Oracle-Node (ON) & Offer Maker Analyzer (OMA)

Explained: A Zero-Fee, Non-Custodial Coordination Layer for Kaspa/KRC-20 Swaps

Whitepaper — v2.1 • October 25, 2025
By
Todd J. Desiato, CEO, *NC SOLAR MINING LLC*

"Signal in. Signal out. Join the Collective."

Executive Summary

The Oracle-Node (ON) and Offer Maker Analyzer (OMA) work together to enable direct, peer-to-peer swaps on the Kaspa network without custodians and without venue fees. OMA is the human-centered app where a person creates and responds to offers. ON is the behind-the-scenes node that safely relays messages, checks basic rules, and helps peers find each other. Users always sign and broadcast their own transactions; ON never holds funds, never auto-executes, and never charges per-trade fees. This design removes two major costs of trading—custody risk and venue fees—while keeping price impact (slippage) to essentially zero by matching exact-for-exact trades.

This expanded edition is written for non-experts. It defines terms in plain language, shows how a swap feels in OMA, and explains the "why" behind each design choice.

Key Concepts (Read This First)

- AMM (Automated Market Maker): Curve-based pricing where your trade moves the price (slippage) and incurs protocol fees.
- Orderbook (centralized): Exchange lists of bids/asks; typically requires custody and charges taker/maker fees.
- Exact-for-exact: Both sides specify exact send/receive amounts—no curve, no partial fills.
- TTL (Time-to-Live): Short timer; if it expires before both legs broadcast, the offer auto-expires.
- Execution drift: Price move during the brief window while both parties finalize; minimized with short TTLs and near-simultaneous broadcast.
- UTXO: The spendable coin outputs in your wallet on Kaspa.

- Signature: Cryptographic proof the stated sender actually created the message; transports are untrusted, signatures are trusted.
- Transport: The channel carrying messages (WSS, Email, Kasia).
- Envelope: The standardized set of fields so all transports look the same to OMA/ON.

OMA is moving beyond a single-wallet, single-asset mindset. As soon as ON is complete, we prioritize OMA for Linux with two wallet connection modules on equal footing: Kastle/Kasia and Ledger (e.g., Ledger Nano S). Kaspa is our transport between ONs, not a limit on what can be traded—any supported wallet and asset can participate. This any-wallet, any-asset design sets the stage for a decentralized digital FOREX that challenges broker-driven gatekeeping and manipulation.

What OMA Is (and Why It Matters)

OMA is your control room for making and accepting offers. Think of it as a private listings tool for exact swaps: “I’ll send 1,000 MWHK and receive 1,000 KAS,” or “I’ll send 500 KAS and receive 5,000 TOKENX.” OMA lets you compose, review, and accept these offers with clear timers, checks, and confirmations.

Why users want OMA

- Clarity: Offers are exact, with no hidden curves or taker fees.
- Control: You sign and broadcast your own transaction—nobody can move your funds for you.
- Confidence: Short time-to-live (TTL) timers and a two-step commit process reduce last-second surprises (execution drift).
- Reach: Your node pools discovery with other ON operators, so more counterparties can see your intent.

What OMA does not do

- No custody: OMA and ON never hold your coins or tokens.
- No auto-trading: OMA does not push buttons for you; you decide when to finalize.
- No advice: OMA shows offers and tools; you choose whether to act.

OMA Charting & Click-to-Trade (Non-Custodial)

Result: fast, intuitive execution with full user control. The chart is for discovery and timing; signatures and broadcasts are performed by the user’s wallet, preserving our zero-custody design.

OMA will embed a TradingView-powered, forex-style charting interface. Users can initiate trades directly from the chart by clicking Accept/Send/Trade buttons. Execution remains non-custodial: OMA only passes the prepared transaction to the connected wallet module (e.g., Kastle/Kasia or Ledger). The wallet prompts the user to review and sign; no auto-sending, no custody.

A Day in OMA: The User Journey

1) See opportunities

OMA shows an Offers feed—simple cards with what's being offered and requested, the timer remaining (TTL), and who can accept (e.g., open to all or to a specific counterparty).

2) Make an offer

- You choose what you'll send and what you want to receive (exact amounts).
- You set a TTL (for example, 60 seconds). If no one matches in time, the offer expires cleanly.
- OMA creates a signed message describing your offer and sends it via your ON node to the broader pool.

3) Get matched

If someone accepts, both sides enter a short Reserve phase. Each side locally "holds" the inputs they plan to spend so those coins aren't accidentally double-used.

4) Commit and finalize

- OMA prompts you to confirm. When you tap Confirm, your wallet signs and broadcasts your leg.
- The counterparty does the same. With both legs broadcast, the swap is complete.

If the other side fails to broadcast in time, OMA releases your local hold and shows what happened. No funds are trapped because there was no escrow.

The Problem We Solve (Explained)

What is an AMM and why can it be costly?

Automated Market Makers (AMMs) are popular decentralized exchanges that use a formula (a bonding curve) to set prices. Trading against a curve means your trade itself moves the price against you—the larger the trade, the worse the price impact (slippage). AMMs also charge protocol fees on each trade. For many retail and mid-size swaps, this combination of curve slippage and fees dominates the total cost of execution.

What about centralized orderbooks?

Centralized exchanges match buyers and sellers, but usually hold customer funds (custody). Custody concentrates risk and may trigger licensing and compliance obligations. They also charge taker/maker fees and can impose withdrawal delays.

Why Kaspa needs a third path

Kaspa's fast, parallelized UTXO layer is ideal for direct settlement between users. What was missing is a simple, wallet-agnostic way to find matches and coordinate timing—that's ON (the messenger) and OMA (the controller). Together, they keep custody with the user and fees at zero while increasing the chance that two peers find each other quickly.

ON + OMA at a Glance

Roles

- OMA (Offer Maker Analyzer): The app you use to draft, publish, view, accept, or cancel offers. It manages timers, local holds, and confirmations.
- ON (Oracle-Node): A small server you or a trusted operator run. It forwards messages between participants, checks basic message validity, and provides status. It never takes custody.

Transports (how messages travel)

Messages can travel over several channels: secure WebSockets (WSS), standard email, and Kasia (a direct, end-to-end messaging tool). OMA doesn't care which one is used; ON normalizes everything so OMA sees one consistent feed.

Non-custodial by design

Every action stops at the edge of your wallet. Only you can sign and broadcast a transaction. This keeps funds in your control and reduces platform risk.

Use Cases & Potential Pairs

OMA and ON are designed to be wallet-agnostic and asset-agnostic. Beyond KAS↔MWHK, Phase 2 targets cross-asset pairs that matter to everyday users and businesses.

- BTC ↔ USDC (exact-for-exact, user-signed, zero venue fees)
- BTC ↔ EURC (decentralized digital FOREX across dollar and euro stablecoins)
- KAS ↔ USDC (on/off-ramps between KRC-20 ecosystem and dollar-pegged stablecoins)
- USDC ↔ EURC (stable-to-stable swaps as a low-friction FX alternative)
- Token X ↔ Token Y (any supported token via adapters; exact amounts, short TTLs)

How Messages Work (Plain English)

All transports are turned into a single, compact message format so OMA and ON can speak one language. Think of it like a standardized postcard with a few fixed fields.

Message fields you'll hear about

Field	What it means (plain English)	Example value
id	A unique number so we can tell messages apart.	b8b7-661-...
ts	Time the message was created (UTC).	2025-10-24T20:31:59Z
type	What kind of message this is.	offer, receipt, notice
from	The sender's public address or identity.	kaspa:qpx...
to	Who it's intended for (can be 'all' for broadcast).	all / kaspa:qpm...
sig	A digital signature proving the sender actually signed it.	MEQCIF...
ttl	How long it should be considered valid.	60s
topic	Which stream this belongs to.	onc.offers
traceId	An optional tag to correlate events.	abc-123
body	The human-meaningful content.	{"send": "1000 KAS", "receive": "1000 MWHK"}

These fields are signed together so they can't be altered in transit. If anything changes, the signature no longer matches and OMA/ON can discard it.

What you actually see in OMA

- A clean offer card: “Send 1,000 KAS ↔ Receive 1,000 MWHK” with a countdown timer.
- A simple Accept button and a Details panel (who posted it, TTL, and any limits).
- Clear outcomes: Accepted, Finalized, Expired, or Canceled.

Timing & “Execution Drift” (Explained)

Because there’s no escrow, both sides must act within a small window. ‘Execution drift’ is what can happen if the market moves during that window. OMA and ON minimize drift by:

- Using short TTLs (e.g., 20–60 seconds).
- Reserving inputs locally so your wallet doesn’t double-spend those coins by accident.
- Prompting both sides to broadcast nearly simultaneously (target ≤ 750 ms apart; never more than 2 seconds).

Security & Privacy (Why This Is Good)

Non-custodial = fewer single points of failure

If a platform never holds your funds, your risk is limited to your own wallet and keys. There’s nothing on ON to seize or misappropriate because nothing is held there.

Signatures and idempotency = authenticity without trust

Every message is signed, and each id is unique. If a duplicate or modified message appears, ON can ignore it. This reduces spam and protects against replay.

Transport-agnostic but verified

Email may be plaintext and WSS is encrypted, but authenticity in our system comes from the signature, not the transport. Kasia provides end-to-end encryption for those who want direct messaging; OMA treats it the same once verified.

Compliance posture (non-legal)

- No custody and no execution on behalf of users.
- Subscription revenue for operators instead of per-trade fees.
- Operators should consult counsel about communications, privacy, and data-retention policies in their jurisdiction.

Operator View & Economics

An ON operator runs a small server (for example, a Raspberry Pi 5) that connects to peers and forwards messages. Costs are modest, and revenue comes from subscriptions to a high-quality, policy-filtered feed. Because there are no per-trade fees, incentives remain aligned with users.

- Estimated OPEX: ~ $30/month.
- Sustainable pricing: $10–$25/month per subscriber can reach break-even with just a handful of users.
- Pooling across operators increases the variety and freshness of offers (better discovery → happier users).

Architecture (Readable)

Clean separation of concerns

- Transports (WSS, Email, Kasia): simple adapters that send/receive messages.
- Analyzer: parses, validates, de-duplicates, applies policy, logs.
- OMA client: presents offers, manages timers and holds, asks you to sign when ready.

Where files live (for operators)

ON converts all transports into normalized ‘inboxes’ and ‘outboxes’ on disk. This makes behavior easy to test and audit without special tools.

- Analyzer inboxes: ~/ON_data/analyzer_inbox/mail/new/, ~/ON_data/analyzer_inbox/wss/new/, ~/ON_data/analyzer_inbox/kasia/new/
- Egress queues: ~/ON_data/egress/mail/send.*.ndjson, ~/ON_data/egress/wss/send.*.ndjson, ~/ON_data/egress/kasia/send.*.ndjson
- Logs & counters: ~/ON_data/logs/*.log, plus counters.jsonl for simple telemetry.

Frozen channels

- onc.offers — broadcast of available offers
- onc.status — status and heartbeat messages
- onc.receipts — acknowledgments and outcomes

Current Status

- Module-1 (Email): IMAP in, SMTP out; owner tests pass.
- Module-2 (WSS): live for subscribers; multicast/unicast verified; systemd unit in place.

- Module-3 (Kasia): UI online at :3000; moving from a headless sender to a clean file-queue bridge for broadcast inbox/outbox parity.
- Module-4 (Analyzer I/O): v0.1 plan frozen; unifies inboxes/outboxes to prove end-to-end flow without complex policy.
- Tooling pins: Node 20.19.5 / npm 10.8.2; rusty-kaspa Web SDK v1.0.1-beta1; kaspa_bg.wasm checksum pinned.

Roadmap & Timeline (Extended)

We advance in measured steps. Each change lands through a numbered Addendum with explicit tests and rollbacks.

Q4 2025 — Transport Parity & Minimal Analyzer

1. Finish Kasia broadcast inbox mirror and outbox runner; remove legacy headless typist.
2. Ship Analyzer I/O v0.1 with health counters and a tiny Operator Dashboard.
3. Publish deterministic message fixtures for testing across transports.

- Post-ON Completion Sprint: OMA Linux wallet adapters with two equal modules — Kastle/Kasia and Ledger (e.g., Ledger Nano S); user-signed, non-custodial, exact-for-exact across KAS, BTC, USDC/EURC, and KRC-20 tokens.

Q1 2026 — OMA Feature Pass & Pooling

4. Release OMA-Mobile (iOS) with WSS transport, watchlists, accept/cancel flows, and accessibility improvements.
5. Federated ON↔ON pooling across onc.offers with latency scoring, revenue sharing, and operator quality metrics.
6. Optional reference-price display (no auto-routing; user remains in control).

Q2 2026 — Phase 2: Wallet-Agnostic Any-Asset Swaps

7. Add intent diffusion and anti-linkability options for high-sensitivity users.
8. Ship SDKs for iOS/Android/Desktop; provide sample apps and fixtures.
9. Direct 1:1 Kasia receive path with normalized inbox files and operator policies.

Q3 2026 — Growth & Operator Tools

10. Fleet observability, rate-limit policies, installer SOPs, and upgrade runbooks.
11. ESP egress adapter for compliant bulk mail (Module-1 parity).
12. Partner marketplace listings under NDA to grow discovery.

Goals & KPIs

- Finalize-within-TTL ≥ 95% excluding user cancels.
- p50 dual-broadcast gap ≤ 750 ms; never above 2 seconds.
- Zero schema drift (additive keys only).
- Operator uptime ≥ 99.5%; crash-safe hold pruning proven in tests.
- No per-trade venue fees—ever.

Competitive Landscape

AMMs offer always-on liquidity but impose slippage and fees. Centralized orderbooks can be deep but require custody and compliance. ON/OMA is a third way for exact, zero-fee swaps where users value control and predictability. For typical retail sizes, effective execution is competitive or better—and the experience is simpler to understand.

Governance & Change Control

Membership & Network Governance

ON (Oracle-Node) access to the pooled discovery network is managed with annual memberships. Each ON ships with a signed Membership Certificate (MC) tied to the device's public key. Licensed ONs participate in the Pro Pool; non-members operate in Solo Mode or can form independent collectives. OMA remains free to maximize user adoption.

Key elements

Issuer-signed broadcasts (over existing onc.status)

Deterministic relay rule

Operator runbook (summary)

- Recover: broadcast membership_snapshot (higher version wins); rotate issuer root if needed.
- Remove: broadcast membership_revoke; peers stop Pro Pool relays to that ON.
- Issue/Renew: deliver MC to ~/ON_config/membership/active.json and broadcast membership_add.
- Community topics remain best-effort and rate-limited; Pro Pool requires valid membership.
- Peers relay Pro Pool topics only if MC verifies (issuerSignature), the operator binds the message to the MC (mbrSig), and expTs > now.
- membership_snapshot — periodic, signed full list with a monotonic version to resolve partitions/replays.

- membership_revoke — removes an ON from the Pro Pool (reason + effective time).
- membership_add — introduces/renews an ON with its MC; takes effect at the stated time.
- Renewals & revocations: managed by issuer-signed broadcasts; boxes are never bricked.
- Solo Mode: without a valid MC, an ON still runs locally; operators may form their own collectives.
- Pooled access: Pro Pool relays only messages from ONs presenting a valid, unexpired MC.
- Membership Certificate (MC): memberId, devicePubKey, tier, issuedTs, expTs, issuerSignature.

We practice Behavior-Lock: single-path endpoints, frozen names, additive extensions, and human-readable proof-packs for every change. This keeps upgrades boring and safe—and makes audits straightforward.

Risks & Mitigations

- Upstream changes (SDKs, transports) → Pin versions, ship fixtures, test additively.
- Spam/flood on public channels → Rate limits, allowlists/blocks, signed messages, and TTLs.
- Operator misconfiguration → Clear runbooks, health counters, and deterministic tests.
- Regulatory ambiguity → Non-custodial design, subscriptions over per-trade fees, NDA-based partner diligence.

FAQ (For Newcomers)

Is this an exchange?

No. It is a coordination layer. You decide whether to sign and broadcast. The system never holds or moves your funds.

Can I lose funds if the other side vanishes?

No. There is no escrow. If the other party doesn’t broadcast in time, your wallet simply releases its local hold.

Do I need special hardware?

No, but operators often run ON on small, inexpensive hardware. Users can run OMA on iOS or desktop when available.

What does ‘exact-for-exact’ mean?

The amounts are fixed on both sides. You know exactly how much you send and receive. There is no curve or partial fills.

Do I pay any fees to swap?

Only network fees to publish your transaction on Kaspa. There are no venue fees to ON/OMA.

Glossary

- AMM (Automated Market Maker): A protocol that sets prices using a formula. Trades move the price and incur fees.
- Orderbook: A list of buy/sell orders. Centralized versions typically hold user funds (custody).
- Exact-for-exact swap: A trade where both sides specify exact amounts, with no curve or partial fills.
- TTL (Time-to-Live): A short timer after which an offer auto-expires if not completed.
- Execution drift: The small window where prices can move while both sides are finalizing.
- UTXO: 'Unspent Transaction Output'—the coins your wallet can spend on Kaspa.
- Signature: A cryptographic proof that a message really came from the stated sender.
- Transport: The channel used to carry messages (WSS, email, Kasia).
- Envelope: The standardized set of fields all messages use so they can be handled consistently.

Appendix — Envelope Extensions for Membership (Additive)

To support membership without changing transports or ports, ON adds optional, backward-compatible fields to the normalized envelope. These extensions are additive and safe for older peers to ignore.

Optional envelope keys

Membership control messages (topic: onc.status)

- type = membership_snapshot — signed full table with monotonic version; resolves partitions/replays.
- type = membership_revoke — removes an ON from the Pro Pool; includes reason and effectiveAt.
- type = membership_add — adds/renews an ON; carries MC and effectiveAt.
- tier — human-readable tier label (e.g., Community, Pro, Enterprise).
- mbrSig — operator's signature binding the current message id to the MC's memberId/devicePubKey.
- mbr — base64(JSON of the Membership Certificate).

Estimated Cost

- OMA: $0 — OMA is a free-to-use app anyone can download and use.
- ON: TBD — Operator pricing for the Oracle-Node subscription is under evaluation.