after end).
  + proposal\_type: u8 – Type of proposal (e.g., 0 for tax rate change).
  + proposal\_values: Vec<u64> – Values associated with the proposal (max 7).
* Constants:
  + MAX\_DESCRIPTION\_LEN: usize = 200 – Maximum description length.
  + MAX\_PROPOSAL\_VALUES: usize = 7 – Maximum number of proposal values.
  + LEN: usize = 8 + 8 + (4 + 200) + 8 + 8 + 1 + 8 + 8 + 8 + 1 + (4 + 8 \* 7) = 298 – Total size in bytes.
* Usage: Created in submit\_proposal, updated in vote, and used in execute\_proposal to manage governance proposals.

Pending Withdrawal

* Purpose: Tracks pending treasury withdrawal requests with a delay mechanism.
* Fields:
  + amount: u64 – Amount to withdraw.
  + initiation\_slot: u64 – Slot when the withdrawal was initiated.
  + delay\_slots: u64 – Number of slots to delay (default: 432,000 for large amounts).
* Constants:
  + LEN: usize = 8 + 8 \* 3 = 32 – Total size in bytes.
* Usage: Created in initiate\_withdrawal, used in complete\_withdrawal to enforce withdrawal delays.

Pending Whitelist Change

* Purpose: Manages pending changes to the whitelist with a 72-hour time-lock for security.
* Fields:
  + contract: Pubkey – Contract to add or remove from the whitelist.
  + add: bool – Flag indicating whether to add (true) or remove (false).
  + initiation\_time: i64 – Unix timestamp when the change was initiated.
* Constants:
  + LEN: usize = 8 + 32 + 1 + 8 = 49 – Total size in bytes.
* Usage: Created in add\_whitelisted\_contract or remove\_whitelisted\_contract, used in confirm\_whitelist\_change to apply changes after a delay.

Pending Multisig Change

* Purpose: Tracks pending changes to the multisig configuration with a 24-hour time-lock.
* Fields:
  + new\_owners: Vec<Pubkey> – Proposed new list of multisig owners.
  + new\_threshold: u8 – Proposed new signature threshold.
  + initiation\_time: i64 – Unix timestamp when the change was initiated.
* Constants:
  + MAX\_OWNERS: usize = 5 – Maximum number of owners.
  + LEN: usize = 8 + 4 + 32 \* MAX\_OWNERS + 1 + 8 = 181 – Total size in bytes.
* Usage: Created in initiate\_set\_multisig, used in confirm\_set\_multisig to apply changes after a delay.

Pending Pause

* Purpose: Manages pending pause actions with a 24-hour time-lock to ensure deliberate action.
* Fields:
  + reason: String – Reason for pausing the contract (max 100 bytes).
  + initiation\_time: i64 – Unix timestamp when the pause was initiated.
* Constants:
  + MAX\_REASON\_LEN: usize = 100 – Maximum length for pause reason.
  + LEN: usize = 8 + 4 + MAX\_REASON\_LEN + 8 = 120 – Total size in bytes.
* Usage: Created in initiate\_pause, used in confirm\_pause to enforce a delay before pausing the contract.

Pending Resume

* Purpose: Tracks pending resume actions with a 24-hour time-lock to ensure deliberate reactivation.
* Fields:
  + initiation\_time: i64 – Unix timestamp when the resume was initiated.
* Constants:
  + LEN: usize = 8 + 8 = 16 – Total size in bytes.
* Usage: Created in initiate\_resume, used in confirm\_resume to enforce a delay before resuming the contract.

## Instructions

This section outlines the instructions (functions) available in the smart contract. These instructions allow users, administrators, and stakers to interact with the contract’s state. Each instruction is documented with its purpose, parameters, behavior, constraints, and emitted events for clarity and ease of use. The instructions are grouped logically based on their functionality (e.g., initialization, token operations, staking, and governance).

Below is a detailed breakdown of key instructions from the provided code. For this response, I’ve included a representative selection, but I can expand to cover all instructions if you’d like—just let me know!

#### Reentrancy Guard

Prevents reentrancy attacks where a malicious contract could recursively call back into the contract mid-operation.

* Parameters:
  + Implemented via ReentrancyGuard struct in Instructions.rs.
  + Uses state.is\_processing flag, set to true during execution and reset to false upon completion.
* Validation:
  + Checks !state.is\_processing at the start of each instruction; fails with ErrorCode::ReentrancyGuardTriggered if already processing.
* Impact:
  + Protects against exploits that could drain funds or corrupt state by ensuring operations complete fully before allowing new calls.

Applied universally across all instructions for robust security.

#### 1. Initialization

1.1 initialize

Sets up the initial state of the contract, including multisig owners and key configurations.

**Parameters:**

charity\_wallet: Pubkey – Address for charity tax allocations.

team\_wallet: Pubkey – Address for team tax allocations.

liquidity\_pool: Pubkey – Address for liquidity pool funding.

launch\_timestamp: i64 – Unix timestamp marking the start of vesting schedules.

initial\_owners: Vec<Pubkey> – List of initial multisig owners (3-5 unique addresses).

threshold: u8 – Minimum signatures required for multisig actions (must be ≥2).

initial\_dex\_programs: Vec<Pubkey> – Initial list of approved DEX program IDs (max 10).

presale\_usdt\_receiver: Pubkey – Address to receive USDT from presale purchases.

**Behavior:**

Validates that initial\_owners contains 3-5 unique addresses and threshold is at least 2.

Ensures all initial\_dex\_programs are executable accounts.

Initializes key accounts like ContractState, Treasury, Multisig, and MigrationState.

Sets initial treasury allocations (e.g., staking, liquidity) and default parameters (e.g., tax rates).

**Treasury Pool Initialization:**

- Allocates treasury reserve to predefined pools:

- Staking Pool: 30% (STAKING\_POOL\_PCT)

- Airdrop Pool: 8% (AIRDROP\_POOL\_PCT)

- Governance Reserve: 16% (GOVERNANCE\_RESERVE\_PCT)

- Marketing Fund: 18% (MARKETING\_FUND\_PCT)

- Emergency Fund: 5% (EMERGENCY\_FUND\_PCT)

- Team Pool: 18% (TEAM\_POOL\_PCT)

- Validation: Calculated with overflow checks (`ErrorCode::ArithmeticOverflow`).

- Impact: Ensures transparent allocation for operational funding and incentives.

**Constraints:**

Can only be called once during contract deployment.

initial\_owners must be unique and within the 3-5 range.

initial\_dex\_programs must be valid executable accounts.

**Events:**

Emits InitializeEvent { launch\_timestamp: i64, threshold: u8, initial\_owners: Vec<Pubkey> }.

#### 2. Token Operation

2.1 buy\_presale

**Purpose:** Enables users to buy tokens with USDT during the presale period.

**Parameters:**

amount: u64 – Amount of USDT to spend.

Behavior:

Confirms the presale is active and the contract is not paused.

Calculates token amount: (amount \* 10^TOKEN\_DECIMALS) / PRESALE\_PRICE\_PER\_TOKEN.

Updates PresalePurchase and PresaleVesting accounts for the buyer.

Transfers USDT to presale\_usdt\_receiver.

**Constraints:**

Total tokens sold must not exceed the presale supply cap.

Per-wallet limit applies (e.g., max purchase per wallet).

Large purchases (e.g., ≥1000 USDT) may require KYC.

**KYC Verification:**

- Required for purchases ≥ $1000 USDT.

- Verified via `PresalePurchase.kyc\_verified` flag.

- Fails with `ErrorCode::KYCRequired` if `usdt\_amount >= 1000` and `kyc\_verified` is false.

- Ensures large presale buyers are vetted, reducing risks of illicit activity and adding trust and compliance. Aswell as Legal Requirements.

**Events:**

Emits PresalePurchaseEvent { buyer: Pubkey, usdt\_amount: u64, token\_amount: u64 }.

2.2 claim\_presale\_tokens

Purpose: Allows users to claim tokens vested from presale purchases.

Parameters: None (claims all available tokens for the caller).

Behavior:

Checks the current time against launch\_timestamp to determine vesting progress.

Calculates claimable tokens based on a vesting schedule (e.g., weekly releases).

Transfers claimable tokens from the treasury to the user.

Updates PresaleVesting with the claimed amount.

Constraints:

Vesting must have started (current\_time >= launch\_timestamp).

Must have unclaimed tokens available.

Events:

Emits ClaimRewardsEvent { user: Pubkey, amount: u64 } on success.

Emits NoRewardsEvent { user: Pubkey, reason: String } if no tokens are claimable.

2.3 transfer

**Purpose:** Transfers tokens between wallets with applicable taxes.

**Parameters:**

amount: u64 – Number of tokens to transfer.

**Behavior:**

Checks if the contract is paused (only zero-amount transfers allowed if paused).

Validates against transaction limits (e.g., max sell per transaction).

Applies tax based on configured rates, adjusting for conditions (e.g., progressive tax).

Distributes tax to burn, treasury, liquidity, incentives, charity, and team pools.

Transfers the net amount to the recipient.

**Constraints:**

Sender must have sufficient balance.

Must respect transaction cooldowns (e.g., 1 minute).

**Events:**

Emits TransferEvent { from: Pubkey, to: Pubkey, amount: u64, tax: u64, burn: u64, ... }.

**Tax Distribution Precision:**

Uses ceiling division: `(amount \* rate \* 10^7 + 9999\_999) / (10000 \* 10^7)` to ensure accurate allocation.

Allocations sum to 10,000 basis points, with overflow checks (`ErrorCode::ArithmeticOverflow`).

Prevents under-allocation due to rounding, maintaining fairness and consistency.

2.4 whitelisted\_transfer

Purpose: Transfers tokens with reduced tax rates for whitelisted contracts.

Parameters:

amount: u64 – Number of tokens to transfer.

**Behavior:**

Verifies the caller and recipient are whitelisted.

Applies a reduced tax rate (e.g., half the standard rate).

Distributes tax as in transfer.

Transfers the net amount to the recipient.

**Tax Distribution Precision:**

Uses ceiling division: `(amount \* rate \* 10^7 + 9999\_999) / (10000 \* 10^7)` to ensure accurate allocation.

Allocations sum to 10,000 basis points, with overflow checks (`ErrorCode::ArithmeticOverflow`).

Prevents under-allocation due to rounding, maintaining fairness and consistency.

**Constraints:**

Caller must be in whitelisted\_contracts.

Recipient must be an allowed destination.

**Events:**

Emits TransferEvent { from: Pubkey, to: Pubkey, amount: u64, tax: u64, ... }.

#### 3. Staking

3.1 stake

Purpose: Enables users to stake tokens for rewards and voting power.

Parameters:

amount: u64 – Number of tokens to stake.

Behavior:

Transfers tokens from the user to the staking account.

Updates or creates a Staker account:

Sets start\_time and last\_claim if new.

Adds amount to staked\_amount.

Assigns a staking tier based on amount and duration.

Updates the user’s voting power.

Constraints:

Contract must not be paused.

Minimum stake amount must be met (e.g., 20,000 tokens).

Events:

Emits StakeEvent { user: Pubkey, amount: u64, tier: u8 }.

Emits VotingPowerUpdated { old\_power: u64, new\_power: u64 }.

3.2 unstake

Purpose: Withdraws all staked tokens and claims pending rewards.

Parameters: None (unstakes everything).

Behavior:

Verifies the staking duration meets the tier’s lock period.

Calculates pending rewards based on tier and time staked.

Transfers staked tokens and rewards to the user.

Resets the Staker account.

Constraints:

Lock period must be satisfied.

Treasury must have sufficient reward funds.

Events:

Emits UnstakeEvent { user: Pubkey, amount: u64 }.

Emits ClaimRewardsEvent { user: Pubkey, amount: u64 } if rewards are paid.

3.3 claim\_rewards

Purpose: Claims staking rewards without unstaking.

Parameters: None (claims all available rewards).

Behavior:

Checks staking duration against the tier’s lock period.

Calculates rewards since last\_claim.

Transfers rewards to the user.

Updates last\_claim timestamp.

Constraints:

Lock period must be met.

Rewards must be available.

Events:

Emits ClaimRewardsEvent { user: Pubkey, amount: u64 } on success.

Emits NoRewardsEvent { user: Pubkey, reason: String } if no rewards.

3.4 stake\_lp

Purpose: Stakes liquidity pool (LP) tokens for incentives.

Parameters:

amount: u64 – Number of LP tokens to stake.

Behavior:

Transfers LP tokens to the staking account.

Updates or creates an LPStaker account:

Sets last\_action\_time if new.

Adds amount to staked\_lp\_amount.

Constraints:

Contract must not be paused.

User must have sufficient LP tokens.

Events:

Emits StakeLPEvent { user: Pubkey, amount: u64 }.

3.5 unstake\_lp

Withdraws staked LP tokens.

**Parameters:** amount: u64 – Number of LP tokens to unstake.

**Behavior:**

Ensures staking duration is at least 7 days.

Transfers amount LP tokens back to the user.

Updates staked\_lp\_amount and last\_action\_time.

**Account Closure Logic:**

If `lp\_staker.amount == 0` after unstaking, the account is reassigned to `system\_program` and excess lamports are refunded.

Ensures correct lamport handling and rent reclamation, optimizing resource usage and returning rent to users.

**Constraints:**

Minimum staking duration of 7 days.

amount ≤ staked\_lp\_amount.

**Events:** Emits UnstakeLPEvent { user: Pubkey, amount: u64 }.

3.6 claim\_lp\_rewards

Purpose: Claims rewards from LP staking.

Parameters: None (claims all available rewards).

Behavior:

Verifies staking duration ≥7 days.

Transfers unclaimed\_rewards to the user.

Resets unclaimed\_rewards to 0.

Constraints:

Must have unclaimed rewards >0.

Events:

Emits ClaimLPRewardsEvent { user: Pubkey, amount: u64 } on success.

Emits NoRewardsEvent { user: Pubkey, reason: String } if no rewards.

3.7 process\_daily\_rewards

**Purpose:** Distributes daily LP staking rewards.

**Parameters:**

batch\_size: u64 – Number of stakers to process (max 50).

**Behavior:**

Confirms rewards haven’t been processed today.

Calculates total eligible stake (duration ≥7 days).

Distributes daily reward pool proportionally to the batch.

Updates each staker’s unclaimed\_rewards.

**Constraints:**

Once per day only.

batch\_size ≤50 for compute efficiency.

Reward pool must be funded.

**Events:**

Emits DailyRewardsProcessed { batch\_size: u64, total\_distributed: u64 }.

***LP Reward Calculation and Distribution:***

*Incentivizes liquidity provision by distributing daily rewards to LP stakers.*

***Parameters:***

*- Eligibility: Staked for ≥ 7 days (`current\_time - lp\_staker.last\_action\_time >= 7 \* 86400`).*

*- Reward: Proportional to `lp\_staker.amount / total\_eligible\_stake \* daily\_reward`.*

*- Batch size capped at 50 stakers.*

***Validation:***

*- Fails with `ErrorCode::LockPeriodNotMet` if staking duration < 7 days.*

*- Ensures `batch\_size ≤ 50` (`ErrorCode::BatchSizeTooLarge`).*

***Impact:***

*- Encourages sustained liquidity provision, supporting token market health.*

*- Batch processing optimizes compute usage on Solana, ensuring scalability.*

#### 4. Governance

4.1 submit\_proposal

Purpose: Submits a governance proposal by multisig owners.

Parameters:

description: String – Proposal description (max 200 bytes).

proposal\_type: u8 – Type (e.g., 0 for tax change).

proposal\_values: Vec<u64> – Values to apply (max 7).

Behavior:

Verifies the caller is a multisig owner.

Creates a Proposal account with a 14-day voting period.

Constraints:

Multisig only.

description ≤200 bytes, proposal\_values ≤7.

Events:

Emits ProposalSubmittedEvent { proposal\_id: u64, description: String, proposal\_type: u8 }.

4.2 vote

**Purpose:** Allows stakers to vote on proposals.

**Parameters:**

proposal\_id: u64 – Proposal to vote on.

in\_favor: bool – Vote direction (true = yes, false = no).

**Behavior:**

Ensures the staker’s tier is ≥1 and the proposal is active.

Calculates voting power from staked amount and tier.

Adds power to votes\_for or votes\_against.

**Constraints:**

Staker must have tier ≥1.

Proposal must be in voting period.

**Events:**

Emits VoteEvent { staker: Pubkey, proposal\_id: u64, in\_favor: bool, power: u64 }.

Voting Power Calculation:

- Determines a staker’s governance influence based on their staking tier and duration.

- Calculated in `calculate\_voting\_power` in `Instructions.rs`.

**Base Power:**

- Tier 0: 0 votes

- Tier 1: 1 vote

- Tier 2: 4 votes

- Tier 3: 20 votes

-**Multiplier:**

- <60 days: 1.0x

- 60-89 days: 1.5x

- ≥90 days: ~1.995x (1.5 *\* 1.33)*

*- Capped by the highest tier the staked amount qualifies for.*

***Validation:***

*- Requires tier >= 1 for voting eligibility (`ErrorCode::InsufficientTierForVoting`).*

*- Ensures amount meets tier thresholds and `time\_staked` meets duration requirements.*

**Impact:**

- Rewards long-term commitment with higher voting power, aligning incentives with project stability.

- Caps influence to prevent dominance by large stakers without sufficient staking duration.

4.3 execute\_proposal

Purpose: Executes an approved proposal post-voting.

Parameters: proposal\_id: u64 – Proposal to execute.

Behavior:

Checks voting period and notice period have ended.

Verifies quorum and approval thresholds are met.

Applies changes (e.g., tax rate updates).

Constraints:

Must be approved and within execution window.

Events:

Emits ProposalExecutedEvent { proposal\_id: u64, status: u8 }.

Governance Proposal Types

The smart contract includes 12 governance proposal types that let stakers vote on how the system operates. These proposals control important settings like taxes, staking rewards, and transfer limits. Each type has a specific job, requires certain inputs, and follows rules to keep the system fair and stable. Here’s what each one does:

Proposal Type 0: Update Tax Rate

**Purpose**: Adjusts the transaction tax rate (state.tax\_rate) applied to transfers.

**Parameters:** proposal\_values[0] (new tax rate in basis points, range: 1% to 10%).

**Validation:** Ensures the new tax rate is between 100 and 1000 basis points (1% to 10%).

**Impact:** Changes the tax percentage applied to all token transfers, affecting burn, treasury, liquidity pool, LP incentives, charity, and team allocations.

Proposal Type 1: Update Staking Tiers

**Purpose:** Modifies the token amount thresholds for staking tiers (state.staking\_tiers).

**Parameters:** proposal\_values[0..4] (four tier thresholds in ascending order).

**Validation:** Ensures the first tier is at least 20,000 tokens, tiers are strictly increasing, and the highest tier is at most 10% of the total supply.

**Impact:** Adjusts the eligibility criteria for staking tiers, affecting reward distribution and voting power.

Proposal Type 2: Update Tax Allocation

**Purpose:** Updates the allocation of taxes (state.burn\_alloc, state.treasury\_alloc, state.liquidity\_pool\_alloc, state.lp\_incentive\_alloc, state.charity\_alloc, state.team\_alloc).

**Parameters:** proposal\_values[0..6] (allocations in basis points for burn, treasury, liquidity pool, LP incentives, charity, team).

**Validation:** Ensures the total allocation equals 10,000 basis points (100%) and each allocation is at most 50%.

**Impact:** Changes how taxes collected from transfers are distributed among different pools.

Proposal Type 3: Update Reduction Thresholds and Factors

**Purpose:** Adjusts the thresholds and factors for reward reduction based on staking pool depletion (state.reduction\_thresholds, state.reduction\_factors).

**Parameters:** proposal\_values[0..3] (thresholds), proposal\_values[3..7] (factors).

**Validation:** Ensures thresholds are increasing, within 10% to 90% range, and factors are between 10% and 200%.

**Impact:** Modifies how staking rewards are scaled based on the staking pool's size, affecting reward payouts.

Proposal Type 4: Update Launch Timestamp

**Purpose:** Changes the presale vesting start time (state.launch\_timestamp).

**Parameters:** proposal\_values[0] (new Unix timestamp).

**Validation:** Ensures the timestamp is within 30 days before or one year after the initial launch timestamp (May 1, 2025).

**Impact:** Adjusts the schedule for presale token vesting, affecting when users can claim their tokens.

Proposal Type 5: Update Maximum Sell Transaction Limit

**Purpose:** Modifies the maximum tokens allowed per sell transaction (state.max\_sell\_txn\_limit).

**Parameters**: proposal\_values[0] (new limit).

**Validation:** Ensures the limit is between 0.1% and 2% of the total supply.

**Impact:** Controls the size of individual sell transactions to DEXes, affecting market liquidity and price stability.

Proposal Type 6: Update Daily Sell Limit

**Purpose:** Adjusts the daily sell limit per user (state.daily\_sell\_limit).

**Parameters:** proposal\_values[0] (new limit).

**Validation:** Ensures the limit is between 0.1% and 2% of the total supply.

**Impact:** Limits the total tokens a user can sell to DEXes daily, preventing large dumps.

Proposal Type 7: Update Maximum Transfer Limit

**Purpose:** Changes the maximum tokens allowed per non-sell transfer (state.max\_transfer\_limit).

**Parameters:** proposal\_values[0] (new limit).

**Validation:** Ensures the limit is between 0.1% and 2% of the total supply.

**Impact:** Controls the size of non-sell transfers, affecting token movement between wallets.

Proposal Type 8: Update Daily Transfer Limit

**Purpose:** Modifies the daily non-sell transfer limit per user (state.daily\_transfer\_limit).

**Parameters:** proposal\_values[0] (new limit).

**Validation:** Ensures the limit is between 0.1% and 2% of the total supply.

**Impact:** Limits the total tokens a user can transfer daily (excluding sells), enhancing security.

Proposal Type 9: Update Progressive Tax Threshold

**Purpose:** Adjusts the threshold for applying a tripled tax rate on large transfers (state.progressive\_tax\_threshold).

**Parameters:** proposal\_values[0] (new threshold).

**Validation:** Ensures the threshold is between 0.1% and 2% of the total supply.

**Impact:** Changes when the progressive tax (2x the normal rate) applies, discouraging large transfers.

Proposal Type 10: Update Staking Rewards

**Purpose:** Modifies the weekly staking rewards for each tier (state.staking\_rewards).

**Parameters**: proposal\_values[0..4] (rewards for each tier).

**Validation:** Ensures rewards are between 100 and 1,000,000 tokens.

**Impact:** Adjusts the incentives for staking, affecting user participation and reward distribution.

Proposal Type 11: Batch Whitelist Update

**Purpose:** Adds or removes multiple whitelisted contracts (state.whitelisted\_contracts).

**Parameters:** proposal\_values[0] (number of contracts to add), proposal\_values[1] (number to remove), proposal\_values[2..] (contract addresses as u64).

**Validation:** Ensures the total number of add/remove operations fits within the proposal values and respects the maximum whitelist size.

**Impact:** Updates the list of contracts allowed to perform privileged operations (e.g., reduced-tax transfers).

#### 5. Airdrop

5.1 airdrop

Purpose: Distributes tokens to multiple recipients.

Parameters:

winners: Vec<(Pubkey, u64)> – Recipients and their amounts.

Behavior:

Requires multisig approval.

Transfers tokens from the airdrop pool to recipients.

Deducts total from the pool.

Constraints:

Multisig only.

Airdrop pool must have sufficient funds.

Limited to ≤ 100 recipients (`ErrorCode::TooManyRecipients`).

Events:

Emits AirdropEvent { total\_amount: u64, recipient\_count: usize }.

#### 6. Pause and Resume

6.1 pause

Purpose: Initiates a contract pause.

Parameters:

reason: String – Reason for pausing.

Behavior:

Requires multisig approval.

Creates a PendingPause account with initiation time.

Constraints:

Multisig only.

Contract must not be paused.

Events:

Emits PauseInitiated { initiation\_time: i64 }.

6.2 confirm\_pause

Purpose: Finalizes the pause after a delay.

Parameters: None.

Behavior:

Ensures delay (e.g., 24 hours) has passed.

Sets ContractState.paused = true.

Constraints:

Delay period must be complete.

Events:

Emits PauseEvent { timestamp: i64, reason: String }.

6.3 resume

Purpose: Initiates resuming the contract.

Parameters: None.

Behavior:

Requires multisig approval.

Creates a PendingResume account.

Constraints:

Multisig only.

Contract must be paused.

Events:

Emits ResumeInitiated { initiation\_time: i64 }.

6.4 confirm\_resume

Purpose: Finalizes resuming the contract.

Parameters: None.

Behavior:

Ensures delay has passed.

Sets ContractState.paused = false.

Constraints:

Delay period must be complete.

Events:

Emits ResumeEvent { timestamp: i64 }.0

#### 7. Vesting

7.1 update\_team\_vesting

Purpose: Configures team vesting schedules.

Parameters:

team\_member: Pubkey – Team member’s address.

total\_amount: u64 – Total tokens to vest.

immediate\_amount: u64 – Tokens released immediately.

Behavior:

Requires multisig approval.

Updates or creates TeamVesting account.

Transfers immediate\_amount to the team member.

Constraints:

Multisig only.

immediate\_amount ≤ total\_amount.

Events:

Emits TeamVestingUpdatedEvent { team\_member: Pubkey, total\_amount: u64 }.

7.2 cancel\_team\_vesting

Purpose: Cancels a team vesting schedule.

Parameters:

team\_member: Pubkey – Team member’s address.

Behavior:

Requires multisig approval.

Marks TeamVesting as canceled.

Constraints:

Multisig only.

Events:

Emits TeamVestingCanceledEvent { team\_member: Pubkey }.

7.3 claim\_team\_vesting

Purpose: Allows team members to claim vested tokens.

Parameters: None.

Behavior:

Calculates claimable tokens based on vesting schedule.

Transfers tokens to the team member.

Updates claimed\_amount.

Constraints:

Vesting must not be canceled.

Claimable amount >0.

Events:

Emits TeamVestingClaimedEvent { team\_member: Pubkey, amount: u64 }.

7.4 update\_freelancer\_vesting

Purpose: Configures freelancer vesting schedules.

Parameters:

freelancer: Pubkey – Freelancer’s address.

total\_amount: u64 – Total tokens to vest.

Behavior:

Requires multisig approval.

Updates or creates FreelancerVesting account.

Constraints:

Multisig only.

Events:

Emits FreelancerVestingUpdatedEvent { freelancer: Pubkey, total\_amount: u64 }.

7.5 claim\_freelancer\_vesting

Purpose: Allows freelancers to claim vested tokens.

Parameters: None.

Behavior:

Checks claim cooldown (e.g., 3 days).

Calculates claimable amount.

Transfers tokens to the freelancer.

Updates claimed\_amount and last\_claim\_time.

Constraints:

Cooldown must have passed.

Claimable amount >0.

Events:

Emits FreelancerVestingClaimedEvent { freelancer: Pubkey, amount: u64 }.

7.6 release\_freelancer\_milestone

Purpose: Releases tokens for freelancer milestones.

Parameters:

amount: u64 – Tokens to release.

Behavior:

Requires multisig approval.

Increases released\_amount in FreelancerVesting.

Constraints:

Multisig only.

released\_amount ≤ total\_amount.

Events:

Emits FreelancerMilestoneReleasedEvent { freelancer: Pubkey, amount: u64 }.

#### 8. Withdrawal

8.1 initiate\_withdrawal

Purpose: Starts a treasury withdrawal request.

Parameters:

amount: u64 – Amount to withdraw.

Behavior:

Requires multisig approval.

Creates PendingWithdrawal with a delay if amount exceeds a threshold.

Constraints:

Multisig only.

Events:

Emits WithdrawalInitiatedEvent { amount: u64, initiation\_slot: u64, delay\_slots: u64 }.

8.2 complete\_withdrawal

Purpose: Finalizes a withdrawal post-delay.

Parameters: None.

Behavior:

Ensures delay has passed.

Transfers amount from treasury to destination.

Constraints:

Delay period must be complete.

Events:

Emits WithdrawalCompletedEvent { amount: u64 }.

#### 9. Multisig Management

9.1 set\_multisig

Purpose: Initiates a multisig configuration change.

Parameters:

owners: Vec<Pubkey> – New owner list.

threshold: u8 – New threshold.

Behavior:

Requires current multisig approval.

Creates PendingMultisigChange.

Constraints:

Multisig only.

Owners must be unique, threshold ≤ owners.

Events:

Emits MultisigChangeInitiated { initiation\_time: i64 }.

9.2 confirm\_set\_multisig

Purpose: Applies the multisig change post-delay.

Parameters: None.

Behavior:

Ensures delay has passed.

Updates Multisig with new owners and threshold.

Constraints:

Delay period must be complete.

Events:

Emits MultisigUpdatedEvent { threshold: u8, owner\_count: usize }.

#### 10. LP Incentives

10.1 distribute\_lp\_incentives

Purpose: Distributes LP incentives.

Parameters:

recipients: Vec<(Pubkey, u64)> – Recipients and amounts.

Behavior:

Requires multisig approval.

Transfers tokens from treasury to recipients.

Deducts from liquidity\_incentive pool.

Constraints:

Multisig only.

Total ≤ liquidity\_incentive pool.

Events:

Emits LPIncentivesDistributedEvent { total\_amount: u64, recipient\_count: usize }.

#### 11. DEX Programs

11.1 update\_dex\_programs

Purpose: Updates approved DEX programs.

Parameters:

dex\_programs: Vec<Pubkey> – New DEX program IDs.

Behavior:

Requires multisig approval.

Validates programs are executable.

Updates dex\_programs in ContractState.

Constraints:

Multisig only.

Programs must be executable.

Events:

Emits DexProgramsUpdatedEvent { program\_count: usize }.

#### 12. Whitelist Management

12.1 add\_whitelisted\_contract

Purpose: Initiates adding a contract to the whitelist.

Parameters:

contract: Pubkey – Contract to add.

Behavior:

Requires multisig approval.

Creates PendingWhitelistChange (add).

Constraints:

Multisig only.

Contract must be executable.

Events: None (pending).

12.2 remove\_whitelisted\_contract

Purpose: Initiates removing a contract from the whitelist.

Parameters:

contract: Pubkey – Contract to remove.

Behavior:

Requires multisig approval.

Creates PendingWhitelistChange (remove).

Constraints:

Multisig only.

Events: None (pending).

12.3 confirm\_whitelist\_change

Purpose: Applies a whitelist change post-delay.

Parameters: None.

Behavior:

Ensures delay has passed.

Adds or removes the contract from whitelisted\_contracts.

Constraints:

Delay period must be complete.

Events:

Emits WhitelistedContractAddedEvent { contract: Pubkey } or WhitelistedContractRemovedEvent { contract: Pubkey }.

#### 13. Migration

Migration Workflow

**Purpose:** Enables secure token upgrades or transitions with lock, unlock, and burn capabilities.

**Parameters:**

- Locking: Via `lock\_for\_migration` when `migration\_active` is true.

- Unlocking: Via `unlock\_for\_migration` when `migration\_active` is false.

- Burning: Via `burn\_locked\_tokens`, admin-controlled with multisig.

**Validation:**

- `ErrorCode::MigrationNotActive` or `MigrationActive` enforces correct state.

- Multisig required for burning (`ErrorCode::InsufficientSigners`).

**Impact:**

- Provides flexibility for future upgrades while protecting user funds.

- Multisig control ensures transparency and security in migration completion.

13.1 lock\_for\_migration

Purpose: Locks tokens for migration.

Parameters:

amount: u64 – Tokens to lock.

Behavior:

Transfers tokens to the migration account.

Updates MigrationRecord and MigrationState.

Constraints:

Migration must be active.

Events:

Emits TokensLockedForMigrationEvent { user: Pubkey, amount: u64 }.

13.2 unlock\_for\_migration

Purpose: Unlocks tokens if migration is inactive.

Parameters: None.

Behavior:

Transfers locked tokens back to the user.

Updates MigrationRecord and MigrationState.

Constraints:

Migration must be inactive.

Events:

Emits TokensUnlockedForMigrationEvent { user: Pubkey, amount: u64 }.

13.3 burn\_locked\_tokens

Purpose: Burns locked tokens during migration.

Parameters:

user: Pubkey – User whose tokens to burn.

Behavior:

Requires multisig approval.

Burns locked tokens and marks migration complete.

Constraints:

Multisig only.

User must have locked tokens.

Events:

Emits LockedTokensBurnedEvent { user: Pubkey, amount: u64 }.

Emits MigrationConfirmedEvent { user: Pubkey }.

13.4 confirm\_migration

Purpose: Confirms migration completion (alias for burn\_locked\_tokens completion).

Parameters: None (implied from context).

Behavior:

Finalizes migration for a user or globally.

Constraints:

Multisig or user-triggered depending on context.

Events:

Emits MigrationConfirmedEvent { user: Pubkey }.

13.5 toggle\_migration\_active

Purpose: Toggles migration state.

Parameters:

active: bool – New state.

Behavior:

Requires multisig approval.

Updates migration\_active in MigrationState.

Constraints:

Multisig only.

Cooldown (e.g., 7 days) applies.

Events:

Emits MigrationToggledEvent { active: bool }.

#### 14. Pool Transfer

14.1 transfer\_between\_pools

Purpose: Transfers tokens between treasury pools.

Parameters:

source\_pool: PoolType – Source pool.

dest\_pool: PoolType – Destination pool.

amount: u64 – Amount to transfer.

Behavior:

Requires multisig approval.

Moves amount from source\_pool to dest\_pool.

Constraints:

Multisig only.

Source pool must have sufficient funds.

Events:

Emits TreasuryPoolAdjusted { pool\_type: PoolType, new\_amount: u64 } for both pools.

#### 15. Query

15.1 query\_pending\_rewards

Purpose: Returns a user’s pending staking rewards.

Parameters: None (uses caller).

Behavior:

Fetches the user’s Staker account.

Calculates rewards based on time and tier.

Returns the amount.

Constraints:

User must have a Staker account.

Events: None.

15.2 query\_state

Purpose: Returns the current ContractState.

Parameters: None.

Behavior:

Retrieves and returns ContractState.

Constraints: None.

Events: None.

## Events

This section details all events emitted by the smart contract, categorized by their functional roles. Events are logged using Anchor's #[event] attribute and are emitted within instruction implementations in Instruction.rs. They provide transparency and enable external systems to track contract activities.

**Overview of Events**

Logs critical actions for transparency and auditability.

**Parameters:** 47 event types (e.g., `TransferEvent`, `StakeEvent`, `ProposalSubmittedEvent`) defined in `Instructions.rs`.Include details like amounts, users, and reasons.

**Validation:** Emitted automatically in instructions; no direct validation required.

**Impact:** Enables tracking of contract activity by users and auditors.Supports debugging and verification of governance outcomes.

### Initialization and Configuration

Events related to the contract's setup and configuration updates.

Initialize Event

* Purpose: Logs the initialization of the smart contract with its initial parameters.
* Fields:
  + launch\_timestamp: i64 – Unix timestamp marking the start of vesting schedules.
  + threshold: u8 – Minimum number of multisig signatures required for administrative actions.
  + initial\_owners: Vec<Pubkey> – List of initial multisig owners.
* When Emitted: At the end of the initialize instruction, after setting up the contract state, treasury, multisig, and migration state.
* Usage: Allows tracking of the contract’s initial configuration, including vesting start time and governance setup.

Tax Rate

* Purpose: Indicates an update to the base tax rate applied to token transfers.
* Fields:
  + new\_tax\_rate: u64 – Updated tax rate in basis points (e.g., 500 = 5%).
* When Emitted: During execute\_proposal when a proposal of type 0 successfully updates the tax rate.
* Usage: Alerts users to changes in transaction costs, aiding in transaction planning.

Staking Tiers

* Purpose: Records updates to the staking tier thresholds.
* Fields:
  + new\_tiers: [u64; 4] – Updated token amount thresholds for each staking tier.
* When Emitted: In execute\_proposal when a proposal of type 1 adjusts staking tiers.
* Usage: Informs stakers of new eligibility requirements for staking tiers.

Tax Allocation

* Purpose: Logs changes to the distribution of transaction taxes among various pools.
* Fields:
  + burn: u64 – Allocation for burning tokens (basis points).
  + treasury: u64 – Allocation to the treasury pool.
  + liquidity\_pool: u64 – Allocation to the liquidity pool.
  + lp\_incentive: u64 – Allocation for liquidity provider incentives.
  + charity: u64 – Allocation to charity.
  + team: u64 – Allocation to the team pool.
* When Emitted: During execute\_proposal when a proposal of type 2 updates tax allocations.
* Usage: Tracks shifts in tokenomics and fund distribution for transparency.

Reduction Factors

* Purpose: Indicates updates to reward reduction thresholds and factors based on pool depletion.
* Fields:
  + thresholds: [u64; 3] – New pool depletion thresholds (in thousandths).
  + factors: [u64; 4] – New reduction multipliers for rewards (basis points).
* When Emitted: In execute\_proposal when a proposal of type 3 adjusts reduction factors.
* Usage: Helps stakers understand changes affecting reward calculations.

Launch Timestamp

* Purpose: Logs updates to the vesting schedule start timestamp.
* Fields:
  + new\_timestamp: i64 – Updated Unix timestamp for vesting start.
* When Emitted: During execute\_proposal when a proposal of type 4 changes the launch timestamp.
* Usage: Notifies users of adjustments to vesting schedules.

Max Sell Limit

* Purpose: Records updates to the maximum token amount per sell transaction.
* Fields:
  + new\_limit: u64 – New maximum tokens allowed per sell transaction.
* When Emitted: In execute\_proposal when a proposal of type 5 updates the sell transaction limit.
* Usage: Alerts traders to changes in selling restrictions.

Daily Sell Limit

* Purpose: Indicates changes to the daily sell limit per user.
* Fields:
  + new\_limit: u64 – Updated daily sell limit in tokens.
* When Emitted: During execute\_proposal when a proposal of type 6 adjusts the daily sell limit.
* Usage: Helps users monitor daily selling constraints.

Max Transfer Limit

* Purpose: Logs updates to the maximum token amount for non-sell transfers.
* Fields:
  + new\_limit: u64 – New maximum tokens per non-sell transfer.
* When Emitted: In execute\_proposal when a proposal of type 7 updates the transfer limit.
* Usage: Informs users of changes in transfer restrictions.

Daily Transfer Limit

* Purpose: Records changes to the daily transfer limit for non-sell transactions.
* Fields:
  + new\_limit: u64 – Updated daily transfer limit in tokens.
* When Emitted: During execute\_proposal when a proposal of type 8 adjusts the daily transfer limit.
* Usage: Tracks adjustments in daily transfer allowances.

Progressive Tax Threshold

* Purpose: Indicates updates to the threshold for applying a tripled tax rate.
* Fields:
  + new\_threshold: u64 – New token amount threshold for progressive taxation.
* When Emitted: In execute\_proposal when a proposal of type 9 changes the progressive tax threshold.
* Usage: Alerts users to shifts in tax brackets for large transactions.

Staking Rewards

* Purpose: Logs updates to the weekly staking rewards for each tier.
* Fields:
  + new\_rewards: [u64; 4] – Updated reward amounts for each staking tier.
* When Emitted: During execute\_proposal when a proposal of type 10 adjusts staking rewards.
* Usage: Informs stakers of changes in potential earnings.

Dex Programs

* Purpose: Records updates to the list of approved decentralized exchange (DEX) programs for sell detection.
* Fields:
  + program\_count: usize – Number of DEX programs in the updated list.
* When Emitted: At the end of the update\_dex\_programs instruction after updating state.dex\_programs.

Usage: Tracks changes in DEX integrations affecting sell transaction identification.

Whitelisted Contract Added

* Purpose: Logs the addition of a contract to the whitelist for privileged operations (e.g., reduced taxes).
* Fields:
  + contract: Pubkey – Public key of the added whitelisted contract.
* When Emitted: In confirm\_whitelist\_change when adding a contract, or in execute\_proposal (type 11) for batch updates.
* Usage: Tracks contracts eligible for reduced tax rates or special permissions.

Whitelisted Contract Removed

* Purpose: Records the removal of a contract from the whitelist.
* Fields:
  + contract: Pubkey – Public key of the removed contract.
* When Emitted: In confirm\_whitelist\_change when removing a contract, or in execute\_proposal (type 11) for batch updates.
* Usage: Updates the list of contracts no longer eligible for privileges.

Multisig

* Purpose: Indicates changes to the multisig configuration (owners or threshold).
* Fields:
  + threshold: u8 – New minimum number of signatures required.
  + owner\_count: usize – New number of multisig owners.
* When Emitted: At the end of confirm\_set\_multisig after updating the multisig account.
* Usage: Tracks changes in governance control for administrative actions.

Migration Toggled

* Purpose: Logs the toggling of the migration state (active/inactive) for token upgrades.
* Fields:
  + active: bool – Indicates if migration is active (true) or inactive (false).
* When Emitted: In toggle\_migration\_active when the migration state is changed.
* Usage: Informs users of the availability or closure of token migration processes.

### Token Operations

Events related to token transfers, presale purchases, and airdrops.

Transfers

* Purpose: Documents a successful token transfer, including tax breakdown.
* Fields:
  + from: Pubkey – Sender’s public key.
  + to: Pubkey – Recipient’s public key.
  + amount: u64 – Net tokens transferred (after tax).
  + tax: u64 – Total tax applied.
  + burn: u64 – Portion of tax burned.
  + treasury\_tax: u64 – Portion allocated to the treasury.
  + liquidity\_pool\_tax: u64 – Portion allocated to the liquidity pool.
  + lp\_incentive\_tax: u64 – Portion allocated to LP incentives.
  + charity\_tax: u64 – Portion allocated to charity.
  + team\_tax: u64 – Portion allocated to the team.
* When Emitted: At the end of transfer or whitelisted\_transfer after a successful transfer.
* Usage: Provides a detailed record of transfers and tax distributions for auditing and tracking.

Transfer Failed

* Purpose: Logs a failed transfer attempt with the reason for failure.
* Fields:
  + from: Pubkey – Sender’s public key.
  + to: Pubkey – Intended recipient’s public key.
  + reason: String – Explanation for the failure (e.g., "Amount too small after tax").
* When Emitted: In transfer or whitelisted\_transfer when the transfer fails (e.g., due to insufficient amount after tax).
* Usage: Helps diagnose transfer issues and informs users of failed attempts.

Presale Purchase

* Purpose: Records a user’s token purchase during the presale phase.
* Fields:
  + buyer: Pubkey – Buyer’s public key.
  + usdt\_amount: u64 – Amount of USDT paid.
  + token\_amount: u64 – Tokens purchased.
* When Emitted: At the end of buy\_presale after a successful purchase.
* Usage: Tracks presale participation and funding contributions.

Airdrop Event

* Purpose: Logs the distribution of tokens via an airdrop.
* Fields:
  + total\_amount: u64 – Total tokens distributed.
  + recipient\_count: usize – Number of recipients.
* When Emitted: At the end of the airdrop instruction after transferring tokens to recipients.
* Usage: Verifies airdrop execution and token distribution scale.

### Staking and LP Staking

Events related to staking, unstaking, and reward claims for both regular and liquidity provider (LP) staking.

Stake

* Purpose: Indicates a user staking tokens to earn rewards.
* Fields:
  + user: Pubkey – Staker’s public key.
  + amount: u64 – Amount of tokens staked.
  + tier: u8 – Assigned staking tier based on amount and duration.
* When Emitted: At the end of the stake instruction after tokens are transferred to the staking account.
* Usage: Tracks staking activity and tier assignments for reward eligibility.

Unstake

* Purpose: Records a user unstaking their tokens.
* Fields:
  + user: Pubkey – Unstaker’s public key.
  + amount: u64 – Amount of tokens unstaked.
* When Emitted: At the end of the unstake instruction after tokens are returned to the user.
* Usage: Confirms staking exits and updates user balances.

Claim Rewards

* Purpose: Logs the claiming of staking rewards by a user.
* Fields:
  + user: Pubkey – Claimant’s public key.
  + amount: u64 – Amount of rewards claimed.
* When Emitted: In claim\_rewards, unstake (if rewards are claimed), or claim\_presale\_tokens after transferring rewards.
* Usage: Verifies reward payouts and updates reward status.

Stake LP

* Purpose: Indicates a user staking LP tokens for incentives.
* Fields:
  + user: Pubkey – Staker’s public key.
  + amount: u64 – Amount of LP tokens staked.
* When Emitted: At the end of stake\_lp after LP tokens are transferred to the staking account.
* Usage: Tracks LP staking activity for incentive eligibility.

Unstake LP

* Purpose: Records a user unstaking their LP tokens.
* Fields:
  + user: Pubkey – Unstaker’s public key.
  + amount: u64 – Amount of LP tokens unstaked.
* When Emitted: At the end of unstake\_lp after LP tokens are returned to the user.
* Usage: Confirms LP staking exits and updates user LP balances.

Claim LP Rewards

* Purpose: Logs the claiming of LP staking rewards.
* Fields:
  + user: Pubkey – Claimant’s public key.
  + amount: u64 – Amount of rewards claimed.
* When Emitted: At the end of claim\_lp\_rewards after transferring rewards to the user.
* Usage: Verifies LP reward payouts and updates reward status.

No Rewards

* Purpose: Indicates no rewards were available for claiming.
* Fields:
  + user: Pubkey – User attempting to claim (or Pubkey::default() for system-wide events).
  + reason: String – Explanation (e.g., "No tokens available to claim").
* When Emitted: In claim\_presale\_tokens, unstake, claim\_rewards, claim\_lp\_rewards, or process\_daily\_rewards when no rewards are due.
* Usage: Informs users or systems of the absence of claimable rewards.

### Governance

Events related to governance proposals, voting, and execution.

Proposal Submitted

* Purpose: Announces the submission of a new governance proposal.
* Fields:
  + proposal\_id: u64 – Unique identifier for the proposal.
  + description: String – Description of the proposal.
  + proposal\_type: u8 – Type of proposal (e.g., 0 for tax rate update).
* When Emitted: At the end of submit\_proposal after creating a new proposal.
* Usage: Notifies stakeholders of new governance proposals for review.

Vote

* Purpose: Records a staker’s vote on a governance proposal.
* Fields:
  + staker: Pubkey – Voter’s public key.
  + proposal\_id: u64 – ID of the proposal voted on.
  + in\_favor: bool – Vote direction (true for approval, false for rejection).
  + power: u64 – Voting power applied to the vote.
* When Emitted: At the end of the vote instruction after updating proposal vote counts.
* Usage: Tracks voting participation and influence on governance outcomes.

ProposalExecuted

* Purpose: Indicates a governance proposal has been executed.
* Fields:
  + proposal\_id: u64 – ID of the executed proposal.
  + status: u8 – Outcome status (1 = approved, 2 = rejected).
* When Emitted: At the end of execute\_proposal after applying or rejecting the proposal.
* Usage: Confirms the result of governance actions.

### Vesting

Events related to team and freelancer vesting schedules and claims.

Team Vesting Updated

* Purpose: Logs updates to a team member’s vesting schedule.
* Fields:
  + team\_member: Pubkey – Team member’s public key.
  + total\_amount: u64 – Total tokens allocated for vesting (excluding immediate amount).
* When Emitted: At the end of update\_team\_vesting after setting up or modifying the vesting schedule.
* Usage: Tracks changes to team vesting allocations.

Team Vesting Claimed

* Purpose: Records a team member claiming vested tokens.
* Fields:
  + team\_member: Pubkey – Claimant’s public key.
  + amount: u64 – Amount of tokens claimed.
* When Emitted: At the end of claim\_team\_vesting after transferring vested tokens.
* Usage: Verifies team vesting payouts.

Team Vesting Cancelled

* Purpose: Indicates the cancellation of a team member’s vesting schedule.
* Fields:
  + team\_member: Pubkey – Affected team member’s public key.
* When Emitted: At the end of cancel\_team\_vesting after marking the vesting as canceled.
* Usage: Tracks vesting schedule terminations.

Freelancer Vesting Updated

* Purpose: Logs updates to a freelancer’s vesting schedule.
* Fields:
  + freelancer: Pubkey – Freelancer’s public key.
  + total\_amount: u64 – Total tokens allocated for vesting.
* When Emitted: At the end of update\_freelancer\_vesting after setting up or modifying the vesting schedule.
* Usage: Tracks changes to freelancer vesting allocations.

Freelancer Vesting Claimed

* Purpose: Records a freelancer claiming vested tokens.
* Fields:
  + freelancer: Pubkey – Claimant’s public key.
  + amount: u64 – Amount of tokens claimed.
* When Emitted: At the end of claim\_freelancer\_vesting after transferring vested tokens.
* Usage: Verifies freelancer vesting payouts.

Freelancer Milestone Released

* Purpose: Indicates the release of tokens for a freelancer’s milestone.
* Fields:
  + freelancer: Pubkey – Freelancer’s public key.
  + amount: u64 – Amount of tokens released.
* When Emitted: At the end of release\_freelancer\_milestone after updating the released amount.
* Usage: Tracks milestone-based payments to freelancers.

### Administration

Events related to contract pausing, resuming, withdrawals, and treasury adjustments.

Pause

* Purpose: Logs the pausing of the contract, halting most operations.
* Fields:
  + timestamp: i64 – Unix timestamp of the pause.
  + reason: String – Reason for pausing the contract.
* When Emitted: At the end of confirm\_pause after setting state.paused to true.
* Usage: Alerts users to operational pauses and their justifications.

Resume

* Purpose: Indicates the resumption of normal contract operations.
* Fields:
  + timestamp: i64 – Unix timestamp of the resumption.
* When Emitted: At the end of confirm\_resume after setting state.paused to false.
* Usage: Confirms the return to normal contract functionality.

Pause Initiated

* Purpose: Records the initiation of a pause request, pending confirmation.
* Fields:
  + initiation\_time: i64 – Unix timestamp when the pause was initiated.
* When Emitted: At the end of initiate\_pause after creating a PendingPause account.
* Usage: Tracks pending pause actions awaiting multisig approval.

Resume Initiated

* Purpose: Logs the initiation of a resume request, pending confirmation.
* Fields:
  + initiation\_time: i64 – Unix timestamp when the resume was initiated.
* When Emitted: At the end of initiate\_resume after creating a PendingResume account.
* Usage: Tracks pending resume actions awaiting multisig approval.

Withdrawal Initiated

* Purpose: Indicates a withdrawal request from the treasury has been initiated.
* Fields:
  + amount: u64 – Requested withdrawal amount in tokens.
  + initiation\_slot: u64 – Slot number when the withdrawal was initiated.
  + delay\_slots: u64 – Number of slots to wait before completion (based on amount).
* When Emitted: At the end of initiate\_withdrawal after creating a PendingWithdrawal account.
* Usage: Tracks pending treasury withdrawals and their delay periods.

Withdrawal Completed

* Purpose: Confirms the completion of a treasury withdrawal.
* Fields:
  + amount: u64 – Amount of tokens withdrawn.
* When Emitted: At the end of complete\_withdrawal after transferring tokens to the destination.
* Usage: Verifies treasury fund movements.

Multisig Change Initiated

* Purpose: Logs the initiation of a multisig configuration change, pending confirmation.
* Fields:
  + initiation\_time: i64 – Unix timestamp when the change was initiated.
* When Emitted: At the end of initiate\_set\_multisig after creating a PendingMultisigChange account.
* Usage: Tracks pending updates to multisig ownership or threshold.

Treasury Pool Adjusted

* Purpose: Records adjustments to treasury pool allocations.
* Fields:
  + pool\_type: PoolType – Type of pool adjusted (e.g., Staking, Airdrop).
  + new\_amount: u64 – Updated amount in the pool after adjustment.
* When Emitted: In transfer\_between\_pools, unstake, claim\_rewards, airdrop, or distribute\_lp\_incentives when pool balances change.
* Usage: Tracks treasury reallocations and pool funding changes.

LP Incentives Distributed

* Purpose: Logs the distribution of liquidity provider incentives from the treasury.
* Fields:
  + total\_amount: u64 – Total tokens distributed.
  + recipient\_count: usize – Number of recipients.
* When Emitted: At the end of distribute\_lp\_incentives after transferring tokens to recipients.
* Usage: Verifies LP incentive payouts and distribution scale.

### Migration

Events related to token migration processes for upgrades or transitions.

Tokens Locked For Migration

* Purpose: Indicates tokens have been locked by a user for migration.
* Fields:
  + user: Pubkey – User’s public key locking tokens.
  + amount: u64 – Amount of tokens locked.
* When Emitted: At the end of lock\_for\_migration after transferring tokens to the migration account.
* Usage: Tracks user participation in the migration process.

Tokens Unlocked For Migration

* Purpose: Logs tokens being unlocked and returned to a user post-migration.
* Fields:
  + user: Pubkey – User’s public key unlocking tokens.
  + amount: u64 – Amount of tokens unlocked.
* When Emitted: At the end of unlock\_for\_migration after returning tokens to the user.
* Usage: Confirms the return of tokens if migration is inactive.

Locked Tokens Burned

* Purpose: Records the burning of locked tokens as part of migration.
* Fields:
  + user: Pubkey – User whose tokens were burned.
  + amount: u64 – Amount of tokens burned.
* When Emitted: In burn\_locked\_tokens after burning the locked tokens.
* Usage: Verifies token supply adjustments during migration.

Migration Confirmed

* Purpose: Confirms a user’s migration process is complete.
* Fields:
  + user: Pubkey – User who completed migration.
* When Emitted: At the end of burn\_locked\_tokens after marking the migration record as complete.
* Usage: Tracks the finalization of individual migrations.

### Miscellaneous

Events for miscellaneous conditions or updates not fitting other categories.

Insufficient Funds

* Purpose: Logs when an operation fails due to insufficient funds in a pool.
* Fields:
  + required: u64 – Amount of tokens required.
  + available: u64 – Amount of tokens available.
* When Emitted: In instructions like claim\_rewards or airdrop when pool funds are insufficient (though not explicitly emitted in the provided code, included for completeness based on error checks).
* Usage: Helps diagnose funding shortages in treasury pools.

Voting Power Updated

* Purpose: Indicates changes in the total voting power due to staking activities.
* Fields:
  + old\_power: u64 – Previous total voting power.
  + new\_power: u64 – Updated total voting power.
* When Emitted: In stake or unstake after recalculating state.total\_voting\_power.
* Usage: Tracks shifts in governance influence based on staking changes.

## Error Codes

This section provides a comprehensive list of all error codes that can be returned by the smart contract. Each error code is accompanied by its message and a detailed description of the conditions under which it is triggered, as well as its significance. These error codes are defined in Errors.rs and utilized throughout the contract (e.g., in Instructions.rs) to enforce rules, handle exceptions, and ensure secure and predictable behavior.

### Authorization Errors

Errors related to permissions, ownership, or multisig requirements.

Paused

* + Message: "Contract is paused"
  + Description: Triggered when an operation (e.g., transfer, stake, claim\_rewards) is attempted while the contract is paused, except for specific admin actions or zero-amount transfers. Pausing is controlled via initiate\_pause and confirm\_pause by multisig owners.

NotPaused

* + Message: "Contract is not paused"
  + Description: Occurs when an operation requiring the contract to be paused (e.g., initiate\_resume) is attempted while the contract is active.

UnauthorizedSender

* + Message: "Unauthorized: Sender does not own the token account"
  + Description: Raised in the transfer instruction if the sender’s public key does not match the owner of the sender\_token\_account.

InsufficientSigners

* + Message: "Unauthorized: Insufficient signers for multisig operation"
  + Description: Thrown when a multisig operation (e.g., initialize, submit\_proposal) lacks the required number of unique signers as specified by the threshold in the Multisig account.

SignerNotOwner

* + Message: "Unauthorized: Signer is not a multisig owner"
  + Description: Occurs in multisig operations when a signer is not listed among the owners in the Multisig account.

CallerNotWhitelisted

* + Message: "Caller is not whitelisted"
  + Description: Triggered in whitelisted\_transfer if the calling program’s ID is not in the whitelisted\_contracts list in ContractState.

InvalidDestination

* + Message: "Invalid transfer destination"
  + Description: Raised in whitelisted\_transfer if the recipient’s token account owner is not in the allowed\_destinations list in ContractState.

VersionMismatch

* + Message: "Version mismatch in whitelisted contract"
  + Description: Occurs in whitelisted\_transfer when the stored version hash of the whitelisted contract does not match the expected hash (e.g., the program ID bytes).

### State Errors

Errors related to the contract’s or accounts’ current state.

AlreadyPaused

* + Message: "Contract is already paused"
  + Description: Thrown in initiate\_pause if the contract is already paused (state.paused is true).

UninitializedAccount

* + Message: "Account not initialized"
  + Description: Raised in claim\_presale\_tokens if the vesting account has a total\_amount of zero, indicating it has not been initialized.

MigrationNotActive

* + Message: "Migration is not active"
  + Description: Triggered in lock\_for\_migration if migration\_state.migration\_active is false.

MigrationActive

* + Message: "Migration is active"
  + Description: Occurs in unlock\_for\_migration if migration\_state.migration\_active is true, preventing unlocking during active migration.

PresaleNotActive

* + Message: "Presale is not active"
  + Description: Raised in buy\_presale if state.presale\_active is false.

AlreadyMigrated

* + Message: "Already migrated"
  + Description: Thrown in lock\_for\_migration or unlock\_for\_migration if the migration\_record.migrated flag is true.

InvalidMigrationUser

* + Message: "Invalid migration user"
  + Description: Occurs in burn\_locked\_tokens if the specified user does not match migration\_record.user.

### Validation Errors

Errors due to invalid inputs, parameters, or conditions.

InvalidThreshold

* + Message: "Invalid threshold"
  + Description: Triggered in initialize if the multisig threshold is less than 2.

TooFewOwners

* + Message: "Too few owners in multisig"
  + Description: Raised in initialize or initiate\_set\_multisig if the number of multisig owners is less than 3 or 2, respectively.

DuplicateOwners

* + Message: "Duplicate owners in multisig"
  + Description: Occurs in initialize or initiate\_set\_multisig if the owners list contains duplicate public keys.

TooManyOwners

* + Message: "Too many owners in multisig"
  + Description: Thrown in initiate\_set\_multisig if the number of owners exceeds Multisig::MAX\_OWNERS (5).

ThresholdExceedsOwners

* + Message: "Threshold exceeds number of owners"
  + Description: Raised in initiate\_set\_multisig if the threshold exceeds the number of owners.

InvalidTaxRate

* + Message: "Invalid tax rate"
  + Description: Triggered in execute\_proposal (type 0) if the proposed tax\_rate is not between 100 (1%) and 1000 (10%) basis points.

InvalidStakingTiers

* + Message: "Invalid staking tiers"
  + Description: Occurs in execute\_proposal (type 1) if staking tiers are not in ascending order, below the minimum (20,000 tokens), or above 10% of TOTAL\_SUPPLY.

InvalidTaxAllocationTotal

* + Message: "Invalid tax allocation total"
  + Description: Raised in execute\_proposal (type 2) if the sum of tax allocations does not equal 10,000 basis points (100%).

InvalidReductionThresholds

* + Message: "Invalid reduction thresholds"
  + Description: Thrown in execute\_proposal (type 3) if reduction thresholds are not in ascending order or outside 10%–90% range.

InvalidReductionFactor

* + Message: "Invalid reduction factor"
  + Description: Occurs in execute\_proposal (type 3) if reduction factors are not between 100 (10%) and 2000 (200%).

InvalidSellLimit

* + Message: "Invalid sell limit"
  + Description: Raised in execute\_proposal (types 5, 6) if the sell limit is not between 0.1% and 2% of TOTAL\_SUPPLY.

InvalidTransferLimit

* + Message: "Invalid transfer limit"
  + Description: Triggered in execute\_proposal (types 7, 8) if the transfer limit is not between 0.1% and 2% of TOTAL\_SUPPLY.

InvalidTaxThreshold

* + Message: "Invalid tax threshold"
  + Description: Occurs in execute\_proposal (type 9) if the progressive tax threshold is not between 0.1% and 2% of TOTAL\_SUPPLY.

InvalidStakingReward

* + Message: "Invalid staking reward"
  + Description: Raised in execute\_proposal (type 10) if staking rewards are not between 100 and 1,000,000 tokens (scaled by decimals).

InvalidProposalType

* + Message: "Invalid proposal type"
  + Description: Thrown in execute\_proposal if the proposal\_type is unrecognized (beyond types 0–11).

InvalidProposalValueCount

* + Message: "Invalid proposal value count"
  + Description: Occurs in execute\_proposal if the number of proposal\_values does not match the expected count for the proposal\_type.

DescriptionTooLong

* + Message: "Description too long"
  + Description: Raised in submit\_proposal or initiate\_pause if the description exceeds Proposal::MAX\_DESCRIPTION\_LEN (200) or ContractState::MAX\_PAUSE\_REASON\_LEN (100), respectively.

TooManyProposalValues

* + Message: "Too many proposal values"
  + Description: Triggered in submit\_proposal if proposal\_values exceeds Proposal::MAX\_PROPOSAL\_VALUES (7).

TooManyRecipients

* + Message: "Too many recipients"
  + Description: Occurs in airdrop or distribute\_lp\_incentives if the number of recipients exceeds 100.

InvalidRecipientAccounts

* + Message: "Invalid recipient accounts"
  + Description: Raised in airdrop or distribute\_lp\_incentives if the number of provided token accounts does not match the recipient list.

InvalidRecipientAccount

* + Message: "Invalid recipient account"
  + Description: Thrown in airdrop or distribute\_lp\_incentives if a recipient’s token account owner does not match the intended recipient.

InvalidContract

* + Message: "Invalid contract"
  + Description: Occurs in add\_whitelisted\_contract if the contract account is not executable.

TooManyWhitelistedContracts

* + Message: "Too many whitelisted contracts"
  + Description: Raised in confirm\_whitelist\_change if adding a contract would exceed ContractState::MAX\_WHITELISTED (20).

ContractNotWhitelisted

* + Message: "Contract not whitelisted"
  + Description: Triggered in confirm\_whitelist\_change (removal) if the contract is not in whitelisted\_contracts.

KYCRequired

* + Message: "KYC verification required for this purchase"
  + Description: Occurs in buy\_presale if the purchase amount is ≥ $1000 USDT and presale\_purchase.kyc\_verified is false.

### Operation-Specific Errors

Errors tied to specific operations like transfers, staking, or governance.

MaxSellTxnLimitExceeded

* + Message: "Maximum sell transaction limit exceeded"
  + Description: Raised in transfer if a sell transaction amount exceeds state.max\_sell\_txn\_limit.

DailySellLimitExceeded

* + Message: "Daily sell limit exceeded"
  + Description: Occurs in transfer if the daily sell total exceeds state.daily\_sell\_limit.

MaxTransferLimitExceeded

* + Message: "Maximum transfer limit exceeded"
  + Description: Thrown in transfer if a non-sell transfer exceeds state.max\_transfer\_limit.

DailyTransferLimitExceeded

* + Message: "Daily transfer limit exceeded"
  + Description: RaisedMason in transfer if the daily non-sell transfer total exceeds state.daily\_transfer\_limit.

CooldownActive

* + Message: "Transfer cooldown active"
  + Description: Occurs in transfer if less than TXN\_COOLDOWN (60 seconds) has passed since the last transaction.

VestingNotStarted

* + Message: "Vesting period has not started"
  + Description: Triggered in claim\_presale\_tokens or claim\_team\_vesting if the current time is before the vesting start time.

NoRewards

* + Message: "No rewards available"
  + Description: Raised in reward-claiming functions (e.g., claim\_rewards, claim\_lp\_rewards) if no rewards are available.

ProposalExpired

* + Message: "Proposal has expired"
  + Description: Occurs in vote if the proposal’s end\_time has passed or its status is not active.

VotingPeriodNotEnded

* + Message: "Voting period has not ended"
  + Description: Thrown in execute\_proposal if the current time is before the proposal’s end\_time.

NoticePeriodNotMet

* + Message: "Notice period for proposal execution not met"
  + Description: Raised in execute\_proposal if the current time is before the proposal’s execution\_time.

ProposalAlreadyExecuted

* + Message: "Proposal has already been executed"
  + Description: Occurs in execute\_proposal if the proposal’s status is not 0 (active).

WithdrawalDelayNotMet

* + Message: "Withdrawal delay not met"
  + Description: Triggered in complete\_withdrawal if the current slot is before the required delay slot.

LockPeriodNotMet

* + Message: "Lock period not met"
  + Description: Raised in unstake, unstake\_lp, or claim\_lp\_rewards if the staking lock period (e.g., 7 or 30 days) has not elapsed.

InsufficientStakedAmount

* + Message: "Insufficient staked amount for operation"
  + Description: Occurs in unstake\_lp if the requested unstake amount exceeds the staked amount.

NoLockedTokens

* + Message: "No locked tokens available"
  + Description: Thrown in unlock\_for\_migration or burn\_locked\_tokens if migration\_record.locked\_amount is zero.

AmountTooSmallAfterTax

* + Message: "Amount too small after tax"
  + Description: Raised in transfer or whitelisted\_transfer if the net amount after tax is zero or negative.

InsufficientTierForVoting

* + Message: "Insufficient tier for voting"
  + Description: Occurs in vote if the staker’s tier is 255 (ineligible) or 0 (no voting power).

InvalidImmediateAmount

* + Message: "Invalid immediate amount"
  + Description: Triggered in update\_team\_vesting if immediate\_amount exceeds total\_amount.

InvalidTeamMember

* + Message: "Invalid team member"
  + Description: Raised in cancel\_team\_vesting if the team\_member does not match team\_vesting.team\_member.

VestingCanceled

* + Message: "Vesting canceled"
  + Description: Occurs in claim\_team\_vesting if team\_vesting.canceled is true.

ClaimCooldownNotMet

* + Message: "Claim cooldown not met"
  + Description: Thrown in claim\_freelancer\_vesting if less than 3 days have passed since the last claim.

ExceedsVestingTotal

* + Message: "Exceeds vesting total"
  + Description: Raised in release\_freelancer\_milestone if the new released\_amount would exceed total\_amount.

MigrationToggleCooldown

* + Message: "Migration toggle cooldown active"
  + Description: Occurs in toggle\_migration\_active if less than 7 days have passed since the last toggle.

BatchSizeTooLarge

* + Message: "Batch size too large"
  + Description: Triggered in process\_daily\_rewards if batch\_size exceeds 50.

WhitelistDelayNotMet

* + Message: "Whitelist delay not met"
  + Description: Raised in confirm\_whitelist\_change if less than 72 hours have passed since initiation.

TimeLockNotMet

* + Message: "Time lock requirement not met"
  + Description: Occurs in confirm\_pause, confirm\_resume, or confirm\_set\_multisig if the 24-hour timelock has not elapsed.

### Technical Errors

Errors related to computational or structural constraints.

ReentrancyGuardTriggered

* + Message: "Reentrancy guard triggered: Operation already in progress"
  + Description: Thrown by the ReentrancyGuard if an instruction is called while state.is\_processing is true.

InsufficientStakingPoolFunds

* + Message: "Insufficient funds in staking pool"
  + Description: Raised in unstake or claim\_rewards if treasury.staking\_pool is less than the reward amount.

InsufficientAirdropFunds

* + Message: "Insufficient funds in airdrop pool"
  + Description: Occurs in airdrop if treasury.airdrop\_pool is less than the total airdrop amount.

InsufficientLiquidityIncentiveFunds

* + Message: "Insufficient funds in liquidity incentive pool"
  + Description: Thrown in distribute\_lp\_incentives if treasury.liquidity\_incentive is insufficient.

ArithmeticOverflow

* + Message: "Arithmetic overflow occurred"
  + Description: Triggered across various instructions (e.g., transfer, stake) when a calculation exceeds u64 or u128 limits.

VectorOverflow

* + Message: "Vector overflow: Too many elements"
  + Description: Raised in initialize if dex\_programs exceeds ContractState::MAX\_DEXES (10).

InvalidAccounts

* + Message: "Invalid accounts provided"
  + Description: Occurs in initialize if the number of remaining accounts does not match initial\_dex\_programs.

InvalidDexProgram

* + Message: "Provided DEX program ID is not executable"
  + Description: Thrown in initialize or update\_dex\_programs if a DEX program account is not executable.

### Presale-Specific Errors

Errors specific to presale operations.

PresaleSupplyExceeded

* + Message: "Presale supply exceeded"
  + Description: Raised in buy\_presale if the total sold would exceed PRESALE\_SUPPLY.

PresaleMaxPerWalletExceeded

* + Message: "Presale maximum per wallet exceeded"
  + Description: Occurs in buy\_presale if the purchase would exceed PRESALE\_MAX\_PER\_WALLET (2M tokens).

# Features

This section highlights the key capabilities of the smart contract, emphasizing its operational functionalities, governance mechanisms, staking and reward systems, security features, and additional utilities.

1. Token Operations

This subsection covers the mechanisms related to token transfers, presale participation, and token migration, which are fundamental to the contract’s tokenomics and lifecycle management.

**Token Transfers**

**Secure Transfers with Taxation:** Enables token transfers with an adjustable tax rate (initially 5%, configurable via governance between 1% and 10%), where taxes are allocated to burn, treasury, liquidity pool, LP incentives, charity, and team wallets (configurable allocations summing to 100%).

**Whitelisted Transfers:** Approved external contracts can perform transfers with a reduced tax rate (half of the standard rate), restricted to pre-approved destinations, enhancing flexibility for ecosystem partners.

**Progressive Taxation:** Applies a double tax rate for transfers exceeding a threshold (initially 0.5% of total supply), deterring large-scale dumps and stabilizing token value.

**Transaction Limits:** Enforces maximum transaction limits (0.5% of total supply per sell/transfer) and daily limits (0.5% of total supply per wallet), tracked via a 24-hour bucket system, to prevent market manipulation.

**Cooldown Mechanism:** Imposes a 1-minute cooldown between transactions and a 24-hour cooldown for large transfers (above 0.1% of total supply), mitigating rapid trading exploits.

**Presale Participation**

**USDT-Based Purchases**: Users can buy tokens during the presale phase using USDT at a fixed rate (0.0016 USDT per token), with a total supply cap of 250 million tokens.

**Purchase Limits:** Caps individual wallet purchases at 2 million tokens to ensure broad participation.

**KYC Enforcemen**t: Requires KYC verification for purchases of $1000 USDT or more, enhancing regulatory compliance.

**Vesting Schedule:** Tokens are vested post-launch (starting May 1, 2025), with 10% unlocked initially and 10% weekly thereafter until fully vested at 100%.

**Token Migration**

**Secure Upgrade Path:** Facilitates token upgrades via a lock-unlock-burn process, allowing users to lock tokens for migration and either unlock them or burn them for new tokens.

**Multisig Control:** Migration activation and token burning are governed by multisig, ensuring transparency and preventing unauthorized actions.

**Toggle Mechanism:** Migration can be toggled on/off by multisig with a 7-day cooldown between toggles, providing controlled transition periods.

2. Staking and Rewards

This subsection details the staking options and reward distribution systems, incentivizing user participation and liquidity provision.

**Multi-Tier Staking**

**Tiered System:** Offers four staking tiers based on token amount (e.g., 20K, 100K, 500K, 5M tokens) and lock duration (7, 14, 30, 30 days), with higher tiers unlocking greater rewards and voting power.

**Dynamic Rewards:** Weekly rewards (e.g., 500, 2,500, 12,500, 125,000 tokens) are adjustable via governance and scaled by pool depletion factors (e.g., 51.2% to 100% based on thresholds like 25%, 50%, 75%).

**Long-Term Incentives:** Rewards and voting power increase with staking duration (e.g., 1.5x multiplier at 60 days, ~2x at 90 days), encouraging sustained commitment.

**Liquidity Provider (LP) Staking**

LP Token Staking: Users can stake LP tokens with a 7-day lock period, contributing to ecosystem liquidity.

**Daily Rewards:** Distributes liquidity incentives daily from a dedicated pool (initially 0, funded by tax allocations), proportional to staked amounts, with batch processing capped at 50 stakers per transaction for efficiency.

**Unclaimed Rewards:** Rewards accumulate as unclaimed balances, claimable after the lock period, optimizing gas usage on Solana.

**Reward Claiming**

**Flexible Claims:** Stakers can claim rewards without unstaking, provided lock periods are met, with rewards calculated based on time staked and pool status.

**Batch Processing:** LP rewards are processed in batches (up to 50 accounts) to manage compute budget, ensuring scalability and fairness in distribution.

3. Governance

This subsection outlines the decentralized governance framework, empowering stakers to influence contract parameters.

**Decentralized Decision-Making**

**Staker Voting:** Stakers with tier 1 or higher (minimum 100K tokens staked for 14 days) can vote, with power scaled by tier (e.g., 1, 4, 20 votes) and duration multipliers (up to ~2x).

**Parameter Control**: Adjustable parameters include tax rates, staking tiers, reward rates, and transaction limits, ensuring adaptability to community needs.

**Proposal System**

**Proposal Types:** Supports updates to tax rates, staking tiers, tax allocations, reward factors, sell/transfer limits, and whitelist management, submitted by multisig with detailed descriptions (up to 200 characters).

**Voting Process:** Proposals undergo a 14-day voting period, requiring 30% quorum, 51% approval, and 20% of total voting power, followed by a 3-day execution delay for transparency.

**Execution:** Approved proposals are executed automatically post-delay, applying changes atomically to maintain contract integrity.

**Multisig Controls**

**Administrative Oversight:** Critical actions (e.g., pausing, airdrops, vesting updates) require approval from a multisig with 2-5 owners and a threshold (minimum 2), preventing single-point failures.

**Time-Locked Changes**: Multisig updates and whitelist changes are subject to a 24-hour or 72-hour delay, respectively, enhancing security.

4. Security and Control Mechanisms

This subsection emphasizes the contract’s robust security features and operational safeguards.

**Reentrancy Protection**

**Scope-Based Guard:** Prevents recursive calls using a reentrancy flag (is\_processing), ensuring atomic operation completion and blocking exploits.

**Transaction Limits and Cooldowns**

**Bucket System:** Tracks sell and transfer volumes in hourly buckets over 24 hours, enforcing daily limits (0.5% of total supply) to stabilize liquidity.

**Cooldowns:** Applies a 1-minute transaction cooldown and a 24-hour cooldown for large transfers, reducing the risk of flash attacks or rapid dumps.

**Time-Locks and Delays**

**Critical Action Delays:** Pausing (24 hours), multisig changes (24 hours), whitelist updates (72 hours), and large withdrawals (48 hours for amounts >0.5% of supply) are time-locked, allowing community oversight.

**Notice Periods:** Governance changes include a 3-day post-voting delay, ensuring deliberate execution.

**Whitelisted Contracts**

**Privileged Operations:** Whitelisted contracts enjoy reduced tax rates and restricted transfer destinations, with versioning (via program ID hash) to revoke outdated contracts.

**Management:** Added/removed via multisig with a 72-hour delay, capped at 20 contracts for efficiency.

**Emergency Controls**

**Pause Functionality:** Multisig can pause the contract (with a 24-hour delay), halting most operations except zero-amount transfers and admin actions, with a reason logged for transparency.

**Resume Process:** Resumption follows a similar 24-hour delay, ensuring controlled recovery.

5. Additional Features

This subsection covers supplementary functionalities enhancing usability and ecosystem support.

**Airdrop and Incentive Distribution**

**Airdrops:** Multisig-controlled airdrops distribute tokens from an 8% treasury pool (initially 52.8M tokens) to up to 100 recipients per transaction, promoting community engagement.

**LP Incentives:** Distributes incentives (funded by tax allocations) to LP stakers, capped at 100 recipients per batch, fostering liquidity stability.

**Vesting for Team and Freelancers**

**Team Vesting:** Tokens vest over time (10% monthly after a 3-month cliff, capped at 20M tokens per claim), with immediate allocations and cancellation options via multisig.

**Freelancer Vesting:** Tokens are milestone-based, released by multisig, with claims limited to 500K tokens every 3 days, ensuring gradual distribution.

**Query Functions**

Transparency: Provides read-only access to pending rewards (via query\_pending\_rewards) and contract state (via query\_state), enabling users to monitor their stakes and the contract’s configuration.

# Tokenomics

1. Total Supply and Decimals

* **Total Supply:** 1,000,000,000 tokens.
  + Scaled Value: 1,000,000,000 \* 10^9 (1 trillion units with 9 decimals).
  + Source: TOTAL\_SUPPLY in Constants.rs.
* **Decimals:** 9.
  + Source: TOKEN\_DECIMALS in Constants.rs.
  + Purpose: Allows fractional token amounts with precision up to 10^-9.

2. Presale Details

* **Presale Supply:** 250,000,000 tokens.
  + Scaled Value: 250,000,000 \* 10^9 (250 billion units).
  + Source: PRESALE\_SUPPLY in Constants.rs.
* **Presale Price:** 0.0016 USDT per token.
  + Scaled Value: 1600 USDT units per 10^9 token units.
  + Source: PRESALE\_PRICE\_PER\_TOKEN in Constants.rs.
* **Purchase Limits:**
  + Maximum per Wallet: 2,000,000 tokens (2 billion units).
  + Source: PRESALE\_MAX\_PER\_WALLET in Constants.rs.
* **KYC Requirement:** Mandatory for purchases ≥ 1000 USDT.
  + Source: buy\_presale instruction in Instructions.rs.
* **Vesting Schedule:**
  + Initial Unlock: 10% at launch (May 1, 2025).
  + Weekly Unlock: 10% per week thereafter until fully vested (10 weeks total).
  + Source: claim\_presale\_tokens instruction in Instructions.rs.
  + Launch Timestamp: Unix timestamp 1746057600 (May 1, 2025).
    - Source: LAUNCH\_TIMESTAMP in Constants.rs.

3. Treasury Allocations

* **Treasury Reserve:** 660,000,000 tokens.
  + Scaled Value: 660,000,000 \* 10^9 (660 billion units).
  + Source: TREASURY\_RESERVE in Constants.rs.
* **Initial Liquidity:** 90,000,000 tokens.
  + Scaled Value: 90,000,000 \* 10^9 (90 billion units).
  + Source: INITIAL\_LIQUIDITY in Constants.rs.
* **Treasury Pool Breakdown:**
  + Staking Pool: 30% of treasury (198,000,000 tokens or 198 billion units).
    - Source: STAKING\_POOL\_PCT in Constants.rs, calculated in initialize instruction.
  + Airdrop Pool: 8% of treasury (52,800,000 tokens or 52.8 billion units).
    - Source: AIRDROP\_POOL\_PCT in Constants.rs.
  + Governance Reserve: 16% of treasury (105,600,000 tokens or 105.6 billion units).
    - Source: GOVERNANCE\_RESERVE\_PCT in Constants.rs.
  + Marketing Fund: 18% of treasury (118,800,000 tokens or 118.8 billion units).
    - Source: MARKETING\_FUND\_PCT in Constants.rs.
  + Emergency Fund: 5% of treasury (33,000,000 tokens or 33 billion units).
    - Source: EMERGENCY\_FUND\_PCT in Constants.rs.
  + Team Pool: 18% of treasury (118,800,000 tokens or 118.8 billion units).
    - Source: TEAM\_POOL\_PCT in Constants.rs.
  + Liquidity Incentive Pool: Initially 0, funded dynamically via tax allocations.
    - Source: initialize instruction sets liquidity\_incentive = 0.
* **Management:** Multisig can transfer tokens between pools using transfer\_between\_pools.
  + Source: TransferBetweenPools instruction in Lib.rs.

4. Tax System

* **Initial Tax Rate:** 5% (500 basis points).
  + Adjustable Range: 1% to 10% (100–1000 basis points) via governance proposal type 0.
  + Source: INITIAL\_TAX\_RATE and execute\_proposal in Instructions.rs.
* **Tax Allocations** (in basis points of tax amount, totaling 10,000):
  + Burn: 20% (2000 bp).
  + Treasury (Staking Pool): 20% (2000 bp).
  + Liquidity Pool: 24% (2400 bp).
  + LP Incentives: 6% (600 bp).
  + Charity: 20% (2000 bp).
  + Team: 10% (1000 bp).
  + Source: initialize sets initial allocations; adjustable via proposal type 2 in execute\_proposal.
* **Progressive Tax:**
  + Threshold: Transfers > 0.5% of total supply (5,000,000 tokens or 5 billion units).
  + Rate: Triples the base tax (e.g., 15% if base is 5%).
  + Source: PROGRESSIVE\_TAX\_THRESHOLD in Constants.rs, logic in transfer instruction.
* **Whitelisted Transfers:**
  + Tax Rate: Half of the standard rate (e.g., 2.5% if base is 5%).
  + Conditions: Must be to allowed destinations, called by whitelisted contracts.
  + Source: whitelisted\_transfer instruction in Instructions.rs.
* **Tax Calculation:**
  + Uses ceiling division for precision (e.g., tax = (amount \* rate \* 10^7 + 9999999) / (10000 \* 10^7)).
  + Source: transfer and whitelisted\_transfer instructions.

5. Staking and Rewards

* **Staking Tiers:**
  + Tier 0: 20,000 tokens (20 million units), 7-day lock.
  + Tier 1: 100,000 tokens (100 million units), 14-day lock.
  + Tier 2: 500,000 tokens (500 million units), 30-day lock.
  + Tier 3: 5,000,000 tokens (5 billion units), 30-day lock.
  + Source: INITIAL\_STAKING\_TIERS and STAKING\_DURATIONS in Constants.rs, adjustable via proposal type 1.
* **Weekly Staking Rewards:**
  + Tier 0: 500 tokens (500 million units).
  + Tier 1: 2,500 tokens (2.5 billion units).
  + Tier 2: 12,500 tokens (12.5 billion units).
  + Tier 3: 125,000 tokens (125 billion units).
  + Source: STAKING\_REWARDS in Constants.rs, adjustable via proposal type 10.
* **Reward Adjustments:**
  + Depletion Factors: Based on staking pool balance relative to initial allocation (198 billion units).
    - Thresholds: 25% (250), 50% (500), 75% (750) in basis points of initial pool.
    - Factors: 51.2% (512), 64% (640), 80% (800), 100% (1000) of base reward.
  + Source: reduction\_thresholds and reduction\_factors in ContractState, set in initialize, adjustable via proposal type 3.
* **LP Staking Rewards:**
  + Source: Liquidity Incentive Pool, funded by tax allocations (6% of tax).
  + Distribution: Daily, proportional to staked LP tokens, processed in batches (max 50 stakers).
  + Lock Period: 7 days.
  + Source: process\_daily\_rewards and stake\_lp/unstake\_lp instructions.

6. Transaction Limits

* **Sell Transactions:**
  + Max per Transaction: 0.5% of total supply (5,000,000 tokens or 5 billion units).
  + Daily Limit per Wallet: 0.5% of total supply.
  + Source: MAX\_SELL\_TXN\_LIMIT and DAILY\_SELL\_LIMIT in Constants.rs, adjustable via proposal types 5 and 6.
* **Non-Sell Transfers:**
  + Max per Transaction: 0.5% of total supply.
  + Daily Limit per Wallet: 0.5% of total supply.
  + Source: MAX\_TRANSFER\_LIMIT and DAILY\_TRANSFER\_LIMIT in Constants.rs, adjustable via proposal types 7 and 8.
* **Tracking:**
  + Bucket System: 24 hourly buckets track sell and transfer volumes, reset daily.
  + Source: TransactionRecord struct and advance\_buckets in Instructions.rs.
* **Cooldowns:**
  + Transaction Cooldown: 1 minute between transactions.
    - Source: TXN\_COOLDOWN in Constants.rs.
  + Large Transfer Cooldown: 24 hours for transfers ≥ 0.1% of total supply (1,000,000 tokens or 1 billion units).
    - Source: TRANSFER\_COOLDOWN and TRANSACTION\_TRACKING\_THRESHOLD in Constants.rs.

7. Vesting Schedules

* **Presale Vesting:**
  + Initial Unlock: 10% at launch (May 1, 2025).
  + Weekly Unlock: 10% per week until fully vested.
  + Source: claim\_presale\_tokens instruction.
* **Team Vesting:**
  + Cliff: 3 months post-launch.
  + Monthly Unlock: 10% per month after cliff.
  + Claim Cap: 20,000,000 tokens (20 billion units) per claim.
  + Cancellation: Multisig can cancel vesting.
  + Source: claim\_team\_vesting and cancel\_team\_vesting instructions.
* **Freelancer Vesting:**
  + Milestone-Based: Released by multisig via release\_freelancer\_milestone.
  + Claim Limit: 500,000 tokens (500 million units) every 3 days.
  + Cooldown: 3 days between claims.
  + Source: claim\_freelancer\_vesting and release\_freelancer\_milestone instructions.

8. Additional Economic Mechanisms

* **Airdrops:**
  + Source: Airdrop Pool (52.8 billion units initially).
  + Distribution: Multisig-controlled, up to 100 recipients per transaction.
  + Source: airdrop instruction.
* **LP Incentives:**
  + Funding: From tax allocations (6% of tax) into Liquidity Incentive Pool.
  + Distribution: Daily to LP stakers via process\_daily\_rewards.
* **Token Migration:**
  + Mechanism: Lock-unlock-burn process for upgrades.
  + Control: Multisig toggles migration with a 7-day cooldown.
  + Source: lock\_for\_migration, unlock\_for\_migration, burn\_locked\_tokens, toggle\_migration\_active instructions.
* **Treasury Management:**
  + Pool Transfers: Multisig can reallocate between pools.
  + Withdrawal: Multisig-initiated with delay (48 hours for amounts > 0.5% of supply).
  + Source: transfer\_between\_pools, initiate\_withdrawal, complete\_withdrawal instructions.

Summary Table

|  |  |
| --- | --- |
| **Aspect** | **Details** |
| Total Supply | 1,000,000,000 tokens (1 trillion units) |
| Presale Supply | 250,000,000 tokens (250 billion units) |
| Treasury Reserve | 660,000,000 tokens (660 billion units) |
| Initial Liquidity | 90,000,000 tokens (90 billion units) |
| Initial Tax Rate | 5% (adjustable 1%–10%) |
| Tax Allocations | Burn: 20%, Treasury: 20%, Liquidity: 24%, LP Incentives: 6%, Charity: 20%, Team: 10% |
| Progressive Tax | 2x tax for transfers > 5M tokens |
| Staking Tiers | 20K, 100K, 500K, 5M tokens; 7–30 day locks |
| Weekly Staking Rewards | 500, 2,500, 12,500, 125,000 tokens |
| Transaction Limits | 0.5% per sell/transfer; 0.5% daily per wallet |
| Presale Vesting | 10% at launch, 10% weekly (10 weeks total) |
| Team Vesting | 10% monthly after 3-month cliff, 20M cap per claim |
| Freelancer Vesting | Milestone-based, 500K every 3 days |

# Governance

Governance allows stakers to propose and vote on changes to contract parameters, such as tax rates, staking tiers, and transaction limits, while a multisig handles administrative actions like airdrops, vesting updates, and emergency pauses.

Key Components:

* + Proposal System: Enables stakers to propose and vote on parameter changes.
  + Voting System: Determines proposal outcomes based on staker voting power.
  + Multisig Controls: Manages sensitive operations with multiple signers.

Source: submit\_proposal, vote, execute\_proposal, Multisig struct, and related instructions in Instructions.rs and State.rs.

1.Proposal Types:

* + 0: Tax Rate:
    - Adjusts tax\_rate (1%–10%, 100–1000 basis points).
    - Requires 1 value.
    - Example: Change tax from 5% to 7%.
  + 1: Staking Tiers:
    - Updates staking\_tiers (minimum 20,000 tokens, max 10% of total supply).
    - Requires 4 values, strictly increasing.
  + 2: Tax Allocations:
    - Adjusts burn\_alloc, treasury\_alloc, liquidity\_pool\_alloc, lp\_incentive\_alloc, charity\_alloc, team\_alloc (sum to 10,000 bp, max 50% per category).
    - Requires 6 values.
  + 3: Reward Reduction:
    - Updates reduction\_thresholds (10%–90%) and reduction\_factors (10%–200%).
    - Requires 7 values (3 thresholds, 4 factors).
  + 4: Launch Timestamp:
    - Adjusts launch\_timestamp (within 30 days before or 1 year after May 1, 2025).
    - Requires 1 value.
  + 5: Max Sell Transaction Limit:
    - Updates max\_sell\_txn\_limit (0.1%–2% of total supply).
    - Requires 1 value.
  + 6: Daily Sell Limit:
    - Updates daily\_sell\_limit (0.1%–2% of total supply).
    - Requires 1 value.
  + 7: Max Transfer Limit:
    - Updates max\_transfer\_limit (0.1%–2% of total supply).
    - Requires 1 value.
  + 8: Daily Transfer Limit:
    - Updates daily\_transfer\_limit (0.1%–2% of total supply).
    - Requires 1 value.
  + 9: Progressive Tax Threshold:
    - Updates progressive\_tax\_threshold (0.1%–2% of total supply).
    - Requires 1 value.
  + 10: Staking Rewards:
    - Updates staking\_rewards (100–1,000,000 tokens per tier).
    - Requires 4 values.
  + 11: Whitelist Update:
    - Adds/removes whitelisted contracts (whitelisted\_contracts).
    - Requires ≥2 values (add count, remove count, contract Pubkeys).
  + Source: execute\_proposal instruction in Instructions.rs.

2. Proposal System

* Submission:
  + Who Can Submit: Multisig owners (2–5 signers, minimum 2 approvals).
  + Requirements:
    - Description: Up to 200 characters (Proposal::MAX\_DESCRIPTION\_LEN).
    - Proposal Values: Up to 7 values (Proposal::MAX\_PROPOSAL\_VALUES) specifying changes.
    - Validation: Must include valid proposal\_type (0–11) and appropriate proposal\_values count.
  + Process: Submitted via submit\_proposal, incrementing state.proposal\_count.
  + Source: submit\_proposal instruction, Proposal struct in State.rs.
* **Timing:**
  + Start Time: Set to current Unix timestamp upon submission.
  + End Time: 14 days after start (14 \* 86,400 seconds).
  + Execution Time: 3 days after voting ends (3 \* 86,400 seconds).
  + Source: submit\_proposal instruction.

3. Voting System

* **Eligibility:**
  + Requirement: Stakers in Tier 1 or higher (≥100,000 tokens staked for ≥14 days).
  + Exclusion: Tier 0 (20,000 tokens, 7 days) has no voting power.
  + Source: vote instruction, calculate\_voting\_power in Instructions.rs.
* **Voting Power:**
  + Base Power:
    - Tier 1: 1 vote.
    - Tier 2: 4 votes.
    - Tier 3: 20 votes.
  + Time-Based Multiplier:
    - ≥90 days: ~1.995x (1995/1000).
    - ≥60 days: 1.5x (1500/1000).
    - <60 days: 1.0x (1000/1000).
  + Calculation: (base\_power \* multiplier + 999) / 1000, capped by highest eligible tier.
  + Source: calculate\_voting\_power function in Instructions.rs.
* **Voting Process:**
  + Instruction: vote with proposal\_id and in\_favor (true/false).
  + Vote Record: Stored in VoteRecord (tracks staker, proposal, vote status).
  + Revoting: Stakers can change votes, subtracting previous power and adding new power.
  + Constraints:
    - Must vote before proposal.end\_time.
    - Proposal must be active (status == 0).
  + Source: Vote accounts struct, vote instruction in Instructions.rs.
* **Approval Criteria:**
  + Quorum: ≥30% of total voting power (VOTING\_QUORUM).
  + Approval: ≥51% of cast votes in favor (VOTING\_APPROVAL).
  + Threshold: ≥20% of total voting power in favor (VOTING\_THRESHOLD).
  + Outcome:
    - Approved: status = 1, changes applied.
    - Rejected: status = 2, no changes.
    - No Voting Power: Auto-rejected if total\_voting\_power == 0.
  + Source: execute\_proposal instruction, Constants.rs.

4. Proposal Execution

* **Execution:**
  + Instruction: execute\_proposal with proposal\_id.
  + Timing: After end\_time and execution\_time (17 days total from submission).
  + Validation:
    - Proposal must be unexecuted (status == 0).
    - Must pass quorum, approval, and threshold checks.
  + Changes: Applied atomically based on proposal\_type (see Proposal Types).
  + Source: execute\_proposal instruction.
* **Events:**
  + ProposalSubmittedEvent: Logs proposal\_id, description, proposal\_type.
  + VoteEvent: Logs staker, proposal\_id, in\_favor, power.
  + ProposalExecutedEvent: Logs proposal\_id, status.
  + Specific Events: E.g., TaxRateUpdated, StakingTiersUpdated for approved changes.
  + Source: Events in Instructions.rs.

5. Multisig Controls

* **Structure:**
  + Owners: 3–5 unique Pubkeys (Multisig::MAX\_OWNERS).
  + Threshold: Minimum 2 approvals (threshold >= 2).
  + Source: Multisig struct, initialize instruction.
* **Initialization:**
  + Set during initialize with initial\_owners and threshold.
  + Validates: No duplicates, executable DEX programs, threshold ≤ owners.
  + Source: initialize instruction.
* **Updates:**
  + Instruction: initiate\_set\_multisig and confirm\_set\_multisig.
  + Process:
    - Initiate with new owners and threshold.
    - 24-hour timelock (TimeLockNotMet error if premature).
    - Confirm to apply changes.
  + Validation: Ensures ≥2 owners, no duplicates, threshold ≤ owners.
  + Source: initiate\_set\_multisig, confirm\_set\_multisig instructions.
* **Actions Requiring Multisig:**
  + Proposal Submission: submit\_proposal.
  + Airdrops: airdrop (up to 100 recipients).
  + Pause/Resume: initiate\_pause, confirm\_pause, initiate\_resume, confirm\_resume (24-hour timelock).
  + Vesting Updates: update\_team\_vesting, cancel\_team\_vesting, update\_freelancer\_vesting, release\_freelancer\_milestone.
  + Withdrawals: initiate\_withdrawal, complete\_withdrawal (48-hour timelock for >0.5% of supply).
  + Whitelist Management: add\_whitelisted\_contract, remove\_whitelisted\_contract, confirm\_whitelist\_change (72-hour timelock).
  + DEX Updates: update\_dex\_programs (max 10 programs).
  + Migration Controls: burn\_locked\_tokens, confirm\_migration, toggle\_migration\_active (7-day cooldown).
  + Pool Transfers: transfer\_between\_pools.
  + Source: Respective instructions in Lib.rs and Instructions.rs.
* **Security:**
  + Validation: validate\_multisig checks signer count and ownership.
  + Timelocks: Enforce delays for critical actions (24–72 hours).
  + Events: E.g., MultisigUpdatedEvent, PauseEvent, WhitelistedContractAddedEvent.
  + Source: validate\_multisig function, event emissions in Instructions.rs.

6. Administrative Actions

* **Pause/Resume:**
  + Pause: initiate\_pause with reason (≤100 characters), confirm\_pause after 24 hours.
    - Effect: Halts most operations except zero-amount transfers and admin actions.
    - Event: PauseEvent with timestamp and reason.
  + Resume: initiate\_resume, confirm\_resume after 24 hours.
    - Effect: Restores normal operations, clears pause\_reason.
    - Event: ResumeEvent with timestamp.
  + Source: initiate\_pause, confirm\_pause, initiate\_resume, confirm\_resume instructions.
* **Airdrops:**
  + Distributes tokens from airdrop\_pool (initially 52.8 billion units).
  + Max 100 recipients per transaction.
  + Requires multisig approval.
  + Source: airdrop instruction.
* **Vesting Management:**
  + Team Vesting: Update/cancel via update\_team\_vesting, cancel\_team\_vesting.
  + Freelancer Vesting: Update/release via update\_freelancer\_vesting, release\_freelancer\_milestone.
  + Requires multisig approval.
  + Source: Vesting-related instructions.
* **Whitelist Management:**
  + Add/remove contracts via add\_whitelisted\_contract, remove\_whitelisted\_contract, confirmed after 72 hours.
  + Max 20 contracts (ContractState::MAX\_WHITELISTED).
  + Source: Whitelist-related instructions.
* **DEX Program Updates:**
  + Updates dex\_programs (max 10) for sell detection.
  + Requires multisig approval.
  + Source: update\_dex\_programs instruction.
* **Treasury Withdrawals:**
  + Initiated via initiate\_withdrawal, completed via complete\_withdrawal.
  + 48-hour timelock for amounts >0.5% of supply (WITHDRAWAL\_THRESHOLD).
  + Source: initiate\_withdrawal, complete\_withdrawal instructions.
* **Pool Transfers:**
  + Moves tokens between treasury pools (e.g., staking\_pool, airdrop\_pool).
  + Requires multisig approval.
  + Source: transfer\_between\_pools instruction.
* **Migration Controls:**
  + Toggles migration via toggle\_migration\_active (7-day cooldown).
  + Burns tokens via burn\_locked\_tokens, confirms via confirm\_migration.
  + Requires multisig approval.
  + Source: Migration-related instructions.

7. Governance Security and Safeguards

To protect the governance system from abuse, manipulation, and centralization risks, several security measures are implemented:

**Voting Power Cap**

* Description: Voting power is capped based on the highest staking tier a staker qualifies for, determined by both the staked amount and duration. This prevents large token holders from gaining disproportionate influence without meeting the required staking commitment.
* Purpose: Mitigates short-term manipulation by ensuring influence scales with long-term participation, balancing power distribution.
* Implementation: The calculate\_voting\_power function in Instructions.rs limits power by applying the multiplier (e.g., 1.995x at 90+ days) to the base power of the highest eligible tier (get\_highest\_eligible\_tier), preventing excessive votes beyond tier constraints.

**Multisig Proposal Submission**

* Description: Only the multisig (2–5 owners, threshold ≥2) can submit governance proposals, requiring pre-approval before community voting.
* Purpose: Reduces spam and malicious proposals by filtering submissions through trusted parties, enhancing governance efficiency and security.
* Implementation: The submit\_proposal instruction in Instructions.rs enforces multisig approval via validate\_multisig, ensuring only authorized submissions proceed.

**Proposal Value Constraints**

* Description: Proposals must adhere to predefined constraints (e.g., tax rate 1%–10%, staking tiers ascending from 20,000 tokens to ≤10% of supply), validated during execution.
* Purpose: Prevents destabilizing or exploitative changes by enforcing safe parameter ranges, protecting the contract’s integrity.
* Implementation: The execute\_proposal instruction in Instructions.rs includes checks like require!(new\_tax\_rate >= 100 && new\_tax\_rate <= 1000, ErrorCode::InvalidTaxRate) for each proposal type.

**Revoting Mechanism**

* Description: Stakers can change their votes during the 14-day voting period, with vote counts dynamically updated to reflect their current voting power.
* Purpose: Enhances flexibility and responsiveness to new information, ensuring accurate representation of staker sentiment.
* Implementation: The vote instruction in Instructions.rs adjusts previous votes (if vote\_record.voted { ... }) before applying the new vote, maintaining tally integrity.

8. Security and Transparency

* **Reentrancy Protection:**
  + ReentrancyGuard ensures atomic operations (is\_processing flag).
  + Source: ReentrancyGuard struct in Instructions.rs.
* **Timelocks:**
  + Multisig changes: 24 hours.
  + Pause/Resume: 24 hours.
  + Whitelist changes: 72 hours.
  + Large withdrawals: 48 hours.
  + Migration toggle: 7 days.
  + Source: Respective instructions with TimeLockNotMet, WhitelistDelayNotMet errors.
* **Transparency:**
  + Events: Log all actions (e.g., ProposalSubmittedEvent, VoteEvent, MultisigUpdatedEvent).
  + Query Functions:
    - query\_state: Returns ContractState (includes proposal\_count, total\_voting\_power).
    - query\_pending\_rewards: Provides staker reward data.
  + Source: Event emissions, query\_state, query\_pending\_rewards instructions.
* **Error Handling:**
  + Comprehensive errors (e.g., InsufficientTierForVoting, ProposalExpired, InvalidProposalType).
  + Source: Errors.rs.

Summary Table

|  |  |
| --- | --- |
| **Aspect** | **Details** |
| Proposal Submission | Multisig (2–5 owners, ≥2 approvals), ≤200-char description, ≤7 values |
| Voting Eligibility | Tier 1+ (≥100K tokens, ≥14 days), Tier 0 excluded |
| Voting Power | Tier 1: 1, Tier 2: 4, Tier 3: 20; Multipliers: ~1.995x (90d), 1.5x (60d) |
| Approval Criteria | 30% quorum, 51% approval, 20% voting power threshold |
| Proposal Types | 12 types (tax rate, staking tiers, allocations, whitelist, etc.) |
| Timing | 14-day voting, 3-day execution delay |
| Multisig Actions | Airdrops, pause/resume, vesting, withdrawals, whitelist, DEX, migration |
| Timelocks | 24h (multisig, pause), 72h (whitelist), 48h (large withdrawals), 7d (migration) |
| Security | Reentrancy guard, timelocks, event logging, query functions |

# Security

1. Reentrancy Protection

* **Mechanism:** A scope-based ReentrancyGuard prevents reentrant calls by setting state.is\_processing to true during critical operations and resetting it upon completion. Attempts to re-enter trigger a ReentrancyGuardTriggered error.
* **Purpose:** Mitigates risks of malicious contracts draining funds or manipulating state via recursive calls.
* **Implementation:** Applied to all major instructions (e.g., transfer, stake, unstake, claim\_rewards) using Rust’s ownership model to ensure the guard is dropped only after completion.
* **Source:** Instructions.rs (ReentrancyGuard struct and usage, e.g., let \_guard = ReentrancyGuard::new(&mut ctx.accounts.state);).

2. Arithmetic Safety

* **Mechanism:** Checked arithmetic operations (checked\_add, checked\_sub, checked\_mul, checked\_div) prevent overflows and underflows. Ceiling division ensures precision in tax and reward calculations (e.g., ((amount \* rate \* 10^7) + 9999999) / (10000 \* 10^7)).
* **Purpose:** Ensures calculations for token amounts, voting power, and allocations remain accurate and secure within u64 limits.
* **Implementation:** Used throughout token transfers, tax calculations, staking rewards, and voting power computations, with ArithmeticOverflow errors for invalid results.
* **Source:** Instructions.rs (e.g., transfer, get\_pending\_rewards, execute\_proposal).

3. Access Controls

* **Multisig Controls:**
  + Administrative actions (e.g., submit\_proposal, airdrop, update\_team\_vesting) require multisig approval with 2–5 owners and a threshold ≥2.
  + Unique owner validation prevents duplication, and signer checks ensure authorization.
  + Source: Instructions.rs (validate\_multisig function); Lib.rs (Multisig struct).
* **Whitelisted Contracts:**
  + Privileged operations like whitelisted\_transfer are restricted to whitelisted contracts, verified by program ID and version hash.
  + Source: Instructions.rs (whitelisted\_transfer, require!(state.whitelisted\_contracts.iter().any(...), ErrorCode::CallerNotWhitelisted)).
* **User Permissions:**
  + Token transfers require sender ownership (UnauthorizedSender error otherwise), and staking/voting enforce tier eligibility (InsufficientTierForVoting error).
  + Source: Instructions.rs (transfer, stake, vote).

4. Timelocks and Delays

* **Purpose:** Prevents rushed or malicious actions by enforcing mandatory delays.
* **Key Timelocks:**
  + Multisig changes: 24-hour delay (initiate\_set\_multisig, confirm\_set\_multisig).
  + Pause/Resume: 24-hour delay (initiate\_pause, confirm\_pause, initiate\_resume, confirm\_resume).
  + Whitelist changes: 72-hour delay (add\_whitelisted\_contract, remove\_whitelisted\_contract, confirm\_whitelist\_change).
  + Large withdrawals (>0.5% of total supply): 48-hour delay (initiate\_withdrawal, complete\_withdrawal).
  + Migration toggle: 7-day cooldown (toggle\_migration\_active).
* **Implementation:** Enforced via Unix timestamps or slot counts, with errors like TimeLockNotMet, WhitelistDelayNotMet, and MigrationToggleCooldown.
* **Source:** Instructions.rs (e.g., confirm\_pause, require!(clock.unix\_timestamp >= pending\_pause.initiation\_time + 24 \* 3600, ErrorCode::TimeLockNotMet)).

5. Pause Mechanism

* **Functionality:** Multisig can pause the contract in emergencies, halting most operations except zero-amount transfers and admin actions, with a logged reason (≤100 characters).
* **Process:** Initiated via initiate\_pause, confirmed after 24 hours via confirm\_pause; resumed similarly with initiate\_resume and confirm\_resume.
* **Impact:** Blocks instructions like transfer, stake, and claim\_rewards when state.paused is true.
* **Source:** Instructions.rs (paused flag checks, e.g., require!(!state.paused, ErrorCode::Paused)); State.rs (ContractState.paused, pause\_reason).

6. Input Validation

* Purpose: Ensures inputs and proposal values adhere to strict constraints, preventing misuse.
* Key Validations:
  + Proposal Values: Tax rate (1%–10%), staking tiers (min 20,000 tokens, max 10% of supply), tax allocations (sum to 10,000 bp, each ≤50%).
    - Source: Instructions.rs (execute\_proposal, e.g., require!(new\_tax\_rate >= 100 && new\_tax\_rate <= 1000, ErrorCode::InvalidTaxRate)).
  + Transaction Limits: Sell/transfer amounts checked against max\_sell\_txn\_limit, daily\_sell\_limit, etc.
    - Source: Instructions.rs (transfer).
  + Vesting Claims: Caps at 20M tokens for team vesting, 500K for freelancers.
    - Source: Instructions.rs (claim\_team\_vesting, claim\_freelancer\_vesting).
* Errors: Specific errors like InvalidTaxRate, MaxSellTxnLimitExceeded, and ExceedsVestingTotal.

7. Transparency and Auditing

* **Events:** Over 40 event types (e.g., TransferEvent, StakeEvent, ProposalSubmittedEvent) log critical data like amounts, timestamps, and reasons.
  + Source: Instructions.rs (event structs and emit! calls).
* **Query Functions:** query\_state provides ContractState details, and query\_pending\_rewards calculates staker rewards, accessible to anyone.
  + Source: Instructions.rs (query\_state, query\_pending\_rewards).
* **Purpose:** Facilitates external monitoring, auditing, and integration with off-chain tools.

8. Migration and Upgradeability

* **Token Migration:** Supports locking (lock\_for\_migration), unlocking (unlock\_for\_migration), and burning (burn\_locked\_tokens) for upgrades, controlled by multisig with a 7-day toggle cooldown.
  + Source: Instructions.rs (migration instructions); State.rs (MigrationState).
* **Whitelisted Contract Versioning:** Ties whitelisted contracts to version hashes (program IDs), preventing unauthorized upgrades.
  + Source: Instructions.rs (whitelisted\_transfer, VersionMismatch error).

9. Error Handling

* **Comprehensive Errors:** Over 60 specific error codes (e.g., Paused, InsufficientFunds, InvalidProposalType) provide clear diagnostics for failures.
* **Purpose**: Enhances debugging, user feedback, and integration reliability.
* **Source:** Errors.rs (full error list); used throughout Instructions.rs.

10. Cooldown Periods for Transactions

* **Mechanism:** A 60-second cooldown (TXN\_COOLDOWN) applies between transactions, and a 24-hour cooldown (TRANSFER\_COOLDOWN) triggers for transfers/sells above TRANSACTION\_TRACKING\_THRESHOLD (0.1% of total supply).
* **Purpose:** Limits transaction frequency to prevent spamming or rapid exploitation.
* **Implementation:** Enforced in transfer via TransactionRecord timestamps, with CooldownActive error.
* **Source:** Instructions.rs (transfer, require!(clock.unix\_timestamp - record.last\_txn\_time >= TXN\_COOLDOWN, ErrorCode::CooldownActive)); Constants.rs (TXN\_COOLDOWN, TRANSFER\_COOLDOWN).

11. Progressive Taxation

* **Mechanism:** Transfers exceeding a threshold (PROGRESSIVE\_TAX\_THRESHOLD, 0.5% of total supply) incur a double tax rate, while whitelisted transfers receive a halved tax rate. This discourages large, potentially manipulative transactions and incentivizes compliance with whitelisted entities.
* **Purpose:** Adds an economic deterrent to large dumps or unauthorized large transfers, enhancing stability.
* **Implementation:** Calculated in transfer and whitelisted\_transfer instructions based on state.progressive\_tax\_threshold and whitelisting status.
* **Source:** Instructions.rs in transfer (tax\_rate = if ... amount >= state.progressive\_tax\_threshold { state.tax\_rate \* 2 }) and whitelisted\_transfer (tax\_rate = state.tax\_rate / 2); Constants.rs defines PROGRESSIVE\_TAX\_THRESHOLD.

12. Immutable PDA Seeds

* **Mechanism**: Program Derived Addresses (PDAs) use fixed seeds (e.g., b"treasury\_authority", b"staking\_authority") with bumps to ensure deterministic and secure authority accounts for token transfers and state management.
* **Purpose:** Prevents spoofing or manipulation of authority accounts by ensuring only the program can derive and sign with these addresses.
* **Implementation**: Used across instructions like transfer, stake, unstake, and claim\_rewards with seeds and bumps validated via seeds = [...] in account structs.
* **Source:** Lib.rs in account structs (e.g., #[account(seeds = [b"treasury\_authority"], bump)]); Instructions.rs in CPI calls with signer\_seeds.

13. Additional Security Measures

* **KYC for Presale:** Purchases ≥1000 USDT require KYC verification (kyc\_verified flag).
  + Source: Instructions.rs (buy\_presale, require!(usdt\_amount < 1000 || purchase.kyc\_verified, ErrorCode::KYCRequired)).
* **Batch Processing:** Reward distribution limits batches to 50 stakers (process\_daily\_rewards) to manage compute budget constraints.
  + Source: Instructions.rs (process\_daily\_rewards, require!(batch\_size <= MAX\_BATCH\_SIZE, ErrorCode::BatchSizeTooLarge)).
* **Account Validation:** PDAs secure treasury, staking, and migration accounts with seeds and bumps.
  + Source: Lib.rs (account structs with seeds attributes).

Summary Table

|  |  |
| --- | --- |
| **Security Aspect** | **Details** |
| Reentrancy Protection | ReentrancyGuard blocks recursive calls during critical operations. |
| Arithmetic Safety | Checked operations and ceiling division ensure precise, overflow-free calculations. |
| Access Controls | Multisig for admin, whitelisting with versioning, and user permission checks. |
| Timelocks | Delays for multisig (24h), pauses (24h), whitelist (72h), withdrawals (48h), migration (7d). |
| Pause Mechanism | Multisig-controlled pause with reason, halts most operations. |
| Input Validation | Strict checks on proposals, transactions, and vesting claims. |
| Transparency | Events and query functions for all actions and state data. |
| Migration | Controlled token migration with multisig toggle and cooldown. |
| Error Handling | Over 60 specific errors for clear diagnostics. |
| Cooldown Periods | 60s between transactions, 24h for large transfers/sells. |
| Progressive Taxation | Double tax for large transfers, halved for whitelisted, deterring manipulation. |
| Immutable PDA Seeds | Fixed seeds with bumps secure authority accounts against spoofing. |
| Additional Measures | KYC for presale, batch processing, and PDA validation. |

# Glossary

The glossary provides concise definitions for key terms and concepts used throughout the smart contract documentation. These terms are critical for understanding the contract’s functionality, governance, tokenomics, and security mechanisms. Entries are listed alphabetically for easy reference.

* **Airdrop:** Distribution of tokens to multiple wallet addresses, typically for free, to incentivize adoption or reward users. Managed via the airdrop instruction with multisig approval.
* **Accounts:** Data storage units on the Solana blockchain, used to hold tokens, program state, or configuration data. Examples include ContractState and Staker accounts.
* **Basis Points (bp):** A unit of measure for percentages where 1 bp equals 0.01% (100 bp = 1%). Used to define tax rates and allocations (e.g., tax\_rate = 500 bp for 5%).
* **DEX (Decentralized Exchange):** A platform enabling peer-to-peer token trading without intermediaries. Identified in the contract via dex\_programs for sell detection.
* **Governance:** The decentralized decision-making system allowing stakers to propose and vote on contract parameter changes, such as tax rates or staking tiers, via the submit\_proposal and vote instructions.
* **Instructions:** Functions or operations within the smart contract that users or administrators can invoke, such as transfer, stake, or execute\_proposal.
* **KYC (Know Your Customer):** A verification process requiring identity confirmation for presale purchases of $1,000 USDT or more, enforced in the buy\_presale instruction.
* **Liquidity Pool:** A reserve of tokens locked in the contract to facilitate trading on DEXes, funded via tax allocations (liquidity\_pool\_alloc) and tracked in the Treasury struct.
* **Multisig (Multi-Signature)**: A security mechanism requiring multiple private key signatures (e.g., 2 out of 5) for administrative actions like pausing or airdrops, configured in the Multisig struct.
* **PDAs (Program Derived Addresses)**: Unique Solana addresses derived from a program ID and seeds (e.g., b"treasury\_authority"), used to secure operations like treasury transfers.
* **Reentrancy:** A potential attack where a malicious contract re-calls the original contract before completion, mitigated by the ReentrancyGuard mechanism.
* **Security:** Protective measures within the contract, including reentrancy guards, timelocks, and input validation, to safeguard against exploits and ensure operational integrity.
* **Smart Contract**: A self-executing program on the Solana blockchain encoding the VeraLux token’s rules for tokenomics, governance, and security.
* **Solana:** A high-performance blockchain platform hosting this smart contract, leveraging its fast transaction processing and low costs.
* **Staking:** The act of locking tokens in the contract to earn rewards or gain voting power, managed via stake and tracked in the Staker struct.
* **State:** The current data stored in the smart contract, such as configuration (ContractState) or user balances (Staker), modifiable through transactions.
* **Tokenomics:** The economic model governing the VeraLux token, including supply (1B tokens), distribution (e.g., 25% presale), and incentives (e.g., staking rewards).
* **Treasury:** A pool of tokens (initially 660M) allocated for various purposes like staking rewards or marketing, managed in the Treasury struct and adjustable via transfer\_between\_pools.
* **Unix Timestamp:** A time-tracking system counting seconds since January 1, 1970, used for vesting schedules (e.g., launch\_timestamp = 1746057600) and timelocks.
* **Vesting:** A mechanism where tokens are locked and released gradually over time, such as 10% weekly for presale buyers, managed via PresaleVesting or TeamVesting structs.

# Appendices

The appendices provide supplementary details to support the main document, including constants, calculation examples, and error codes. These enhance transparency and assist developers, auditors, and users in understanding the contract’s mechanics.

## Appendix A: Constants

This section lists key constants defined in Constants.rs, critical for the contract’s configuration and operation, along with their values and purposes.

TOTAL\_SUPPLY:

* + Value: 1,000,000,000 \* 10^9 (1 trillion units)
  + Description: The total VeraLux token supply, fixed at deployment.

TOKEN\_DECIMALS:

* + Value: 9
  + Description: Number of decimal places for token precision (e.g., 1 token = 1,000,000,000 units).

PRESALE\_SUPPLY:

* + Value: 250,000,000 \* 10^9 (250 billion units)
  + Description: Tokens allocated for presale, sold at 0.0016 USDT each.

PRESALE\_PRICE\_PER\_TOKEN:

* + Value: 0.0016 USDT
  + Description: Price per token during presale, used in buy\_presale.

PRESALE\_MAX\_PER\_WALLET:

* + Value: 2,000,000 \* 10^9 (2 billion units)
  + Description: Maximum tokens a wallet can buy in presale.

LAUNCH\_TIMESTAMP:

* + Value: 1746057600 (May 1, 2025)
  + Description: Unix timestamp marking the vesting start for presale tokens.

INITIAL\_TAX\_RATE:

* + Value: 500 (5%)
  + Description: Default tax rate on transfers, adjustable via governance.

STAKING\_POOL\_PCT:

* + Value: 30%
  + Description: Percentage of treasury allocated to staking rewards (198M tokens).

AIRDROP\_POOL\_PCT:

* + Value: 8%
  + Description: Percentage of treasury for airdrops (52.8M tokens).

GOVERNANCE\_RESERVE\_PCT:

* + Value: 16%
  + Description: Percentage of treasury for governance costs (105.6M tokens).

MARKETING\_FUND\_PCT:

* + Value: 18%
  + Description: Percentage of treasury for marketing (118.8M tokens).

EMERGENCY\_FUND\_PCT:

* + Value: 5%
  + Description: Percentage of treasury for emergencies (33M tokens).

TEAM\_POOL\_PCT:

* + Value: 18%
  + Description: Percentage of treasury for team allocation (118.8M tokens).

MAX\_DEXES:

* + Value: 10
  + Description: Maximum number of DEX program IDs in dex\_programs.

MAX\_WHITELISTED:

* + Value: 20
  + Description: Maximum number of whitelisted contracts.

MAX\_ALLOWED\_DESTINATIONS:

* + Value: 10
  + Description: Maximum transfer destinations for whitelisted transfers.

MAX\_PAUSE\_REASON\_LEN:

* + Value: 100
  + Description: Maximum length (bytes) for pause reason strings.

MAX\_DESCRIPTION\_LEN:

* + Value: 200
  + Description: Maximum length (bytes) for proposal descriptions.

MAX\_PROPOSAL\_VALUES:

* + Value: 7
  + Description: Maximum number of values in a governance proposal.

VOTING\_QUORUM:

* + Value: 30%
  + Description: Minimum percentage of total voting power required for a valid vote.

VOTING\_APPROVAL:

* + Value: 51%
  + Description: Minimum percentage of cast votes needed for proposal approval.

VOTING\_THRESHOLD:

* + Value: 20%
  + Description: Minimum percentage of total voting power in favor for approval.

TXN\_COOLDOWN:

* + Value: 60 seconds
  + Description: Cooldown period between consecutive transactions.

TRANSFER\_COOLDOWN:

* + Value: 24 hours
  + Description: Cooldown for transfers exceeding 0.1% of total supply.

TRANSACTION\_TRACKING\_THRESHOLD:

* + Value: 0.1% of total supply (1M tokens)
  + Description: Threshold triggering the 24-hour transfer cooldown.

PROGRESSIVE\_TAX\_THRESHOLD:

* + Value: 0.5% of total supply (5M tokens)
  + Description: Threshold for applying a doubled tax rate on transfers.

WITHDRAWAL\_THRESHOLD:

* + Value: 0.5% of total supply (5M tokens)
  + Description: Threshold triggering a 48-hour withdrawal delay.

## Appendix B: Calculation Examples

These examples illustrate key computations within the smart contract, aiding developers and auditors in verifying its behavior.

**Tax Calculation Example**

Scenario: A user transfers 10,000 tokens (10,000 \* 10^9 = 10 trillion units) with an initial tax rate of 5% (500 bp).

* **Tax Amount:**
  + Formula: tax = (amount \* tax\_rate + 9999) / 10000 (ceiling division for precision)
  + Calculation: (10,000,000,000,000 \* 500 + 9999) / 10000
  + Step-by-step:
    - 10,000,000,000,000 \* 500 = 5,000,000,000,000,000
    - 5,000,000,000,000,000 + 9999 = 5,000,000,000,009,999
    - 5,000,000,000,009,999 / 10000 = 500,000,000,000 (500 tokens)
  + Result: Tax = 500,000,000,000 units (500 tokens)
* **Net Transfer:**
  + Formula: net\_amount = amount - tax
  + Calculation: 10,000,000,000,000 - 500,000,000,000 = 9,500,000,000,000 (9,500 tokens)
* **Tax Allocation:**
  + Allocations (bp of tax):
    - Burn: 20% (2000 bp)
    - Treasury: 20% (2000 bp)
    - Liquidity Pool: 24% (2400 bp)
    - LP Incentives: 6% (600 bp)
    - Charity: 20% (2000 bp)
    - Team: 10% (1000 bp)
  + Per allocation: (tax \* allocation\_bp) / 10000
    - Burn: (500,000,000,000 \* 2000) / 10000 = 100,000,000,000 (100 tokens)
    - Treasury: (500,000,000,000 \* 2000) / 10000 = 100,000,000,000 (100 tokens)
    - Liquidity Pool: (500,000,000,000 \* 2400) / 10000 = 120,000,000,000 (120 tokens)
    - LP Incentives: (500,000,000,000 \* 600) / 10000 = 30,000,000,000 (30 tokens)
    - Charity: (500,000,000,000 \* 2000) / 10000 = 100,000,000,000 (100 tokens)
    - Team: (500,000,000,000 \* 1000) / 10000 = 50,000,000,000 (50 tokens)

Source: transfer instruction in Instructions.rs.

**Voting Power Calculation Example**

Scenario: A user stakes 600,000 tokens (600M units) for 70 days.

* **Tier Determination:**
  + Tiers:
    - Tier 0: 20K tokens, 7 days
    - Tier 1: 100K tokens, 14 days
    - Tier 2: 500K tokens, 30 days
    - Tier 3: 5M tokens, 30 days
  + Result: Qualifies for Tier 2 (500K tokens, 30 days met)
* **Base Voting Power:**
  + Tier 2: 4 votes
* **Time-Based Multiplier:**
  + Rules:
    - <60 days: 1.0x
    - 60–89 days: 1.5x
    - ≥90 days: ~1.995x
  + 70 days: 1.5x (1500/1000)
* **Final Voting Power:**
  + Formula: (base\_power \* multiplier + 999) / 1000
  + Calculation: (4 \* 1500 + 999) / 1000 = (6000 + 999) / 1000 = 6999 / 1000 = 6 (integer division, floored)
  + Result: 6 votes

Source: calculate\_voting\_power in Instructions.rs.

## Appendix C: Error Codes

This section lists all error codes defined in Errors.rs, providing a reference for diagnosing failures during contract execution.

* Paused: "The contract is paused" - Most operations blocked when state.paused is true.
* InsufficientFunds: "Insufficient funds" - Not enough tokens in an account or pool.
* InvalidProposalType: "Invalid proposal type" - Unrecognized proposal\_type in execute\_proposal.
* ReentrancyGuardTriggered: "Operation already in progress" - Recursive call detected.
* TimeLockNotMet: "Time lock requirement not met" - Delay period not elapsed (e.g., 24h for pause).
* WhitelistDelayNotMet: "Whitelist delay not met" - 72-hour delay for whitelist changes not met.
* MigrationToggleCooldown: "Migration toggle cooldown active" - Less than 7 days since last toggle.
* BatchSizeTooLarge: "Batch size too large" - Exceeds 50 in process\_daily\_rewards.
* KYCRequired: "KYC verification required" - Presale purchase ≥ $1000 USDT without KYC.
* PresaleSupplyExceeded: "Presale supply exceeded" - Exceeds PRESALE\_SUPPLY.
* PresaleMaxPerWalletExceeded: "Presale maximum per wallet exceeded" - Exceeds PRESALE\_MAX\_PER\_WALLET.
* InvalidTaxRate: "Invalid tax rate" - Outside 1%–10% range in proposal type 0.
* InvalidStakingTiers: "Invalid staking tiers" - Not ascending or out of bounds in proposal type 1.
* InvalidTaxAllocationTotal: "Invalid tax allocation total" - Sum ≠ 100% in proposal type 2.
* InvalidReductionThresholds: "Invalid reduction thresholds" - Not ascending or outside 10%–90% in proposal type 3.
* InvalidReductionFactor: "Invalid reduction factor" - Outside 10%–200% in proposal type 3.
* InvalidSellLimit: "Invalid sell limit" - Outside 0.1%–2% of supply in proposal types 5, 6.
* InvalidTransferLimit: "Invalid transfer limit" - Outside 0.1%–2% of supply in proposal types 7, 8.
* InvalidTaxThreshold: "Invalid tax threshold" - Outside 0.1%–2% of supply in proposal type 9.
* InvalidStakingReward: "Invalid staking reward" - Outside 100–1M tokens in proposal type 10.
* ProposalExpired: "Proposal has expired" - Voting past end\_time.
* VotingPeriodNotEnded: "Voting period has not ended" - Execution before end\_time.
* NoticePeriodNotMet: "Notice period not met" - Execution before execution\_time.
* ProposalAlreadyExecuted: "Proposal already executed" - Status ≠ 0 in execute\_proposal.
* WithdrawalDelayNotMet: "Withdrawal delay not met" - Less than 48h for large withdrawals.
* LockPeriodNotMet: "Lock period not met" - Staking duration insufficient (e.g., <7 days).
* InsufficientStakedAmount: "Insufficient staked amount" - Unstake amount exceeds staked.
* NoLockedTokens: "No locked tokens" - Migration action with zero locked amount.
* AmountTooSmallAfterTax: "Amount too small after tax" - Net transfer ≤ 0.
* InsufficientTierForVoting: "Insufficient tier for voting" - Tier < 1 in vote.
* InvalidImmediateAmount: "Invalid immediate amount" - Exceeds total in update\_team\_vesting.
* InvalidTeamMember: "Invalid team member" - Mismatch in cancel\_team\_vesting.
* VestingCanceled: "Vesting canceled" - Claim attempt on canceled vesting.
* ClaimCooldownNotMet: "Claim cooldown not met" - Less than 3 days since last freelancer claim.
* ExceedsVestingTotal: "Exceeds vesting total" - Release exceeds total in freelancer vesting.
* TooManyRecipients: "Too many recipients" - Exceeds 100 in airdrop or incentives.
* InvalidRecipientAccounts: "Invalid recipient accounts" - Mismatch in recipient list.
* InvalidContract: "Invalid contract" - Non-executable in add\_whitelisted\_contract.
* TooManyWhitelistedContracts: "Too many whitelisted contracts" - Exceeds MAX\_WHITELISTED.
* ContractNotWhitelisted: "Contract not whitelisted" - Removal attempt on non-whitelisted contract.
* VersionMismatch: "Version mismatch" - Whitelisted contract hash mismatch.
* MaxSellTxnLimitExceeded: "Maximum sell transaction limit exceeded" - Exceeds max\_sell\_txn\_limit.
* DailySellLimitExceeded: "Daily sell limit exceeded" - Exceeds daily\_sell\_limit.
* MaxTransferLimitExceeded: "Maximum transfer limit exceeded" - Exceeds max\_transfer\_limit.
* DailyTransferLimitExceeded: "Daily transfer limit exceeded" - Exceeds daily\_transfer\_limit.
* CooldownActive: "Transfer cooldown active" - Within TXN\_COOLDOWN or TRANSFER\_COOLDOWN.
* VestingNotStarted: "Vesting period not started" - Claim before launch\_timestamp.
* NoRewards: "No rewards available" - No claimable rewards.
* ArithmeticOverflow: "Arithmetic overflow occurred" - Calculation exceeds u64 limits.
* VectorOverflow: "Vector overflow" - Exceeds vector size limits (e.g., MAX\_DEXES).
* InvalidAccounts: "Invalid accounts provided" - Mismatch in initialize.
* InvalidDexProgram: "Invalid DEX program ID" - Non-executable in update\_dex\_programs.