GENESIScoin Core Token

What it is
- ERC-20 on BSC. Fixed supply. No mint. Trading gated via enableTrading().

Ownership
- Two-step: transferOwnership → acceptOwnership. Events emitted.

AMM & trading
- AMM registry (setAMM), router/pair refs. Trading toggled once. Invalid AMM guarded.

Fees (modular)
- Auto-burn, Liquidity, Rewards.
- Wallet fee buckets: Dev, Expansion, Team.
- Each has <bucket>AddressSetup, <bucket>FeesSetup, and emits updates.

Exemptions
- excludeFromFees, excludeFromDividends, excludeFromTradingRestriction.

Dividend tracker (built-in)
- Read: balances, holders count, total distributed.
- Payout: claim(), process(gas) with claimWait and gasForProcessing controls.

Swap & liquidity
- swapThresholdRatio with updateSwapThreshold.
- addLiquidityFromLeftoverTokens() with LiquidityAdded events.

Limits & safety
- Trading restriction list.
- burn, burnFrom. Token recovery for foreign tokens.
- Robust error set for invalid states and SafeERC20 failures.

Operator quick hits
- One-time trading enable.
- Keep fee addresses set before enable.
- Monitor dividend gas and claim wait for throughput.

GENESIScoin — Pre-Sale Contract

Purpose
- Handles the entire GENc Pre-Sale: stage logic, token purchases, bonuses, B100D whitelist, daily payouts, final bonus, and 24-hour BNB distribution.

Roles and Addresses (fixed in constructor)
- Token: GENc
- OracleBot: controls timing, stages, whitelist, B100D, and Final Bonus.
- B100DBot: executes daily 100-day rewards.
- FinalBonusBot: triggers the one-time final bonus after the loyalty cycle.
- Owner: fixed; no transferOwnership.
- All operational addresses immutable post-deployment.

1. Stage System
- 5 stages with own price, bonus %, duration, and supply.
- Active stage: currentStageId().
- Auto-close:
    - ≤ 4 000 GENc left → auto close.
    - 4 000–5 000 GENc → manual close window (“1k window”).
    - Sold-out → StageSoldOut.

2. Token Purchase
- buyTokens() payable. Converts BNB → GENc + stage bonus.
- Emits TokensPurchased(buyer, tokens, bonus, totalHoldingAfter).
- Enforces: MAX_PURCHASE_GENc, MAX_HOLD_PER_WALLET, MIN_GENc_BUY.

3. BNB Price and Oracle
- bnbPrice() + updateBNBPrice() by Oracle.
- lastPriceUpdateTime() for transparency.

4. Whitelist B100D Program
- Oracle opens/closes: WhitelistOpened / WhitelistClosed.
- registerToB100DWhitelist() payable saves initialHold.
- ejectFromWhitelist(address) removes.
- After WL close → B100D starts.

5. B100D Cycle (100 Days)
- triggerB100DStart().
- triggerDailyB100DPayout() batched.
- Events: B100DBatchProcessed, B100DDayCompleted.
- Progress: b100dCurrentDay().

6. Final Bonus
- triggerFinalBonusDistribution() batched.
- Events: FinalBonusStarted, FinalBonusBatchProcessed, FinalBonusDistributed, FinalBonusCompleted.
- Guard: finalBonusClaimed(addr).

7. Token Pools
- preSaleGENcPool — sale supply.
- preSaleBonusPoolGENc — stage bonuses.
- b100dPoolGENc — B100D + Final Bonus.

8. Dividend Link
- Timeout/forceClose → 35% of stage leftovers → existing Dividend Pool (GENc).
- Sold-out → no leftovers.

9. BNB Distribution
- distributeFunds() daily by Oracle.
- Routes BNB to wallets (LP, marketing, team, dev, community, expansion, audit).
- Does not fund dividend pool.

10. Safety & Batch Control
- ReentrancyGuard on payouts/transfers.
- Batch start: 25 addrs (adjustable).
- Validations: active stage, supply ≥ buy+bonus, wallet limits, WL/B100D windows.

11. Events to Watch
- TokensPurchased, StageSoldOut, StageTimeout, ForceStageCloseByOwner.
- WhitelistOpened, WhitelistClosed, B100DRegistered, B100DDayCompleted.
- FinalBonusStarted, FinalBonusCompleted, FundsDistributed.

In short
- Event-driven, Oracle-timed, strict caps, daily BNB routing, full B100D autonomy.

GENESIScoin — PublicSaleGENc

Purpose
- Manages the public sale phase: start signal, daily LP bonuses, dividends, and airdrops. No buy logic or BNB distribution on-chain.

1. Core Logic
- 4 phases by day count since start.
- Phase thresholds (days): 100 / 200 / 300.
- Bonus per phase: 100k / 130k / 200k / 300k GENc.
- THRESHOLD and MAX_USERS defined but inactive.

2. Roles and Addresses (fixed in constructor)
- Token: GENc
- Oracle: startPublicSale() once.
- EligibleBot (eliB): daily LP bonuses.
- DividendBot (divT, divV): dividend tracking and payout cycles.
- AirdropBot: airdrop batches.
- Owner fixed; no transferOwnership.
- All operational addresses immutable.

3. Token Pools
- publicSaleBonusPoolGENc — eliB.
- dividendPoolGENc — divV.
- airdropPoolGENc — airdrops.
- fund<PoolName>(amount) via transferFrom.

4. Start and Timing
- Oracle calls startPublicSale() → sets publicSaleStartTimestamp.
- Day/phase helpers:
  - getCurrentDay() = ((now - start)/86400)+1
  - getPhase(day) → 1–4
  - getBonusAmountForPhase(phase) → amount.

5. Daily Bonus (eliB)
- processDailyBonus(address buyer, uint256 day).
- Requires: sale started, valid day, not claimed.
- Transfers from publicSaleBonusPoolGENc.
- Updates hasClaimed[day][buyer], dailyBonusClaimed[day].
- Emits DailyBonusGranted(recipient, day).
- onlyEligibleBot, nonReentrant.

6. Dividends (divV)
- distributeDividendsAdjusted(address[] recipients, uint256[] payouts).
- Rules: equal lengths, ≤100 recipients.
- Transfers GENc and decrements pool.
- Emits DividendPaid(user, timestamp).
- onlyDividendBot, nonReentrant.

7. Airdrops
- airdropByList(address[] recipients, uint256[] amounts).
- Equal lengths, ≤100 per batch, pool sufficient.
- Emits AirdropExecuted(recipient, timestamp).
- onlyAirdropBot, nonReentrant.

8. Security and Integrity
- ReentrancyGuard everywhere.
- Fixed roles; no replacement.
- No owner bypass. Full transparency via events and pool views.

Summary
- Fixed-role, non-custodial distributor. All GENc bonuses, dividends, and airdrops run on on-chain time and events.

GENESIScoin — Pre-Sale BOTs

1. Oracle Bot
   Purpose
- Controls Pre-Sale timing and transitions: stages, WL B100D, daily B100D payouts, Final Bonus, daily BNB distribution.

Inputs
- System clock.
- On-chain: currentStageId, remaining tokens, WL timestamps, B100D day, Final Bonus flags, BNB price.

Monitored events
- StageSoldOut, StageTimeout, WhitelistOpened/Closed, B100DBatchProcessed, B100DDayCompleted, FinalBonus*, FundsDistributed.

Actions (on-chain)
- closeStageByOracle().
- triggerWhitelistOpen(), triggerWhitelistClose().
- triggerB100DStart(), triggerDailyB100DPayout().
- triggerFinalBonusDistribution().
- updateBNBPrice().
- distributeFunds().

Timing rules
- ≤ 4 000 GENc left → auto-close.
- 4k–5k → owner may forceClose.
- 35% leftovers at close → Dividend Pool (GENc).
- WL opens at Stage 5; closes 24 h after Pre-Sale end.
- B100D starts 24 h after WL close.
- Public Sale starts 48 h after WL close.
- BNB distribution every 24 h.

Safety
- State checks before actions.
- Idempotent triggers.
- Gas cap ~5M on batch calls.
- Logs with timestamp, stageId, batch index.

2. B100D Bot
   Purpose
- Automates 100-day rewards for all WL addresses.

Inputs

- WL list with initialHold.
- b100dCurrentDay().

Monitored events
- WhitelistClosed, B100DBatchProcessed, B100DDayCompleted.

Actions (on-chain)
- triggerB100DStart() (24 h after WL close).
- triggerDailyB100DPayout() in batches.

Batch system
- Start 25 addrs/tx (expandable).
- Persists lastProcessedIndex per day.
- End-of-day on full coverage → B100DDayCompleted.

Controls
- Skips wallets below required hold.
- Idempotent resume; retry on revert; gas ~5M.

Telemetry
- Day number, payout count, end index, total value.

3. Final Bonus Bot
   Purpose
- One-time final bonus after 100 B100D days for valid holders.

Inputs
- B100D completion flag.
- finalBonusClaimed(addr), totalInitialHoldForFinalBonus.

Monitored events
- B100DDayCompleted (day 100), FinalBonusStarted/BatchProcessed/Distributed/Completed.

Actions (on-chain)
- triggerFinalBonusDistribution(); then batch until done.

Batch
- 50 addrs/tx, ~6M gas. Skips already claimed. Stops at Completed.

Controls
- Sum validation. No double send. Resume from last index.

Telemetry
- Batches, duration, total GENc distributed.

4. Tracking Layer (divT & airT) — starts at Pre-Sale start
   Purpose
- Build and maintain the holder database and eligibility sets from the beginning of Pre-Sale, then carry them into Public Sale.

Tasks
- divT: listens to all Transfers; maintains list of holders ≥ MIN_HOLD; prepares snapshots for future dividend cycles.
- airT: tracks new buyers and activity; maintains airdrop-eligible addresses; exports lists for Airdrop Console.
- Persistence: `div_tracked.json`, `trackedHolders.json`, `lastProcessedBlock.json`.

Summary
- Oracle sets tempo. B100D bot handles daily rhythm. Final Bonus bot closes the program. divT/airT track from Pre-Sale day one. All autonomous and verifiable on-chain.

GENESIScoin — Public Sale BOTs

1. Oracle (PublicSale Start)
   Purpose
- Controls the transition from Pre-Sale to Public Sale — the only direct on-chain call by Oracle.

Action
- startPublicSale() → sets publicSaleStartTimestamp in PublicSaleGENc.

Inputs
- WhitelistClosed (from Pre-Sale).
- 48-hour delay after WL close.

Outputs
- Public Sale active; other bots sync to timestamp.
- Event: PublicSaleStarted(timestamp).

2. EligibleBot (eliB)
   Purpose
- Automates daily LP-based bonuses for new buyers.

Logic

- Reads LP Transfer events.
- Validates minimum buy per phase.
- One bonus per wallet per day.
- Phases 1–3: rollover unused bonuses.
- Phase 4: no cap.
- Resets daily at midnight.

Contract Interaction

- processDailyBonus(address buyer, uint256 day).

Pool / Events / Batch

- Pool: publicSaleBonusPoolGENc.
- Event: DailyBonusGranted(recipient, day).
- Batch: dynamic (hundreds/day).

3. DividendVerifier (divV)

Purpose

- Runs randomized GENc dividend cycles.

Cooldown / Duration

- 240 h after publicSaleStartTimestamp.
- 21–28 days per cycle (random).

Snapshot Logic

- Capture holders ≥ MIN_HOLD.
- Record initialHold and entryCycle.
- Auto-update initialHold at ≥151% balance.

Contract Interaction

- distributeDividendsAdjusted(address[], uint256[]).

Progress / Batch / Events

- 2% base; +1% every 2 cycles up to 10%.
- ≤100 addrs/tx, gas ≈ 6M.
- Event: DividendPaid(user, timestamp).
- Autonomy: self-loop to next cycle.

4. DividendTracker (divT) — continuity from Pre-Sale

Purpose

- Off-chain observer for dividend eligibility that has been building the holder database since Pre-Sale start and continues through Public Sale.

Tasks

- Track all Transfers; maintain holder list ≥ MIN_HOLD.
- Produce snapshots for divV cycles.

Data / Output

- div_tracked.json.
- Eligibility proofs per cycle.

5. AirdropTracker (airT) — continuity from Pre-Sale

Purpose

- Maintains the airdrop-eligible list from the beginning of Pre-Sale, then updates it during Public Sale.

Inputs / Tasks

- LP buys and post-sale transfers.
- Detect new buyers, prune inactive, export lists to Airdrop Console.

Files

- trackedHolders.json, lastProcessedBlock.json.

6. Airdrop Console

Purpose

- Executes GENc airdrops from airdropPoolGENc.

Modes

- Flat (fixed GENc per address).
- Proportional (by balance).

Contract Interaction

- airdropByList(address[] recipients, uint256[] amounts).

Batch / Validation / Event

- ≤100 addrs/tx, gas ≈ 6M.
- Pool ≥ batch sum; no duplicates.
- Event: AirdropExecuted(recipient, timestamp).

7. System Autonomy

Purpose

- Defines the post-start automated workflow and transition from Pre-Sale tracking to Public Sale LP monitoring.

Sequence

- Oracle triggers `startPublicSale()`.
- eliB activates and begins LP monitoring immediately.
- divV waits 240h → creates snapshot → starts first 21–28 day dividend cycle.
- divT & airT, which have tracked since Pre-Sale start,
  automatically switch after Whitelist Close to LP-pair transfer monitoring (post-sale logic).
  They continue continuous tracking of holders and qualifying addresses.
- divV distributes dividends at the end of each cycle.
- eliB grants daily LP bonuses.
- airT feeds verified address lists → Airdrop Console executes periodic airdrop batches.

8. Self-Regulation

Purpose

- Ensures stability without manual control.
- Only Oracle start is manual.
- divV auto-updates initialHold at ≥151%.
- eliB resets daily; divV loops.
- Unused bonuses and dividend tokens roll forward.

9. Fail-Safe Mechanisms
   Purpose

- Guarantees recovery and continuity.
- Bots persist last index and tx hash.
- Auto-resume after restart.
- ENV-tuned gas and batch size.
- Idempotent, state-checked calls.

10. Proof Layer

Purpose

- Verifiable transparency.
- All actions emit on-chain events (DailyBonusGranted, DividendPaid, AirdropExecuted).
- Off-chain JSONs (div_cycle.json, air_queue.json) mirror on-chain state for proof of autonomy.