TECHNICAL PAPER · XRP LEDGER + FLARE

Astryum, explained whole

What exists today on the XRP Ledger and Flare, how the infrastructure works from the inside, each product, what comes next, and what comes after — with every claim labelled and every number dated.

Written 15 September 2026 **Source Tag** 2607090002

LIVE on mainnet today BUILT code with tests; behind a wrapper or not yet exercised on mainnet

NEXT the V2 plan, decided, not coded LATER needs primitives not yet on mainnet, or legal work

---

> **15 September 2026.** This is the master document: what Astryum is, how the infrastructure works on the XRP Ledger and on Flare, each product from the inside, what comes next, and what comes after. Everything here was checked against the repository, the deployment artifacts and live reads of both networks on 15 September; where something is not verified, it says so.
>
> Internal sources (Spanish): the end-to-end thesis (2 September), the system map and the V2 plan (15 September), and the repository canon `INVARIANTS.md` · `ARCHITECTURE.md` · `DECISIONS.md`. If this paper and that canon ever disagree, the canon wins.

How to read this

Three planes, never mixed. Every claim in this document carries one of these labels:

LABEL	MEANING
LIVE	Running on mainnet today (XRPL Mainnet + Flare, chain 14). Checkable on an explorer without asking us.
BUILT	Code with tests in the repository, behind a founders-only wrapper or a switch, or not yet exercised on mainnet.
NEXT	The V2 plan: decided and sequenced, not coded yet — on purpose.
LATER	Depends on primitives that are not on mainnet yet (Flare protocol-managed wallets and confidential compute; XRPL permission delegation), or on legal work that is not code.

Every number carries the time it was read. What we do not claim is in §13.

1. Astryum in one page

What it is. Astryum is non-custodial mission control for XRP. A person connects the wallet they already have and sees all their capital in one place. From there they can put it to work, hold it together with other people under rules, or delegate its management — and in every case *they* sign, and the rules are enforced by the ledger and by a contract, never by us. We never hold a key.

The problem, in the words of the people it is for. *"My money is sitting still."* Today XRP has two options: stay idle, or be handed to someone who custodies it. *"We are several and one person has the keys."* Capital held by a family or a group has a design flaw that has nothing to do with crypto: one person can always move it alone. *"I want someone to manage it without being able to rob me."* Delegating the decision should not mean delegating possession. All three answers need primitives that only exist on a blockchain — a quorum the network enforces, credentials with expiry that anyone can check, contracts that refuse what does not fit the rules. The product consists of giving those capabilities without demanding the vocabulary.

The mechanism in one line. ***XRPL governs. Flare executes. The user signs.*** The XRP Ledger holds *who* has authority (a council with a quorum, a credential with an expiry, a designation from a root to a subordinate account), the constitution the group wrote, and the settlement of XRP itself. Flare holds the contracts that obey those orders or refuse them, and the venues where capital actually works. Between the two sits Flare's Data Connector acting as a notary: it attests that *this* XRPL account signed *this* commitment, and the contract on Flare checks the proof. There is no custodian in the path. What travels through the tunnel is the **order**, never the capital (except XRP itself, as FXRP, which is its legitimate representation on Flare).

The sentence that governs everything. *A person decides. The code decides what they can't.*

What is live on mainnet today (detail in §4):

- **Personal account** — one XRPL signature turns XRP into FXRP and executes DeFi on Flare, with no second wallet, no FLR for gas, no manual bridge; and the way back to XRP.
- **Legacy** — a real council of four with a quorum of three, master key switched off, constitution anchored on the ledger; four council orders executed end to end, the last one in 1 min 40 s.
- **Managed vaults** — a manager whose credential is verified on the ledger directs client capital inside a contract that cannot pay out principal to anyone; the full cycle (create, deposit, direct, recall, redeem) proven on mainnet.
- **Reinforced accounts** — the same ceremony as Legacy, applied to a personal wallet: no single key moves anything.
- **Credentials** as the access layer, **MoneyFlows** (rules that prepare and never sign), and the **/proof** page that verifies our claims in the visitor's own browser.

What comes next (§5–§7). One system: one web, one kernel, N account types, where the credential on the root account decides the type. The V2 plan builds the verification machine for venues, the enterprise

account and its integration kit, composition with the worst case sealed into the signature, and a third generation of the contract with an external audit.

What comes after (§8). Flare's protocol-managed wallets and confidential compute as the cage's remote arm; one XRPL root governing many personalized accounts across chains; the agent as a sixth account type that lives inside its own compartment; private-but-verifiable exchange operations; fiat in and out; a card that cannot freeze your balance.

What we do not claim. The contracts have no external audit, so our own policy is small and written down: our own capital, small amounts, caps in configuration. The product is not open to the public yet. As of 13 September Astryum has **5 active accounts and 289.51 XRP** attributed to its Source Tag: few signatures, each one a council ceremony or a movement of real capital. We are not chasing account counts; we are building the evidence that the mechanism works with real money.

---

2. The base every product shares: the XRPL → Flare tunnel

Every product in Astryum is a combination of two rails over one kernel. Understanding the two rails is understanding the whole system.

2.1 The axiom

The user's signature is always an XRPL transaction in their own wallet (Xaman). Everything else is composition and transport. Astryum prepares an **unsigned** transaction, shows all of it — including every fee — simulates it, hands it to the wallet, and stops. **Astryum never signs, never custodies, never executes with discretion.** This is not a policy; it is an absence of capability (§10.2).

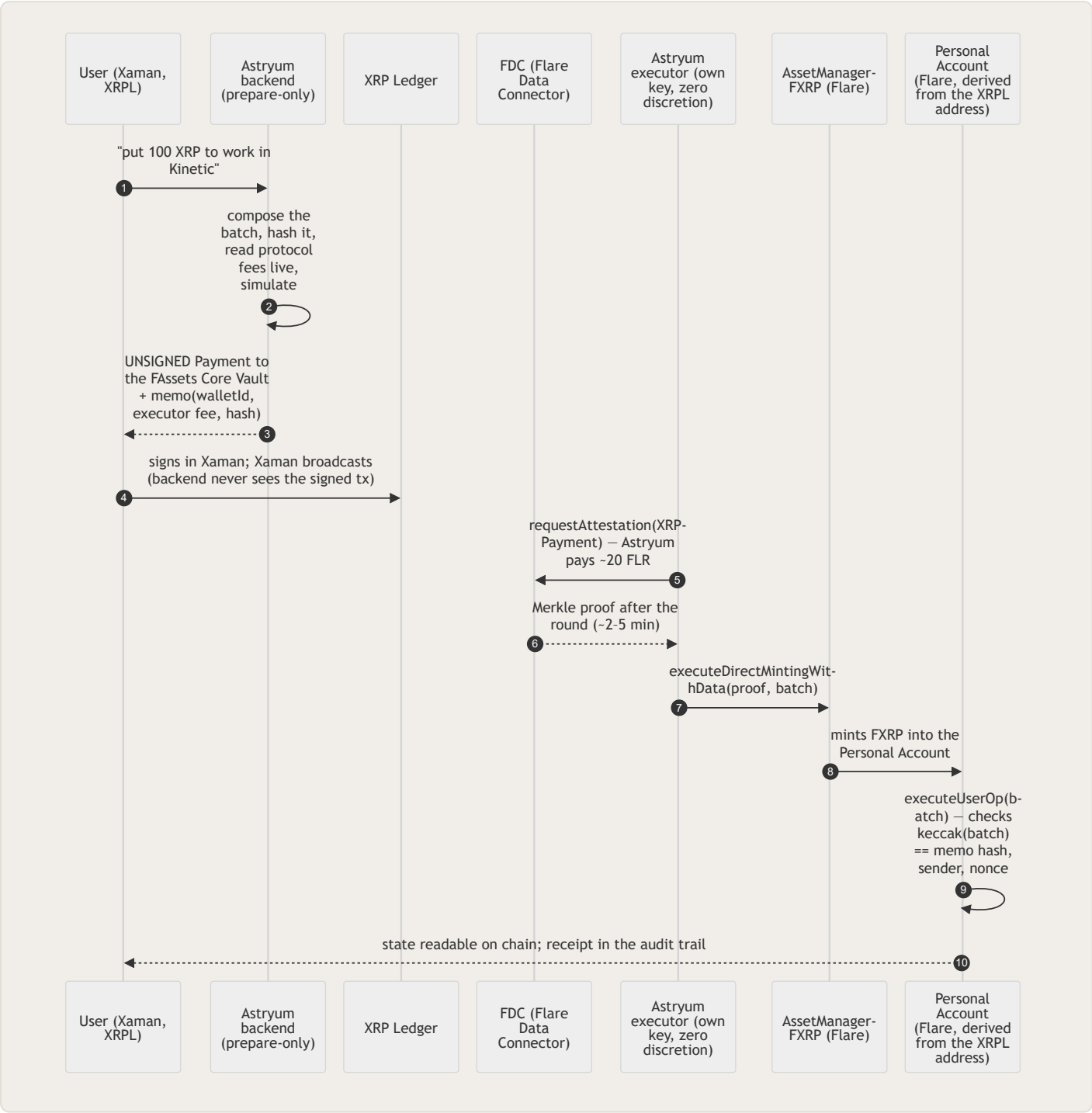
2.2 Rail A – the personal path (the memo instruction) — LIVE

This is how one XRPL signature becomes a sequence of actions on Flare without the user owning a Flare wallet or a single FLR.

1. **The Flare account is derived, not created.** Every XRPL address has a deterministic Personal Account on Flare, derived by Flare's `MasterAccountController` (CREATE2). There is no sign-up: the account exists the moment it is used, and it is deployed by its first instruction. In the product it is shown as "your Astryum account", folded inside the row of the XRPL wallet that governs it.
2. **Astryum composes the operation** the user asked for (for example: mint FXRP and supply it to Kinetic; or the carry FXRP → RLUSD with collateral in a Sentora vault, five transactions in one signature). The operation is a batch of calls the Personal Account will execute, packed as a standard account-abstraction user operation (EIP-4337 v0.7 shape), and hashed.

3. **The unsigned XRPL Payment.** Astryum returns a `Payment` from the user's own XRPL account to the **FAssets Core Vault** (`rfkXSaCZKTg1EZzec2rLDyrWHxRVJdtVXj`), carrying a **Memo** built with Flare's official Smart Accounts encoder: the instruction type, the wallet identifier, the executor fee, and the **hash of the user operation**. The `Account` field is pinned to the signer (a July incident taught us that without it the wallet signs from whichever account is active). A `LastLedgerSequence` bounds the signing window. Our Source Tag goes on the transaction. **No DestinationTag is ever set on this Payment: a tag would misroute a FAssets direct mint** — this is enforced in code and in the exchange's signing policy.
4. **Before the QR, the user sees everything.** The operation is simulated ("we tested this operation without signing it"), and the fees appear in human units: the FAssets minting fee (10 basis points, minimum 0.1 XRP) and the executor fee (0.2 XRP) — both **parameters of the protocol read live from the AssetManager**, never hardcoded, never ours. Fee disclosure is an invariant of the system: the object that is signed carries `disclosedToUser: true` and the policy engine rejects anything that does not.
5. **The user signs in Xaman.** Xaman broadcasts. The backend never sees the signed transaction.
6. **Flare's Data Connector attests the payment.** Astryum's executor — an operational key of our own, not the user's — requests an `XRPPayment` attestation from the FDC hub (paying the round's fee, ~20 FLR, in FLR we hold), waits for the round to finalize (typically 2–5 minutes), collects the Merkle proof, checks it against the transaction, memo, amount and success status, simulates, and delivers it to `AssetManager FXRP.executeDirectMintingWithData(proof, data)` .
7. **FAssets mints FXRP into the Personal Account, and in the same transaction the Personal Account executes exactly the batch whose hash was in the memo.** The contract checks `keccak256(data) == u serOpHash` , the sender and the nonce. The executor carries bytes the user already committed to; it cannot change a byte, an amount or a destination. **One signature, N actions.**
8. **The way back** inverts the path: redeeming FXRP to XRP (`redeemAmount` , or `redeemWithTag` when the destination is an exchange box) is itself a call inside a memo-authorized batch, so the XRPL signature authorizes it too. The redemption fee (18 basis points, read live) and the minimum (5 FXRP today) are shown before signing. Venue exit queues (Firelight, ~24 h) are counted with their real time.
9. **The receipt.** The FDC proof on chain plus the hand-off record form the `ExecutionReceipt` in the audit trail. The state is *read* from the chain, never assumed: no screen turns green before real settlement.

The same path, drawn.



Who pays what. The user pays the protocol's minting fee and executor fee in XRP, deducted from the gross amount and shown first. Astryum pays the FDC round and the gas of its two transport transactions in FLR. The economics close: the executor fee, converted at cost with Flare's native price oracle (FTSO), refuels the executor — proven on chain on 25 July.

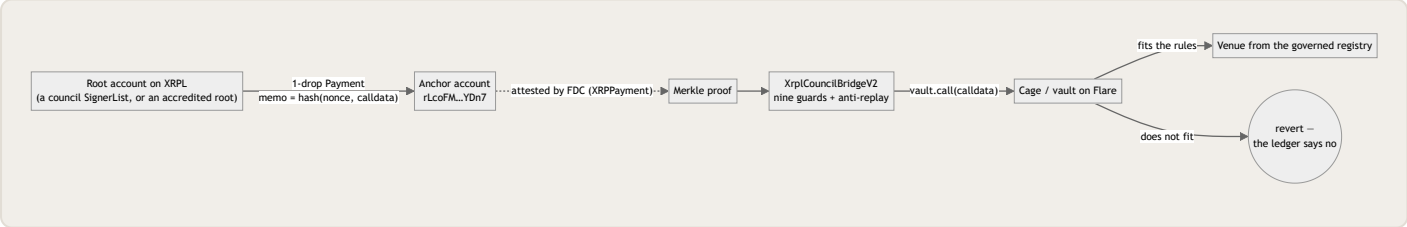
Two reserved variants exist in code and are deliberately not used in production: the *registered custom instruction* memo (needs a registration transaction with FLR gas that no clean non-custodial signer can pay) and the *inline user operation* memo (measured, never composed). The catalogue of memo routes is therefore finite and explicit; we document that as a regulatory virtue, not a limitation.

2.3 Rail B — the council order — LIVE

This is how a group of people, or an accredited root account, governs a contract on Flare without anyone holding a key to it.

1. **The root is an XRPL account.** For a family it is a council: a weighted `SignerList` with a quorum, master key disabled. For a manager or an exchange it is a fresh, virgin account whose credentials live on it (§4.6).
2. **The order is a 1-drop Payment** from the root to the **anchor account** (for the v2 stack: `rLcoFM9XF8CL5GoFyMACguhdnDtwkNYDn7`), with a Memo equal to the hash of `(nonce, calldata)`. What travels is the **commitment**, not the instruction: whoever transports the proof cannot alter it, because the destination contract compares the hash.
3. **FDC attests it**, as in Rail A. Astryum's relayer (same operational key as the executor) requests the attestation and delivers the proof. The relayer is a messenger: impotent by construction.
4. **The bridge on Flare (`XrplCouncilBridge`, v2 `XrplCouncilBridgeV2`) checks nine guards** before executing: the FDC proof verifies; the attestation type and source chain are right and the proof was requested for this bridge; the source address hash equals the council's (immutable at deployment); the XRPL transaction succeeded; the 32-byte memo equals the hash of the order; the transaction id has not been consumed; the nonce is the next one; and — new in v2 — the receiving address equals the anchor's hash (`WrongAnchor` otherwise), so that the anchor can later become a *gate* the ledger itself closes (§4.6). Then it calls the vault with exactly those bytes.
5. **If the order does not fit the rules, the contract reverts.** *The "no" is given by the ledger, not by Astryum.* The bridge is literally the `council` of the vault: no other address can order it.

Four such orders have been executed end to end on mainnet against the Legacy stack (bridge `nextNonce = 4`, read live on 15 September), and the v2 cage bridges show nonces of 3, 1 and 3 across the three cages born so far.

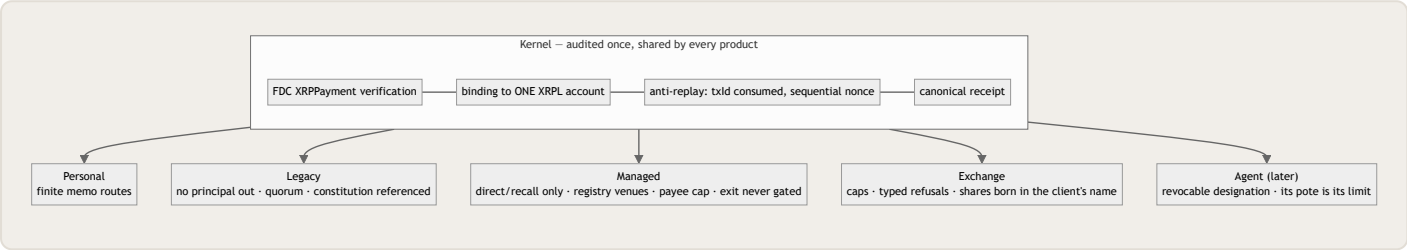


2.4 Kernel + rule modules

The two rails share a **kernel**: FDC `XRPPayment` verification, the binding to one XRPL account, anti-replay (consumed transaction ids, sequential nonce) and a canonical receipt. What changes per product is only the **rule module** the tunnel enforces:

TUNNEL	RULE MODULE
Personal	the finite catalogue of memo routes; personal limits

TUNNEL	RULE MODULE
Legacy	the family cage: principal can only go to work and come back; quorum of heirs; constitution referenced by every governing call
Managed (mandate)	the manager directs and recalls, never extracts; venues only from a governed registry; payee cap; client exit that no one can gate
Exchange / operator	per-transaction and daily caps, a dozen typed refusals, clients attributed by tag; shares born in the client's name
Agent (later)	a designation the root can revoke; the agent's compartment is its limit



The kernel is audited once; each new product is a small module. This is what "one kernel, N occupants" means in the code.

2.5 What the user signs — and what our own keys do

The user signs an XRPL transaction in Xaman: a Payment with a memo (Rail A or B), a `SignerListSet`, an `AccountSet` disabling the master key, a `DIDSet`, an `EscrowCreate`, a `CredentialAccept`. In a council, each member signs their part from their own phone, asynchronously (Tickets keep the account's sequence free); **the browser combines the signatures and broadcasts to public nodes**; the backend never touches a signature. The Xaman payload is created server-side with our API key and the transaction JSON goes to Xaman untouched.

Astryum's own keys sign only Astryum's own transactions, and it is an inventory, not a vague carve-out: the executor of Rail A; the relayer of council orders (same key); the keeper that pushes time-based escrows to their fixed destination (permissionless `EscrowFinish` / `EscrowCancel`); the notary that issues a credential after re-checking public facts; the service that refuels the executor from the anchor; and, only with its switch on, the omnibus key of the **simulated** exchange (§4.5). All of them carry bytes the user already committed to, pay gas, or push an outcome the ledger already fixed. None can change a destination or an amount. None holds user capital. A boot guard (`assertNoCustodialKeys`) enforces at start-up that no key capable of custodying user funds is configured.

Prepare-only is in the type system. The intent preparation engine cannot broadcast (a compile-time guard); the calldata builder marks every authorization `defibroRelays: false`; the regulated relay boundary records an `authorizationProof` — explicitly *not* a transaction hash — and stops.

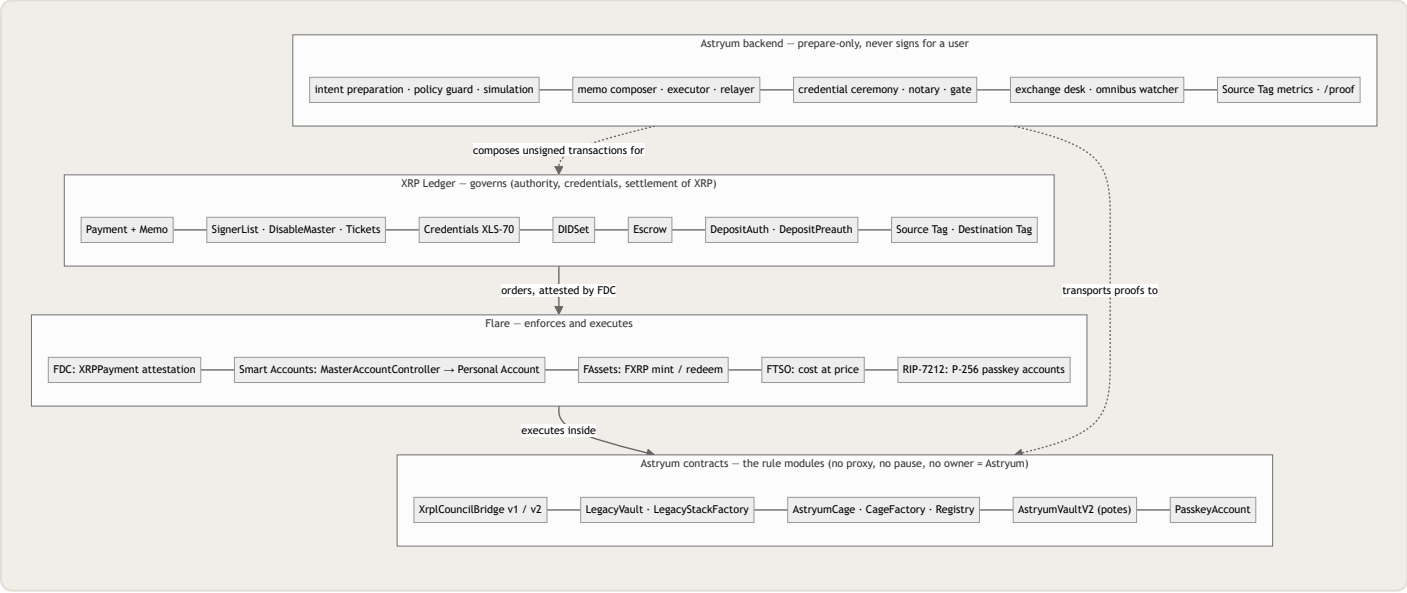
2.6 Why it takes both networks — and the capital rule

The uncomfortable question from anyone in the XRPL ecosystem is legitimate: *why isn't everything on XRPL?* Because the XRP Ledger is an excellent settlement and governance ledger and is not a general-purpose state machine. The rules this product needs — *"this capital can only go to this list of destinations, never to an arbitrary address; the manager may direct but never extract; the client leaves whenever they want"* — must live as a **contract**, not as a setting in an application. So: **XRPL governs** (who commands, with what title, with what quorum, and the settlement of XRP itself) and **Flare enforces and executes** (the contract that obeys or reverts, and the venues where capital works). The product exists only in the overlap. Remove XRPL and you lose the root of authority; remove Flare and you lose automatic enforcement.

The rule that separates us from most bridges: through the tunnel travels the *order*, never the capital. We never route capital that is not XRP through the XRP Ledger, and we never bridge something *to* XRPL first in order to move it later. Every asset works where it lives. (EVM capital, when it comes, executes directly on its chain via a partner; it is never detoured through XRPL.)

2.7 The infrastructure inventory — what we build products out of

Every product above and below is *defined* by which of these pieces it switches on. Nothing in the list is ours except the contracts and the backend; the primitives belong to the two networks, and that is the point: if Astryum disappears, the primitives and the contracts keep working.



PIECE	NETWORK	PERSONAL	REINFORCED	LEGACY	MANAGED	EXCHANGE
Payment + Memo (committed intent)	XRPL	•	•	•	•	•
SignerList · DisableMaster · Tickets	XRPL		•	•	optional	optional

PIECE	NETWORK	PERSONAL	REINFORCED	LEGACY	MANAGED	EXCHANGE
Credentials XLS-70 (compliance / designation)	XRPL				●	●
DIDSet (constitution)	XRPL			●	●	●
Escrow (scheduled payments)	XRPL	● (MoneyFlows)		●		
DepositAuth · DepositPreauth (built, not switched on)	XRPL				○	○
Source Tag / Destination Tag	XRPL	● / –	● / –	● / –	● / –	● / ● (client box)
FDC XRPPayment	Flare	●		●	●	●
Smart Accounts (Personal Account)	Flare	●	●	● (council's PA)	● (root's PA)	● (K1 / K2 PAs)
FAssets FXRP	Flare	●	●	●	●	●
FTSO	Flare	● (executor at cost)				
RIP-7212 passkeys	Flare					● (clients)
XrplCouncilBridge v1 · LegacyVault	Astryum			●		
Bridge v2 · Cage · Registry · Potes	Astryum				●	●
PasskeyAccount	Astryum					●

● used today · ○ built, waiting to be switched on · – not used by design.

3. Why XRPL is load-bearing: the counterfactual, primitive by primitive

The test worth applying: *remove the XRP Ledger — which user capability disappears?* Our answer, primitive by primitive. None is decorative; those without a consumer yet are marked as such.

PRIMITIVE	CAPABILITY THAT DISAPPEARS WITHOUT IT	STATE
Weighted SignerList + master key disabled	The council, the heirs, the reinforced account. "Three of four agreed" is a fact of the network, not a row in our database we could edit. The master key can be re-enabled only by the same quorum: cure is possible, the shortcut is not.	LIVE (real council 3-of-4 on mainnet since July)
Payment with Memo (the committed intent)	One signature in the wallet the user already has orders N actions on another chain, with no second wallet, no gas token, no manual bridge. What travels is the commitment; the transporter cannot change a byte.	LIVE (both rails)
Credentials (XLS-70)	Licence and identity as ledger objects with expiry, accepted by the subject, checkable by anyone without asking us. The credential *is* the account type. Without it, accreditation is a list of ours: trust inverts back to Astryum.	LIVE as the manager gate; the on-ledger door is BUILT (below)
Designation by credential (root → subordinate)	Hierarchy between accounts as a ledger object (a council names its omnibus). Revoking or letting it expire beheads the subordinate **without touching any client's exit**.	LIVE on mainnet for the exchange rehearsal
DepositAuth + DepositPreauth by credential	An account that only accepts what comes from a holder of a valid title — the door closed **by consensus itself**, with no code of ours in the middle. Without it, the door is a check on our server, i.e. a door that opens by switching our server off.	BUILT: builders, routes and API client exist; no screen calls them yet; not exercised on mainnet. The v2 bridge already verifies the anchor for the day it becomes this door.
DIDSet (the anchored constitution)	The rules the group wrote are anchored in the account's own record (hash + pointer). They cannot be rewritten quietly, and the contract references the DID data, not a loose transaction hash.	LIVE
Escrow	Scheduled payments to beneficiaries that settle themselves: finish and cancel are permissionless for time-based escrows, so no process of ours needs to be awake and holding keys.	LIVE (keeper pushes them; it is our own key, no Source Tag)
DestinationTag	The exchange client who participates **without an XRPL account, reserves or gas**: a tag to enter. (This is the *only* place a DestinationTag appears — as a client's box on an omnibus, the standard exchange practice — never on a memo instruction.)	BUILT, rehearsed
SourceTag	Verifiable attribution without asking anyone for data. It goes in the transaction body and survives multisig: each signer counts. House rule that lowers our own numbers on purpose: no operational account of Astryum ever stamps the project tag (that would be self-attribution; it is codified with two explicit modes and fixed by tests).	LIVE
RLUSD	Settlement and collateral in the same currency as the institutional infrastructure that already exists — what makes us complementary rather than parallel to it.	configured as an admissible asset; the Ethereum lending lane exists and is not in the production allow-list

PRIMITIVE	CAPABILITY THAT DISAPPEARS WITHOUT IT	STATE
Permissioned Domains (XLS-80)	(nothing yet: composed and tested, no route consumes the domain id)	BUILT, decorative — and we say so
Permission delegation (XLS-75)	(nothing yet: the amendment is not active)	house rule: **not one line of code** until it activates on mainnet; and then it is *transport* of a delegation, never the security boundary — the limit is always the cage
Batch	(not on mainnet)	not used, not announced

The proof is not the assertion. All of the above can be checked without asking us: the `/proof` page verifies in the visitor's browser against the ledger and the contracts (§12).

---

4. The products today

Each product follows the same template: what the user lives, what happens underneath, which XRPL primitives and which Flare pieces carry it, who signs what, what Astryum never does, and its real state.

4.1 Personal account — your XRPL wallet commands your account on Flare — LIVE

What the user lives. Logs in with Xaman (the wallet is the identity). Sees their capital in one map — XRPL, Flare and EVM positions in read-only. Puts XRP to work on Flare with one signature: no FLR, no EVM wallet, no visible bridge. Leaves from the same screen: "Convert to XRP". Configurable things are *named options*, never technical fields ("Prudent · Balanced · At the limit"); the user is asked only what only they know: how much, and where.

Underneath. Rail A exactly as in §2.2. Venues live in production today, each behind its own scanner and its own switch, listed in code (not in an environment variable — a lesson from an incident on 14 September, §10.3): Kinetic (lend, and the carry that borrows dollars against FXRP), Firelight (stXRP staking), earnXRP (Clearstar), Monarq, and FLR delegation to the FTSO. Adapters for SparkDEX, Ēnosys and Sceptre exist behind their flags. The carry FXRP → RLUSD with collateral in a Sentora vault composes five transactions in one signature.

XRPL primitives: Payment with Memo and Source Tag; multisig valid on the same rail (proven). It does not use Escrow, DEX, AMM, NFT, MPT or Batch. **Flare pieces:** Smart Accounts (`MasterAccountController` + Personal Account), FAssets (FXRP, direct minting and redemption), FDC (`XRPPayment`), FTSO (executor cost at price), the venues above. **Who signs:** the user, one XRPL Payment. (In the carry, optionally the repayment from MetaMask.) **What Astryum never does:** sign with a user key, broadcast a user transaction, decide the operation. The executor is transport with zero discretion; its cost is charged at

cost. **State:** proven on mainnet — mint, deposit, withdraw, repay; FXRP → XRP redemption end to end on 31 July; the full memo rail end to end on 22 August. The bridge of FXRP from Flare to Ethereum (LayerZero OFT) exists as verified calldata that the user signs from a Flare EOA; reaching it *from the XRPL signature* is a later step.

4.2 Reinforced account – a personal wallet where no single key moves anything – LIVE (same ceremony as Legacy)

What the user lives. Their personal account, with the dials turned up: two of three of *their own* keys (for example two phones and a backup) must sign, and the original single key is switched off. It stays a personal wallet — no constitution, no cage, no inheritance. It is the middle step of a spectrum: simple (1-of-1) → reinforced (2-of-3, master off) → council (3-of-4 plus constitution and cage).

Underneath. Exactly the ceremony Legacy proved on mainnet in July: `SignerListSet` with weights and quorum → a rehearsal in which every signer actually signs a multisig transaction (an `EscrowCreate`) so nobody closes a door they cannot open → `AccountSet{asfDisableMaster}`, with the irreversibility disclosed. The ledger does not distinguish "reinforced" from "council"; the owner marks it, and the health rules refuse to close the door without a rehearsed quorum with a margin of at least one signer.

Landmines we refuse, and say so: a RegularKey is an OR, never a second factor; a pure 2-of-2 is a permanent freezer if you lose one factor (default is 2-of-3); closing the master key requires the master's own single signature, reopening it is governed by the quorum — verified on xrpl.org.

State: the multisig coordinator and the ceremony have been live on mainnet since 14–15 July; the reinforced surface (a shorter ceremony that skips the constitution) shipped in September. Founder-facing today, waves later.

4.3 Legacy – a family governs by quorum – LIVE on mainnet

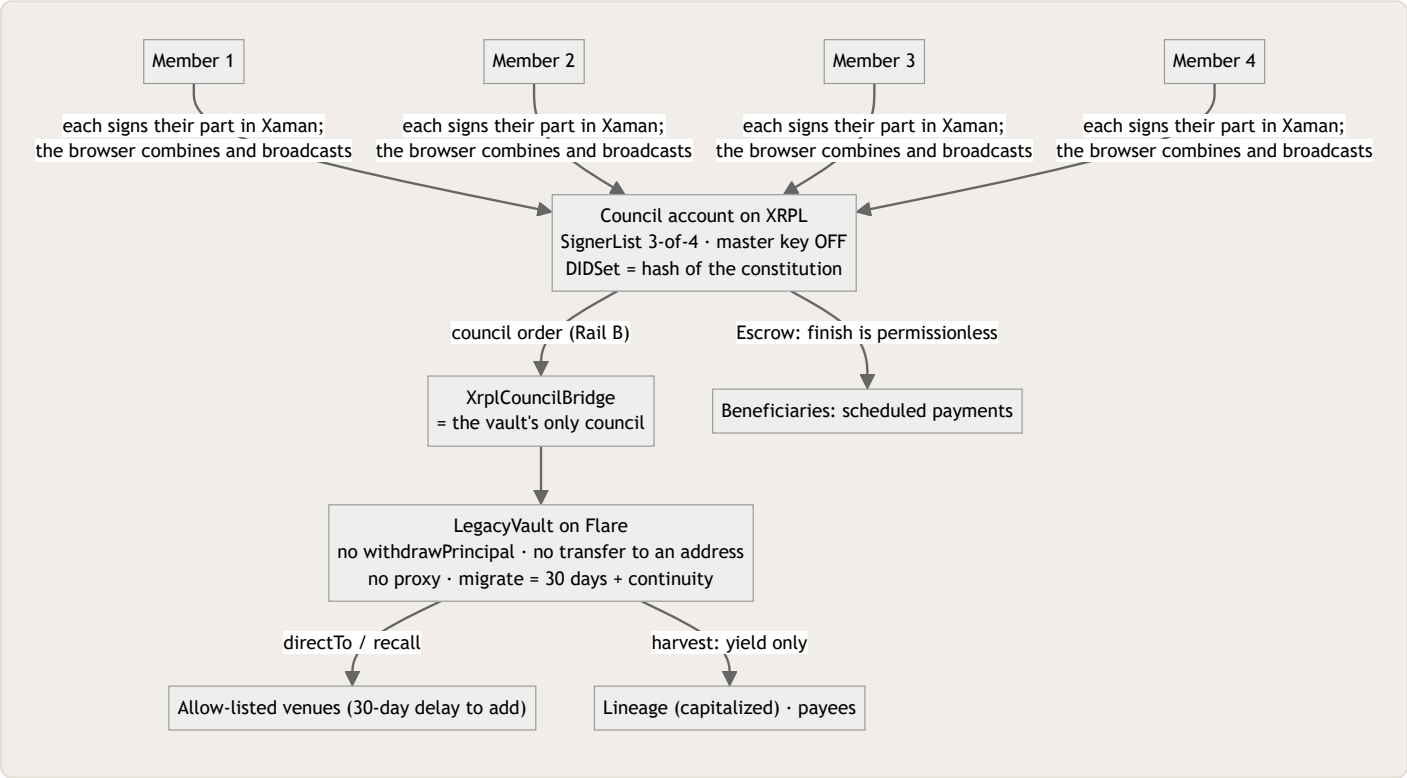
What the user lives. A council of real people (for example 3 of 4) holds the family's capital. They write the constitution in plain language — who signs, for what, within which limits. Astryum checks the plan while it is still reversible, anchors it on the ledger, and from then on every decision needs the quorum. Then the council closes the door: the original single key is switched off. Nobody moves the capital alone: not a member, not us. **The heirs do not wait for an event: they already are the quorum.** Scheduled payments to beneficiaries settle themselves.

Underneath.

1. `SignerListSet` (weights, quorum), `asfDisableMaster`, `DIDSet` with the constitution (SHA-256 of the text plus a URI; the vault references the DID data). The multisig coordinator fixes sequence and fee and never signs, combines or broadcasts.
2. An **asynchronous signing tray**: each member signs their part in Xaman when they can; the browser combines and broadcasts. Tickets prevent sequence blocking.
3. **The cage is born with one memo instruction** (Rail A) whose batch is `[LegacyStackFactory.create, FXRP.approve(predicted vault), LegacyVault.deposit]`; CREATE2 makes the vault's address predictable.

The factory accepts as creator **only the Personal Account of the council itself**: on a multisig-only account, producing that payment *is* the quorum. Nothing is born without an anchored constitution.

4. **Orders travel by Rail B** to `LegacyVault` through `XrplCouncilBridge`: a catalogue of twelve actions (direct capital to a venue, recall it, move, evacuate, propose or retire a venue with a 30-day delay, set the lineage fee, set payees, cede the director's seat temporarily, end the cession, update the constitution reference).
5. **What does not exist in the vault, by design:** no `withdrawPrincipal`, no transfer to an arbitrary address, no proxy, no upgrade, no pause. The only relocation is `migrate` to a successor, and it is the opposite of a back door: the council proposes it, it waits **30 days**, and it requires verified continuity of council, constitution and asset. Changing the rules means moving to a successor, not editing a setting. Only the *yield* leaves to people: `harvest` splits it between the lineage (capitalized, floor 10 % ceiling 40 %), a protocol fee capped hard at 10 % (default 0) and payees.



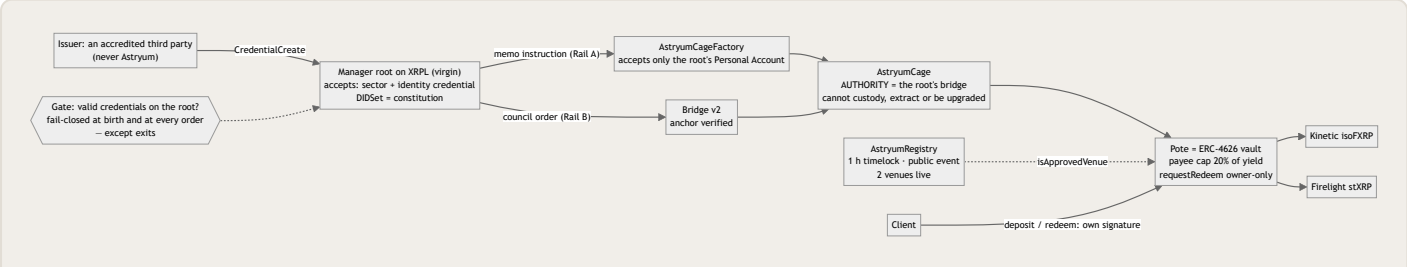
XRPL primitives: SignerList, DisableMaster, DIDSet, Tickets, Escrow, 1-drop Payment with memo, Source Tag surviving multisig. Batch: no. **Flare pieces:** `XrplCouncilBridge`, `LegacyVault`, `LegacyStackFactory` (one cage per Legacy), FDC, the relayer. **Who signs:** the quorum signs everything, in Xaman. Astryum composes and relays the FDC proof with its own account: the messenger is impotent. **State:** proven on mainnet on 29 July (real council, three signers, `directTo` Kinetic, `nextNonce` 0 → 1) and four orders end to end by 4 August (the last in 1 min 40 s). Live reads on 15 September: bridge `nextNonce = 4`; vault holding ≈ 4.70 FXRP in one venue, lineage fee 30 %. No external audit, so our own cap is small: 5 XRP per cage, in configuration. Designed and not built: the "release vessel", and signature ≙ person enforcement. When protocol-managed wallets exist, the family moves to the cage v2 (§8.1).

4.4 Managed vaults — a manager with a credential directs inside a cage — LIVE on mainnet

What the two users live. The **manager** opens their vault from their Xaman, picks venues from a list and directs the capital; they see a desk with orders. The **client** deposits and leaves whenever they want, without asking anyone; they see their position and an exit button. Neither sees a contract. The surface is called "Managed vaults", eyebrow "Run by a third party": a name names the thing, it does not push an act.

Underneath.

1. **The manager's root is a fresh, virgin XRPL account** (one council = one cage, forever). On it live its credentials: a sector credential plus an identity credential, each with an `Expiration` and accepted by the subject. The gate is configured as OR-groups (`AIFM|CASP` and `KYC|KYB`: *the licence of your sector, and your identity*), is **fail-closed**, and bites at birth and at every order: **without valid credentials on the root, the cage is not born.** In public we say only "a credential verified on the ledger": the credential *type* names the licence the manager states they hold; the issuer attests it; Astryum verifies the object on the ledger and never certifies anything.
2. The constitution of the vehicle is anchored with `DIDSet` on the root.
3. **Birth is a memo instruction (Rail A)** that deploys, through `AstryumCageFactory`, the cage (`AstryumCage`, with an immutable `AUTHORITY` = the council's own v2 bridge) and its first "pote" (`AstryumVaultV2`, an ERC-4626 vault). The factory accepts as deployer only the Personal Account of that root.
4. **Every manager order** (`create-pote`, `direct-to`, `recall`, `set-payees`, `cede`…) is a council order (Rail B). The cage only accepts venues from `AstryumRegistry` (a governed list with a timelock — 1 hour, immutable — and a public event; two venues live: Kinetic isoFXRP and Firelight stXRP), and the manager **can never extract**: the cage has no `receive`, no `fallback`, no `sweep`, no generic `call`, no function over shares. Capital only goes to work and comes back.
5. **The manager's cut is a payee on chain, and its ceiling is set by the cage**, not by the pote: the deployed cage carries `MAX_PAYEE_BPS_ALLOWED = 2000`, so no pote of it distributes more than **20 % of the yield** — never the principal — and succession cannot raise it. A pote asking for more than 20 % is born private with a permanent gate: whoever charges more cannot be open to the public.
6. **The client deposits with their signature and redeems with their signature**; `redeem` unwinds venues by itself if needed. `requestRedeem` is owner-only with no allowance path: *"the director is never approved" is not a rule, it is unrepresentable.* And an explicit invariant with a tripwire test: **no flow ever asks the client to approve the operator over their shares** — a single `approve` to the director would dissolve the cage without touching the contract.
7. The pote has **no protocol fee hook**: Astryum's fee to the client is zero, enforced by absence.



XRPL primitives: virgin root, Credentials XLS-70 (`CredentialCreate` / `CredentialAccept`, `Expiration`, `URI` to the public register or the attestation), `DIDSet`, memo Payment, 1-drop Payment with memo, optional SignerList. **Flare pieces:** `AstryumCageFactory`, `AstryumCage`, `AstryumVaultV2`, `AstryumRegistry`, `XrplCouncilBridgeV2` (anchor verified, `WrongAnchor`), `PoteDeployer` (exists because of the contract-size limit), FDC. **Who signs:** the manager signs in Xaman the birth, each order and the credential ceremonies. The client signs deposit and exit. **What Astryum never does:** sign, occupy the director's seat, issue the real licence (it only chooses which issuers it accepts, with a uniform check), ask the client to approve the operator. **State:** Foundry suite green, **168 of 168** test functions, of which **62 cover the cage v2** against real venues on a mainnet fork — including a theft reproduced in v1 and blocked in v2. Stack v2 deployed on mainnet (registry, factory, live anchor; 3–8 September). **Full cycle proven on mainnet**: create pote, deposit, direct and recall by council order, redeem. Live reads on 15 September: registry with 2 venues; factory with **3 cages and 4 potes** (all with zero supply today, consistent with everything having been redeemed); 3 free potes per cage, then a creation fee of 1 FXRP paid directly to the treasury in the birth transaction. The manager's desk and the directory are published; cage creation for non-founders is still behind the founders-only wrapper. **The catalogue of the pote is exhausted today:** a new venue must be a typed shape (ERC-4626, Compound-v2, or queued ERC-4626), on the same chain, with the same underlying asset — anything else needs a new audited branch of the contract, and we do not promise venues we cannot add.

4.5 Exchange — the structure of one, adopted by an operator — BUILT (infrastructure on mainnet; rehearsed; not open)

The framing, verbatim from the product page (15 September): *Astryum is not an exchange. It is the structure of one, built on the network, so that a company holding the credential to serve clients can create its own exchange here and operate it with them.* What the company creates: a root of authority on XRPL, an omnibus it names and controls with its own keys, a register of the clients it approves, and the potes on Flare where their capital works. What Astryum never does: sign, hold anyone's funds, approve a client.

What the operator lives. An exchange is born by stations from its Xaman: accredited root → anchored constitution → cage → pote → KYC door → designation of its omnibus with two signatures. Then it runs a multi-client desk with per-transaction and daily caps, twelve typed refusal codes, a double evaluation before signing (on the intended payment and on the real bytes), and receipts verified on chain. **The DENIED is the proof:** the order that exceeds the limit reverts before moving, and the refusal is archived as evidence.

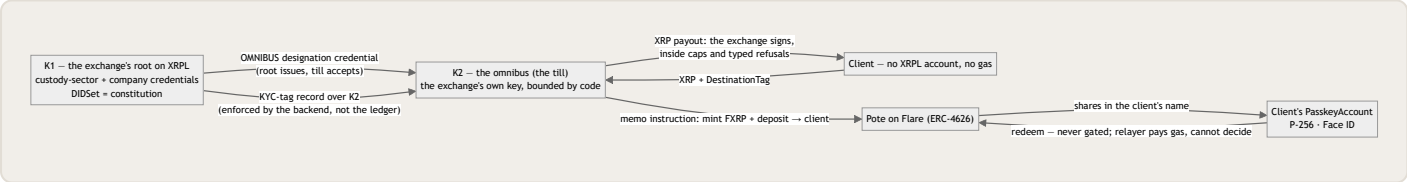
What the exchange's client lives. No XRPL account, no XRP for reserves, no gas: a DestinationTag to enter, Face ID to leave, and vault shares in their own name from the first block.

Underneath.

1. **Two XRPL accounts, always:** K1 (the council: a new accredited root, with the constitution in its DID) and K2 (the omnibus, the till). Reasons: one memo instruction in flight per account; the credential never on a hot key; one Personal Account per account; an auditable separation of duties.
2. **K1 names K2 with a designation credential** of type `OMNIBUS` (XLS-70: the root issues, the till accepts; a short `Expiration` means re-appointment). Verification is **relational** (issuer == the root of *this*

structure), never a global allow-list. Revoking it beheads the till without touching any client's exit.

3. **The client deposits XRP into K2 with their tag.** The omnibus watcher detects it. **K2 signs, with its own key and in its own infrastructure**, the memo instruction `[approve, deposit → the client's passkey account]` : the minted FXRP enters the pote and the shares are born in the name of the client's `PasskeyAccount` (a P-256 account on Flare via the RIP-7212 precompile, counterfactual, live on Flare since August).
4. **KYC per box.** The root issues a credential `KYC—<tag>` whose subject is the omnibus: a public, dated, expiring **notarized record of the exchange's own process**. Precision matters here: **the ledger does not enforce it** — XRPL conditions nothing on a DestinationTag — the backend does, re-reading its validity on every entry. It gates what comes in, never what goes out. It is not the client's consent and not a portable credential; calling it either would be lying.
5. **Exits.** Two kinds, and they differ in who custodies. The client's XRP in the omnibus is **custodied by the exchange** with the exchange's key — that is the nature of the product, exactly as in any exchange; the XRP payout to the client's registered wallet is a Payment signed by the omnibus, within the caps and refusals. The client's **shares in the pote** are theirs: the exit is signed by their passkey, a relayer pays the gas and cannot decide, and **no credential — revoked, expired or missing — can ever block it.** "Close the entry, never the exit" is the rule for both products.
6. **The signing key is bounded by code, not by policy.** The exchange's signer refuses by construction: signing from any other account or run, paying anywhere but the FAssets Core Vault or a registered client wallet, minting shares to anyone who is not a client of the run, a mint with a DestinationTag, and anything above the caps. In the demo Astryum plays the role of the exchange with that key (`Astryum prepares, the exchange signs`); any other tenant signs from its own Xaman or its own backend — the autopilot key only opens its own omnibus.



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

State: built and green in tests; the exchange's root (credentials and DID), designated omnibus, cage and two potes **exist on mainnet** (one of the three cages on the v2 factory carries "Exchange pote B", 72 h cooldown); the client circuit has been **rehearsed with the founders' own accounts** in XRPL-only mode. What is not yet true: an on-chain per-client registry (`ExchangeKycRegistry` was written and never deployed; all live potes have `userGate = 0x0`), a tenant model, an orderly wind-down of a run with clients inside (decided for the next wave: declared return addresses plus prepared payments plus evidence). The interface lives behind the founders-only wrapper and its own switch; reads and the `verify` of a run are public. No third-party exchange has run on it.

4.6 Credentials — the transversal access layer — LIVE as the gate; the on-ledger door BUILT, not switched on

Doctrine. The credential lives on the **root of authority**; subordinates (potes, Personal Accounts, passkey accounts) never carry credentials — their obedience is proven by a chain of immutable bindings (credential → root → bridge → cage → subordinates) that an auditor or a judge can walk. Two classes that never mix: **COMPLIANCE** (an accredited third party → the root) and **DESIGNATION** (the root → its non-captive subordinate). The check is at the entry: revoking freezes governance and **never the client's exit**. *The ledger carries the YES; the NO is never published* — a refusal written on a public chain would be a leak of personal data and a serious legal problem. And **the credential is the account type**: direct user (identity, if any), manager (sector + identity), exchange (custody sector + company identity), the exchange's client (a KYC record issued by *its* exchange).

How it works. The issuer signs `CredentialCreate` (type ≤ 32 characters with no personal data, `Expiration` **mandatory** — 1 to 366 days, default 180 — and a `URI` to the public register or the attestation); the subject signs `CredentialAccept`, and that accept is the consent. The verifier is issuer-agnostic, never turns green without the accepted flag, and treats an expired credential as absent; which issuers are accepted is configuration. The **notary robot** (`ManagerNotaryIssuer`) re-checks facts anyone can re-check — today the Coinbase EAS attestation on Base for the identity credential; the domain-set, domain↔toml binding and register entry for the sector credential — and issues with typed refusals; it never exercises judgment, and "could not read" is never "no". In the rehearsal, sector credentials are **demo issuance** with the register's link as `URI`: Astryum brings on chain what the register says, it never certifies. The corporate jewel, not yet composed: when the root is the `SignerList` of a company's board, a multisigned `CredentialAccept` **would be** the board's resolution, natively on the ledger.

The on-ledger door. `AccountSet{asfDepositAuth}` plus `DepositPreauth{AuthorizeCredentials}` on the anchor account: without a valid credential from an accredited issuer, the order dies in consensus, with no code of ours in the middle. Builders, two routes and the API client exist; no screen calls them yet, and the circuit has not been exercised on mainnet. This is the one technical change that would move Credentials from "meaningful" to "essential" in a strict reading, and it is first in line. Permissioned Domains (XLS-80) are composed and wait for their first consumer.

Where the gate bites today: cage creation, and every cage order except accepting a pote and the **exits** (`recall`, `evacuate`), which are never gated. The manager's desk reads only the identity and sector credentials whose subject is the account itself. The exchange's KYC-per-box gate bites on deposit instructions, put-to-work and desk payments — never on a withdrawal.

4.7 MoneyFlows — rules that prepare and never sign — LIVE

Standing orders and automations inside signed limits: XRPL Escrow fixes destination and dates (TTL ≤ 90 days; finish and cancel permissionless), instant revocation, pauses. **A rule never signs:** when it triggers, the automation engine creates a pending authorization session and brings the transaction to the wallet that owns the position for that person to approve. The price trigger has been real since 16 August (live oracle price against the rule's own reference; legacy rules without a reference refuse with an explicit reason rather than guess). The escrow keeper signs with an account of our own, never user capital, and never with the Source Tag.

4.8 Proof — the ledger's word, not ours — LIVE (read)

Every claim is checked **in the visitor's browser** against the chain: transactions, memos, credential objects, the state of registry, factory, cage and pote. The `verify` of an exchange run and its `proof.md` are built. Transactions of Astryum's operational keys are excluded from the Source Tag on purpose — a rule that lowers our own numbers and is shown.

4.9 Copilot and MCP — the agent compiles; a person signs — LIVE as copilot

The agent turns natural language into an intent; the person reviews and signs. **There is no tool to sign, execute or custody in its surface, and that absence is the product.** The assistant routes have no tools at all; a flow of AI origin cannot carry a token address (`ai_wrote_address` is a typed violation); the MCP server exposes 19 tools — reads, simulation, drafts and two `prepare_*` — and none that signs or submits. The only agent that already runs is the notary robot, and it only issues credentials after re-checking public facts. The MCP server is local (stdio) and not exposed over HTTP; per-user isolation is fixed before it is (§7, wave 0).

---

5. One system: one web, one kernel, N account types

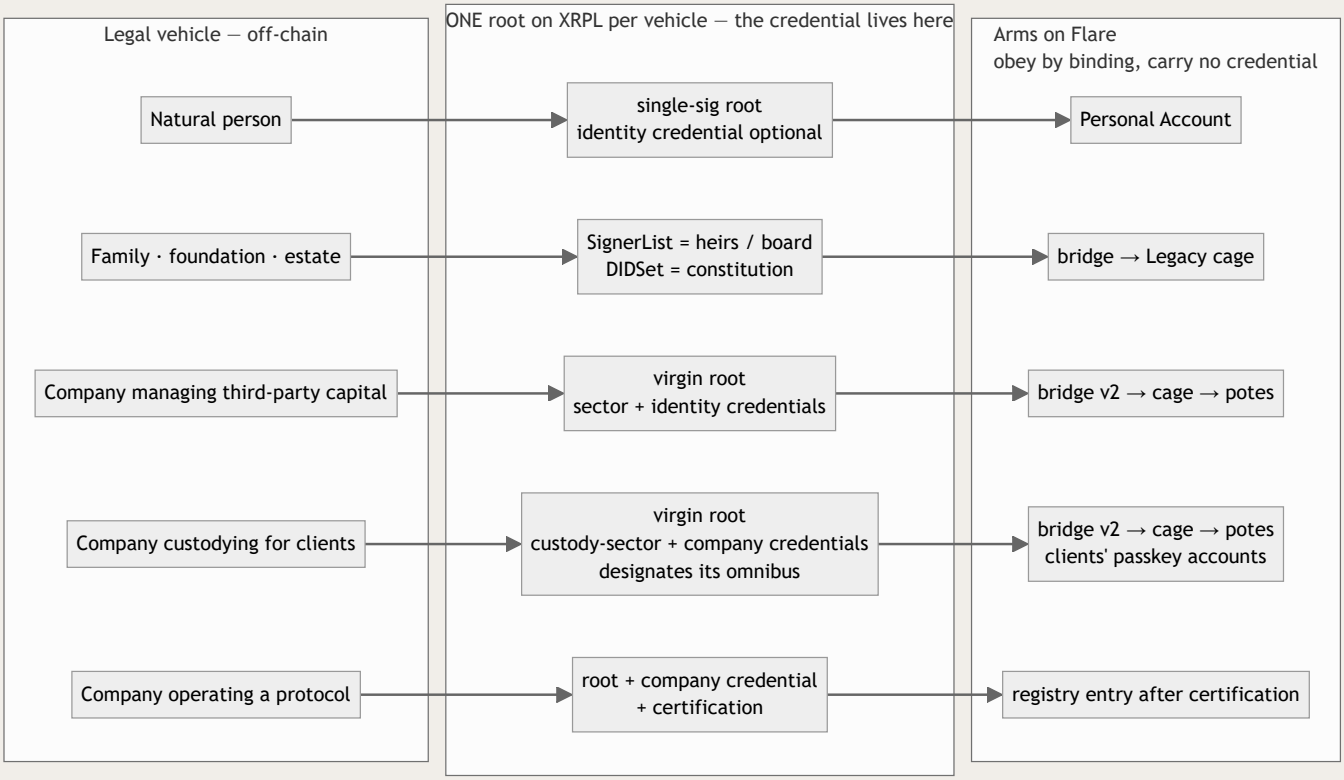
This is the map that orders everything above and everything that follows (fixed 15 September).

> **One web. One kernel. N account types.** The account type is decided by the credential its root carries, and the account type decides which functions of the same kernel are active.

The kernel — identical for everyone. Identity (the XRPL root; the Personal Account on Flare as its arm; credentials, notary, ceremony, signing tray). Reading (the capital map, portfolio, positions — observe wide, all chains). Catalogue (a taxonomy of venue classes; the venue card; the on-chain registry). Verification (a test battery on a fork, a certificate, drift monitoring). Language (canonical money flows and translators; a route planner; swap and batches). Automation (rules, sign-at-trigger; pre-signed sealed envelopes). Surfaces (explainer, filter, composition, one modal per verb, disclosure). Goals (compartments by objective). Economy (treasury, executor, sponsorship and cost recovery on exit; the distribution fee). Operation (watchdog, alerts, admin panel, audit trail). **Exit** (rescue without a gate, redemption, every product's way out — the guarantee that makes everything else sellable). *The rule of the kernel: if a piece has to ask "which account type are you?", it is misplaced.*

The account types — what changes.

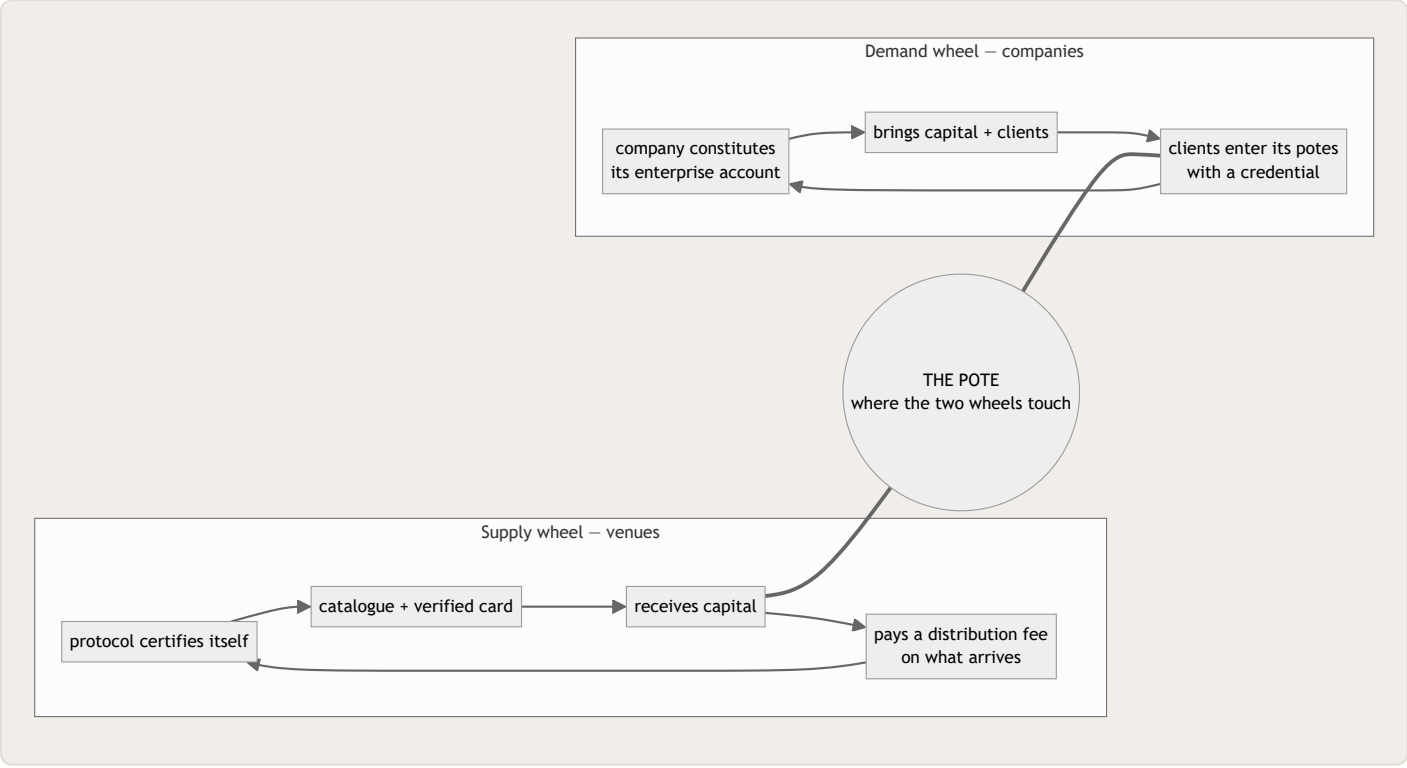
ACCOUNT TYPE	LEGAL VEHICLE	CREDENTIAL ON THE ROOT	WHAT IT ACTIVATES	STATE
Personal	natural person	none; identity for fiat later	personal account, goal compartments, composition, MoneyFlows, own potes	LIVE; compartments and composition NEXT
Legacy	family, foundation, estate	the council's	multisig council, Legacy cage, constitution, succession, governed payments	LIVE on mainnet
Manager	a company managing third-party capital	sector + identity	cage v2, grouped potes, directing capital, the directory	LIVE on mainnet
Exchange	a company custodying for clients	custody sector + company identity	its instance: omnibus with its key, boxes per client, KYC per box, potes for clients	BUILT; rehearsed; tenant model NEXT
Protocol / venue	the company operating the protocol	company identity + certification	profile, proposed addresses, catalogue presence, **pays the distribution fee**	NEXT (does not exist)
Agent	designated by any of the above	a revocable designation	operating **one** pote inside its limits	LATER



The enterprise account is the mould of the four company types (manager, exchange, protocol and, structurally, Legacy): a root, a board as `SignerList`, credentials, a domain with its `toml`, designations, an anchor with authorization by credential, and a ledger of acts (hash + pointer, never personal data). The

integration kit — verify third parties, sign as a board, designate and revoke, receive through a gate, renew and watch — is offered by API with keys and by a remote MCP. Honesty test: *if Astryum disappears, the root, its board, its credentials, its domain and its ledger of acts keep their value exactly. The only thing the company loses is the convenience of the kit. That is what is charged.* Public wording rule: never "legal account" or "legal identity" — *"an account whose board signs and whose credential anyone can check"*. What the technology does not give is legal imputability: Astryum provides the evidence; the lawyer provides the imputation.

The flywheel has two wheels, and they touch in the pote. The supply wheel: a protocol certifies itself → appears in the catalogue with its card → users and managers send it capital → it pays a distribution fee on what arrives. The demand wheel: a company constitutes its enterprise account → brings its capital and its clients → its clients enter its potes with a credential → more clients look for accredited companies. They touch where a manager, an exchange or a person **directs capital from their pote to a certified venue**: there demand pays supply, the venue pays Astryum for the capital received, and Astryum charges for the act of connecting in the directory. That point of contact is the business; everything else exists so that act is safe, verifiable and easy. The axis that starts both wheels is the company, not the catalogue: the manager arrives with capital and clients; the venue does not.



Rules that do not change with the account type: the user signs at the end of every action; the limits are imposed by the contract; the exit is never gated; Astryum never issues identity, never custodies, never proposes "for you"; the fee never comes out of the user's or the manager's capital; the order of the catalogue and the directory cannot be bought; nothing is plugged in without its scanner, its switch and its certificate — and the scanner must be able to actually cut.

Composition and autonomy are two different axes (decision of 13 September). Composition is who designs the allocation (manual → composed → assisted). Autonomy is who triggers and with what authority. One can climb in composition without climbing one step in autonomy: however elaborate the

composition, it ends the same way — the user signs, and the contract imposes the limits (a cap per venue, a liquid floor, an allow-list with a delay, an exit no one can gate). Astryum compiles and explains; it never proposes "for you", because a personalized proposal is advice, and advice belongs to an occupant with a credential, never to the tool. The system does not set weights the user did not state.

---

6. Legal vehicles ↔ credentials ↔ accounts: the link lives on the root

This section is the hinge between the technology and the world it is for. Everything Astryum enforces on chain is *about* a legal vehicle that exists off chain — a person, a family, a company, a foundation — and the link between the two is not a database row of ours: it is a credential on an XRPL account.

6.1 The idea in one sentence

Every real-world legal vehicle has ONE root account on the XRP Ledger, and the XLS-70 credentials on that root ARE the link between the legal vehicle and its on-chain existence. Everything the vehicle does on chain — its cage, its potes, its subordinate accounts, tomorrow its protocol-managed wallets and its agents — hangs from that root and inherits its status, because subordinates only obey: gating the root's order gates the whole tree. We do not credential wallets; we credential **the vehicle, once, on its root**. The forest has one tree per vehicle, and never an "account for everything": that would destroy the 1:1 mapping between account and entity that gives the structure its legal force.

6.2 The mapping: which credentials each vehicle carries

The credential is not "one": it is *the set of credentials the law already requires from that vehicle*, translated to XLS-70. The gate reads groups — *the licence of your sector, and your identity* — implemented today as OR-groups (`AIFM|CASP` and `KYC|KYB`).

VEHICLE	THE ROOT (WHO SIGNS)	CREDENTIALS ON THE ROOT	WHAT IT ANCHORS (DID)	NATURAL ISSUER
Natural person	single-sig (their Xaman)	identity (`KYC`) — only when fiat or a credentialed lane needs it	—	a qualified trust-service provider (eIDAS); today the notary robot from Coinbase's on-chain attestation
Self-employed / professional	single-sig	identity + professional licence if applicable	—	professional body, trust-service provider

VEHICLE	THE ROOT (WHO SIGNS)	CREDENTIALS ON THE ROOT	WHAT IT ANCHORS (DID)	NATURAL ISSUER
Operating company (SL, SA, LLC, GmbH…)	**SignerList = the board of directors** (a real quorum)	company existence (`KYB`) + power of representation + sector licence (`CASP` for custody, `AIFM` for management…)	the by-laws (SHA-256 in `DID.Data`)	commercial register, notary, sector regulator
Holding / patrimonial company	SignerList of the board	`KYB` + beneficial ownership	shareholders' agreement, by-laws	notary + register
Foundation / Legacy	SignerList = trustees / heirs (the quorum *is* the structure)	`KYB` + trustee / protector credential	the whole constitution (built: `constitutionRef = DID.Data`)	notary, supervisory authority
AI agent (later)	the root of the vehicle that employs it	an agent credential ISSUED BY its vehicle + the vehicle's own, inherited	the agent's mandate (hash)	the vehicle itself + its notary

What already exists in code fits without changes: the XLS-70 ceremony (create + accept), the OR-group gate by sector, the notary robot (identity from a real attestation; sector credentials as demo issuance), the anchored constitution of Legacy, and the anchor door (`DepositAuth` + `DepositPreauth{AuthorizeCredentials}`) as consensus enforcement.

6.3 The exact XRPL mechanics — why XLS-70 and not something else

1. **`CredentialCreate` by the issuer.** The regulated third party (notary, register, trust-service provider) verifies **off-ledger** and signs. On the ledger goes only the minimal attestation: type + `Expiration` + `URI` .
2. **`CredentialAccept` by the subject** — and here is the corporate jewel: when the root is the SignerList of the board, **the multisigned accept IS the board's resolution.** Corporate consent becomes native to the ledger, with a real quorum. (Today our acceptance route signs with a single subject; composing it through the multisig coordinator is design, not yet code — and we say so.)
3. **Mandatory `Expiration` is the revocation.** A licence is withdrawn in the real world and the ledger does not find out; the only honest exit is that it expires and is not re-issued. Revocation freezes **new orders** (the check lives at the entry); the capital's exit is never gated.
4. **`URI` is a pointer, never the document** — to the issuer's verifiable credential or attestation. **Never personal data on the ledger.**
5. **The ledger carries the YES, never the NO.** It records who *can*; a compliance refusal written on a public chain would be a leak and, in some regimes, an offence. The "no" lives off chain, with the issuer.
6. **The enforcement is not Astryum, it is consensus.** The order anchor demands the credential through `DepositPreauth{AuthorizeCredentials}` : an order from a root without a valid credential dies in consensus, before any backend has an opinion. Astryum only does the pre-flight — "do not sign a doomed order". (Built; not switched on yet.)

6.4 The vehicle's "account for everything" is its ledger of acts

The root is not just money: it is **the vehicle's book of acts**. Every relevant act — a board resolution, a power granted, the purchase of a property, a document — is recorded as **hash + pointer** (memo or DID), never the content. Over time the vehicle holds, on one account: its **constitution / by-laws** anchored; its **credentials** in force (who it is, what it may do, until when); its **cage** (what it is allowed to do with capital — one cage per vehicle); its **acts** (the auditable history); its **subordinates** (the hands: Personal Accounts, passkey accounts, potes — captive, without credentials of their own, obeying by binding; and the one non-captive subordinate, the exchange's omnibus with a free key, which carries only the root's **designation**, never compliance); and tomorrow its protocol-managed wallets and its agents. The consequence for the law: when a vehicle's **statutes adopt the chain as their layer of action**, the gap between the two worlds disappears by definition. That requires a lawyer per jurisdiction — *legal before architecture*.

6.5 Who issues — never Astryum

Astryum **never** issues the real title and never verifies identity: it chooses which issuers it accepts (a technical allow-list with no ranking) and checks the object on the ledger with a uniform rule. The issuers are third parties — notary, lawyer, register, trust-service provider; today humans with a robot (the notary robot already issues the identity credential from a real third-party attestation), tomorrow agents that verify and sign with their own agent credential. The chain of trust is recursive and always ends in a regulated third party in the real world. Sector credentials in the rehearsal are **demo issuance**, with the public register's link as `URI`: Astryum brings on chain what the register says; it certifies nothing.

6.6 The concrete legal shapes of Legacy (typologies)

TYPOLOGY	WHO	LEGAL VEHICLE	STATE
T1 Personal with rules (reinforced)	one person, three keys	none (+ a will)	LIVE (the ceremony)
T2 Family without an entity	a family, a council	private agreement	**LIVE on mainnet** (the `familiar` template)
T3 Minors / education	parents → child	parental authority / gift	constitution template exists; a coming-of-age credential is the missing piece
T4 Marital	a couple	marriage contract + regime	not templated; legal opinion pending
T5 Operating company	a commercial company	company + by-laws mirrored on chain	designed; Astryum's own company is the dogfood
T6 Patrimonial holding	partners WITH ownership	closed holding company	designed
T7 Foundation	beneficiaries WITHOUT ownership	private foundation	template exists; the target model

Ten axes that every typology has to settle, and the mechanism for each: ownership (the vehicle, or its declared absence) · organs and quorum (`SignerList` with weights) · membership (credentials + a rotation ceremony) · beneficiaries (beneficiary credentials + governed MoneyFlows) · continuity (a succession ceremony on a legal-fact credential) · supremacy when law and ledger diverge (a clause in the text + the anchored hash + a curing ceremony with a delay) · exit of all capital when the law demands it (the "release vessel" via `migrate` — not built) · privacy (hashes only) · tax (an opinion; we never advise) · **Astryum's perimeter** (software, templates, ceremonies, a verifier — **never** custodian, signer, administrator, trustee, issuer or registrar).

6.7 State today → what is missing

PIECE	TODAY	MISSING
XLS-70 create / accept with `Expiration`	LIVE (ceremony, notary robot)	—
Gate by sector with OR-groups	LIVE (`AIFM|CASP,KYC|KYB`)	more types (power of representation, beneficial owner) = configuration + issuers
Constitution anchored (DID)	LIVE (Legacy, manager, exchange root)	templates per vehicle type
Anchor door (consensus enforcement)	BUILT	mainnet rehearsal, a screen that calls it
SignerList as the board	LIVE (Legacy multisig)	a "constitute a company" ceremony in the UI; the multisigned `CredentialAccept`
Real issuers (trust-service providers, notaries, registers)	demo robot + one real identity attestation	partnerships pending; the KYB provider under contract (wave 2)
Ledger of acts (hash + pointer)	pattern exists (memos, DID)	a product surface

Guardrails, non-negotiable: Astryum is never the issuer, administrator or organ of any vehicle (the legal wrapper belongs to the client and their notary; Astryum provides the rail). Never personal data on the ledger; never the document — only hash and pointer. In public, never "licensed", "regulated" or the name of any regulation: the demo issuance is not a regulatory attestation; only *"a credential verified on the ledger"*.

7. What comes next: the V2 plan (decided 15 September; not coded yet)

The plan is fifty-three items in order, grouped in waves. The method: one round by hand before automating; build in place behind the founders-only wrapper; short branches; **vertical delivery** — one venue class complete end to end before the next. Delivered by the same hands that built what is live.

Wave 0 — the fourteen urgent items that exist today in production code (~2–3 weeks). Found by our own audit of 15 September and said here before anyone finds them: the risk scanner's chain guard (the third-party scanner does not cover Flare and "unknown" passes the gates today — the invariant "nothing is plugged in without its scanner" holds on paper and not in fact, on the chain where the product lives); moving the venue registry's key to a multisig (today it is the treasury key; the registry timelock is one hour); per-user isolation in the MCP server; proof of subject in the notary; credential renewal without a gap; succession of a cage with an expired credential; credential groups in the anchor door; and seven smaller ones. Four of them are candidates for an early hotfix.

Wave 1 — the first vertical pass + the credential schema (5–7 weeks). Declare the class "deposit with shares" once, and hang from it the venue card, the verbs, the modal, the test battery and the contract branch (*the class is the hinge*). The venue card served from the server with provenance per field (measured / declared / attested). A minimal battery on a mainnet fork for Kinetic and Firelight, producing a certificate with method version, date and block. The explainer reads only the verified card (if a fact is not there, it says it does not know). A filter by exit time. Normalized disclosure. A versioned credential schema, and the notary fed by it.

Wave 2 — the enterprise account, the distribution fee, the compartments (8–12 weeks). A KYB provider under contract. The on-ledger anchor door switched on in mainnet. A single register of accepted issuers. Expiry sweeps ("do not admit new entries, do not cut off those inside"). The kit: remote MCP and API with keys. Volume attribution and disclosure of the venue's fee. **Goal compartments:** the exchange's own infrastructure (boxes by tag on XRPL, potes on Flare) turned into a feature for every personal user — *with the user's key only; at no point in any compartment is there a key that is not the user's*. One onboarding for all account types (three assistants become one). Who pays the transport per account type.

Wave 3 — composition (6–9 weeks). Several actions per step in the canonical language; swap in both directions as calldata the user signs; one form of call and one preflight per chain; one disclosure; one modal per verb; a composition draft. Done when "convert and repay" is one signature with the worst case sealed inside it.

In parallel, permanently: the third generation of the contract (a branch per venue class, starting with the queued venues; the registry key in a multisig) with an **external audit** — the number-one blocker, budgeted as the main line of the grant applications; and the legal work that is not code (§13).

Wave 4 — the planner, the opening, the fiat last mile. A route planner (today the multi-step patterns are written by hand — and the planner is where discretion would creep in, so the user sees the route and the worst outcome before signing, and the floor travels sealed in the signature). The protocol's profile

and domain file. The directory with a fixed fee per act. A published specification. Fiat in and out with regulated partners.

Wave 5 — Ethereum and the agent. The verification attestation anchored on Ethereum (EAS) with a published schema — the measurement layer as a product there, for curators; *not* the cage (gas breaks its economics, and the XRPL → Flare authority tunnel does not reach Ethereum). Pre-signed sealed orders from XRPL. The agent's pote. Protocol-managed wallets as a remote arm.

The calendar, honestly: the core (waves 0–3) is 104–154 person-days — five to seven months with one pair of hands, three to four with two. The first revenue appears in wave 2.

---

8. What comes after: the future, with precision

Everything in this section depends on primitives that are not on mainnet yet, or on legal work. It is told on the map; it is never shown as if it worked.

8.1 Flare protocol-managed wallets and confidential compute — the cage's remote arm — LATER

What they are, verbatim from Flare's documentation: protocol-managed wallets are keys split across several TEE machines that act as k-of-n signers on *native* multisig accounts of **XRPL and BTC**; instructions reach the TEE only after more than half of the signing weight of Flare's data providers has approved them; executions are proven back by FDC. Confidential compute (FCC) runs verifiable extensions inside a TEE and is, in Flare's own words, not yet a public production system.

What they change for Astryum. Today the cage holds capital *inside* the contract. With protocol-managed wallets the enforcement moves from "custody in the contract" to "the contract's instruction set": the TEE signs only what the registered instruction sender — the cage — tells it, and the cage has no instruction of the kind "pay an arbitrary address" (the same principle as *no* `withdrawPrincipal`, carried to the arm). Capital can then sit in wallets on other chains that the cage governs — a **locked arm**: it cannot be taken out, it can be put to work. The cage already has the frame for this (register a remote wallet, mint into a pote from it) and **the arm that would dispatch it reverts on purpose** — forward compatibility decided on 22 August, so that the day the primitive exists the migration is cheap, without depending on it or showing it as if it worked.

The honest conditions. Keys must be generated *inside* the TEE, never imported (a delivered key existed outside, and its deletion is unverifiable). The trust model is Flare's data providers plus hardware attestation plus the cage's code — *trust-minimized, never "trustless"*. Because TEE-managed custody is a form of delegated signing, it lives as an **isolated module, gated by jurisdiction**, never coupled to the kernel. It does not reach the user's own MetaMask or Phantom — only arms created under the authority. Zero confidential-compute code in the repository today, by decision.

And one thing it makes native: a TEE key can be *one signer among people* in an XRPL `SignerList`. The ledger already supports N weighted signers collecting independent signatures in any order; what Flare governs is *when* the TEE signs. That is the cleanest version of a protocol-managed arm for a family or a company.

8.2 The single root: one XRPL account governing many — LATER

The product we are most excited to build. Today a person has one Xaman account and, from it, one Personal Account on Flare — already presented as **one account** with three verbs from the same signature (mint, use what is already inside, convert back). The next step generalizes it: **one master XRPL root** governing several personalized XRPL accounts — the family's, the company's, a reinforced one — each of which, through protocol-managed wallets, governs wallets on other chains and their assets. One human key, N seats in N `SignerList`s; the same governance system for every account (a personal quorum can be 1-of-1 as the minimum dial). Everything binds to the **address**, not to the keys, so rotating signers never re-points a vault, a Personal Account or an arm.

Two rules already fixed: **one root per legal vehicle**, never an "account for everything" (it would destroy the 1:1 mapping between account and entity that gives the structure its legal force); and Astryum never generates or hands over a seed — the wallet does, on the device.

8.3 Agents — the sixth account type, inside its own compartment — LATER

The agent is not a product; it is an account type designated by any of the others. The chain, stated whole: a credentialed human root → a designation (XLS-70 today; XLS-75 as transport when it activates) → an agent with its own credentials → executing **inside the cage**, in **its own pote** — *the agent's limit is its pote*; revoking the designation switches it off; the user's exit never depends on it → and confidential compute certifying the **pipeline** (exact prompt template, pinned model version, hashes of input and output), never "the intelligence". Payment to an agent or its company can sit in a native XRPL escrow whose release is gated by the credential: if the credential is revoked between the order and the delivery, the payout fails on the ledger, with no code of ours.

Three lines that do not move: **the root is always human**; the agent is never on a root's `SignerList`; and *we credential the authority and the act — never the thought*: a register of "bad prompts" would be a register of NO, forbidden by our own doctrine. Permission and settlement on the same rail: a permission system without control of payment is a recommendation; a payment system without permissions is a bank; together they are the only layer that can say no and have the no carry consequences. Astryum does not issue credentials, does not evaluate work, does not choose judges — it transports and executes third-party attestations. Legal work (escrow, payment-services and operational-resilience questions) comes **before** architecture on the leg that touches money. Today's copilot is the first caller of that rail.

8.4 The exchange, privately but verifiably — LATER

With confidential compute, the exchange's operations that must be private — trading, purchases, client-level operations — can run inside the TEE with verifiable outputs, while the structure (root, designation, potes, shares in the client's name, exits no one can gate) stays public. The merge with protocol-managed

wallets is the roadmap step: the live omnibus key enters the TEE and the subordinate passes from *designated* to *captive* — the same root governs more wallets, multichain, without issuing a new credential.

8.5 Fiat in and out, and the card — LATER

On-ramp and off-ramp through regulated partners (decided, not built): the partner holds the licence and does the regulated part; Astryum gives the access. The **card**: spend from your own wallet with a balance no one can freeze — a *pull-at-authorization* mandate with no float, a contract with no owner and no pause, two signatures forever (create, revoke), only e-money tokens (USDC, EURC, RLUSD), fees disclosed. Zero code today; no partner under contract; an external audit is blocking. What we will not build: consumer credit dressed as a card.

8.6 More networks — through Flare's primitives, honestly — LATER

Observation is already wide (XRPL, Flare, EVM via indexers). Execution widens by phases and only where a primitive reaches: FDC's `EVMTransaction` supports exactly three sources today (Ethereum, Flare, Songbird), so *the common layer is not a chain, it is the signature*: a fact goes on a chain only when a contract has to obey it; authority stays on XRPL; enforcement stays on Flare where there is a cage; measurements are signed and anchored where curators live. XRP → EVM yield through the XRPL EVM sidechain and Axelar is a later phase; Solana later still.

---

9. How Astryum earns — and what it never charges

Today nothing is charged. The machinery is wired and dormant on purpose. Four rules govern any future charge: the fee never biases what is shown (the engine that ranks strategies does not receive the fee as an input); the fee is visible before signing, with amount and payer (a literal type in the code, and a policy that rejects any undisclosed extraction); sponsor first and charge after — **never a prepaid user balance** (that would be e-money and would make us what we are not); at cost, not by eye (infrastructure costs converted with the native oracle).

Where the money comes from, by solidity.

1. **The distribution fee, paid by the venue** on the capital that reaches it from Astryum — the principal source, decided 15 September, not built: the user deposits 100 and the venue receives 100; the venue pays Astryum; the same percentage for the whole class, no effect on order or visibility, always disclosed ("this venue pays Astryum X %"). The non-custodial introducing-broker model.
2. **Fixed fees per act between companies:** the cage charges a creation fee in FXRP from the fourth pote onwards, directly to the treasury in the birth transaction (LIVE: 1 FXRP on the deployed factory); designating a manager, listing a venue in a cage, joining the directory — a fixed fee per act, disclosed before signing, collected by the code or the partner, never intercepted by Astryum in the path of other people's money. Not a "marketplace"; no paid positions; no percentage on flows between companies.

3. **Transport at cost:** each order from XRPL to Flare costs a real, measured FDC round (~20 FLR); a fixed fee like the protocol executor's (0.2 XRP) inside the Payment the user signs — built, switched off, and it fails loudly if misconfigured rather than charging zero silently.
4. **Referral basis points via partners on EVM** (15 bps on the main aggregators, embedded in the partner's transaction, paid directly to a fee wallet; zero on XRPL native by design) — wired, dormant until a public fee schedule, a UI disclosure row, a test that the ranking cannot see the fee, and a legal opinion exist.
5. **Enterprise membership and the kit**, and a review fee for venue certification that covers cost and buys no position.

What is never charged. Nothing on shares or on the client's yield in the managed lane (the pote has no fee hook — zero enforced by absence); the manager's cut is capped at 20 % of yield by the cage; nothing deducted from a defensive operation; no token of our own; no charge in a flow that does not pass through a registered regulated partner. Treasury doctrine: revenue is not consumed, it becomes productive patrimony — the treasury is Astryum's first Legacy, verifiable on chain.

---

10. The trust model: what happens if we fail, are attacked, or disappear

10.1 If Astryum disappears tomorrow

The user loses nothing and needs no permission from us. Their keys were never on our servers (we never generate or hand over a seed). Their capital was never at an address of ours (no Astryum contract receives user assets; no omnibus is ours). Their rules live on the ledger and in contracts **without proxy and without pause** (checkable: zero `upgradeTo`, `delegatecall`, `whenNotPaused` or `Initializable` in `contracts/src`); the owner is the user's own council. Their exit does not depend on us: the quorum signs from any XRPL client, and the vault unwinds positions by itself on redeem. Their credentials were issued by third parties and live on their own account. **This is the design test, applied to every new piece before it is merged.**

10.2 Who can harm whom

- **Can Astryum rob the user?** No — by absence of capability, not by policy. No function to sign, broadcast or custody exists; the seven operational keys (§2.5) sign only Astryum's own transactions.
- **Can the manager rob the client?** They cannot extract: the cage accepts two verbs — send capital to a listed destination, recall it. The payee cap lives in the cage. The trap closed before it existed: no flow ever asks the client to approve the operator over shares (tripwire test). What the manager *can* do, and we say it: choose badly within the list. **There is a person deciding**; it is never sold as autonomous.
- **Can a council member move the capital alone?** No: the network enforces the quorum and the master key is off.

- **Can an exchange operator trap their client?** The XRP in the omnibus is the operator's custody — that is the product. The operator's key is bounded by code (caps, refusals, double evaluation); the client's shares in the pote are incongelable and exit by the client's passkey; revoking credentials freezes the operator's governance, not the clients' exits.
- **Can the AI decide anything?** It has nothing to decide with: no tool to sign, execute or custody.

10.3 Failures we already had, and what they built

A signed order took forty minutes to execute while the screen said "done" → *signing is not executing*: no modal turns green before real settlement. A repeated signature could cause a double dispatch → fail-safe classification: only an error that *proves* nothing left allows a retry; the unknown stays "unconfirmed" and a second signature is never offered. Rules that failed silently and looked green → "could not read" is a first-class verdict, distinct from "fine". The escrow keeper signed with our own account *and* stamped the project tag → two explicit attribution modes, fixed by tests. On 14 September two untested vaults appeared on the public site because an environment variable cloned from staging became effective on the first successful backend deploy in days → **what decides whether something is visible in production now lives in code (an allow-list of tested vaults), never in an environment variable; defaults open outside production and close inside**. The legal pages claimed things the product did not do → an accuracy audit, published with the detail against us.

10.4 The limits we set on purpose

No token of our own. No paid or covert promotion. No asking for transactions "to support the project" (it would falsify the metrics, and the refusal is codified). No regulatory claims in public — only "a credential verified on the ledger"; we say *we are structured to stay outside* the perimeter of regulated activity, and the boundary map lives in the repository. No coupling of enclave custody to the kernel.

---

11. The abstraction: what the user never sees, and what we refuse to hide

The thesis. One co-founder holds XRP and does not live inside DeFi; the other operates positions daily. That asymmetry is the product thesis, written on the `/about` page since July. The bar, written in the glossary: *if a screen needs explaining before it makes sense, it isn't finished.*

What is absorbed. "I don't have the other chain": the Flare account is derived, gas is paid by the executor or relayer ("pays the gas · cannot decide"), XRP becomes FXRP inside the same flow, and "Convert to XRP" inverts the whole path. "Everything costs and I don't know how much": the expensive part is not XRPL (12 drops per signature) but the transport; the answers are cost recovery on exit (built, inert), amortization of one signature across N actions (live), the tag channel where the exchange's client has no account, no reserves, no gas, no fees (built), less XRPL surface (working accounts live on Flare where

there is no reserve), pricing at cost via the oracle (live), and later sponsored fees when the amendment activates. "I don't understand what the screen says": one error translator, transaction types named by what they do, waits shown in real time, stuck transactions watched and retried with the *same* proof.

What we refuse to abstract, because it is the product. The user's signature. The fees, before signing — with the cost, when recovered inside the operation, *inside the hash the user signs*. The cage, before it receives capital: disclosure at the irreversible step, with server-hashed text so an audit can prove what was read. The reserve of the root account (1 XRP + 0.2 per object): the notarial cost of constituting a vehicle, disclosed and paid. And physics: between two incompatible wallets there is no "same transaction" — it is N signatures, and we say so; the catalogue of routes is finite and documented as such.

What the user reads, and what it hides: *your Astryum account* (the Personal Account on Flare) · *Convert to XRP* (FAssets redemption) · *the cushion* (health factor) · *transport* (the memo instruction's carrier) · *Put to work / Working* (supply to a venue) · *We tested this operation without signing it* (simulation) · *a rule of the network, not a fee* (XRPL base reserve) · *Saving moves no money and signs nothing* (the automation engine in prepare-only mode) · *You never touch a wallet, gas or FLR* (passkey + relayer + tag channel). Forbidden on the product's surface: `payload`, `userOp`, `calldata`, `intent`, `mint/unmint`, `bps`, `drops`, `shares` as the main unit, `TVL` unexplained, and any internal enum.

---

12. Evidence: how to verify without asking us

XRPL Mainnet (xrpscan.com/account/...): FAssets Core Vault `rfkXSaCZKTg1EZzec2rLDyrWHxRVJdtVXj` · Legacy anchor `rK4tsuGhmbhaNQuvucL8n1RKLtARBCp3qm` · v2 anchor `rLcoFM9XF8CL5GoFyMACguhdnDtwkNYDn7` · notary issuer `rHKxjrGRrCegQhLrdnXEPAeGyeJ1JR4Hae` (operational: its transactions carry no Source Tag, on purpose). Project Source Tag: **2607090002**.

Flare (chain 14) (flarescan.com/address/...): `MasterAccountController` `0x434936d47503353f06750Db1A444DBDC5F0AD37c` · `AssetManagerFXRP` `0x2a3Fe068cD92178554cabcf7c95ADf49B4B0B6A8` · FXRP `0xAd552A648C74D49E10027AB8a618A3ad4901c5bE` · Legacy: `XrplCouncilBridge` `0x02aE9fcB76768e42b8D3Ed9FE842238A6616b26F` (nextNonce 4), `LegacyVault` `0xc8379c79779cCE3B738424892709fe0D4339E3b1`, `LegacyStackFactory` `0xF93A8A0bd93e95514fF02285349b0b1c1a5a3e0a` · Managed v2: `AstryumRegistry` `0x18c807d107EF9d337aFA5BdD39b190C42Dc50Aec` (2 venues, 1 h timelock), `AstryumCageFactory` `0xE897fFef10F950Abc2d9c759107410F713e941D9` (3 cages, 4 potes), `PoteDeployer` `0x5f84c5F4805c29298AF71Fc6657D3E780eF27E9E` · v1: `AstryumStackFactory` `0xc221E9F447E65EdAe16A4Af64D3a7a67E2EE3783`, pote A `0x21d4ccf29EEB61E573c2037F2B29B727168c3da9` · `PasskeyAccountFactory` `0xfa559cE04135835Cb8f7d3CFad2b8d14525ad932` · FXRP OFT adapter `0xd70659a6396285BF7214d7Ea9673184e7C72E07E`. All reads above dated 15 September 2026.

The product: `https://astryum.xyz` — `/proof` verifies claims in your browser. The public repository has a README and a licence (audit yes, exploit no: public does not mean permissive).

Tests: Foundry 168/168 (62 on the cage v2 against real venues on a fork); backend 4,275 tests in 329 suites and frontend 2,169 in 160 files, green on 15 September — with the caveat we apply to ourselves:

green tests prove the pieces, not that the chain exists in the real domain, which is why every fix in the last cycle had to carry a test of the *phase* or the *consumer* that failed.

Attribution: Source Tag metrics are computed with a fixed definition (active user = signer of ≥ 1 validated `tesSUCCESS` transaction carrying the tag, multisig `Signers[]` included, de-duplicated by hash, operational accounts excluded, truncation declared). **5 active accounts · 289.51 XRP** (13 September, afternoon). The transaction-by-transaction read of 19 September is in Astryum on mainnet.

---

13. What we do not claim — and the open blockers

We do not claim: an external audit (none yet; hence own capital, small amounts, caps in configuration); users beyond the founders' circle; that the tag is on *every* transaction of ours (it is on every *user-signed* one; our operational keys carry none, on purpose); that the exchange has run with a third party; that the on-ledger credential door is switched on; that any swap, planner, fiat ramp, card, agent, protocol-managed wallet or confidential-compute path exists in code; that yields are anything but live protocol data with a source.

Open blockers, in order.

1. **External audit of the contracts** — the number-one blocker; budgeted as the main line of the grant applications.
2. **The company** — the legal wrapper is in progress; without it there is no KYB provider contract, no distribution agreement, no listing agreement.
3. **What production shows.** Legacy, the manager's desk and the Exchange are closed in production by code until they are opened by hand. Until then a visitor sees the product without those doors.
4. **Key concentration.** The registry governor and the factory treasury are the same key — and, per our own audit of 15 September, the executor's; moving the registry to a multisig is wave 0.
5. **Regulatory boundary document** to be extended for portfolio management (the answer of substance: the manager holds the licence; Astryum is the tool). **Data-protection representative** designation pending signature.
6. **Written technical debt:** row-level isolation in the database before opening watch-only to third parties; self-service recovery of stuck operations; external incident alerting; the executor's daily budget as the physical ceiling of adoption.

Appendix A — status table (15 September 2026)

PRODUCT / PIECE	STATE	PROOF
Personal account (memo rail, carry, redemption)	LIVE	mainnet: mint, deposit, withdraw, repay; redemption 31 Jul; E2E 22 Aug
Reinforced account	LIVE (ceremony)	same multisig pipeline live since 14–15 Jul; surface shipped Sep
Legacy	LIVE	council 3-of-4; 4 orders E2E; bridge nonce 4; vault ≈ 4.70 FXRP
Managed vaults (manager)	LIVE	cage v2 on mainnet; 3 cages, 4 potes; full cycle proven; 168 Foundry tests
Exchange (operator + client)	BUILT	root, designation, cage and potes on mainnet; rehearsed; founders-only
Credentials — the gate	LIVE	fail-closed at cage birth and orders; notary from Coinbase EAS (9 Sep)
Credentials — on-ledger door (DepositAuth + Preauth)	BUILT	routes and client exist; no screen calls them; not exercised on mainnet
Permissioned Domains	BUILT	no consumer
MoneyFlows	LIVE	escrows + keeper; price trigger since 16 Aug
/proof	LIVE	run `verify` not exercised on mainnet
Copilot / MCP	LIVE as copilot	no sign/submit tools; MCP local only
Watch-only	BUILT	row-level isolation pending
Enterprise account, venue certification, distribution fee, compartments, planner	NEXT	V2 waves 1–4
Fiat ramp, card	LATER	partners decided; zero code
Protocol-managed wallets / confidential compute	LATER	frame in the cage; arm reverts on purpose; zero TEE code
Single root, agent account	LATER	designed; depends on the above and on XLS-75

Appendix B — terms used here

Root — the XRPL account that governs (a person's Xaman account, a family council, a company's board as a `SignerList`). **Personal Account** — the account on Flare derived from an XRPL address, the root's arm. **Memo instruction / Rail A** — a Payment to the FAssets Core Vault whose memo commits to the batch the Personal Account executes. **Council order / Rail B** — a 1-drop Payment to an anchor whose memo commits to calldata a bridge executes against a vault. **Anchor** — the XRPL account that receives council orders; with DepositAuth it becomes a gate. **Bridge** — the contract on Flare that verifies the FDC proof and is the vault's only council. **Cage** — the contract that governs potes for one root and cannot custody, extract or be upgraded. **Pote** — an ERC-4626 vault inside a cage: shares to depositors, venues from the registry, exit no one can gate. **Venue** — a protocol where capital works (a line in the registry, never the product). **Credential** — an XLS-70 object on a root, with expiry, accepted by the subject. **Designation** — a credential from a root to its subordinate (a council naming its omnibus). **FDC** — Flare Data Connector, the notary between the two ledgers.

One kernel. N tunnels. The order travels; the capital does not. And in every product the same sentence: Astryum prepares, the user signs, the ledger enforces.