

Razor Agent (\$RZAG)

The genesis your agent can verify

Version 1.0, 31 October 2026. Network: BNB Smart Chain (chainId 56). The deployed contracts supersede this document on any disagreement.

1. Abstract
2. The problem
3. System overview
4. The genesis mechanism
5. Quota economics
6. Post-open economics
7. Failure and refund semantics
8. Trust model and security posture
9. Parameter table
10. What this paper does not claim
11. Glossary

00 ABSTRACT

Razor Agent is a genesis mint whose entire rule set is fixed in three smart contracts before the sale begins, published in machine-readable form, and checkable by anyone, including software agents acting for buyers, without trusting the operator. Participation is gated by Binance Agent ID. The genesis offers 10,000 shares at a flat 1.15 USDT, with the first share of each Agent ID free for the first 3,333 minters. All genesis payments, including the 20 percent treasury cut, stay locked inside the mint contract until the pool opens. If 10,000 shares are not minted within 10 days of the start, the genesis fails by rule and every payment becomes refundable in full, claimable forever, with the treasury receiving zero. If the threshold is reached, the transaction that mints share 10,000 creates the pool in the same transaction, at exactly the genesis price, with LP custodied in a vault releasable to the treasury Safe only after a public 30-day on-chain notice. After opening, a

nine-band ladder continues to 4,444,000 shares and the deployer's administrative power is renounced. This paper specifies the mechanism, the trust model, and the known limitations.

81 THE PROBLEM

A genesis launch is a contract between strangers. Buyers send money first and receive a token whose value depends on later events: whether a pool is created, at what price, with what liquidity, under whose continuing control. In its common form, three properties make that contract hard to evaluate.

1.1 The parameters are not fixed in a checkable place. Prices, caps, deadlines, and splits are announced in posts and documents that can be edited, deleted, or contradicted. Even when a summary is published, it is prose, with the operative numbers scattered across channels. A buyer, and especially a buyer's software agent, cannot cheaply confirm that the marketed terms are the terms the code will execute.

1.2 Administrative power is discretionary and open-ended. The operator typically retains the ability to change rules, move liquidity, or extract value after the sale, with no expiry written anywhere. The buyer must price not only the mechanism but the operator's future restraint, which cannot be verified in advance, only falsified afterward.

1.3 Failure is unpriced. If a launch undersubscribes, the usual outcomes are improvised: an extension, a refund process run by hand, a partial return, or abandonment. The buyer cannot know, before paying, exactly what failure pays back, in what asset, on what schedule, and whether the operator keeps a cut of a failed raise.

Razor Agent answers these three properties in the same order. Every parameter is fixed in contract constants and republished in structured form (Section 2.4). Every administrative power is enumerated, bounded, and given an expiry that executes in code (Section 7). Failure is a defined state with defined payouts, computed from records written at payment time (Section 6). The mechanism does not ask for trust in the operator's intentions; it asks the reader to check the code paths that constrain them.

1.4 How to verify this paper. Each section ends at checkable facts: contract constants, public state views, and the parameter file. Section 8 is the complete parameter table, matching `params.json`. An agent that reads nothing else should read Section 8 and the Agent Pack guardrails, then confirm both against the contracts.

82 SYSTEM OVERVIEW

2.1 Components

The system has three contracts, one external identity dependency, one off-chain verification service, and a published data surface.

COMPONENT	ROLE
RazorToken (\$RZAG)	The token. Mintable only by the mint contract. Transfers frozen until the pool opens. Applies the trade taxes and pool-blocking rules.
RazorMint	The genesis and post-open mint. Holds all genesis funds in escrow, enforces the gate and quota, runs the deadline, opens the pool, records and pays refunds, runs the ladder and curve.
RazorLpVault	Custodies all LP tokens. Releases them only to the treasury Safe, after the sunset notice procedure of Section 5.5.
Agent ID registrar (external)	The Binance Agent ID registry at <code>0x60B9262456dE011F7d15a54CdD8c48bAaa8B022F</code> . The mint contract resolves the caller's Agent ID from it on every mint and claim.
Verification service (off-chain)	A claim-only web service (a Cloudflare Worker) that checks the three social tasks and signs quota vouchers. It can sign vouchers and nothing else: no mint, refund, fund-transfer, or contract administration capability.
Published surface	The website, <code>params.json</code> , <code>agent-pack.json</code> , and <code>llms.txt</code> : the human and machine renderings of the same parameter set.

Deployment order is token, then LP vault, then the mint contract, then token initialization pointing at the mint contract. The token constructor creates the RZAG/USDT pair; the pair holds no liquidity until the opening event of Section 3.8.

2.2 The Agent ID gate

Minting requires a Binance Agent ID. On every mint and social claim, `RazorMint` resolves the caller's identity from the registrar: the caller's registered handle is converted to the numeric ID used throughout the contract (`uint256(keccak256(handle))`), and the contract confirms the caller owns that ID. A wallet the registrar cannot resolve cannot mint, and there is no manual override, whitelist, or administrative path around the gate. All accounting is per Agent ID, not per wallet, so changing wallets resets nothing.

2.3 The verification service

Three quota points (Section 4.3) depend on off-chain facts: whether an X account follows the project account, whether it liked and quoted a specific post, and whether a Telegram account belongs to the project channel. The verification service attests exactly those facts and no others.

On a pass it signs an EIP-712 voucher that the holder submits to `claimSocial`; the contract verifies the signature against the immutable signer of Section 7.2 and applies its own one-time and binding rules, so an improperly obtained voucher still cannot be replayed, transferred, or used twice. Its failure mode is refusal: an uncompletable check issues nothing.

2.4 The machine-readable surface

The same parameter set is published four ways, from one source of truth:

- `params.json`: every locked parameter (addresses, ladder bands, quota rules, tax routes, dependencies, risk statements) in structured form for machines.
- `agent-pack.json`: the Agent Pack: addresses, function signatures, the public state views with their meanings, action recipes for minting and claiming, and the guardrails an agent must observe (verify chainId 56 and addresses, approve exactly the quoted cost, read state views before acting, respect the human's budget).
- `llms.txt`: the reading order for language-model agents: this file, then `params.json`, then the Agent Pack, then the leaderboard, then the verified contract source as the final authority.
- The website: the human rendering, whose live figures are contract reads, not stored values. When every public RPC endpoint fails, it shows the last successful read under a STALE plate with its read time and disables minting; on a first-load failure it shows a read error and no figures.

2.5 Lifecycle

The contract is deployed dormant: minting reverts until the deployer calls `startGenesis()` once, so code and parameters can be examined before any clock runs. From the start, the system is in exactly one state: genesis running, paused (Section 3.7), opened (Section 3.8), or failed (Section 6). Opening and failure are terminal; there is no extension function and no restart.

83 THE GENESIS MECHANISM

3.1 Shares and supply

The unit of sale is the share. One share pays out 50,000,000 RZAG to the buyer at purchase. Tokens are minted at purchase against payment; there is no pre-mint and no team allocation (the allocation is zero; the team's income is the fee streams of Sections 3.6 and 5.4, never minted tokens). For each paid genesis share, an additional 40,000,000 RZAG of pool inventory is minted into the contract's pending pool leg. The total sale is 4,444,000 shares (genesis plus the post-open ladder of Section 5.1). The token supply cap is 444,000,000,000,000 RZAG, enforced by a `SoldOut` revert, covering all buyer tokens and pool inventory. Derivation: buyer tokens at a complete sellout are 4,444,000 times 50,000,000 equals 222,200,000,000,000 RZAG, exactly half the cap; genesis pool inventory adds at most 400,000,000,000, and post-open pool minting (Section 5.2) draws on the remaining headroom.

A purchase is at most 10 shares per transaction (`MAX_PER_TX`). Every purchase, in genesis and after opening alike, passes the same checks: the caller resolves to an Agent ID, and the caller's cumulative minted count plus the new shares does not exceed the caller's quota (Section 4). The secondary market, not the mint contract, is the uncapped path after opening.

3.2 Start and deadline

`startGenesis()` is a one-shot deployer call. The Deadline is the start time plus 10 days, extended second for second by any pause (Section 3.7). Before the start, nothing can be minted; after the deadline, if the pool has not opened, genesis minting ends permanently and the failure state of Section 6 applies.

3.3 The free first share

The first share minted by each Agent ID during genesis is free, while a global cap lasts: at most 3,333 free shares in total (`freeSharesGiven` against a cap of 3,333), including mid-purchase (if the cap is reached inside a multi-share first purchase, that share is free and the rest are charged). A free share:

- pays out the same 50,000,000 RZAG to the buyer;
- consumes 1 of the ID's quota and counts toward the 10,000 opening threshold;
- adds no pool inventory and no USDT to the pending pool leg;
- never earns invite credit, never binds a referrer, and never sets the buyer's paid flag (Section 4.2).

The free share is genesis-only. Once the cap is exhausted, every first share costs the genesis price; an ID whose first-ever mint happens after opening pays the ladder price, with no retroactive free share. The pool-ratio invariant holds for any mix, by construction: free shares add nothing to either leg, so the ratio is set entirely by paid shares (Section 3.8).

3.4 Payment in USDT (exact)

The primary entry is `mint(shares, referrer, maxCost)`. `quote(user, shares)` returns the exact USDT cost for that buyer, accounting for the buyer's unused free share if the cap remains. The buyer approves the mint contract for the quoted amount; the contract measures its own USDT balance delta around the transfer and books the table cost only on exact receipt, reverting (`BadPayment`) if the delivered amount differs, so a short delivery reverts the whole mint and books nothing. `maxCost` is a slippage guard: the transaction reverts if the execution cost exceeds it. There is no conversion and no friction on this path.

3.5 Payment in BNB (converted, friction on the minter)

The second entry is `mintWithBnb(shares, referrer, maxCost)`, payable in BNB. `bnbQuote(user, shares)` returns the BNB amount to send: the USDT cost converted at the Chainlink BNB/USD feed (`0x0567F2323251f0Aab15c8dFb1967E4e8A7D42aeE`) plus 1 percent

headroom (`BNB_EXTRA_BPS = 100`). The contract swaps the received BNB to USDT on the WBNB pair internally. The published fee-risk rule, restated by the site at the point of sale, is:

- All conversion friction is borne by the minter, never by the pool or the treasury.
- If the swap obtains less than the USDT table cost, the transaction reverts (`SwapShort`) and no BNB is taken.
- Pool and treasury amounts are computed from the price table alone, never reduced by friction.
- Any conversion surplus (obtained minus cost) goes to the pre-open escrow or, after opening, to the treasury. Excess BNB above the quote is refunded in the same transaction.
- A BNB buyer's refund record is the swap-obtained USDT value, paid in USDT (Section 6).
- A stale or out-of-range feed reading makes `bnbQuote` revert (`StalePrice`) and closes the BNB method until the feed is healthy; USDT is unaffected.
- A zero-cost purchase (a free share) performs no transfer, consults no feed, and swaps nothing, so a feed outage cannot block it.

3.6 Escrow and the 80/20 split

Each paid genesis share at 1.15 USDT splits as follows: 0.92 USDT (80 percent) to the pool leg, 0.23 USDT (20 percent) as the treasury cut (1.15 times 0.80 and 0.20 respectively). Before opening, 100 percent of genesis payments, both legs, stay locked inside the mint contract: the pool leg as `pendingUsdt` against `pendingTokens`, the cut as `escrowedCut`. The treasury receives nothing before opening, and nothing at all if genesis fails (Section 6). The cut is therefore not a fee collected during the sale; it is a contingent claim that settles only if the pool opens.

3.7 Pause

The deployer, and only the deployer, may pause minting before opening, for a cumulative budget of 48 hours across the whole genesis. Every second paused extends the deadline by one second, so a pause can only stop the clock. Pause authority dies permanently at opening. A pause cannot move funds, change prices, or alter any record.

3.8 The opening event

The transaction that mints share 10,000 opens the pool in the same transaction; no separate opening transaction exists, and no callable function before that moment can create a pool or set its price. At opening, the contract settles what genesis accumulated:

- Pool USDT = paid shares times 0.92 USDT.
- Pool tokens = paid shares times 40,000,000 RZAG.
- Opening price = exactly the genesis price, 1.15 USDT per 50,000,000 RZAG, for any mix of free and paid shares, because both legs scale with paid shares only (derivation: 0.92 divided by 40,000,000 equals 1.15 divided by 50,000,000).
- The escrowed treasury cut is transferred to the treasury Safe inside the same transaction.

- All LP tokens are minted to `RazorLpVault`, never to a wallet and never burned at opening, and trading is enabled under the rules of Section 5.

Worked example (the standard fill in the project's test suite): 1,000 Agent IDs minting 10 shares each gives 1,000 free and 9,000 paid shares. The pool opens with 9,000 times 0.92 equals 8,280 USDT against 9,000 times 40,000,000 equals 360,000,000,000 RZAG, and the escrow release is 9,000 times 0.23 equals 2,070 USDT. Across all fills, the escrowed cut at exactly 10,000 shares ranges from about 1,533 USDT (6,667 paid shares, maximum free usage) to 2,300 USDT (no free shares used).

84 QUOTA ECONOMICS

4.1 Composition and ceiling

Each Agent ID has a quota: the maximum number of shares it may mint through the mint contract. The composition is additive and capped:

- Base: 10 per Agent ID.
- Invites: up to +20 (Section 4.2).
- Social tasks: up to +3 (Section 4.3).
- Ceiling: 33 in total (10 + 20 + 3), enforced in the mint path itself: a mint that would take the ID's cumulative count above its quota reverts (`Quota`).

Quota can grow after opening (invite credit and social claims are not phase-locked), but the ceiling binds every contract mint at all times. Quota is the only referral reward in the system: no money, commission, or tokens ever flow to a referrer.

4.2 Invite credit

A buyer passes a referrer address with any mint. The credit rule is deliberately narrow:

- Credit is granted only on the invitee's first paid mint: the first mint by that Agent ID whose cost is greater than zero.
- On that mint, if a referrer was passed and resolves to a different Agent ID, the referrer earns +1 quota, once. The inviter binding recorded then is permanent.
- A free mint never credits anyone, never binds a referrer, and does not consume the first-paid-mint event: an ID whose first mint is free can still credit an inviter on its first paid mint.
- Credits stop at the cap: the 21st qualifying invitee grants nothing.

4.3 Social tasks and automatic verification

Three tasks, each worth +1 quota, each claimable once per Agent ID: follow the official X account (@Razor_Agent); like and quote the official announcement post (one task, two halves, +1 in total, both halves required); and join the project Telegram (@razoragentai).

Verification is automatic and uses the claimant's own access, because the underlying facts are not public: X likes have been private platform-wide since 2024.

- The claimant connects a wallet, then connects X through OAuth 2.0 (Authorization Code with PKCE) with read-only scopes only: `users.read`, `tweet.read`, `follows.read`, `like.read`. No write scopes and no offline access: the access token lives only inside an encrypted, HTTP-only session cookie of at most 30 minutes, never exposed to page JavaScript, logs, URLs, or any database, and the service stores no copy.
- The service checks the follow against the claimant's own following list, and the like against the claimant's own liked posts. For the quote half, it searches for the claimant's quote of the announcement; if the automatic lookup is refused by the X API, the claimant may paste a link to their quote post, and the service verifies it is theirs and does quote the announcement. The fallback can only ever prove the quote half; the like half has none.
- The Telegram task binds through the project bot. Binding creation requires proof of wallet ownership: the claimant signs a canonical message with EIP-191 `personal_sign` (containing the literal line "Razor Agent Telegram binding", the checksummed wallet address, and an issued-at timestamp); the service verifies recovery and a 300-second freshness window, and each signature is single-use. The claimant then opens the bot through a single-use deep link; the bot reads channel membership through the Telegram Bot API (`getChatMember`). The binding response never discloses the raw Telegram user ID.
- Before signing, the service re-reads the contract state (task already done for this ID; account already bound to another ID), requiring agreement between at least two independent RPC endpoints when two or more are configured, and enforces a single-flight, minimum-interval gate per wallet, account, and task.
- On a pass, the service signs an EIP-712 voucher over `Social(address user, bytes32 task, bytes32 account, uint256 deadline)`, with a 10-minute deadline. The task identifiers are the contract constants `keccak256("FOLLOW")`, `keccak256("QUOTE_TWEET")`, and `keccak256("JOIN_TELEGRAM")`. The account identifier is `keccak256` of the immutable numeric platform user ID, never of a changeable handle.
- The claimant submits the voucher to `claimSocial`. The contract checks the signature, the deadline, that this ID has not done this task, and that the account is not bound to another ID. On success the account binds to the ID permanently and quota increases by 1.

Two properties deserve emphasis. First, binding is permanent and one-to-one in both directions: one social account serves exactly one Agent ID forever, and a second ID presenting the same account is refused by the contract, not by the service. Second, there is no manual issuance path anywhere: not in the site, not in the service, not by the bot, not by any administrator. An unavailable check yields no voucher; the claimant retries later.

4.4 What the caps do and do not prevent

The per-ID caps bind an Agent ID, not a human being, and this paper does not claim otherwise. The identity root is the Agent ID registrar (Section 7.5): whatever it costs, in money or effort, to

control an additional Agent ID is the real price of an additional 33-share allowance and one additional free share. The contract cannot distinguish one human holding five IDs from five humans, and no rule in it pretends to.

What the caps do bound, regardless of how many IDs one actor controls:

- The free-share pool is global. Maximum extraction from it is 3,333 free shares, worth 3,333 times 1.15, about 3,833 USDT of genesis price; every free share one actor takes is one the rest of the market cannot.
- Every non-free share is paid at the full table price. Multiplying IDs multiplies allowances, never discounts.
- Invite credit cannot be farmed from free mints, and one ID's invite earnings stop at +20 no matter how many invitees it produces.
- The opening ratio and the refund promise are invariant under any distribution of buyers, because both pool legs and all refund records are computed per paid share.

The design judgment, stated plainly: the per-ID cap is a fairness and pacing rule, not an anti-sybil proof.

85 POST-OPEN ECONOMICS

5.1 The ladder

After share 10,000, minting continues on a nine-band ladder to 4,444,000 shares. Within each band, the price in cents for share n is linear between the endpoints: $\text{price} = \text{start} + (\text{end} - \text{start}) * (n - \text{bandStart}) / (\text{bandEnd} - \text{bandStart})$, exact at both endpoints. The bands are:

BAND	SHARES	PRICE START (USDT)	PRICE END (USDT)
Genesis	1 to 10,000	1.15	1.15
2	10,001 to 20,000	1.25	1.50
3	20,001 to 50,000	1.50	2.00
4	50,001 to 100,000	2.00	2.60
5	100,001 to 300,000	2.60	3.40
6	300,001 to 700,000	3.40	4.20
7	700,001 to 1,500,000	4.20	5.00
8	1,500,001 to 3,000,000	5.00	5.70

BAND	SHARES	PRICE START (USDT)	PRICE END (USDT)
9	3,000,001 to 4,444,000	5.70	6.30

Minting never expires; the ladder ends only at the supply cap or at share 4,444,000.

5.2 The curve mint: 99 percent of pool price, 80/20 split

A post-open share costs its ladder price from Section 5.1, split 80/20 exactly as in genesis: 20 percent to the treasury Safe directly, 80 percent paired into the pool. The tokens the buyer receives are not the flat 50,000,000 of genesis; they are priced against the live pool at 99 percent of the pool price: $\text{tokens} = \text{payment} \times \text{pool token reserves} \times 10,000$, divided by $(\text{pool USDT reserves} \times 9,900)$. The buyer thus mints at a 1 percent discount to the pool's spot price while the payment's pool leg deepens the same pool, and the paired leg mints pool tokens in proportion to the reserves so the pairing does not dilute the price. Yield is computed per share against reserves as they settle, so a batched purchase yields exactly what the same shares bought one by one would; packaging confers no advantage.

5.3 The floor mechanism

The ladder fixes what minting costs; it does not fix what the pool trades at. The one intervention in pool pricing is a floor. The contract tracks `totalPaid`, the cumulative USDT table value of all shares ever minted; the floor line is $\text{floorValue} = \text{totalPaid} * 50,000,000 / 27,727,060$, approximately 1.8034 times `totalPaid`. After each post-open share settles, if the pool-implied value of the circulating supply (circulating tokens times pool USDT reserves divided by pool token reserves) is below the floor line, the contract burns pool-held tokens toward the line, under two caps: at most a 0.8 percent price rise per share (the rise cap, 1,008/1,000, re-anchored after every share), and pool token reserves never burned below 11,666,666,667 RZAG. The burn consumes pool inventory tokens and touches no holder's balance. The floor operates only when post-open mints occur; it is not a price guarantee, a buyback promise, or a peg.

5.4 Taxes

Taxes apply to pool trades only. Wallet-to-wallet transfers and mints are untaxed.

- Buy tax: 1 percent of the traded tokens, routed to the treasury Safe.
- Sell tax: 2 percent: 1 percent burned to the dead address, 1 percent routed to the treasury Safe.

Tax tokens accruing to the mint contract are converted for the treasury under hard caps: the conversion swap runs at most once per block and swaps at most 0.1 percent of the pool's token reserves per run. The rates themselves are bounded in code: buy and sell taxes are each capped at 2 percent, permanently, hardcoded so the caps survive renounce. No function can raise a tax above its cap.

5.5 The LP vault and The Sunset Clause

LP tokens are never burned in this design, deliberately: a permanent burn would strand the liquidity forever, and a fixed calendar lock would be a date chosen in advance for a market of unknown lifetime. Instead, all LP is minted to `RazorLpVault`, whose states are Locked, Notice, and Released, and whose sole beneficiary is the treasury Safe address, hardcoded in the vault.

The release procedure, The Sunset Clause, is:

1. While Locked, LP cannot move, for anyone, including the Safe.
2. The treasury Safe, and only the Safe, may call `beginSunset()`, starting a public 30-day notice, visible on-chain immediately. During the notice, the pool stays 100 percent intact and any holder may exit through it.
3. The Safe may cancel the notice at any time before day 30, returning the vault to Locked. Nobody, including the Safe, can shorten the notice.
4. After 30 full days, LP can be released, and only to the Safe.

Liquidity can therefore never move without at least 30 days of public warning, and the warning is an on-chain event, not an announcement that can be missed or backdated.

5.6 Scale arithmetic (not a forecast)

If every share sold at its ladder price, total mint revenue would be approximately 22.68 million USDT (the sum over bands of band size times midpoint price, exact under linear interpolation, plus genesis), of which the 20 percent cut would be approximately 4.54 million USDT. A property of the price table, not a projection: most bands may never be reached.

86 FAILURE AND REFUND SEMANTICS

The failed state is entered in exactly one way: the deadline passes with `totalMinted` below 10,000 and the pool unopened. From then:

- The token never trades. Transfers remain frozen, and no pool is ever created from the genesis funds.
- Every buyer's refund record, written at payment time, becomes claimable through `claimRefund()`. The record pays the exact basis of the payment method used: for USDT payments, the exact USDT amount paid; for BNB payments, the recorded swap-obtained USDT value (the table cost plus any conversion surplus the contract actually received for that buyer). Refunds are always paid in USDT, never in BNB, so BNB price movement between payment and refund is the payer's, as disclosed at purchase.
- Claims have no deadline: no window, no expiry, and no sweep function. Unclaimed balances stay locked in the contract forever and cannot be taken by anyone, including the team. The project's own tests claim a refund 400 days after the deadline and are paid in full. A buyer whose only mint was free has a zero record and claims nothing, having paid nothing.

- The treasury receives zero. The escrowed cut of Section 3.6 is part of the same locked balance that funds refunds; the opening transfer that would release it can never execute in the failed state.
- Each record can be claimed exactly once; the record is zeroed before payment.

The refund promise does not depend on the operator being present, solvent, or cooperative. It depends only on the contract holding the funds, which it does by construction: before opening, no code path moves genesis payments except a buyer's own refund claim.

87

TRUST MODEL AND SECURITY POSTURE

7.1 Who can do what, and when it ends

ACTOR	POWERS	BOUNDS AND EXPIRY
Deployer	Call <code>startGenesis()</code> once; pause minting pre-open; execute the renounce ceremony after opening	Pause: 48 hours cumulative maximum, extends the deadline 1:1, dead at opening. Renounce: requires the pool open, disables pause first, then renounces ownership; the manual pool blocklist dies at renounce. After renounce, no powers remain.
Treasury Safe (2-of-3)	Receive the treasury cut and tax routes; call <code>beginSunset()</code> and cancel it on the LP vault; set the manual pool blocklist before renounce	Cannot mint, pause, change prices, touch escrow before opening, or reach refunds. Sunset release pays only to itself, after 30 full days of notice.
quotaSigner	Sign social-task vouchers, through the verification service only	Immutable (Section 7.2). Cannot move funds or change any state except, indirectly, +1 quota per verified task.
Verification service	Run task checks, sign vouchers	Claim-only. No keys to the contracts' administration, no custody of funds or tokens.
Anyone	Mint within quota, claim refunds, trade after opening, read all state	No registration beyond the Agent ID gate.

Two absences are part of the trust model. There is no snapshot-import or seed function: no code path can mint outside the public price rules, not a locked one and not a gated one, but one absent from the deployed bytecode. And there is no sweep, withdraw, or rescue function over genesis funds in any state.

7.2 The immutable signer and its blast radius

The quotaSigner address is set in the constructor and can never be changed by anyone. This is a deliberate trade: rotation would require redeployment, accepted in exchange for a signer whose authority no administrator can rotate, seize, or extend. The risk is the mirror image: a compromised signer key cannot be rotated either. The damage is bounded by the contract, not by custody: a leaked key can only sign social vouchers, each worth +1 quota, claimable once per task per ID, each permanently binding one social account to one ID on first use; no voucher can move funds, mint directly, or exceed the 33 ceiling. The key is held as a service secret and never appears in the website's code or configuration.

7.3 The treasury Safe

The treasury is a published 2-of-3 multisignature Safe at

`0xC82d07450301152779112eFFF5c867bcf75D1537`, owned by the project's deployer, owner, and cold wallets. It receives the genesis cut at opening, the post-open mint cut, the buy tax, and the sell-side treasury percent, and is the sole possible recipient of LP at a sunset release. Its threshold and address are public from before the sale, so every fee flow here terminates at an address any reader can watch.

7.4 Verification the project submits to

- **Source verification.** Contract source is verified on the chain explorer from deployment day, before the announcement.
- **Published parameters.** The complete parameter set is public in structured form before genesis starts (Section 2.4), including dependency addresses and risk statements.
- **Contract-read state.** Every live figure on the site is a contract read (Section 2.4); the project's claims about its own sale can be falsified by a single `eth_call`.
- **The project's own testing record, stated factually.** Pre-launch testing included: a local contract suite of 44 tests (genesis fill, opening composition, refunds claimed 400 days out, both payment paths, ladder boundaries, vault notice flow); an internal line-by-line security review of contracts, service, site, and indexer (findings F-01 to F-15, all closed, with a re-review that lifted its initial deploy hold); an acceptance pass redeeming real service vouchers on a local deployment for all three tasks (43 of 43 checks); and an internal adversarial red-team pass: 70 contract attack cases (63 stopped by the contracts; two economic findings fixed and the suite re-run; the rest accepted with reasoning published in the project's review files) and 10 web-surface proofs of concept (the one medium finding, an unauthenticated Telegram binding path, closed by requiring the wallet signature of Section 4.3). These are

internal processes. **No third-party audit is claimed, and none of the above should be read as one.**

7.5 Limitations, stated plainly

- **Pool blocking is best-effort.** The token hard-rejects the PancakeSwap v3 PoolManager and the Infinity Vault permanently, rejects other pairs detected by a standard probe, and carries a manual blocklist that lives until renounce. Adversarial contracts can evade probe-based detection (for example by making the probe fail), so unofficial pairs cannot be fully prevented on-chain. Only the pair linked from the official site is official; trades elsewhere sit outside this paper's tax and price design. Mitigation is operational: watch for pair creation and blocklist quickly.
- **The BNB path depends on the Chainlink feed.** A stale or out-of-range reading closes the BNB method until the feed recovers; the USDT path has no such dependency. All conversion friction is the minter's by rule (Section 3.5).
- **Social verification depends on the X API and the Telegram Bot API.** Outages, rate limits, or policy changes can make tasks temporarily unverifiable; the consequence is no voucher, never a manual grant.
- **The identity root is external.** Quota fairness is exactly as strong as the registrar's issuance. If Agent IDs become cheap to multiply, per-ID caps dilute accordingly; the global bounds of Section 4.4 still hold.
- **Public RPC dependence.** Site and service read chain state from public RPC endpoints with fallback and, in the service, cross-endpoint agreement for signing decisions. Endpoint failure degrades to the labelled stale and unavailable states of Section 2.4, never to fabricated figures.
- **Tax conversion timing.** A self-funded price crash timed against the capped tax conversion can degrade one tranche's conversion rate (about 10 percent in the project's red-team exercise); the caps bound exposure per block, unconverted tax remains for later blocks, and the attacker's round-trip cost exceeded the damage in the tested scenario.
- **Pre-open transfer freeze.** Tokens cannot move before opening except sends from the mint contract itself; buyers who need liquidity before the deadline cannot have it, by design.

PARAMETER TABLE

Every locked number, matching `params.json` version 1 (31 October 2026). The contracts govern on any disagreement.

PARAMETER	VALUE
Network	BNB Smart Chain, chainId 56

PARAMETER	VALUE
Token	RAZOR AGENT (\$RZAG)
Tokens per share (buyer)	50,000,000 RZAG
Pool inventory per paid genesis share	40,000,000 RZAG
Total shares	4,444,000
Token supply cap	444,000,000,000,000 RZAG
Genesis opening threshold	10,000 shares
Genesis price	1.15 USDT per share (flat)
Free first share	1 per Agent ID, genesis only
Free-share global cap	3,333
Genesis window	10 days from <code>startGenesis()</code>
Split (genesis and post-open mints)	80 percent pool / 20 percent treasury
Per paid genesis share	0.92 USDT pool leg, 0.23 USDT treasury cut
Opening composition	paid shares x 0.92 USDT against paid shares x 40,000,000 RZAG
Opening price	1.15 USDT per 50,000,000 RZAG
Payments	USDT exact (<code>mint</code>); BNB converted (<code>mintWithBnb</code>)
BNB conversion	Chainlink BNB/USD feed plus 1 percent headroom (100 bps)
BNB friction rule	Minter bears all friction; shortfall reverts (<code>SwapShort</code>); surplus to escrow, then treasury
BNB refund basis	Recorded swap-obtained USDT value
USDT refund basis	Exact amount paid
Refund claims	<code>claimRefund()</code> , any time, forever, no sweep

PARAMETER	VALUE
Treasury on failed genesis	0
Pause	Deployer only, pre-open only, 48 hours cumulative, deadline extends 1:1
Max purchase per transaction	10 shares
Quota base	10 per Agent ID
Invite credit	+1 per invitee's first paid mint, cap +20 per ID
Social tasks	3 (follow; like and quote; join Telegram), +1 each
Quota ceiling	33 per Agent ID
Voucher	EIP-712 <code>Social(user, task, account, deadline)</code> , 10-minute deadline, immutable signer
Ladder	9 bands, 1.15 flat genesis, then 1.25 to 6.30 (table in Section 5.1)
Post-open mint price basis	Ladder price paid; tokens priced at 99 percent of pool price (9,900 / 10,000 bps)
Floor line	$\text{totalPaid} \times 50,000,000 / 27,727,060$ (about 1.8034 x totalPaid)
Floor rise cap	0.8 percent per share (1,008 / 1,000), re-anchored per share
Pool token floor	11,666,666,667 RZAG minimum pool token reserves under the burn
Buy tax	1 percent, to the treasury Safe
Sell tax	2 percent (1 percent burn, 1 percent to the treasury Safe)
Tax ceilings in code	2 percent per side, hardcoded, survive renounce
Tax conversion caps	At most 1 swap per block; at most 0.1 percent of pool token reserves per swap
LP custody	RazorLpVault; release only to the treasury Safe

PARAMETER	VALUE
LP sunset notice	30 days, public, cancellable by the Safe, never shortenable
Treasury Safe	2-of-3, 0xC82d07450301152779112eFFF5c867bcf75D1537
quotaSigner (immutable)	0xa7F6091B833Dd3db57293b49cB6c24CA0c487CD9
Agent ID registrar	0x60B9262456dE011F7d15a54CdD8c48bAaa8B022F
Chainlink BNB/USD feed	0x0567F2323251f0Aab15c8dFb1967E4e8A7D42aeE
USDT (BEP-20)	0x55d398326f99059fF775485246999027B3197955
Team token allocation	0
Mint, token, LP vault, Published in params.json at deployment pair addresses	

89 WHAT THIS PAPER DOES NOT CLAIM

- No returns, yields, or price predictions, for the mint or for secondary trading. The ladder is a mint price schedule, not a valuation.
- No affiliation with Binance. Agent ID is an external identity system the project depends on and does not operate; Razor Agent is not a Binance product and is not endorsed by Binance.
- No third-party audit, certification, or regulatory approval; Section 7.4 describes internal testing and review only.
- No guarantee that the genesis succeeds, that any ladder band is reached, or that a secondary market exists after opening.
- No claim that per-ID quota prevents one actor from controlling multiple Agent IDs (Section 4.4), or that trading can be confined to the official pair with certainty (Section 7.5).

AZ GLOSSARY

- **Share.** The unit of sale: 50,000,000 RZAG in genesis; a ladder-priced, pool-priced amount after opening.
- **Quota.** The maximum shares one Agent ID may mint through the mint contract: base 10, plus invite and social earnings, ceiling 33.

- **Agent ID.** A Binance Agent ID identity, resolved on-chain from the registrar; the unit of gating, quota, and binding.
 - **The Vault.** The escrow state of the mint contract before opening, in which 100 percent of genesis payments, pool leg and treasury cut alike, remain locked.
 - **The Deadline.** The genesis end: start plus 10 days, extended second for second by pauses. Missing 10,000 shares by it triggers the failed state.
 - **The Sunset Clause.** The LP release procedure: a public 30-day on-chain notice started by the Safe, cancellable, never shortenable, after which LP releases only to the Safe.
 - **Agent Pack.** The machine-readable instruction file (`agent-pack.json`): addresses, function signatures, state views, action recipes, and guardrails.
 - **Floor line.** The post-open reference value, $\text{totalPaid} \times 50,000,000 / 27,727,060$, toward which the contract burns pool inventory (Section 5.3).
 - **Voucher.** An EIP-712 signature by the quotaSigner attesting one verified social task for one user, account, and deadline; the only way social quota is granted.
-

*End of whitepaper v1.0. The deployed contracts supersede this document on any disagreement.
Verify: razoragent.fun/params.*