ASTRYUM · PRODUCT DECK · SEPTEMBER 2026

Exchange

Yield for the clients of a custodial exchange — with the structure of the exchange on the ledger. The client enters with a tag and leaves with Face ID. Their shares are theirs from the first block.

BUILT infrastructure on mainnet · rehearsed with the founders' accounts · not open to third parties yet

THE PROBLEM

Exchanges have lost their clients' assets. The client had no way to know, and no way out.

Who holds what lives in the operator's books

Attribution of client balances is a database. When the operator fails — or lies — the ledger has nothing to say.

Yield means leaving the exchange

A client who wants their XRP to work must learn wallets, gas and bridges, or trust the operator's in-house product with no rules the client can read.

Exit is a permission

Withdrawals can be paused, gated, delayed. The client's claim is a promise, not a property.

Enforcement by code was born from this: put the exchange's *structure* on the ledger, give the client shares in their own name, and make the exit something nobody can block.

WHAT IT IS

Astryum is not an exchange. It is the structure of one.

What the company creates — with its own keys

- A **root of authority** on XRPL with the credentials to serve clients (custody sector + company identity) and its constitution anchored.
- An **omnibus** it names by a designation credential and controls with its own key.
- A **register of the clients it approves**, box by box.
- The **potes on Flare** where its clients' capital works — the same cage and pote as a managed vault.

What Astryum never does

- ✕ Sign. Hold anyone's funds. Hold any omnibus.
- ✕ Approve a client, or administer any operator's KYC register.
- ✕ Decide who may be an exchange: a mechanical credential check, zero discretion.
- ✕ Sign for third parties: the demo key only opens its own omnibus; any other tenant signs from its own Xaman or its own backend.

Astryum prepares. The exchange signs. Built on the same kernel as the non-custodial products, replicated: the exchange **adopts** the infrastructure, it does not consume an API.

TWO ACCOUNTS, ALWAYS

K1 governs. K2 holds the till. A credential on the ledger binds them.

K1 — the root (council)
New, virgin, accredited: custody-sector + company credentials with expiry, accepted by K1. Constitution in its DID. Never a hot key.

OMNIBUS designation credential
K1 issues · K2 accepts
→

K2 — the omnibus (the till)
The exchange's own operating key, bounded by code. Receives client deposits by tag. Signs the memo instructions and the payouts.

rail A · rail B
→

Cage + potes on Flare
Born from K1 by council order. Same contracts as managed vaults. Shares in the clients' names.

Why two accounts

One memo instruction in flight per account; the credential never on a hot key; one Personal Account per account; a separation of duties an auditor can read.

Why a credential, not a config

The hierarchy is a ledger object. Verification is **relational** — issuer == the root of *this* structure — never a global allow-list. A short expiry means re-appointment.

Revoking it

Beheads the till: no new put-to-work. It never touches a client's exit from the pote.

THE CLIENT'S PATH

A tag to enter. Face ID to leave. No wallet, no reserves, no gas, no fees.

1 · Client deposits
XRP to K2 with their DestinationTag — the standard exchange practice. No XRPL account needed.

omnibus watcher detects →

2 · K2 signs the memo instruction
With its own key, in its own infrastructure: [approve, deposit → the client's passkey account]. A mint with a DestinationTag is refused by construction.

FDC attests · executor delivers →

3 · FXRP minted, deposited in the pote
The shares are born in the name of the client's PasskeyAccount — a P-256 account on Flare (RIP-7212), counterfactual.

whenever they want →

4 · Exit by Face ID
The client's passkey signs the redeem; a relayer pays the gas and cannot decide. No credential — revoked, expired or missing — can block it.

or →

4b · XRP payout
From the omnibus to the client's registered wallet — signed by the exchange, inside caps and typed refusals.

The client sees: *your exchange, your tag, your withdrawal wallet, your position*. Never a chain, a bridge or a gas token. Cost to the client: zero.

WHO CUSTODIES WHAT — SAID PRECISELY

Two kinds of exit, and they differ in who holds the key.

ASSET	WHERE	WHO CONTROLS IT	WHO CAN BLOCK THE EXIT
The client's XRP before it goes to work	the omnibus (K2) on XRPL	**the exchange**, with its key — the nature of the product, as in any exchange	the exchange, inside its own caps and refusals; Astryum cannot
The client's shares in the pote	the pote on Flare, in the client's PasskeyAccount	**the client** (Face ID)	**nobody** — not the exchange, not Astryum, not a revoked credential
The tag ↔ client attribution	the exchange's books + a KYC-per-box record on the ledger	the exchange	—
The exchange's governance (new put-to-work, new potes)	K1's credentials on XRPL	the issuers (expiry) and K1	a lapsed credential freezes *entries*, never exits

Close the entry, never the exit — the rule for both products. The shares are incongelable: any block a regulated operator must apply lives in its custody layer and its books, never on the ledger.

THE BOUNDED KEY

The operator’s key is limited by code, not by policy. The DENIED is the proof.

The signer refuses, by construction

- ✕ Signing from any other account or any other run.
- ✕ Paying anywhere but the FAssets Core Vault (memo instruction) or a client’s registered own wallet.
- ✕ Minting shares to anyone who is not a client of the run.
- ✕ A mint carrying a DestinationTag.
- ✕ Anything above the per-transaction and daily caps (entries only — never a client’s exit).

Before every signature

- ✓ Twelve typed refusal codes on the key’s door.
- ✓ Double evaluation: on the intended payment, then on the real bytes.
- ✓ Receipts: a receipt is a hash plus a promise — the backend never marks anything as done by itself; the verifier reads only from chain.
- ✓ The order that exceeds the limit reverts *before* moving, and the refusal is archived as evidence.

Whose key signs is answerable on chain: master key in cold storage, a regular key operating, revocable in one transaction — and tomorrow a TEE in that same seat.

KYC PER BOX — WHAT IT IS AND WHAT IT IS NOT

A notarized record of the exchange's own process. Not consent. Not enforced by the ledger.

What it is

- The root (K1) issues a credential `KYC—<tag>` whose **subject is the omnibus**; the omnibus accepts it. Two signatures of the exchange; the client signs nothing and can operate with only a passkey.
- Public, dated, with expiry: a record that the exchange ran its process for that box.
- Mandatory on every **entry** — deposit instructions, put-to-work, desk payments. On by default.

What it is not — and we say it first

- ✕ Not enforced by the ledger: XRPL conditions nothing on a DestinationTag. **The backend enforces it**, re-reading validity on every entry. Calling it anything else would be lying.
- ✕ Not the client's consent, not a portable credential.
- ✕ Never on an exit. An unreadable ledger is a 503, never a refusal.

The consensus-enforced alternative (client wallet as subject + DepositAuth on the omnibus) would require every client to sign on XRPL and would block the FAssets agent's returns; the Flare-side registry that reverts per client (`ExchangeKycRegistry`) is written and, by decision, not deployed.

THE OPERATOR'S PATH

An exchange is born by stations, from its own Xaman.

1 · Accredited root
K1: credentials accepted on the ledger

→

2 · Constitution
DIDSet on K1

→

3 · Cage
born by memo instruction; AUTHORITY = K1's bridge

→

4 · Pote
by council order; venues from the registry

→

5 · KYC door
per-box record, backend-enforced

→

6 · Designation
K1 issues OMNIBUS, K2 accepts: two signatures

Then, the desk

Multi-client: deposits by tag, put-to-work orders, payouts, caps per transaction and per day, the twelve refusals, receipts verified on chain, and a guided rehearsal the moment the exchange is born. "Entering is not creating": the front page always shows what this is and what it is not.

Two commercial lanes

ADOPT: a new, dedicated till. **CONNECT**: the exchange's already-live omnibus, bound by designation — zero re-onboarding of its clients. CONNECT is not a product lane yet: tag collisions, tenant boundary, cursor-less scanning and a nonce shared with the exchange's own traffic are open, and we never demo it against a real third-party account until they are closed.

LEGAL VEHICLE

A company that custodies for clients — an enterprise account with a custody credential.

The link lives on K1

- One root per legal vehicle; the board can be K1's SignerList — then the multisigned acceptance of a credential *is* the board's resolution.
- Credentials the law already requires, as XLS-70 objects: **custody-sector licence + company identity**, issued by the register, the regulator's attestation or a trust-service provider. Expiry is the revocation.
- By-laws anchored (DIDSet). K1 is the vehicle's book of acts: hash + pointer, never content.
- The designation of K2 is the one credential that goes *down* the tree — designation, never compliance. Everything else obeys by binding.

The credential is the account type

Anyone holding valid exchange credentials on their root creates *their* exchange self-service, approves *their* clients in *their* register. Astryum does not operate, does not decide who may be an exchange (a mechanical check), and holds no omnibus. The instance carries the operator's brand, never ours.

Copy rule, non-negotiable: never "licensed", "regulated" or the name of any regulation. The rehearsal's sector credentials are demo issuance with the register's link as URI. In public: *a credential verified on the ledger*.

STATE, PRECISELY

The structure is on mainnet. The client circuit is rehearsed. Nobody but us has run it.

LIVE on mainnet

- K1 with credentials and constitution; K2 designated by OMNIBUS credential.
- The exchange's cage on the v2 factory with two potes ("Exchange pote B", 72 h cooldown).
- PasskeyAccountFactory (P-256 accounts) since 23 Aug.

BUILT and rehearsed

- Desk, stations, autopilot, watcher, refusals, receipts, `verify` of a run and its proof file — green in tests.
- Client circuit rehearsed with the founders' own accounts in XRPL-only mode; real FAssets redemption fee read live (18 bps).
- Reads and `verify` are public; the interface is founders-only.

NEXT before a third party

- A tenant model (per-operator data, caps, keys).
- Orderly wind-down of a run with clients inside: declared return addresses + prepared payments + evidence.
- On-chain per-client registry (written, never deployed; `userGate = 0x0` on all live potes).
- The CONNECT lane's blockers.

Own audit, said first: some neighbouring exit lanes still "stay silent" or paint settled on a mined-but-ineffective transaction; they are the entry list of the next urgent wave. No external audit; small caps.

ECONOMICS

The client pays nothing. The operator pays its reserves. Astryum charges at cost or per act.

The client

Zero. No XRPL account, no reserves, no gas, no fees of their own: the tag channel is the total abstraction for the small user. Protocol fees (mint, executor, redemption) are read live and disclosed.

The operator

Its own reserves and carriers. Astryum advances the transport (attestation rounds) and recovers it at cost inside the order — built, inert until switched on. A fixed fee per act: creating potes beyond the free three, designating, joining the directory.

Never

A percentage of client capital or yield (the pote has no fee hook). Prepaid balances. A position bought in a directory. A token.

Tomorrow the principal source is the same as for managed vaults: a distribution fee paid by the venue on the capital that reaches it — the user deposits 100, the venue receives 100.

NEXT AND AFTER

From a rehearsed structure to a product an operator adopts — and then private, verifiable operations.

NEXT

- Tenant model and orderly wind-down (wave 2 of the V2 plan).
- The enterprise account as the operator's entry: KYB provider under contract, the anchor door switched on, the kit (API keys + remote MCP) so the operator integrates from its own UI.
- Goal compartments for personal users — the exchange's own infrastructure (boxes by tag, potes) with the user's key only.
- External audit of the contracts before any third-party capital.

LATER **— on Flare's confidential compute and protocol-managed wallets**

- Operations that must be private — trading, purchases, client-level actions — run inside a TEE with verifiable outputs, while the structure stays public.
- The omnibus key enters the TEE: the subordinate passes from *designated* to *captive*; the same root governs more wallets, multichain, without a new credential.
- Zero code for either today, by decision; an isolated, jurisdiction-gated module when it exists.

EVIDENCE AND WHAT WE DON'T CLAIM

Verify the structure; don't take the circuit on faith.

On chain

`AstryumCageFactory 0xE897fFef10F950Abc2d9c759107410F713e941D9`
`PasskeyAccountFactory 0xfa559cE04135835Cb8f7d3CFad2b8d14525ad932`
`v2 anchor rLcoFM9XF8CL5GoFyMACguhdnDtwkNYDn7`
`FAssets Core Vault rfkXSaCZKTg1EZzec2rLDyrWHxRVJdtVXj`

Code: `services/demoExchange/*`, `DemoExchangeSigner.ts`, `clientCredentialGate.ts`, `PasskeyAccount.sol`.

We do not claim

- ✕ That a third-party exchange has run on it.
- ✕ That the ledger enforces the KYC record — the backend does.
- ✕ That the client's XRP in the omnibus is anything but the exchange's custody.
- ✕ That the on-chain per-client registry is deployed.
- ✕ Any regulatory status, for us or for any operator.

Product: astryum.xyz/app/exchange · /proof · code: GitHub.

ASTRYUM EXCHANGE

The structure of an exchange, on the ledger. Shares in the client’s name. An exit nobody can block.

Astryum prepares. The exchange signs. The client leaves whenever they want.