



PRODUCT / BUSINESS / TOKENOMICS

Communities that build, play and belong.

A community workspace on Monad.
Useful contributions. Wallet-controlled actions.
A business built around real participation.

MONAD METROPOLIS AND HELIX ARE TESTNET PILOTS.
ORB, missions, Council and Helix are deployed on Monad testnet 10143. No mainnet token or token sale.

Holder Pro and rewards are test pilots. The AI assistant is metered. No price forecast or promise of profit.

01 / A reason to exist

Give communities a shared purpose.

A project needs people to test its game, improve its documentation, answer questions and help newcomers. A member needs a place to belong, useful opportunities and a contribution record that does not disappear when they leave a platform. Today, these activities often happen across chats, forms, spreadsheets, wallets and payment tools.

Orb's product thesis is to connect that journey: find a community, contribute something useful, have the work reviewed, receive a funded reward and keep a public wallet record. Voice, chat, trading and Arena support the community around that journey rather than compete as unrelated headline features.

Who comes first

- Community organizers: coordinate contributors and see what was actually delivered.
- Web3 projects and game studios: commission playtests, tutorials and community work with a clear budget.
- Members and creators: discover work, collaborate in the community, track reviews and withdraw accepted rewards.

The return loop

Projects fund useful missions. Members contribute. Sponsors review against published criteria. Accepted contributions create a public record. Organizers publish follow-up work and contributors return for relevant opportunities, community relationships and play.

Retention is a hypothesis to measure, not a claim that rewards automatically create loyal users. Empty missions, repetitive check-ins, AI-generated spam and paid reputation would undermine this loop.

What ORB is for

ORB pays for Orb's own services. It is not a share, a claim on revenue or an investment product, and testnet ORB has no promised value.

- Fund work: sponsors prefund mission rewards in ORB; contributors withdraw accepted credits.
- License a community: the verified owner locks 100 ORB for 30 days (Holder Pro, section 06).
- Back a proposal: Council votes lock ORB until the vote closes, then return it (section 08).
- Cover costly surfaces: a license lifts the assistant's daily allowance and the Open Ledger's daily write allowance.

Reputation and Passport scores never depend on ORB.

First validation goal: three real communities, ten useful missions and interviews with both sponsors and contributors. A mission must solve an actual project problem before it deserves distribution or a token incentive.

02 / One coherent journey

From conversation to contribution.

Community missions

The testnet pilot provides a public mission board, task templates, acceptance criteria, a fixed reward per approved contribution, deadlines and a prefunded test-ORB budget. Each wallet can submit one public proof link. The sponsor reviews it; an approved reward becomes a credit that only its recipient can withdraw. Unused funds become refundable after the seven-day review window.

The board includes Discover, My work and Organize, mobile layouts and wallet-scoped receipts. A sponsor-chosen label is not proof of community ownership or endorsement, and acceptance is subjective sponsor review, not certification.

Trade with context

The existing trading workspace includes charts, spot and perpetual interfaces, an assistant that prepares trade drafts, and community trade-card sharing. A user reviews the order in Orb and signs through the supported embedded-wallet/session flow. Supported wallet calls avoid a duplicate Privy transaction popup; initial authorization, recovery and MFA remain separate. An AI response is not an executed order, and an accepted order is not necessarily a fill. This whitepaper does not introduce a hidden execution fee or broaden signing authority.

The assistant is the one surface whose marginal cost is real money, so it is the one surface that is metered: five reviews per wallet per rolling day, then an active Holder Pro licence. It reads markets and creates alert-only price watchers; it never signs, never holds a key and cannot spend.

Play together

Arena gives communities a shared strategy game. Its practice mode and onchain flows have different trust assumptions. Existing sponsored-reward experiments are separate from mission rewards. Neither reward system makes a player's identity unique or turns game participation into a guaranteed income.

Keep a record

Orb Passport provides portable, wallet-linked evidence and independent verification tools. Mission approvals add a separate public contribution record in the mission contract. They do not currently feed Passport scoring automatically. Reputation remains separate from ORB holdings: buying or holding a token must never buy a better contribution record. Separately, the Open Ledger lets an owner save a community's text on Monad, privately (sealed to a key only the owner holds) or publicly (plain text, permanent), readable with only an RPC; the operator's keeper reads the text before sealing it, so it is not end-to-end encryption between members (see Open Ledger (orb-ledger.md)). A community leaving Discord can import its exported history into read-only channels first (import (orb-discord-import.md)). Authors may also switch on opt-in tips: a reader sends a little testnet MON from their Orb Wallet to the author's, and the post shows it only after the server reads the transfer back from the chain; Orb never holds the money (tips (orb-tips.md)).

Implementation boundary: ORB and its mission contract are deployed on Monad testnet. Browser tests use simulated Privy callbacks and a real local EVM. These tests do not certify a production wallet journey or launch a mainnet asset.

03 / Why Monad, and what is decentralized

Public settlement. Private conversations.

Monad supports Ethereum-compatible addresses and transaction formats. That lets Orb connect its existing wallet workflow to smart contracts without inventing a new custody system. The useful scaling unit is many communities settling independent rewards and game actions, not a bot manufacturing transactions to advertise TPS.

Layer	Where it lives	Practical boundary
Mission budgets, approvals and credits	Smart contract	Public; sponsor review remains subjective
Mission titles, criteria and proof links	Smart contract	Deliberately public; never paste private data
Passport evidence commitments	Onchain anchors and proofs	Commitments are not plaintext messages
Wallet signing and session authority	Wallet / Privy flow	Keys and signer permissions still matter
Chat, discovery and account services	Application backend	Server availability and access controls matter
Voice, video and screen distribution	Media server	Current media transport is not decentralized

Mission contract boundaries

Wallet-controlled communities can export owner-signed descriptions and channel templates as IPFS CARs and verify metadata and owner through Monad; a downloadable static client with an RPC picker reduces dependence on Orb's web host. Neither migrates hosted chats nor removes wallet/RPC dependencies. See independent communities ([orb-independent-communities.md](#)).

- Testnet-only deployment guard; the client verifies chain and exact runtime bytecode.
- Exact prefunding, fixed reward slots and a disclosed 5% fee on accepted rewards.
- Only the sponsor reviews; only a wallet owner withdraws that wallet's credits, and accepted credits cannot be refunded to the sponsor. One failed withdrawal cannot block other users.
- No upgrade, arbitrary-call function or admin sweep; ORB funding uses exact budget allowances.

Public briefs and links are permanently observable, and contributors confirm a submission holds no private messages, personal data or secrets. This does not certify all chats as end-to-end encrypted or the app as unhackable. Local contract and simulated-wallet tests are useful evidence, not an external audit, a production Privy acceptance test or a measurement of Monad's throughput.

04 / A business before a token

Revenue from useful community services.

Joining, chatting and playing stay free. Orb's first customers are organizers and projects with a recurring coordination problem. Pilot prices are proposals; subscriptions and fiat AI billing are not active.

Revenue source	Pilot proposal	What the customer pays for
Community Pro	USD 19 / month	Organizer tools, managed bots, archives and analytics with usage limits
AI review packs	Holder Pro licence today; USD 5 / 100 reviews is the fiat proposal	Useful assistant reviews with transparent usage accounting
Sponsored campaigns	USD 99 / campaign	Campaign setup, sponsor labeling and aggregated delivery reports
Mission service fee	5% of accepted gross rewards	Prefunding, workflow and settlement; no fee on unused/refunded budget

A mission reward of 1 test ORB allocates 0.95 to the contributor and 0.05 to the fee recipient; approval reserves both, withdrawal is separate. Testnet tokens are not revenue, and a production reward currency and launch require separate review.

AI-assisted trading fees

A later execution fee must be venue-supported, shown before signing and tied to a verified execution: never on cancelled or failed orders, never altering the recipient, never a hidden spread, never authorizing an AI to spend freely. The current implementation charges no such fee and does not assume Perpl/Kuru pays Orb.

What is metered today is inference, not execution: five assistant reviews per wallet per rolling day, then an explicit refusal unless an active Holder Pro licence covers the member - their own, or one an organizer bought for a community they belong to. No card, no in-app purchase, no silent downgrade; an unreadable licence counts as absent.

An illustrative month, not a forecast

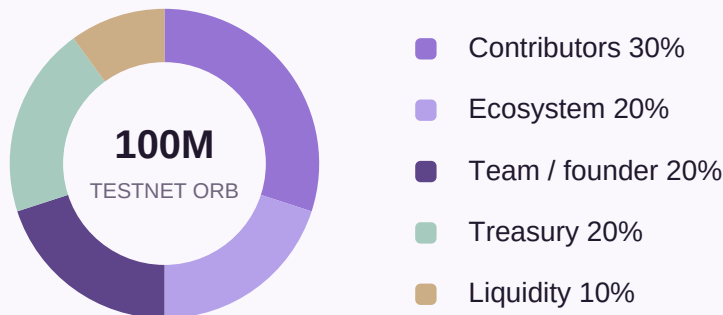
Ten Pro subscriptions, twenty AI packs, two campaigns and USD 1,000 of accepted mission rewards would produce USD 538 in hypothetical gross revenue, excluding token sales and token price changes. AI inference, RPC, gas sponsorship, voice egress, storage, support, acquisition and taxes must be deducted before discussing profit.

The app saves account-scoped pilot-interest requests, with cancellation and no payment authorization: a way to test willingness to pay before promising unlimited infrastructure.

05 / ORB allocation proposal

Fixed supply. Visible ownership.

Testnet name: Orb. Ticker: ORB. Fixed supply: 100,000,000 tokens with 18 decimals. The public deployment record identifies all contracts and beneficiaries. No mainnet token, sale price, market capitalization or trading pool is established by this document.



Allocation	Share	Tokens	Intended use
Community contributors	30%	30,000,000	Reviewed, useful contributions; capped campaign releases
Ecosystem and partnerships	20%	20,000,000	Integrations, community pilots and partner support
Team and founder	20%	20,000,000	Founder/team compensation, subject to vesting
Protocol treasury	20%	20,000,000	Budgeted operations, development and contingency
Liquidity	10%	10,000,000	Transparent initial liquidity allocation if a market launches
Total	100%	100,000,000	No additional inflation proposed

The 20% team allocation is intended for the founder/team, including the current founder. It is compensation and a material concentration of supply. On testnet it is held in a deployed vesting contract. It is not community-owned or already vested.

Contributor and ecosystem allocations are spending budgets, not automatic emissions for clicks, messages, referrals or trading volume. Treasury tokens are not cash revenue. Liquidity also needs a paired asset budget; merely reserving ORB does not guarantee liquidity, a price floor or appreciation.

Deployed testnet token: fixed-supply ERC-20, with no transfer tax, mint extension, blacklist or upgrade administrator. Runtime is checked against compiled source. This is not an independent security audit.

06 / Holder Pro

A community operating license.

Holder Pro on testnet requires the verified community owner to lock 100 ORB for 30 days to activate one community license. Members do not need ORB to enter, chat or play. Conventional subscription access remains a business proposal.

What the license should unlock

- Shipped: community counts and organizer report exports, gated by current ownership and a verified license.
- Shipped: assistant reviews past the free daily allowance, for the licensee and for members of a community whose organizer holds the license.
- Shipped: four times the daily Open Ledger write allowance, because every byte written to Monad costs gas.
- Planned: community customization and automation tools within explicit quotas.
- Planned: contribution analytics and campaign coordination.
- Planned: limited managed bot events; additional resource use remains metered.

Storage, inference and media traffic cannot become unlimited lifetime promises funded by a one-time token purchase. The license is a 30-day lock that expires and must be renewed, and infrastructure-heavy usage still needs a quota or top-up. Today the assistant meter bounds abuse and provider limits as much as cost. Holder access does not grant moderation powers, private-channel access or treasury authority.

Eligibility and renewal

The testnet pilot locks 100 ORB for 30 days per community. One reviewed activation completes exact approval/revoke and locking, with simulations and receipt checks for each transaction. A bounded faucet distributes 100 ORB once per wallet from a 100,000 ORB contributor budget. Wallets are not unique humans. Expired stakes return to the original wallet, including after a community ownership change. Production thresholds need cost and demand validation.

The server verifies current ownership, the account-bound wallet, exact contract runtime and a fresh license record. Expired or unverifiable licenses cannot open organizer reports. Holding never grants community ownership, moderation, private-message access or a better Passport score.

What can create demand

The hypothesis is that useful communities attract organizers, organizers want the tools, and token-backed licenses create service demand. More active licenses could increase tokens held or locked. This does not guarantee that purchases exceed sales, that liquidity remains healthy or that the token appreciates.

ORB should never confer passive revenue rights, guaranteed returns, pay-to-win advantages or a better Passport score. No automatic buyback or burn is promised. Product revenue belongs in operating accounts, not in a narrative that token holders are guaranteed income.

07 / Distribution and vesting

Long-term commitment, explicit limits.

Testnet schedules start at the deployed token launch timestamp. One contract month equals 365 days / 12. Team vesting and first-year budget caps are enforced by deployed allocation vaults. Any mainnet TGE remains unscheduled and requires a separate launch review.

Allocation	Proposed release policy
Team / founder: 20M	Zero release for 12 months after TGE; equal monthly vesting across the following 36 months; final rounding remainder in month 48
Contributors: 30M	At most 1M per quarter during the first year, only for reviewed campaign work; unused budgets remain undistributed
Ecosystem: 20M	At most 2M during the first year; after year one, remaining tokens are operator-controlled
Treasury: 20M	Held in a transparently identified treasury; first-year operating release capped at 2M, with a disclosed operator
Liquidity: 10M	Separate custody and paired-asset budget; disclose deployed pools and LP control before launch

Founder vesting example

- Months 0-12: no founder/team token release.
- Months 13-24: approximately 6.667M vested cumulatively by month 24.
- Months 25-36: approximately 13.333M vested cumulatively by month 36.
- Months 37-48: the full 20M allocation becomes vested by month 48.

First-year contributor, ecosystem and operating caps sum to 8M tokens. The 10M liquidity allocation is separate. These caps do not define circulating supply: unlocked, issued, distributed, pool-held and actually circulating tokens are different measures and must be reported separately.

Treasury control

The testnet uses the existing deployer as founder beneficiary and operator; it is single-operator custody, not independent multisig governance. Before mainnet, establish genuinely independent treasury signers, delayed transfers and published budgets. Token votes should initially guide product priorities rather than directly control user funds or wallet permissions.

Published rules must not be silently weakened. Material changes need notice, visible approval and a migration plan for existing holders. Team locks are enforced on testnet. Non-team budgets become operator-controlled after year one; a production policy needs further review.

08 / Orb Council

A community voice, with team accountability.

The deployed testnet Council lets members propose useful product improvements, vote with temporarily locked ORB and see an explicit team decision. It guides the roadmap; it cannot transfer treasury funds, grant roles, execute code or access private messages.

Vote and review

- Publish a public problem, solution and acceptance criteria and a topic, with a refundable 10 ORB bond. The contract accepts voting windows from 10 minutes to 7 days; the author can withdraw a proposal nobody has voted on and take the bond back at once.
- Each wallet votes once with at least 5 ORB, locked until closing. A voice is the square root of the ORB locked, so four times the ORB buys twice the voice.
- Quorum requires three wallets and 100 ORB raw stake. The voice for must strictly exceed the voice against. A tie fails. A vote that flips the result in the last ten minutes extends the vote by ten minutes, at most three times.
- After closing, an authorized Orb team member records approval or rejection and a public reason. Approval requires a passed vote and review of feasibility, security, scope and product fit.
- Approved items may receive one public delivery link. A team decision is not automatic implementation or a delivery guarantee.

How a voice is counted

The first Council capped the largest wallet at the sum of everyone else's stake, which still let a 1,000 ORB wallet plus a 5 ORB friend outvote any opposition worth less than about 1,005 ORB in total. It was replaced, before any proposal existed, by square-root voice: the same two wallets are worth about 33.9, and seven wallets locking 50 ORB each are worth about 49.5, so a broad opposition wins. The tally is kept as two running sums, so it never scans voters.

The voice is per wallet, not per person: a holder can split across wallets and add voice. The team can mark a proposal as spam; the mark moves no funds and erases nothing. Team review is mandatory; the pilot does not claim Sybil-proof or fully decentralized governance. Initial reviewer administration belongs to the existing testnet operator, who can add or revoke team reviewers. Reviewer authority affects Council records only.

Refunds and safety

Authors and voters can reclaim their own bond/stake after the deadline, independently of the team's response. Withdrawals do not change the finalized tally. No Council fee, admin sweep or upgrade exists; network gas remains payable. Exact token allowances, runtime checks and wallet-scoped recovery protect the client flow.

Proposal text, addresses, votes and decisions are public and permanent. The interface asks for consent and never imports chat text automatically. Refundable bonds add friction, not complete spam protection. Independent contract review and stronger operational governance remain prerequisites for mainnet.

09 / Prove the product

Earn demand before launching supply.

What exists in this build

Orb connects communities, wallets, reviewed AI trade drafts, Arena and Passport. ORB vaults, Holder Pro, faucet, missions, Council and the archive index are deployed on testnet. Local EVM tests cover permissions, accounting, voting and refunds; browser tests use simulated Privy callbacks. The business pilot saves account-scoped interest without charging users.

Also shipped here: the Open Ledger (private and public modes, a keeper that journals every transaction, an independent reader and a published protocol), tested against a local EVM; five interface languages behind a coverage gate that fails the build on an untranslated string; a metered assistant; anonymous rate limits keyed per client address that fail closed in production instead of sharing one bucket; membership required before an installed community app's command runs; and a documented path to fork the client, run your own copy and recover an archive without Orb. Measured on a local fixture: 20,595 bytes of plaintext became a 4,771-byte encrypted block (4.3x), and full batching cut five transactions to two, 846,064 gas to 723,025 (-14.5%).

Next milestones

1. Run a controlled Monad testnet mission pilot with real sponsors and contributors; verify funding, submission, approval, withdrawal and independent contract reads.
2. Measure weekly contributor return rate, repeat sponsor rate, review time, accepted/rejected work and cost per active community, separating test traffic from real participants.
3. Deliver quota-aware Pro features and measure willingness to pay. Connect real billing only after the product, prices and support flow are ready.
4. Validate venue-supported trading revenue with explicit execution fees and receipt-based accounting. AI must remain within each user's authorized wallet/session limits.
5. Before a mainnet launch, independently review fixed supply, vesting, license eligibility and treasury controls, and publish addresses and commercial/regulatory terms.

Measures that matter

Repeat sponsor spend and useful accepted work matter more than raw transaction count. Daily active wallets are not unique humans, and token price is not a product-success metric. A meaningful demo shows a complete journey, public receipts and honest boundaries between local tests, testnet behavior and plans.

09 / Prove the product

References and implementation.

- Monad transaction compatibility: <https://docs.monad.xyz/developer-essentials/transactions>
- OpenZeppelin ERC-20 and vesting references:
<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v5.4.0/contracts/token/ERC20/ERC20.sol> and
[.../finance/VestingWallet.sol](https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v5.4.0/contracts/finance/VestingWallet.sol)
- Council deployment: <https://orb.168-138-150-148.sslip.io/token/council.deployment.monad-testnet.json>
- Public testnet deployment: <https://orb.168-138-150-148.sslip.io/token/deployment.monad-testnet.json>
- Orb implementation: `contracts/OrbMissions.sol`; `components/orb-missions.tsx`; `scripts/test-missions.mjs`;
`scripts/test-missions-ui.mjs`; `scripts/test-orb-business.mjs`.

This draft does not promise investment performance, audited security, custody guarantees or a hackathon outcome. Missions still depend on sponsor decisions, and spam, collusion, false work and link abuse require operational moderation. Mainnet token and payment deployment are separate launch decisions, not authorized by publishing this whitepaper.

October 2 / Local journey addendum

The source checkout now connects normal community settings to the independent public registry. Only the current hosted owner, authenticated with the profile-bound Privy Orb Wallet that owns the onchain identity, can create/link that association. The owner signs a public description, links and channel blueprints; readers can verify the capsule and current owner/CID using a downloadable client and their chosen RPC. Hosted messages, membership and moderation are not migrated. IPFS retention still requires actual pin operators; an ID alone does not retain a file.

The contribution journey links a funded mission to an organizer's editable session plan. Members can return to the event from the mission panel, follow sponsor review and explicitly withdraw approved credits within that panel. The independent client checks regular ORB mission/token runtimes, contribution status and a withdrawal receipt without Orb API or login. Contract-bound links distinguish ORB contributions from the separate native-MON escrow. Withdrawals can combine mission credits and refunds; sponsor approval establishes the sponsor's decision, not objective work quality. The existing 5% approved-work testnet fee remains visible.

The local EVM/browser tests passed with two wallets and simulated Privy sessions, including the independent viewer with Orb API blocked and rejection of forged receipts. This is not evidence of two real Privy accounts on public testnet, actual IPFS replication or an independent security audit. See the complete rehearsal checklist ([orb-hackathon-rehearsal.md](#)) and portability implementation ([orb-community-portability.md](#)). These changes have not been pushed or deployed by this iteration.

October 3 / Portable chat addendum

Orb now has a local testnet implementation for wallet-recoverable encrypted personal archives and a separate onchain inbox. Public Monad testnet receipts confirm key registration, a personal archive and a wallet-to-wallet message with two controlled EOA wallets. A downloadable client on another origin can recover the same chat key with the original wallet and read confirmed ciphertext without Orb API. It publishes full ciphertext, not merely evidence commitments. Six-hour incremental preparation is opt-in and requires a visible unlocked client and active chat session; an operator keeper can publish queued signed parts within their expiry. A scoped delegate cannot spend user wallet funds.

This is an exit path for historical text, not a migration of hosted membership, moderation or voice/video. Personal copies do not prove other authors' statements. Unreadable legacy text and attachment files are excluded. Ciphertext, addresses and traffic remain permanent/public. Original keys remain intact; no ratchet, independent cryptographic audit or universally available free archival RPC is claimed. Privy recovery uses the original app on authorized origins and still depends on Privy/MFA; an unrelated app does not recover the same embedded wallet. Actual Privy acceptance remains pending. This UI has not been pushed or deployed to Oracle by this iteration. See the implementation and receipts ([orb-portable-chat.md](#)).

October 4 / Scalable archive addendum

The local backup interface now prefers a separate hybrid index on Monad testnet 10143. Ciphertext lives in bounded, content-addressed blocks on configured replicas/IPFS pins; the chain stores signed digest/stream/sequence/epoch references. Existing wallet recovery identities and older full-onchain archives remain intact. Independent readers verify selected pages against these references and decrypt locally. A CID alone does not provide retention or key recovery.

Export checkpoints, authenticated upload quotas, atomic bounded queues and leased workers keep each operation bounded. Independent community streams support one encrypted content block plus per-wallet epoch grants. That shared protocol does not automatically export hosted private-channel messages or replace their access rules. Previously authorized readers retain their old keys and readable history.

Two folders on one development computer are not geographically independent replicas. Offsite storage, real Privy acceptance, load/failure testing and an independent security review are release gates. No 120-million-message capacity, production database cluster or mainnet launch is claimed. The public origin does serve this work - measured October 5, /proofs, /independent and the downloadable kit answer 200 and the live bundle manifest matches the working tree - but it is the October 4 build, so the newest local changes are not deployed there yet. See the current scalable design and measured fixture scope (orb-chat-scale.md).

October 5 / Metering, languages and documentation addendum

The assistant now has a price. Every wallet gets five reviews per rolling day; after that the request returns an explicit refusal that names Holder Pro, unless an active 100 ORB licence covers the member - their own, or one an organizer bought for a community they belong to. The check reads the same licence record the organizer dashboard reads, and an RPC that does not answer means "not unlocked", never "unlocked". Nothing in this path signs, holds a key or spends; the model still only sees the member's own message and whitelisted tool output, and it can only propose alert-only watchers and reviewed tickets.

Language coverage became a gate, not a claim. The interface ships in English, Brazilian Portuguese, Spanish, French and Mandarin through one inline catalog, and `npm run i18n:test` now exits non-zero on an untranslated string or a hand-rolled two-language helper, so coverage cannot be asserted while quietly rotting. Commands members type stay English-only in every language, the way they type an API: one spelling, stable autocomplete, copy-pasteable documentation.

A promise the product could not keep was deleted. The `.nad` handle/name-service integration depended on a third-party registrar that is not live for Orb on Monad testnet, so it was removed end to end rather than shipped as a dead field; migration 0050 drops the column. Onchain identity remains what actually verifies today: the wallet link and ERC-721 avatars with ownership checks.

Abuse limits are enforced, and fail closed. An installed community app's command now requires membership before it runs, so a stranger cannot reach an app's webhook (a regression test asserts the refusal happens at the channel and again for a non-member). Anonymous limits are keyed per client address instead of one shared bucket, and in production a request whose client address cannot be determined is refused with a 503 naming `ORBITA_TRUST_LOCAL_PROXY=1` rather than being lumped into a global counter. Open reads (GIF search, feed profile) are throttled per member.

Documentation is English-only now, because the judging panel reads English. The Portuguese records were translated rather than deleted, so the disclosure of the pre-existing Orbita foundation required by section 4.1.4 of the rules stays intact, and the submission checklist (`orb-submission-checklist.md`) maps every mandatory requirement to the file or manifest that satisfies it. Testnet 10143 is compliant as written: sections 4.1.3 and 9.2 require deployment to "Monad mainnet or testnet".

Still open and not claimed as done: real Privy acceptance on public testnet, actual IPFS retention, an independent security review, and the founder-only steps of making the repository public, recording the three-minute demo and submitting.

October 10 / Council, apps and one English addendum

The Council's tally was wrong, so it was replaced before it was used. The first contract capped the largest wallet at the sum of everyone else's stake, which still let a large wallet and one friend outvote any opposition worth less than their own stake, however many people it was spread across. Version 2 counts a voice as the square root of the ORB a wallet locks, closes the last-block advantage (a vote that flips the lead near the deadline extends it, at most three times), lets an author withdraw a proposal nobody has voted on and gives the team a public spam flag that moves no funds. No proposal had been made on the old contract; its address stays in the deployment manifest. An approved proposal now opens a mission brief with its title filled in, so an idea can become funded work.

Apps became a platform with a public record. An Orb app is a service identity with its own token, never a person. It reaches the API through two gates that different people hold: the scopes its developer asked for and the role a community admin accepted. Apps can be private, unlisted or public, with invite links and a directory; they listen by long poll or webhook, register commands, and there is a JavaScript SDK and a developer page. An app can also be put, by its developer's own wallet, on the Orb Bot Registry on Monad testnet: a hash of its manifest that anyone can recompute from the public listing, the scopes each community accepted, and an optional one-time price in the test token, split 95% to the developer and 5% to Orb in code nobody can change. A priced app is installed only after the community's payment is on chain; the contract can pause, upgrade or take back nothing.

The assistant is a conversation, and the trading channel is a desk. A review is a conversation with up to twelve messages instead of one; answers stream; a market read costs no model call at all; and the trade desk shows the chart with the order, levels and position on it. The assistant still reads and drafts: it never signs, never holds a key and cannot spend.

English is the canonical language of the product. Commands, the API, the documentation and the messages the server sends are English; Portuguese is a translation of the interface, and a reader of Portuguese gets the original wording of a server message back from a table fetched only for them. A test fails when server code throws Portuguese or when that table drifts from the code.

Still open and not claimed: the voice stage and the role editor are being rewritten in another change, the testnet token has no value, nothing here was audited independently, and the founder-only steps are unchanged.