BSReward Contract: Decentralized Reward Distribution System

BSReward is a smart contract designed to distribute BUSO tokens to users based on their weekly performance in individual and club leaderboards. The system utilizes Merkle proofs for off-chain data validation, DAO-controlled parameters, and anti-abuse mechanisms.

Core Features

1. **Weekly Reward Cycles**
    a. Operates on weekly intervals starting from `s_startTime`.
    b. Leaderboard data is stored off-chain in Merkle trees; users claim rewards with proofs.
2. **Dynamic Reward Tiers**
    a. DAO selects reward tiers from a predefined table (`rewardLevels`), ranging from 1 token/week (Level 0) to 512,695 tokens/week (Level 14).
3. **DAO-Controlled Parameters**
    a. Adjustable parameters: individual/club reward splits, max eligible users/clubs, banned lists, etc.
4. **Security & Transparency**
    a. On-chain bans for abusive users/clubs.
    b. Reward calculations are fully on-chain.
    c. Weekly Merkle roots and historical data stored on-chain.

Reward Distribution Logic

Individual Rewards

- **Formula**:
  `Individual Reward = Reward Piece × (Total Users − Rank + 1)`
    - *Reward Piece*: Individual reward pool divided by the sum of ranks (1 to total users).
    - *Example*: For 100 users, 1st place → 100x, 2nd → 99x, ..., 100th → 1x.

Club Rewards

1. **Shared Distribution**:
   Club rewards are split equally among members.
   `Member Reward = Club Reward / Member Count`
2. **Performance-Based**:
   Members earn rewards based on intra-club rankings.
   `Member Reward = Reward Piece × (Total Members − Member Rank + 1)`

Key Components

1. Data Structures

- **`Snapshot`**: Contains user performance data (rank, score, club details).
- **`WeekData`**: Stores weekly reward pools, Merkle root hash, and statistics.
- **`RewardConfig`**: DAO-controlled parameters

2. Critical Functions

- **`addWeekData()`**: Initializes a new week with Merkle root (called by signer).
- **`useSnapshot()`**: Claims rewards using Merkle-validated snapshots.
- **`calculateRewardPiece()`**: Computes reward allocation (core mathematical logic).

3. Security Mechanisms

- **Reentrancy Guards**: Prevents reentrancy attacks.
- **Nonce & Hash Checks**: Ensures valid weekly and snapshot data.
- **Used Snapshot Tracking**: Prevents duplicate claims.

Usage Scenario

1. **Setup**:
    a. Platform generates weekly leaderboards and Merkle trees off-chain.
    b. Signer initializes the week via `addWeekData`.
2. **Claim Process**:
    a. Users call `useSnapshot` with valid Merkle proofs to claim rewards.
    b. Rewards are transferred instantly to their wallets.

Example User Snapshot Data

Individual Snapshot

```
{
  "id": 12345,
  "weekIndex": 42,
  "weekNonce": 987654321,
  "user": "0xAbC...123",
  "individual": {
    "score": 8500,
    "rank": 5
  },
  "club": {
    "id": 0,  // No club participation
    "score": 0,
    "rank": 0,
    "distributionMethod": 0,
    "memberCount": 0,
    "memberRank": 0
  }
}
```

Sample Snapshot with Club Reward

```
{
  "id": 67890,
  "weekIndex": 42,
  "weekNonce": 987654321,
  "user": "0xXyZ...456",
  "individual": {
    "score": 7200,
    "rank": 15
  },
  "club": {
    "id": 789,
    "score": 15000,
    "rank": 3,
    "distributionMethod": 1,
    "memberCount": 50,
    "memberRank": 2
  }
}
```

DAO-Controlled Variables

1. `RewardConfig` (DAO-Controlled Parameters)

The `RewardConfig` struct contains critical parameters governing the reward distribution system. These values are **exclusively updatable by the DAO** and affect future reward cycles:

- **`rewardLevel`**: Determines the total weekly reward pool (pulls values from the `rewardLevels` array).
- **`rewardIndividualMax`**: Maximum number of users eligible for individual rewards (e.g., top 100 users).
- **`rewardClubMax`**: Maximum number of clubs eligible for club rewards (e.g., top 50 clubs).
- **`rewardToIndividualPercent`**: Percentage of rewards allocated to individual users (e.g., 25% → 75% to clubs).
- **`maxClubMembers`**: Maximum allowed members per club for reward eligibility (prevents oversized clubs).
- **`allowClaimsForOthers`**: If `false`, only the snapshot owner can claim rewards.

2. Banned Users (`s_bannedUsers`)

- **Purpose**: Block specific addresses from claiming **any rewards** (individual or club).
- **DAO Control**: Updated via `setBannedUser(address user, bool isBanned)`.
- **Effect**:
    - A banned user **cannot claim rewards.**

3. Banned Clubs (`s_bannedClubs`)

- **Purpose**: Block specific clubs from distributing rewards to their members.
- **DAO Control**: Updated via `setBannedClub(uint64 clubId, bool isBanned)`.
- **Effect**:
    - Members of a banned club **cannot claim club rewards.**
    - Users in banned clubs **can still claim individual rewards** if eligible.