SagaHalla

Oracle Handbook

VALIDATOR · OPERATOR · ALLOCATOR

Version v1.4 · 2026-07-25

Customer guide — public documentation

support@sagahalla.io · https://sagahalla.com/oracle-handbook

PUBLIC DOCUMENTATION · NON-ADVISORY · VERSIONED

SagaHalla Oracle Handbook

Version / Date: v1.4 / 2026-07-25 **Audience:** Validator · Operator · Allocator **Handbook URL:** https://sagahalla.com/oracle-handbook **Support:** support@sagahalla.io

Welcome to the SagaHalla Oracle Handbook — your guide to using the Oracle as a **Validator**, **Operator**, or **Allocator**.

This handbook is product documentation. It is not investment advice and not a substitute for the terms and acknowledgments you accept in the app.

By the end, you should know **which tier you are using**, **which data timing applies**, **which controls are safe to use now**, and **which actions require additional gates**.

When the app and this handbook disagree, **what you can see and click in the app wins**.

Start here

If you are…	Do this first
Evaluating the Oracle	Open **Proof**, then **Decisions**, then the **Paper sandbox** (A.5–A.9, A.13)
Integrating alerts	Read A.2 timing, then C.3–C.6; create a test webhook and verify the delivery log
Preparing Allocator automation	Read D.8–D.14; complete the before-you-connect checklist; begin in **Manual Review** with live submission **off**

Who this handbook is for

Reader	Start with
Anyone evaluating the Oracle	Part A (Shared) + Part B (Validator)
Integrators connecting decisions to their stack	Shared + Part C (Operator)
Users deploying capital with brokerage automation	Full handbook, especially Part D (Allocator)

Availability legend

Marker	Meaning
Available	Live for the entitlement in normal product use
Coming soon	Described for planning; not usable yet

How to read tiers

Entitlements are **additive**: **Validator ⊂ Operator ⊂ Allocator**.

- Later Parts assume earlier ones.
- Pricing is summarized in A.3; what **Settings** shows for your account is authoritative.

- **Decision timing** is the main product difference between tiers (see A.2).

Figures & screenshots

Screenshots are **examples** from the product. Your account may show fewer controls depending on your plan and which live controls are enabled. When screenshots and the app differ, **the app is authoritative**.

Some figures show the Oracle’s **Fresh $100K Book** (institutional paper track). That is not your personal brokerage account. If a surface is not available yet, it is marked **Coming soon**.

Part A — Shared (all signed-in users)

These sections apply to every signed-in user. Tier-specific depth follows in Parts B–D.

A.1 What a decision is (and is not)

A SagaHalla **decision** is a structured, end-of-day artifact for a supported instrument on a **closed daily bar**. It reflects the Oracle's **consensus** — the combined view of its models for that bar — and typically includes an action context (buy / sell / hold framing), explainability (why the Oracle reached that state), and related portfolio or proposal context where your entitlement allows.

A decision is **not**:

- a real-time or intraday trade alert;
- a personalized recommendation for your account;
- investment advice, a suitability determination, or a promise of profit;
- by itself an order at your broker (orders require your permission mode and additional live gates — see Part D).

Treat decisions as **general structural analytics** you may use in your own process — not as instructions SagaHalla is giving you to trade.

A.2 End-of-day / closed-bar timing model

SagaHalla decisions are **end-of-day / closed-bar** artifacts, not real-time trade alerts. A decision reflects a completed bar; it is not a live tick or an instruction to act immediately.

Delivery cadence depends on your entitlement and surface:

Entitlement	In-app decisions (Markets / Decisions)	Webhook delivery
Validator	T+1 (prior closed bar)	—
Operator	At publish (latest closed bar)	T+1 (prior closed bar)
Allocator	At publish	At publish

Why the Operator split? In-app audit stays on the latest closed bar so you can watch the Oracle run. Outbound webhooks stay on T+1 so stack integration is an explicit **delayed information feed**, not a same-bar trading signal.

This is a **delayed feed**, not an execution guarantee. Market data, paper fills, consensus state, and webhook delivery can be delayed, retried, or occasionally skipped.

Check freshness before acting. Use the latest publish time and any app health / freshness indicators shown in the product. If data looks stale, pause interpretation and contact support rather than assuming the last value is current.

A.3 Plans at a glance

Plan	Who it's for	Core value	Decision timing (summary)
Validator	Evaluate the Oracle	Paper proof, full decision explainability	In-app: **T+1**
Operator	Integrate into your stack	Everything in Validator **+ decision webhooks** + Fill Ledger / Open Proposals	In-app: **at publish**; webhooks: **T+1**
Allocator	Prepare and, when gated, deploy capital	Everything in Operator **+ at-publish webhooks + native brokerage automation after live gates**	In-app and webhooks: **at publish**

Pricing:

Plan	Price
Validator	$0
Operator	$125 / mo
Allocator	$250 / mo **+** ~$3,000 one-time platform license

What **Settings** shows for your account is authoritative.

What **Settings** shows for your plan is authoritative for entitlements and license terms.

A.4 Core surfaces: Markets (the scan lens of Decisions)

Markets is the *scan-the-book* lens — the Oracle's current view across supported assets with buy / sell / hold markers at **your tier's in-app timing** (see A.2). It is not a separate destination: the Markets scan and the per-asset explainability trace are two views of the same **Decisions** surface.

Use the list to scan the instruments the Oracle covers, then open one to reach its Decision explainability (A.6). Neither view places trades, and neither knows your personal portfolio unless you are on a surface that shows your book (paper or live).

The Decisions list below is this scan lens in the current app.

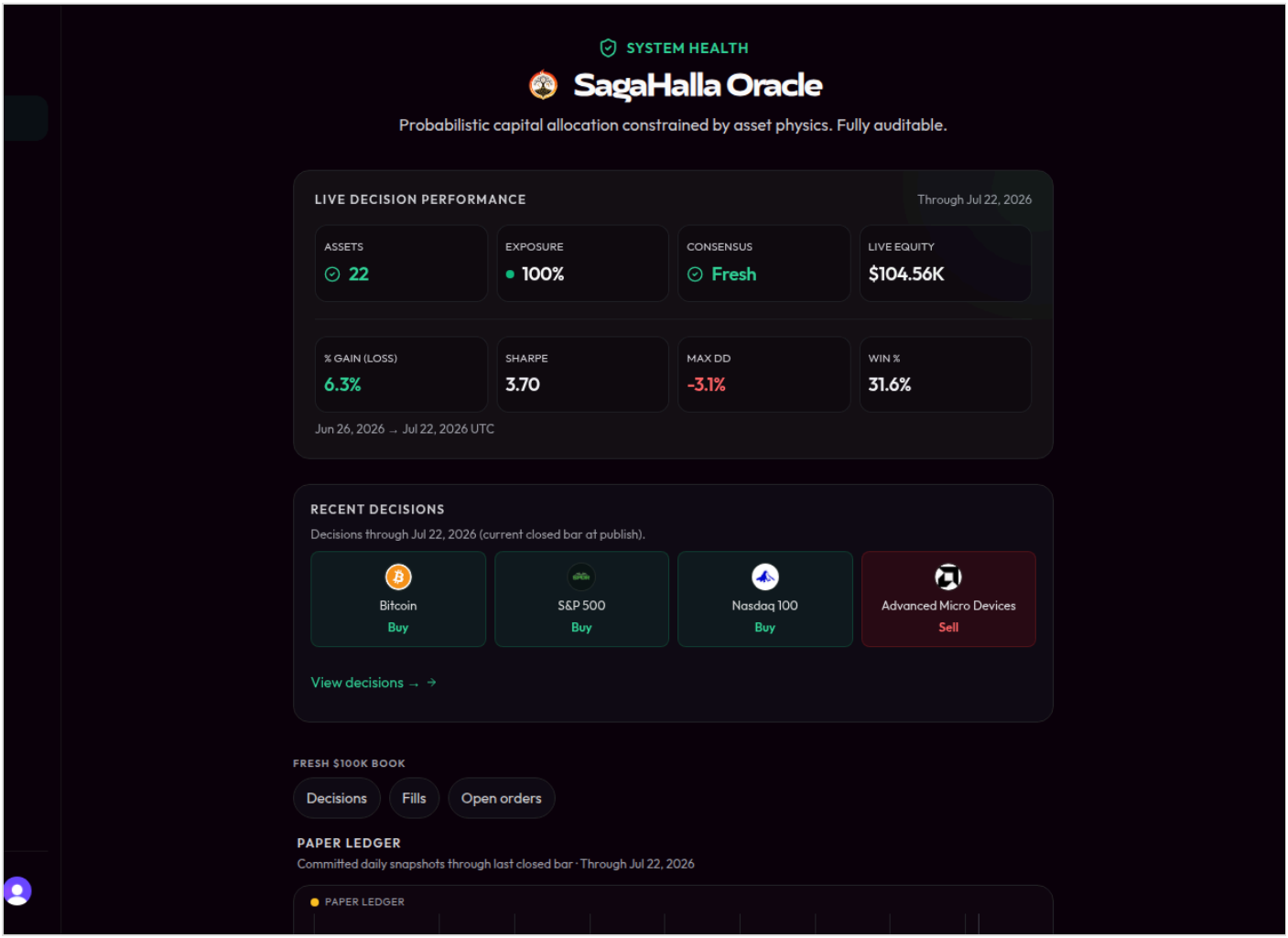
A.5 Core surfaces: Proof

Proof is the validation / performance surface for the Oracle's paper reference book — the **Fresh $100K Book** (the institutional proof track), not your personal brokerage.

What to look for:

- Headline metrics for the Fresh $100K Book
- Equity / activity context that shows how the strategy has behaved in paper
- Freshness / which closed bar the view reflects

Proof answers “is the Oracle’s paper track credible and current?” — not “what should I buy today?”



Proof dashboard for the fresh institutional validation book

Proof (app home) — headline metrics for the Fresh $100K Book.

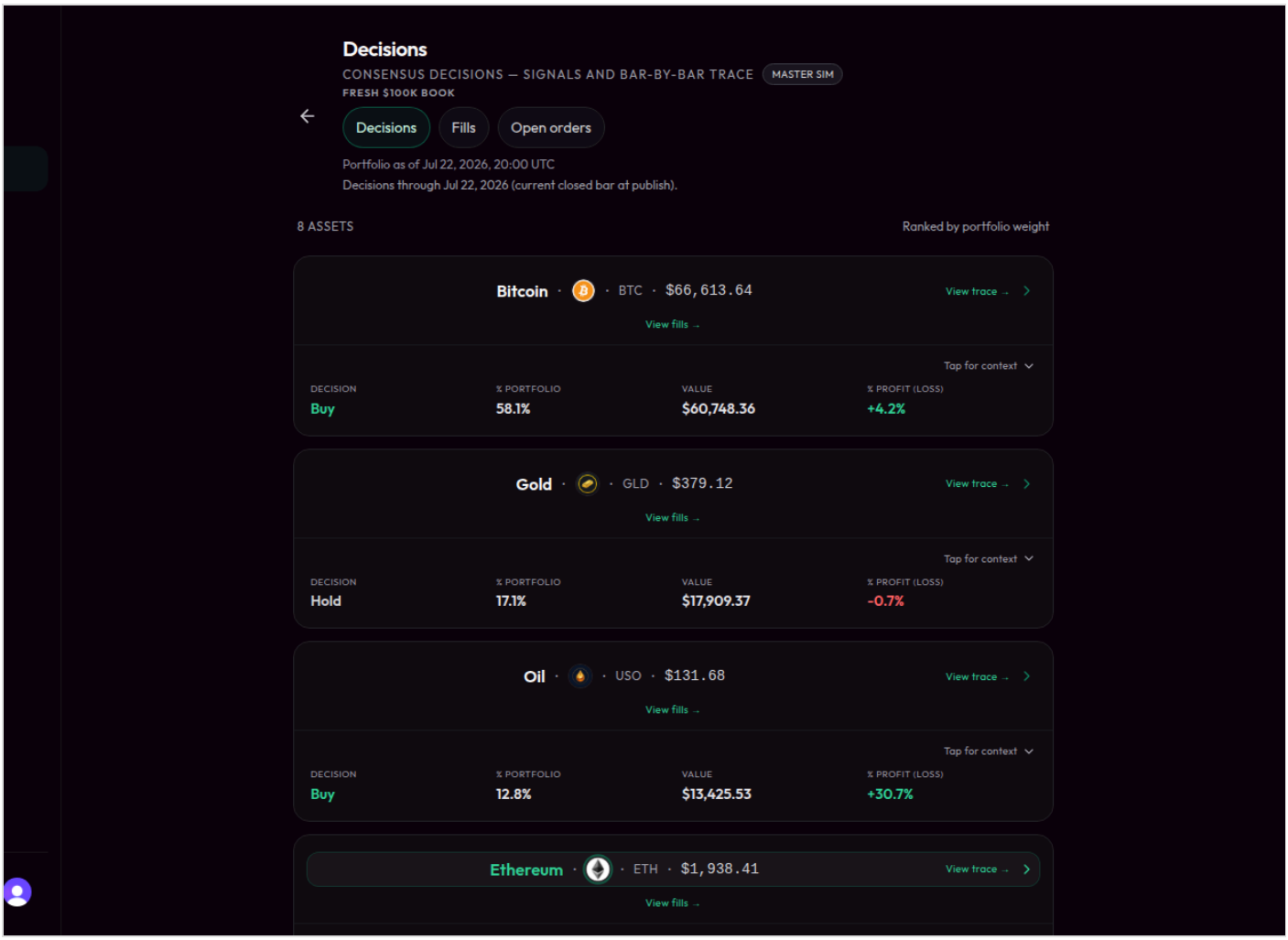
1. Headline metrics for the Fresh $100K Book
2. Freshness / which closed bar the view reflects
3. Paper ledger lens — institutional validation track, not your brokerage

Common confusion: Proof and Fill Ledger show SagaHalla’s institutional paper validation book, not your own brokerage account or a promise of your results.

A.6 Core surfaces: Decisions & explainability

Decisions is where you open a specific buy / sell / hold item and read **why** the Oracle reached that state: confidence, conviction, thresholds, constraints, and supporting context (the explainability trace).

- Timing follows your entitlement (A.2).
- A timing / freshness notice in the app tells you which closed bar you are seeing.
- Explainability is structural context — not a recommendation tailored to you.



Decisions list with consensus buy/hold/sell and portfolio weights

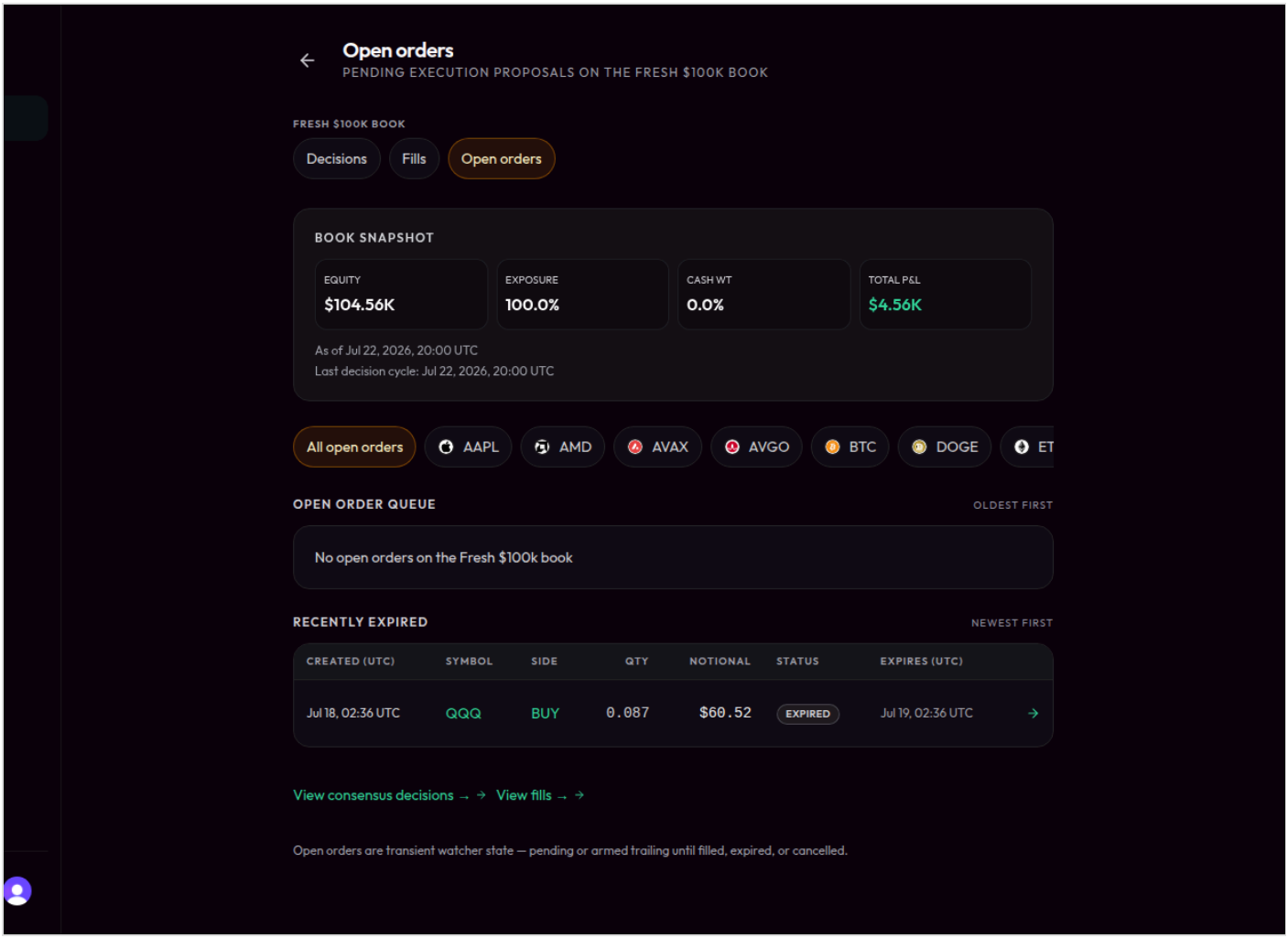
Decisions — open an asset for the explainability trace.

1. Freshness banner / closed-bar label (trust this over assumptions)
2. Consensus action (buy / hold / sell) on each asset
3. **View trace** — open explainability; timing follows your entitlement (A.2)

A.7 Core surfaces: Open Proposals

Open Proposals shows transient paper proposals still waiting — intended trades on the institutional paper book that have not yet filled or expired.

- Available on **Operator** and **Allocator**.
- This is **not** your personal brokerage order blotter.
- On Allocator with Manual Review Mode, a related **pending trades** queue on Allocate is where *you* approve or reject proposals for *your* book (Part D).



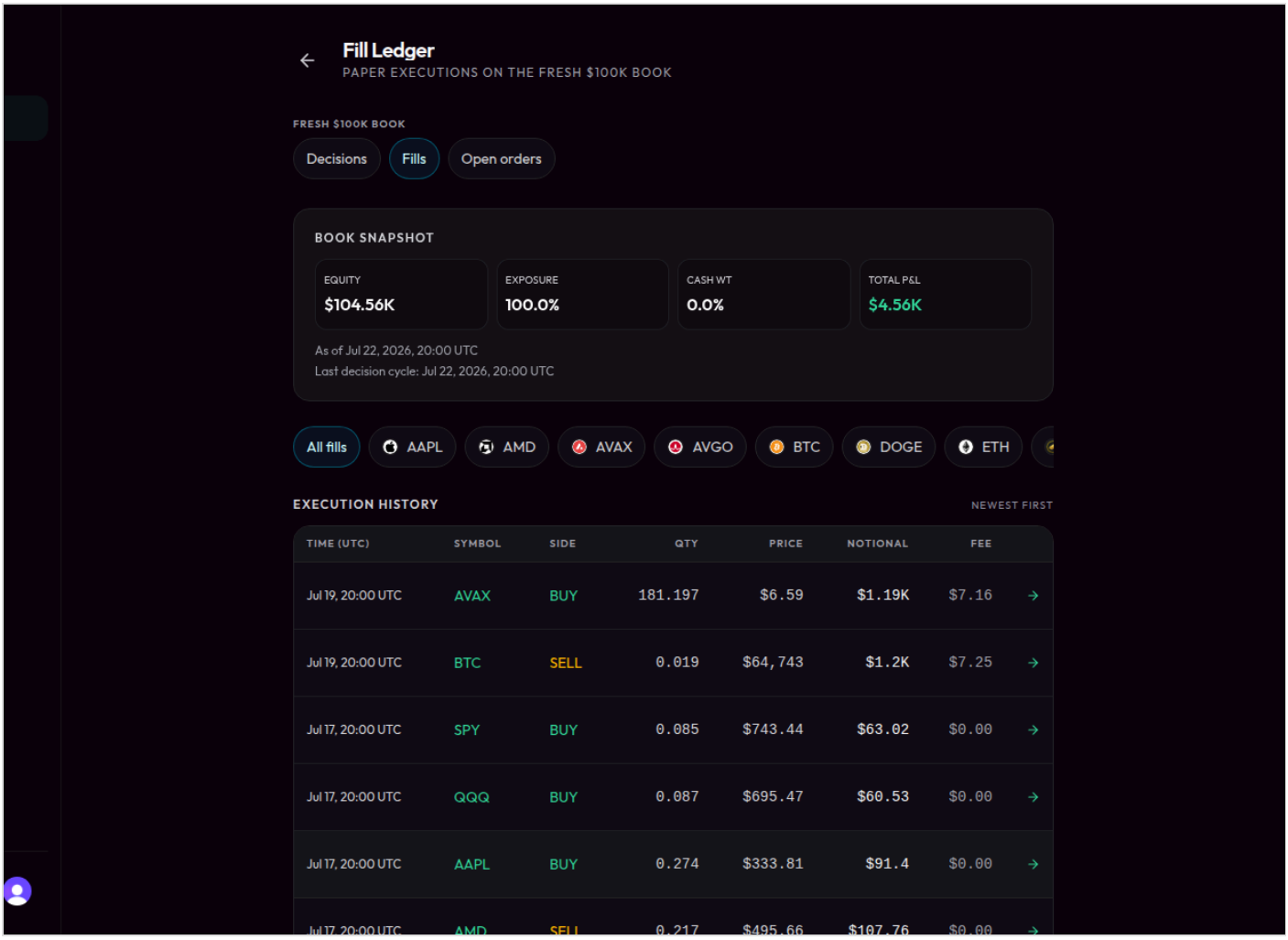
Open Proposals / open orders lens on the fresh paper book

Open Proposals — pending paper proposals on the institutional book, not your broker blotter.

A.8 Core surfaces: Fill Ledger

Fill Ledger is the institutional paper ledger of what has executed on the Oracle's paper book — fills, sizes, and related receipts.

- Available on **Operator** and **Allocator**.
- Use it to distinguish **what executed** (Fill Ledger) from **what is still waiting** (Open Proposals).
- It is not a statement from your live broker.



Fill Ledger execution history for the fresh paper book

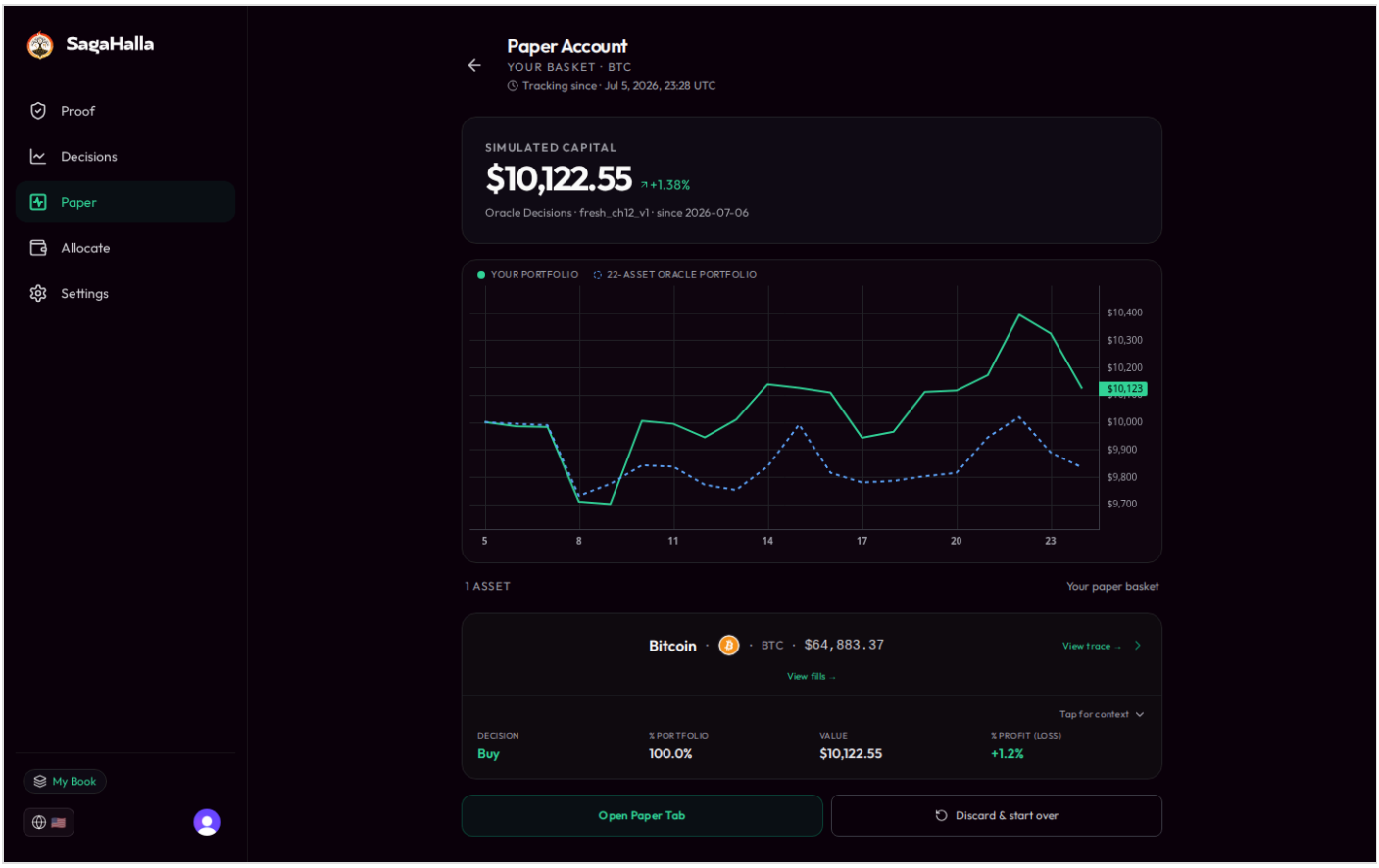
Fill Ledger — what executed on the institutional paper book.

A.9 Paper portfolio / sandbox overview

The **paper portfolio (sandbox)** lets you watch strategy behavior with **no real capital** at risk. Capabilities by tier (the single matrix for the whole handbook):

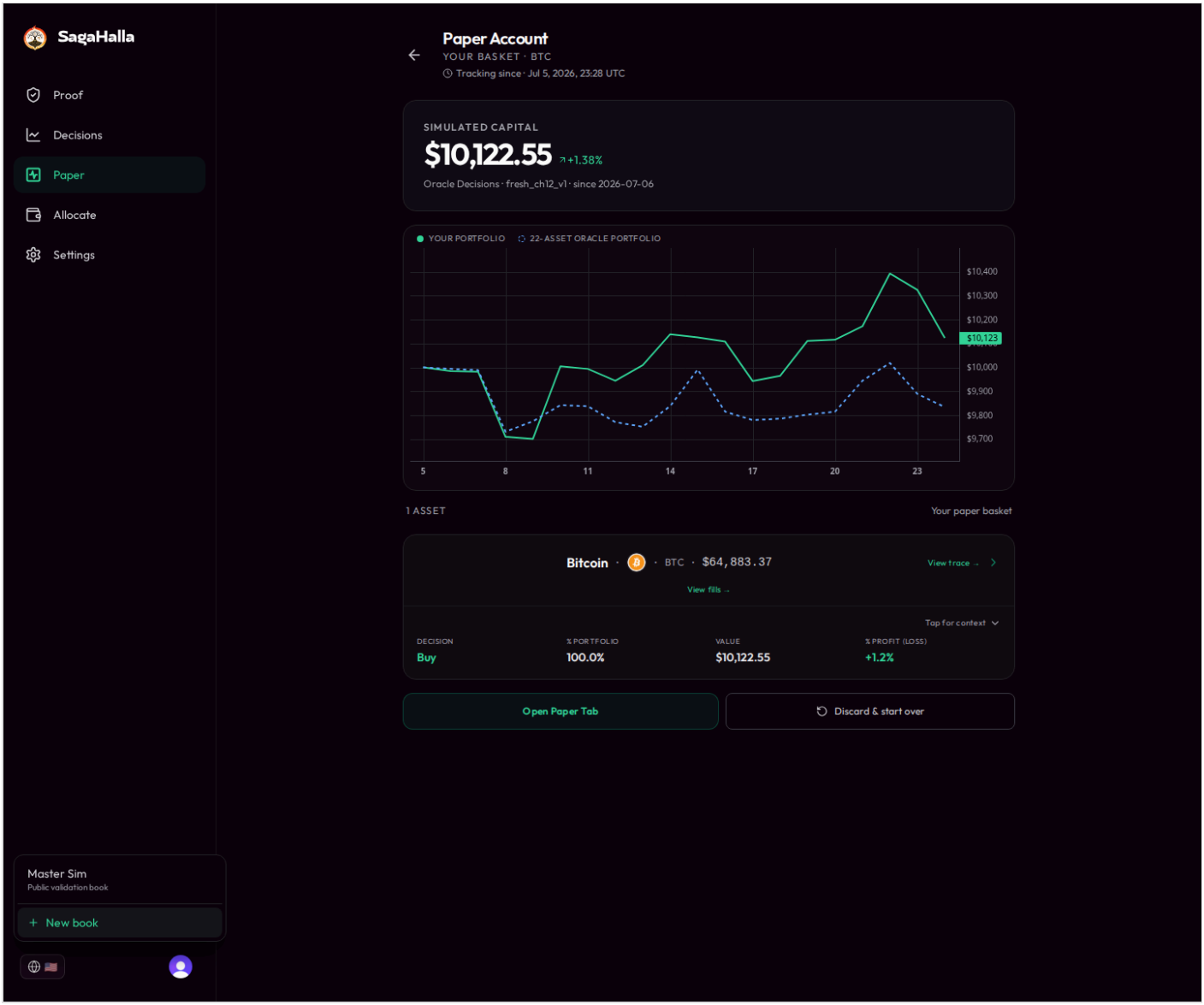
Capability	Validator	Operator / Allocator
Paper sandbox	**Available** — multi-book via **BookSwitcher** (Master Sim + your customer books)	**Available**
Forward observation	✅ Watch the selected book go forward	✅
Full history replay	❌	✅
Fill Ledger / Open Proposals (paper book)	❌	✅

Paper is simulated. Nothing in the sandbox places a real brokerage order. Use **BookSwitcher** to select **Master Sim** (institutional matrix / `dry_run` track — no customer vault) or a customer book. Each customer book binds **exactly one** venue (**Alpaca** or **Coinbase**). Paper connections in **Settings → Beta Integrations** (when offered) are **Alpaca paper only** — Coinbase has no paper/sandbox connect path today — and are sandbox-only and distinct from Allocator **live** broker connect. Broker panels always act on the book selected in **BookSwitcher** and name that book in the panel header; Master Sim holds no vault keys, so its panels stay disabled.



Paper sandbox account with simulated equity curve and basket

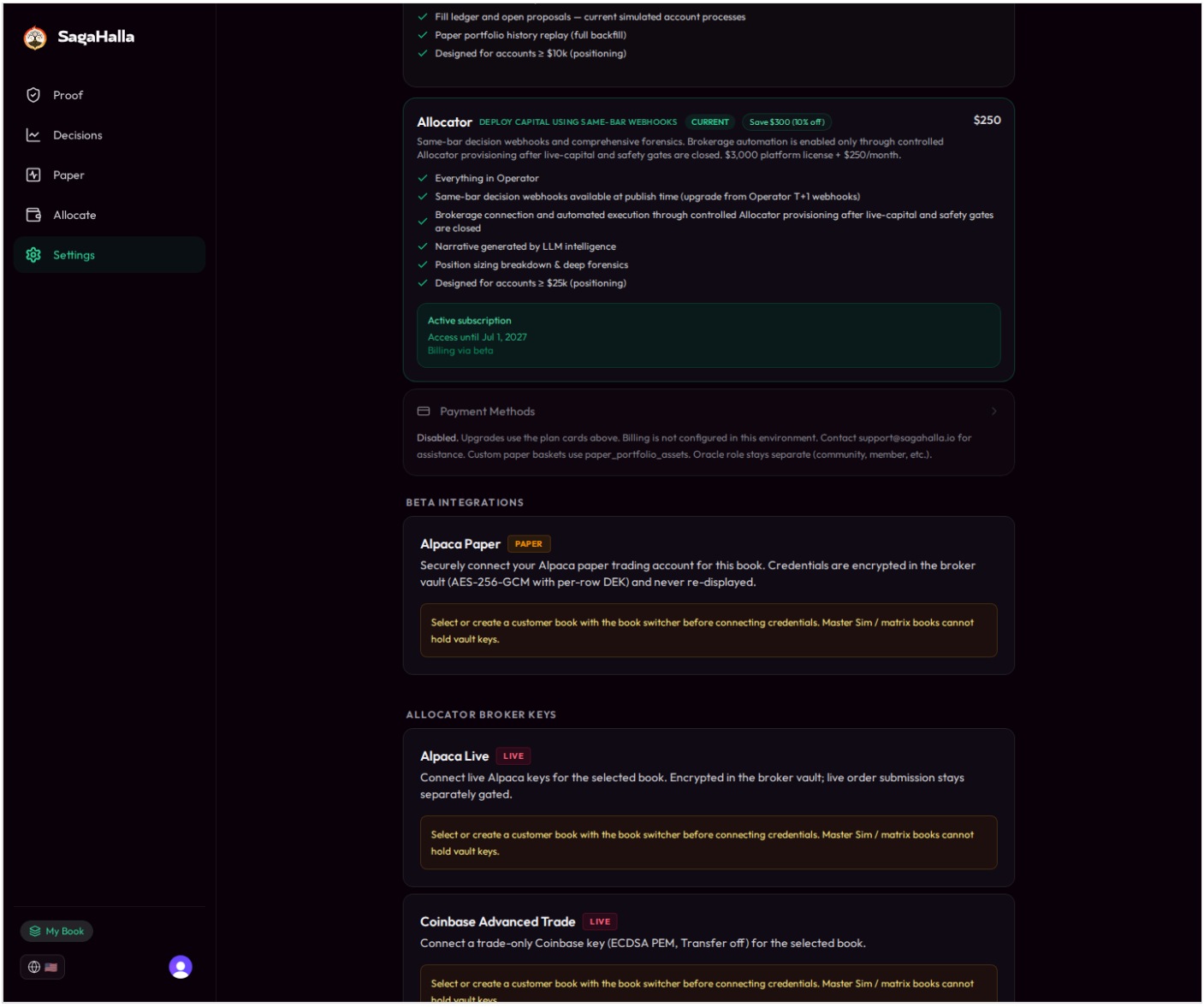
Paper sandbox — simulated capital only; no real brokerage orders. BookSwitcher is in the left nav footer.



BookSwitcher open — Master Sim and customer books

*BookSwitcher — choose **Master Sim** (institutional matrix; no vault) or a customer book (one venue each).*

1. Active book label in the nav footer
2. **Master Sim** = public validation / matrix track
3. Create or select a customer book before vault connect



Settings → Beta Integrations paper broker path

Settings → Beta Integrations — Alpaca paper/sandbox broker connect when offered; distinct from Allocator live connect on Allocate.

A.10 User responsibilities & non-advisory posture

SagaHalla is a **user-directed** structural analytics and automation platform. You choose the instruments, rules, thresholds, sizing, exposure limits, and execution permissions. SagaHalla shows you structural analytics and, when you configure and enable it, alerts you or acts only on the permissions **you** authorized.

SagaHalla is **not**:

- a robo-adviser, discretionary manager, or managed account;
- a source of personalized investment advice, suitability determinations, or recommendations for your specific circumstances;
- a broker-dealer, and it does not custody your assets.

Nothing SagaHalla displays is a recommendation to buy, sell, or hold any instrument. Decisions and signals are **general structural analytics** for supported instruments.

What SagaHalla does not guarantee. SagaHalla does not guarantee, and you should not rely on:

- profits, returns, or protection against loss;
- order fills, fill prices, or freedom from slippage;
- data accuracy, completeness, or continuous availability (uptime);
- suitability of any instrument, rule, or configuration for you;
- that a signal, decision, or automation will behave as expected in any given market condition.

You are responsible for your own trading decisions and for the configuration you choose. Connecting a brokerage or webhook alone does not start trading — live routing has additional explicit gates.

In addition, you are responsible for:

- Keeping API keys, webhook secrets, and account identifiers private
- Using the product within the entitlements and disclosures shown in-app
- Not representing SagaHalla output as personalized advice to others

In product terms, also assume:

Area	Not guaranteed
Markets / decisions	Completeness for every instrument every day; uninterrupted publish cadence
Webhooks	Instant delivery; zero duplicates; zero retries; broker fill quality
Paper ledger	Identity with any live broker’s fills or your personal P&L
Brokerage automation (Allocator)	Fills, prices, or outcomes; connecting keys alone never starts live trading

A.11 Disclosures & acknowledgments (overview)

Depending on what you enable, the app may require you to acknowledge:

Acknowledgment	When it appears
Paper / sandbox first-run disclosure	Early sessions for paper evaluation
Webhook general acknowledgment	Before creating / using decision webhooks
Webhook use-and-risk (execution-oriented formats)	Before enabling TradingView-compatible / execution-ready formats
Standing Instruction acknowledgment	Before leaving Manual Review for advanced automation (Allocator)
Broker / live-submission confirmations	Before storing live keys or enabling live submission intent

Always read the in-app acknowledgment text at the time of acceptance — that wording is authoritative.

A.12 Support expectations

Channel	Use for
Settings → Send Feedback	In-app bugs, confusion, product feedback
support@sagahalla.io	Access, billing, entitlement, or issues you cannot file in-app

Support helps with product access, entitlement mismatches, and clarifying how surfaces work. Support is **not**:

- investment advice or trade coaching;
- a discretionary manager of your account;
- a guarantee of response times suitable for time-critical trading.

Support response times are not guaranteed for time-critical trading decisions.

A.13 Getting started (sign-in, first session)

1. **Sign in** at app.sagahalla.com (or via the Sign in link on the marketing site). Use the email associated with your account or invite.
2. **Read first-run disclosures** (paper / sandbox boundaries) before exploring.
3. **Open Proof** — confirm you can read the headline validation-book metrics and freshness.
4. **Open Decisions** — pick any item and find the explainability trace.
5. **Explore the paper sandbox** — understand simulated state with no real capital.
6. **Visit Settings** — plan / billing, support, push notifications; Operator and Allocator also see webhooks (and related integrations when entitled).

A timing notice in the app tells you which closed bar you are seeing. If something looks stale or blocked, pause and contact support rather than assuming the last value is current.

Part B — Validator

Validator is the free signed-in tier. It is for evaluating the Oracle — paper proof, Markets, and full decision explainability — without webhooks or live brokerage automation.

B.1 Who Validator is for

Choose Validator when you want to:

- See whether the Oracle's paper track and decision traces are credible
- Learn the product surfaces (Proof, Markets, Decisions, paper sandbox)
- Stay on a **$0** plan while you evaluate before integrating or deploying capital

Validator is **not** for piping decisions into your stack (that is Operator) or for native brokerage automation (that is Allocator).

B.2 T+1 in-app decision cadence

On Validator, in-app **Markets**, **Decisions**, and related consensus views show the **prior closed bar (T+1)** relative to the Oracle's latest publish — not the bar that just published.

What you see	Validator meaning
Timing / freshness notice	Which closed bar the current view reflects
Buy / sell / hold markers	Based on the T+1 bar
Explainability trace	Full structured context for that T+1 decision

Why T+1? Validator is an **information / evaluation** experience on the prior closed bar. Operator and Allocator see the latest published closed bar **in-app**; see A.2 for the full matrix.

If the freshness banner and the numbers disagree, trust the banner, pause, and contact support — do not assume you are looking at today's just-published bar.

B.3 Proof in the app

Use signed-in **Proof** (app home) to evaluate the Oracle — the Fresh $100K Book metrics and paper context described in **A.5**. That is the Validator Proof experience. Marketing pages may mention outcomes for discovery; they are not a substitute for the signed-in app view.

B.4 Paper sandbox: forward vs history replay

On Validator, use the sandbox to understand simulated behavior **forward** with no real capital. **Full history replay** and the Operator paper surfaces (Fill Ledger / Open Proposals) unlock when you upgrade. The full capability matrix is in **A.9**.

B.5 Free-tier limits

At a glance, Validator includes and excludes:

Included (Available)	Not included
Proof landing, Markets, Oracle status	Decision webhooks
Full decision explainability **at T+1**	At-publish in-app decisions
Fresh $100K Book view	Paper history replay
Paper sandbox (forward)	Fill Ledger / Open Proposals
Equity outcome context on Proof	Allocator forensics (sizing breakdown / narrative)
	Brokerage automation / live connect

Price: **$0**. What **Settings** shows for your plan is authoritative.

B.6 Upgrade path to Operator / Allocator

You want…	Upgrade to
Decision webhooks into Slack / Discord / your stack; paper history + Fill Ledger / Open Proposals; at-publish **in-app** decisions	**Operator**
Everything in Operator **plus** at-publish webhooks and native brokerage automation (Alpaca or Coinbase per book)	**Allocator**

Pricing for each plan is in A.3. Entitlements are additive. Start on Validator, add Operator when you are ready to integrate, then Allocator when you are ready to deploy capital under explicit permission modes and live gates (Parts C–D).

Do **not** upgrade because a single decision looks attractive. Upgrade when the timing, integration, or automation controls match the workflow you actually intend to run.

Upgrade and billing controls live in **Settings**. What Settings shows for your account is authoritative.

Part C — Operator

Operator adds **integration**: decision webhooks, paper history and paper book surfaces, and **at-publish in-app** decisions — while outbound webhooks stay on **T+1** so your stack receives a delayed information feed.

Deep schema, code samples, and retry details live in the **Webhook integration guide** (in-app: Settings → Webhooks → Integration guide). This Part summarizes operations; it does not replace the in-app Integration guide.

C.1 Who Operator is for

Choose Operator when you want to:

- Watch the Oracle **in-app on the latest closed bar**
- Pipe daily **decision artifacts** into Slack, Discord, n8n, Make, Zapier, Pipedream, or your own HTTPS endpoint
- Use paper history replay, Fill Ledger, and Open Proposals

Operator is **not** native brokerage automation (Allocator). If you automate trades off webhooks yourself, you accept the use-and-risk posture in C.10.

C.2 T+1 vs Allocator at-publish (timing)

Surface	Operator	Allocator
In-app Markets / Decisions	**At publish**	At publish
Decision webhooks	**T+1** (prior closed bar)	**At publish**
Brokerage automation	—	At publish

Operator webhooks are deliberately **delayed** relative to in-app Decisions so integration is an information feed, not a same-bar execution signal. Full matrix: A.2.

C.3 Decision webhooks overview

Decision webhooks `POST` a signed JSON **decision artifact** to a URL you register.

- Carry decisions only — not trade fills, broker confirmations, or intraday ticks
- **Operator cadence:** T+1
- **Formats:** `standard` (notification / workflow) and, when enabled for your account, `tradingview` (execution-oriented; see C.8)
- Entitlement: Operator and Allocator

Area	Availability
Self-serve setup (create, test, delivery log, rotate secret)	**Available** in Settings → Webhooks
Production webhook delivery	**Available** — schema, cadence, and retry details are in the Integration guide

C.4 Create / test / delivery log / rotate secret

1. Go to **Settings → Decision webhooks**.
2. Accept the one-time **webhook general acknowledgment** if prompted.
3. **Create webhook:** name, HTTPS callback URL (HTTP is allowed only for `localhost` during local testing), events (`SIGNAL_buy` / `SIGNAL_sell` / both — shown as **BUY signals** / **SELL signals** in the UI), asset filter (`*` or a list), and format.
4. **Copy the signing secret** — shown **once** at creation. Store it securely.
5. **Send test** and confirm a `POST` arrives.
6. Open the **delivery log** for status codes, timestamps, and truncated errors.
7. **Rotate secret** when needed — the new secret is shown once; previous signatures stop validating immediately.

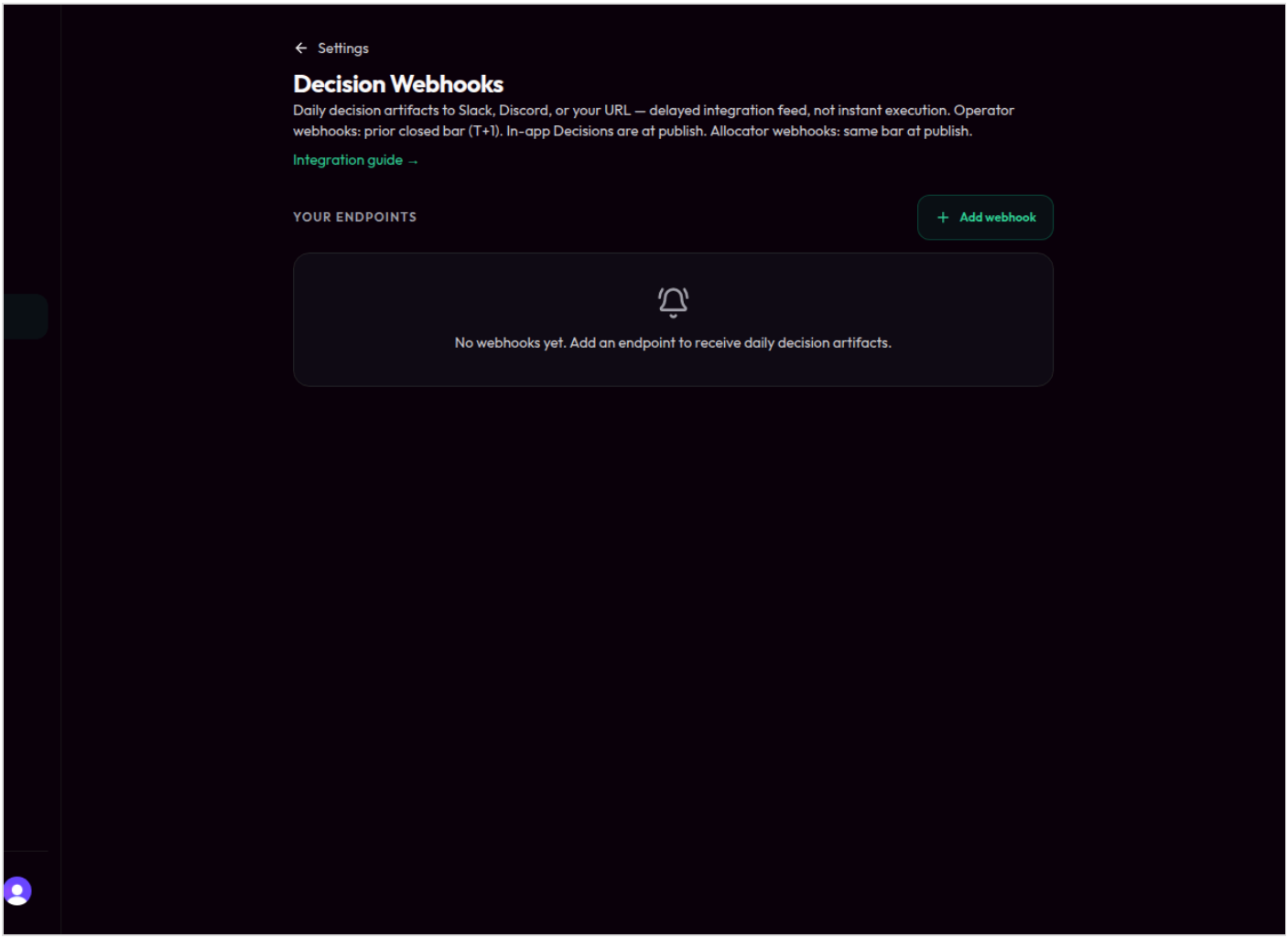
Limits: up to **10** webhooks per account (see integration guide if this changes).

First webhook test (checklist)

1. Create an HTTPS endpoint you control (or a provider inbound URL).
2. Create the webhook in Settings → Webhooks; copy the signing secret once.
3. Click **Send test** and confirm a `POST` arrives.
4. Verify the signature before you trust the body (C.5).
5. Open the **delivery log** for status codes and truncated errors.

Slack / Discord: almost always need a **transform** step — they expect their own message shape. Do not raw-forward the SagaHalla JSON to a Slack/Discord webhook URL and expect a readable channel post. See C.7.

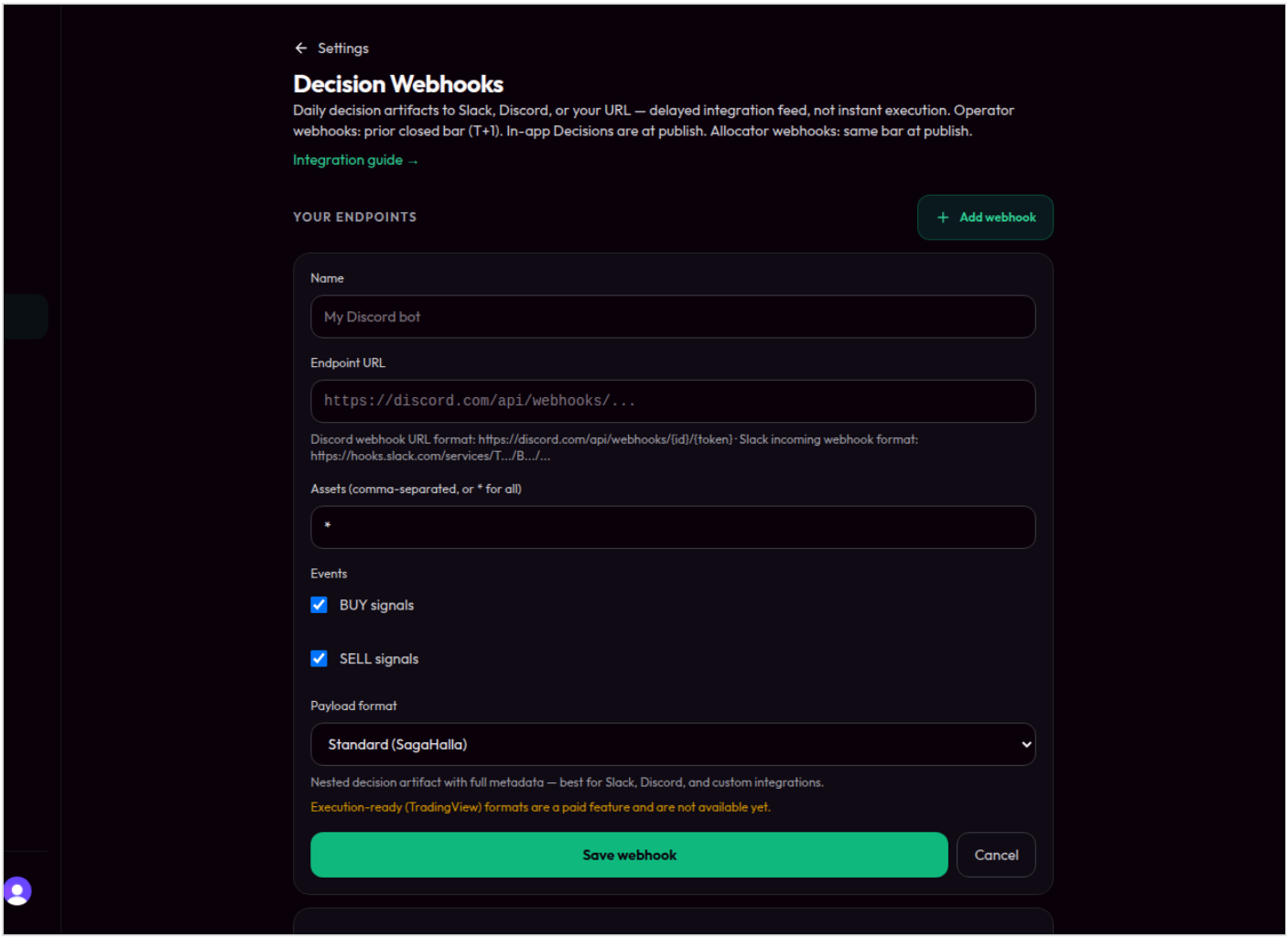
Minimal payload shape and field names: **Webhook integration guide** (in-app).



Decision Webhooks list — empty endpoints state after acknowledgment

Decision Webhooks — manage endpoints after acknowledgment.

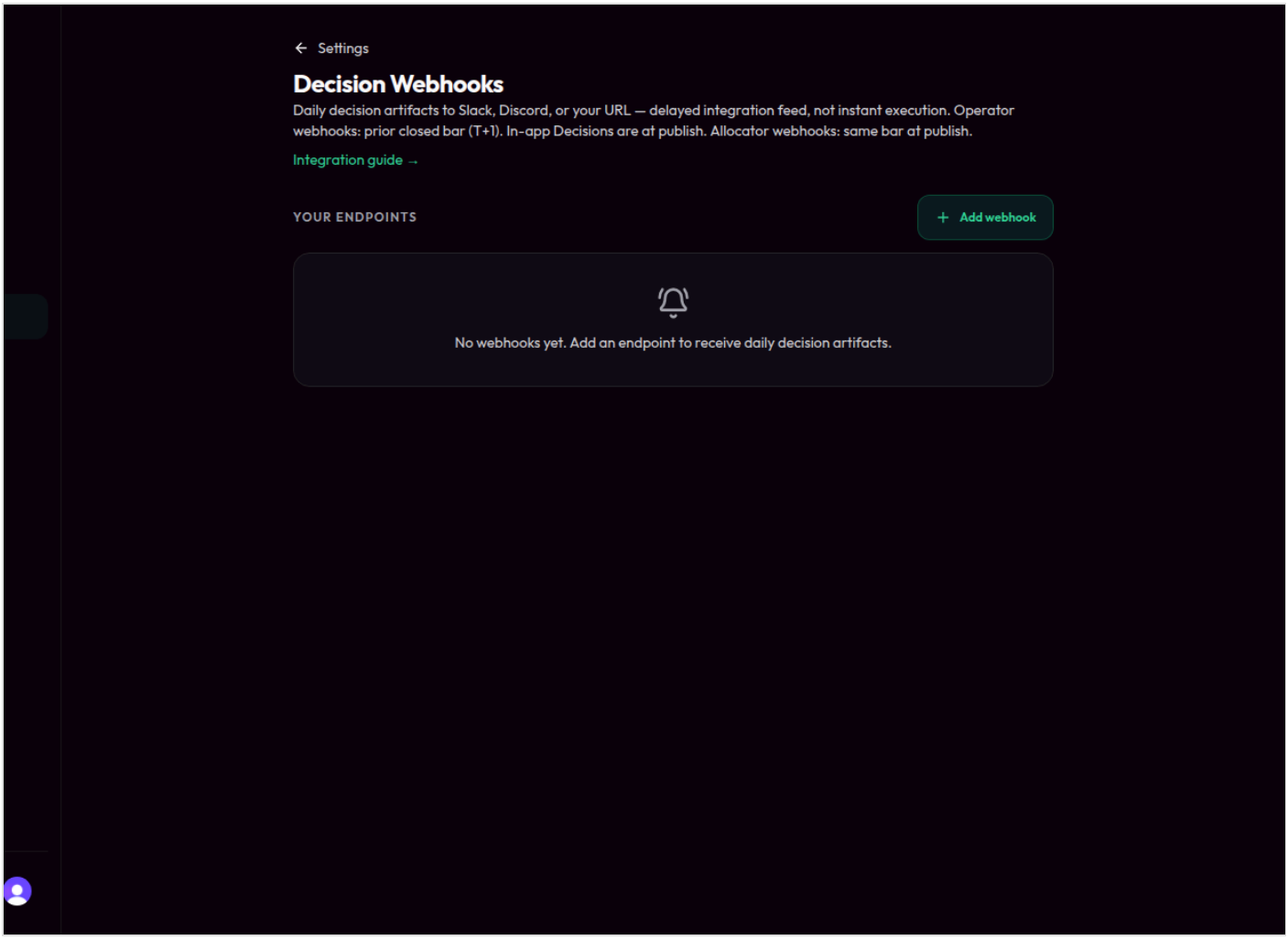
1. One-time webhook acknowledgment
2. Endpoint list / empty state
3. Add webhook entry point



Create webhook form with Discord/Slack URL and event filters

Create webhook — copy the signing secret when shown (once only).

1. Callback URL and event filters (`SIGNAL_buy` / `SIGNAL_sell` — shown as BUY / SELL in UI)
2. Format selection
3. Signing secret — copy when shown; once only



Decision Webhooks endpoints area (delivery log lives per endpoint)

Webhook delivery — status codes and truncated errors after traffic.

1. Per-endpoint delivery log
2. Status codes and timestamps
3. Truncated errors after traffic

C.5 Signature verification (HMAC concept)

Every delivery includes signature headers so you can prove the payload came from SagaHalla. Verify with a **constant-time** comparison and reject anything missing or invalid.

- Prefer Standard Webhooks headers (`webhook-id`, `webhook-timestamp`, `webhook-signature`) when present
- An older `X-Saga-Signature` (HMAC-SHA256 of the raw body) may also be sent for compatibility

No major automation host verifies HMAC for you automatically — add an explicit verify step (or a verifying gateway) before you act on a delivery.

Full algorithms and code samples: **Webhook integration guide**.

C.6 Retries, duplicates, stale signals

Topic	Behavior
Retries	Failed deliveries retry with backoff (see integration guide for attempt count / timing). Success = HTTP 2xx.
Duplicates	Treat deliveries as **at-least-once**. Design receivers to be idempotent (e.g. key on webhook id / decision identity).
Stale signals	T+1 means the bar is already prior. Your automation and broker add more lag. Never assume “just published” on an Operator webhook.
Outages	If SagaHalla or your endpoint is down, deliveries may delay or exhaust retries. Check the delivery log; do not invent fills.

C.7 Notification & workflow providers

Route SagaHalla webhooks to the provider’s **inbound URL**, verify the signature first, then transform for the destination.

Provider	Best for
Pipedream	Developers; verify in a Node/Python step
n8n (self-hosted)	Keeping decisions on your infra
Make	Visual / no-code scenarios
Zapier	Broad app catalog (often paid for webhooks)
Slack / Discord	Channel notifications (**require a transform** — raw SagaHalla JSON is not their message shape)
Custom HTTPS	Your own service

Recommended pattern: **verify → transform → act**. Treat the raw SagaHalla payload as the trusted source.

C.8 Execution-oriented formats (TradingView-compatible)

When enabled for your account, you may select a **TradingView-compatible** format for receivers such as TradersPost, SignalStack, or 3Commas.

Format	Role
`standard`	Decision artifact for notifications / workflows
`tradingview`	Flat alert-style JSON for execution receivers

Availability: Execution-ready formats may be **paid / gated**. Enabling them requires the **use-and-risk** acknowledgment (C.10). Check Settings → Webhooks for what your account can select.

C.9 Sizing fields & subscriber patterns

Execution-oriented payloads may include several sizing lenses. They answer different questions:

Lens	Fields	Meaning
Execution	size basis / fraction / provenance	How SagaHalla expressed the trade (e.g. fraction of cash or position)
Allocation	current / target / delta weight	Strategic intent on SagaHalla's book
Reference absolutes	quantity / notional	Size on **SagaHalla's** book — not yours

Common subscriber patterns (descriptions, not recommendations):

- **Cash-deploy receivers** — map fraction fields into percent-of-cash / position orders
- **Target-weight rebalancers** — read target / delta weight into your own optimizer
- **Information consumers** — read action + context; no automation

Exact field names and schemas: **Webhook integration guide**.

C.10 Information-only / use-and-risk posture

Decision webhooks are an **information feed**, not a trade-execution guarantee.

If you automate orders off webhooks, you accept at least:

- **Timing lag** — publish → delivery → your automation → broker
- **Unknown slippage** — fills at prices SagaHalla does not control
- **Signal ↔ portfolio misalignment** — a decision does not know your positions, cash, or risk limits

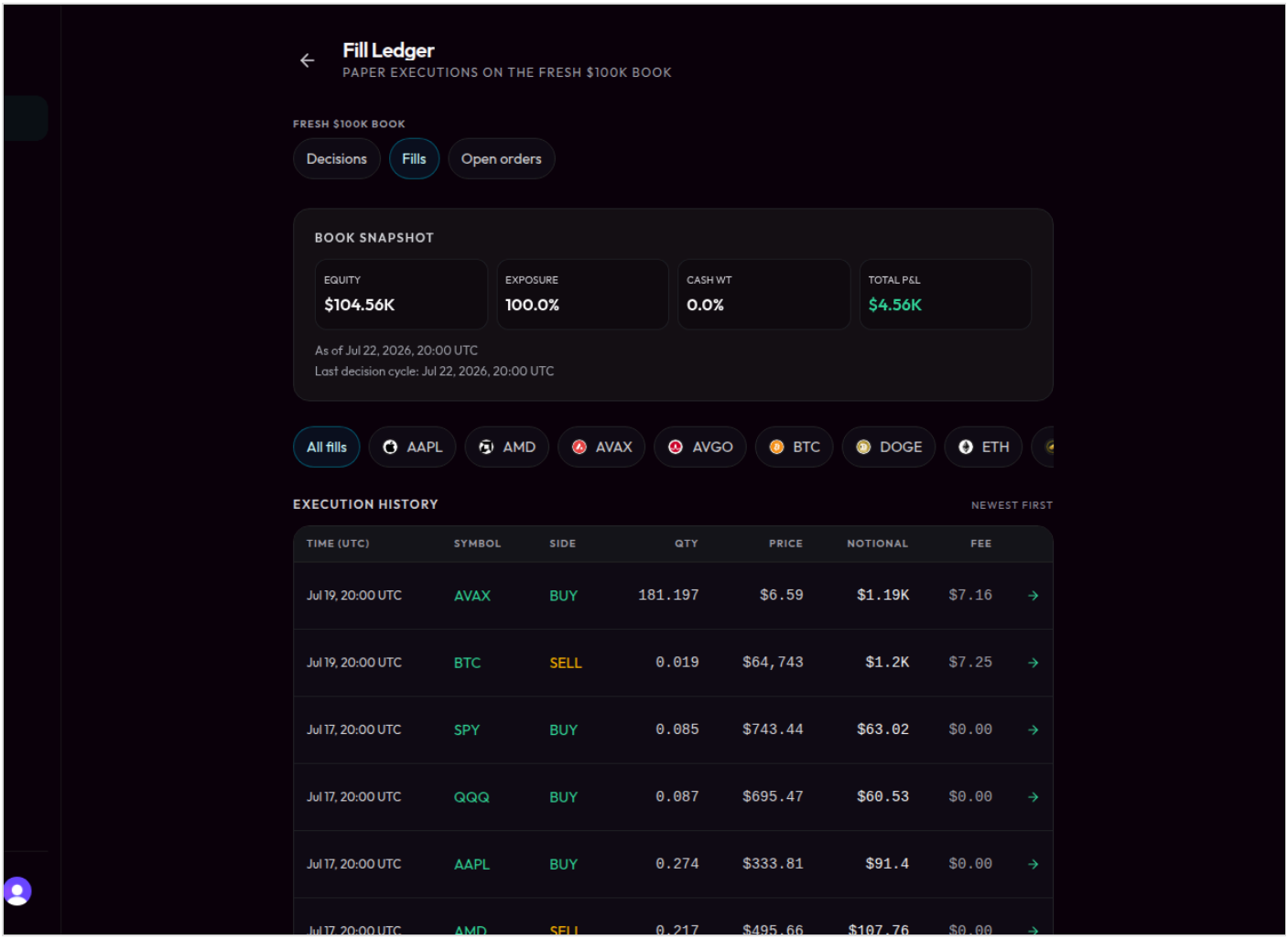
For server-side risk gates, sizing parity, and native brokerage controls, see **Allocator** (Part D). SagaHalla does not recommend trades; you choose the instruments, rules, and permissions you enable.

C.11 Paper book surfaces (Operator)

Operator includes paper book surfaces beyond Validator:

Surface	Role
Paper history replay	History from when the paper book started, where enabled
Fill Ledger	What executed on the institutional paper book
Open Proposals	What is still waiting

These are audit surfaces for the Oracle's paper track — not your live brokerage. If a screen is marked **Coming soon** in-app, trust the app.



Fill Ledger as Operator paper book audit surface

Operator paper book — Fill Ledger / Open Proposals are audit surfaces, not live brokerage.

C.12 Manual review vs standing authorization

Operator's primary "permission" boundary is **webhook acknowledgment + format choice** (information vs execution-oriented). Native per-trade **Manual Review** and **Standing Instruction** modes for brokerage routing are **Allocator** controls — see **D.12** for the definitions and the Standing Instruction acknowledgment.

On Operator alone: you authorize outbound information (and, if enabled, self-directed execution receivers). SagaHalla does not place broker orders for you on the Operator tier.

C.13 Failure modes (Operator path)

Failure	What to do
Webhook 4xx/5xx or exhausted retries	Fix the endpoint; use Send test; read the delivery log

Failure	What to do
Signature failures after rotate	Update your verifier to the new secret immediately
Stale or unexpected bar	Check T+1 vs in-app at-publish; do not force a trade
Provider transform bugs	Keep verify-first; fix the transform; SagaHalla cannot reconcile your downstream tool
Partial automation / missed day	Assume gaps happen; reconcile against in-app Decisions before acting

Missed delivery does not mean missed trade. Operator webhooks are an information feed — they do not place or cancel broker orders. A failed or exhausted delivery is a notification gap, not a brokerage event. Reconcile against in-app Decisions before you act.

Operator webhooks do not provide broker fill reconciliation — that is outside this tier’s product boundary.

C.14 Kill switches / pause paths (Operator)

Action	Effect
Disable or delete a webhook	Stops deliveries to that URL
Rotate secret	Invalidates prior verifiers until you update them
Turn off your automation provider	Stops *your* side from acting (SagaHalla may still deliver if the webhook is live)
Downgrade / remove Operator entitlement	Webhook APIs become unavailable for the account

Operator **cannot**: engage an Allocator brokerage kill switch, pause live Alpaca submission, or revoke Standing Instruction — those are Part D controls.

Part D — Allocator

Allocator is the **deploy** tier: everything in Operator, plus **at-publish** webhooks, Allocator forensics where enabled, and **native brokerage automation** on your account — under explicit permission modes and live gates.

> **Allocator automation is Available for entitled Allocator accounts (Alpaca or Coinbase Advanced Trade, one venue per book). Connecting broker keys never starts live trading by itself — live routing requires your permission mode and all safety gates in D.12. Begin in Manual Review with live submission off and the kill switch engaged.**

SagaHalla does **not** custody your assets. Funds stay at your broker.

D.1 Who Allocator is for

Choose Allocator when you want to:

- Receive decisions **at publish** in-app and on webhooks
- Connect a supported broker (**Alpaca** or **Coinbase**) for automation on **your** account — one venue per SagaHalla book
- Use Manual Review (default) or Standing Instruction (advanced opt-in) for how orders are authorized
- Access Allocator forensics (sizing breakdown, intelligence narrative) where the product surfaces them

Allocator is **not** a managed account, robo-adviser, or suitability service. You remain responsible for instruments, rules, thresholds, sizing, exposure limits, and permissions you choose.

D.2 At-publish cadence

On Allocator, both **in-app** Decisions and **webhooks** use the **latest published closed bar** (at publish) — still end-of-day, not intraday.

Surface	Allocator
Markets / Decisions	At publish
Decision webhooks	At publish
Brokerage automation	At publish (only if live gates allow)

Compare with Operator (in-app at publish, webhooks T+1) in A.2 / C.2.

D.3 Screening vs allocation vs operation

Keep three jobs distinct:

Job	Meaning
Screening	Which instruments / structures are admissible under the Oracle’s structural rules

Job	Meaning
Allocation	How weight / sizing intent is expressed for an admissible set under your constraints
Operation	How (and whether) that intent becomes orders on **your** broker under your permission mode and live gates

SagaHalla's analytics inform screening and allocation context. **Operation** requires your authorization mode and the safety model in D.12. Do not treat a screen or a decision artifact as an order.

D.4 Structural admissibility & pillar interpretation

Structural admissibility is the Oracle's way of saying whether a structure passes its integrity / pillar checks for the closed bar — not whether it is "good for you." Pillars and related scores are general structural analytics (the non-advisory posture in A.10 applies), and passing or failing admissibility does not by itself place a trade.

Read pillar / admissibility context on Decisions explainability as *why the Oracle's machinery classified the bar that way* — then apply your own process.

D.5 Screening methodology & portfolio construction

At a high level:

1. The Oracle evaluates supported instruments on closed bars
2. Structural screens and portfolio construction rules produce decision / proposal context
3. Your book's constraints and permission mode determine what can be routed

Your day-to-day loop: **understand the decision → authorize per your mode → observe gates and broker results**.

D.6 Policy selection, constraints, risk budgets, benchmarks

You direct the book. Where the product exposes them, you (not SagaHalla as adviser) select or accept:

- Instruments and universes you allow
- Constraints, exposure limits, and stop / pause conditions
- Risk budgets and benchmark references you care about for *your* monitoring

SagaHalla does not certify that your policy is suitable. Change or revoke permissions when your policy changes.

D.7 Mandate versioning & rebalancing

A **mandate** is your standing configuration for how automation may act — instruments, limits, and permission mode — versioned over time so you can see what was in force when a decision was authorized.

- Rebalancing follows closed-bar decisions and your authorization mode
- Changing a mandate does not rewrite history; it changes what may happen next

- Pause / kill switch (D.12) stops routing without necessarily deleting the mandate

Trust the labels shown in-app for mandate status.

D.8 Native brokerage automation overview

Native automation means SagaHalla may route **your preconfigured order instructions** to **your** connected broker API when your permission mode and live gates allow — at at-publish timing.

Point	Rule
Custody	Broker holds funds; SagaHalla does not
Keys	Trading-scoped API keys in the **broker vault** (AES-256-GCM with a per-row DEK); never shown back; customer keys are **vault-only** (not customer `.env` files)
Default	Live submission **off**; kill switch **engaged** (safe defaults)
Permission	Manual Review by default; Standing Instruction only with acknowledgment
Fail-closed	Without a valid vault connection for the book's venue, live routing does not fall back to process environment keys

This is distinct from Operator webhook self-execution (C.8–C.10), where *you* wire a third-party receiver. (Treasury / institutional ops may use separate operator tooling; customers never configure broker secrets via env files.)

Area	Availability
Allocate UI (Alpaca or Coinbase connect, modes, approval queue, kill / live controls)	**Available** for Allocator entitlement
Live order routing	**Available** only when **all** live gates clear (D.12). Connection and UI visibility never equal live trading

D.9 Supported venues

Alpaca and **Coinbase Advanced Trade** are supported. Other venues remain **Coming soon**.

Venue	Status	Notes
Alpaca	**Available**	Live or paper connect + automation when gates clear
Coinbase	**Available**	Advanced Trade; ECDSA PEM; Transfer disabled. **Live keys only** — no paper/sandbox connect path

Venue	Status	Notes
Kraken	**Coming soon**	
Schwab	**Coming soon**	
Multi-venue on one book	**Coming soon**	One book ↔ one venue today

You may create **many** SagaHalla books. Each book binds **exactly one** venue and **exactly one** broker account for that venue. Use **BookSwitcher** to select Master Sim (institutional matrix / `dry_run` — simulated; **no** customer vault) or a customer book. Paper pilots in Settings (when offered) are **Alpaca paper only**, are sandbox-only, and are **not** Allocator live connect. Coinbase is a live-only connect and therefore an Allocator path.

D.10 Connecting a broker (Alpaca or Coinbase)

Open and fund your account on the broker's site — the broker owns KYC and key management. Then connect for the **selected book** in Allocate (or Beta Integrations for Alpaca paper pilots). Connect panels name the target book in their header — confirm it matches the book you selected before pasting a secret. Connecting never starts live trading by itself (D.12).

Before you connect (checklist)

- You understand this is **not** custody or advice (A.10).
- You have **trading-only** API keys — never withdrawal / transfer-capable keys.
- You understand live routing remains **off** until all gates clear.
- You intend to begin in **Manual Review**.
- You know how to pause, disconnect, and rotate keys.
- The selected book's bound venue matches the broker you are connecting.

Alpaca

Step	Where	Rule
1. Open / fund an Alpaca **individual** brokerage account	alpaca.markets	Complete Alpaca KYC on Alpaca's site.
2. Create **trading-scoped** API keys	Alpaca dashboard → API keys (Alpaca docs)	**Enable trading. Disable withdrawal / transfer.** SagaHalla never needs withdrawal access. Alpaca does not expose scopes via API — create trading-only keys yourself.
3. Keep the secret offline	Your password manager	The secret is shown once in Alpaca; SagaHalla will never show it back after connect.

Coinbase Advanced Trade

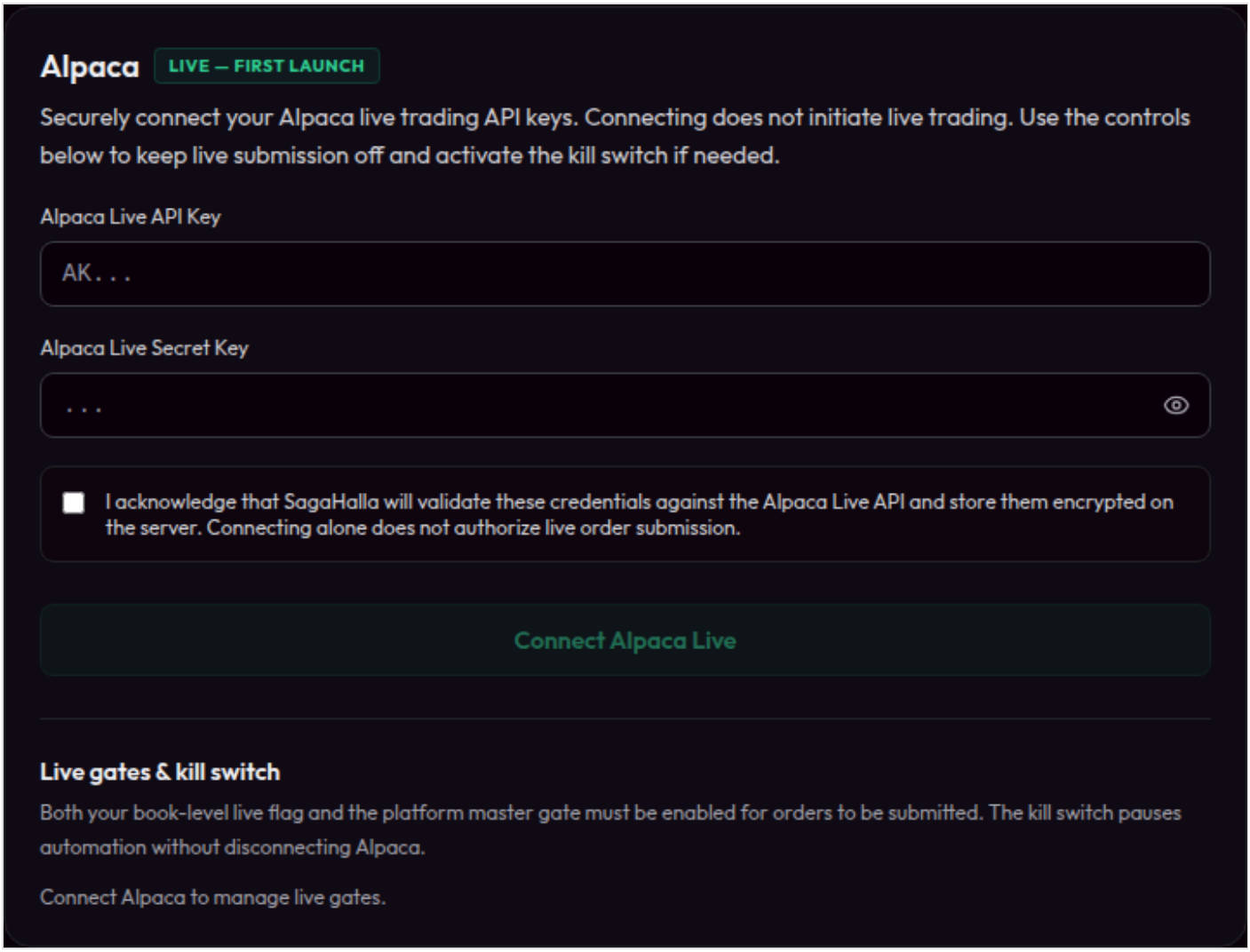
Step	Where	Rule
1. Open / fund Coinbase Advanced Trade	coinbase.com / Coinbase Developer Platform	Complete Coinbase KYC on Coinbase's site.
2. Create a **trade-only** API key	CDP → API keys	**View + Trade only. Transfer must be off** (`can_transfer=false`). SagaHalla rejects transfer-capable keys at connect.
3. Use an **ECDSA (ES256) PEM** secret	Key download / CDP	Paste the **full multi-line PEM**. **Ed25519 keys are rejected** (current SDK accepts ECDSA only). Keep the PEM in your password manager; SagaHalla never shows it back.

Do **not** paste withdrawal- or transfer-capable keys. If you already created broad keys, revoke them at the broker and mint trading-only keys.

In the SagaHalla app

1. Select the customer book in **BookSwitcher** (not Master Sim).
2. Open **Allocate** (live / Allocator) or **Settings → Beta Integrations** (paper pilot).
3. Enter credentials for the book's bound venue; acknowledge storage/validation terms.
4. Keys are validated and stored in the encrypted broker vault (AES-256-GCM + per-row DEK) — **never** in the browser, never in a customer `.env`, and never shown back after submit.
5. The panel shows connection status and metadata only (account label, last verified) — never the secret.

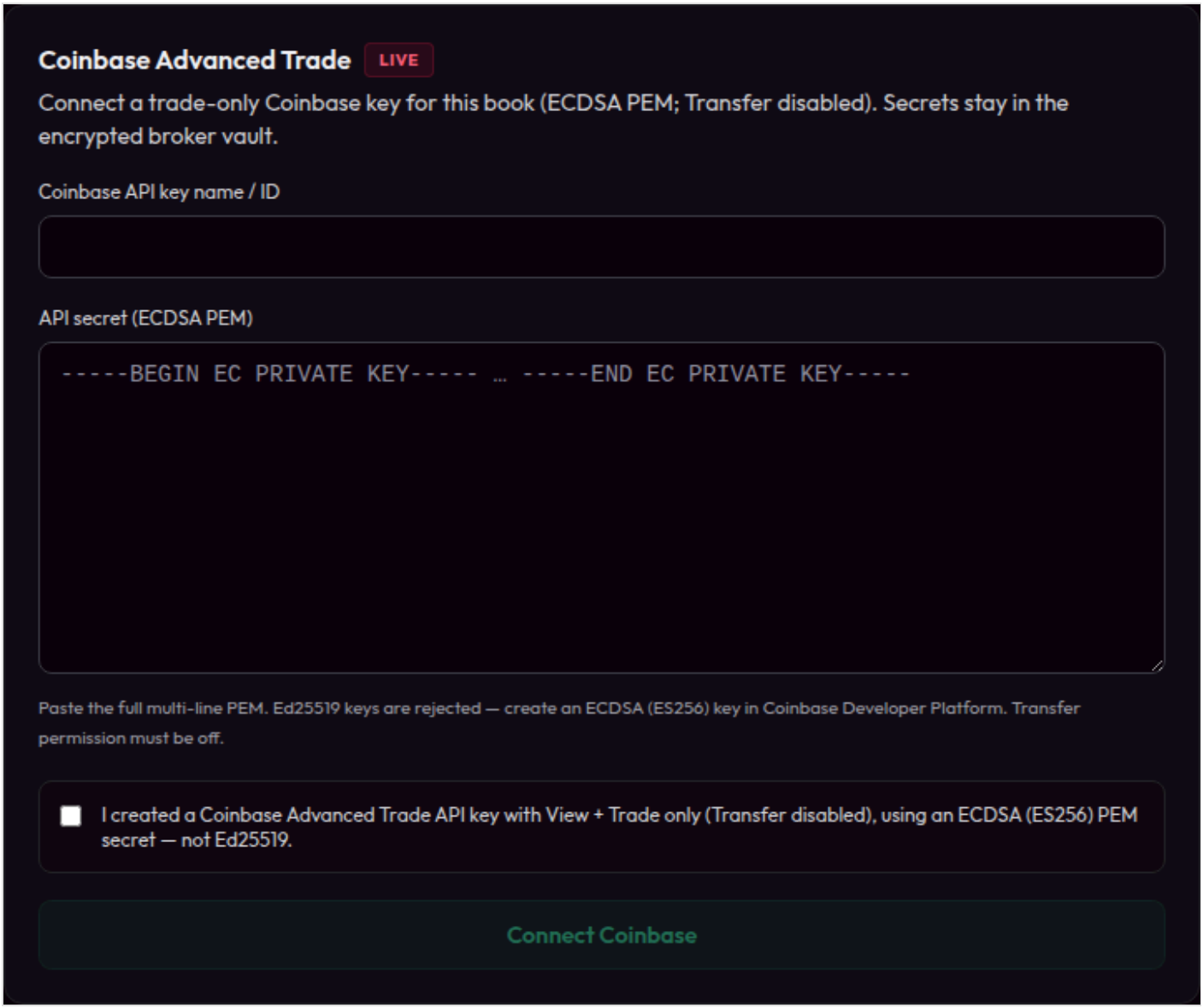
Disconnect removes credentials from SagaHalla; rotate keys at the broker to invalidate them everywhere.



Allocate Alpaca and Coinbase connect panels — live first launch

Allocate — Alpaca live connect plus Coinbase Advanced Trade panel. Connecting vaults keys; it does not turn on live routing.

1. Connection does **not** enable routing
2. Alpaca key/secret fields (secret never shown back)
3. Coinbase panel (ECDSA PEM / Transfer-off); select a customer book in BookSwitcher first
4. Acknowledgment that connect ≠ live submission



Allocate Coinbase Advanced Trade connect — ECDSA PEM

Allocate — Coinbase Advanced Trade connect (trade-only ECDSA PEM; Transfer disabled). Same vault rules as Alpaca.

1. Customer book required (Master Sim cannot hold vault keys)
2. Paste full multi-line ECDSA PEM (Ed25519 rejected)
3. Connect ≠ live submission

D.11 Platform license & execution authorization

Allocator has two commercial components — a recurring subscription and a one-time platform license (an activation filter for deployers, in addition to the monthly). Amounts are in the pricing table in A.3.

What unlocks what:

Gate	Unlocks
Allocator entitlement (+ license when required)	Tier features and Allocate surfaces
Broker connect (Alpaca or Coinbase)	Credential vault entry for that book — **not** live routing
Execution mode	Manual Review vs Standing Instruction
Book live-submission intent	Your preference to allow live routing
Platform master live gate	SagaHalla-wide allow for live submission
Kill switch cleared	Automation not paused

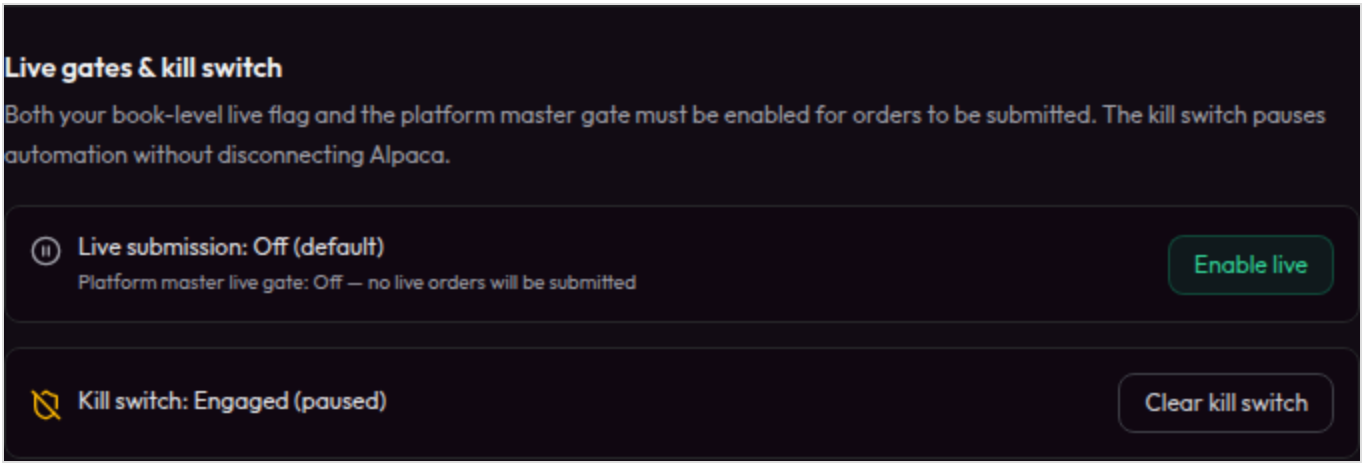
Checkout and license collection follow what **Settings** and your onboarding agreement show for your account.

D.12 Going live: safety model & kill switches

Live routing requires **all** of the following (safe defaults in parentheses). **If any one of these is false, live routing should not occur.**

#	Gate	Safe default
1	Valid vault live connection for the book’s venue (Alpaca or Coinbase)	Disconnected — **fail-closed** (no env-key fallback)
2	Execution mode set	**Manual Review**
3	Your book **live submission** intent	**Off**
4	Platform **master live gate**	May be off during platform maintenance
5	**Kill switch**	**Engaged** (paused)

Control	Effect
Turn live submission off	Stops new live intent without deleting keys
Engage kill switch	Pauses automation; connection can remain
Disconnect broker	Removes vault credentials for that book / venue



Allocate live gates and kill switch with live submission off by default

Live gates & kill switch — live submission off and kill switch engaged by default.

1. **Live submission: Off** (default)
2. **Kill switch: Engaged** (paused)
3. Connection can remain while routing stays off

Execution mode & approval queue

No order is routed without an explicit permission mode that **you** select:

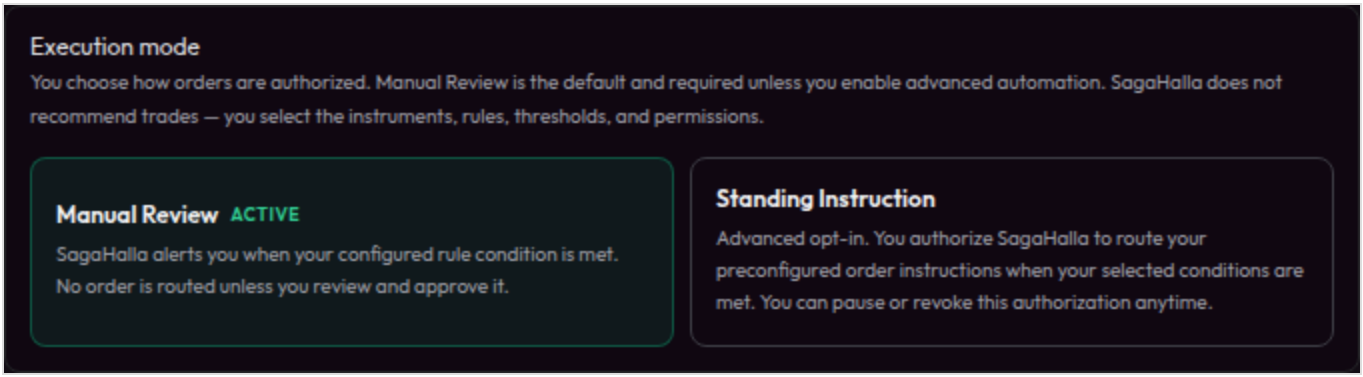
- **Manual Review (default).** SagaHalla alerts you when your configured rule condition is met. No order is routed unless you review and approve it — one at a time or all at once.
- **Standing Instruction (advanced opt-in).** You may authorize SagaHalla to route your preconfigured order instructions when your selected conditions are met. You remain responsible for the instruments, rules, thresholds, sizing, exposure limits, and permissions you choose, and you can pause, modify, or revoke authorization at any time.

Enabling Standing Instruction requires this acknowledgment:

> **I understand that SagaHalla is not recommending this trade. I selected the instrument, rule, threshold, sizing, exposure limit, and execution permission. Orders may execute automatically according to my standing instruction.**

Connecting a brokerage or webhook alone does not start trading — live routing has additional explicit gates.

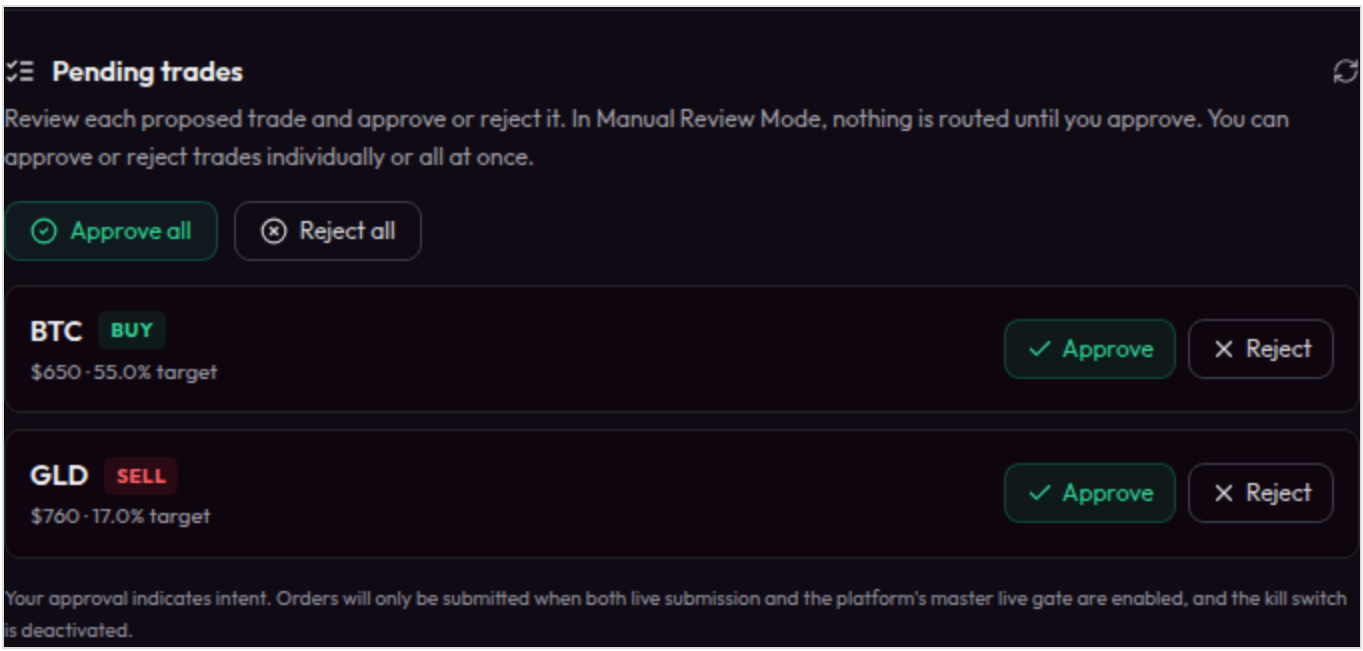
In **Manual Review**, use the Allocate **pending trades** queue to approve or reject proposals **individually** or **all at once**. Your approval records intent; orders still submit only when live gates allow.



Execution mode — Manual Review active vs Standing Instruction

Execution mode — Manual Review is the default; Standing Instruction needs acknowledgment.

1. **Manual Review** active (default)
2. Standing Instruction requires explicit acknowledgment
3. Mode choice ≠ live routing (gates in D.12 still apply)



Pending trades approval queue with approve/reject per trade and approve-all

Approval queue — approve or reject pending trades in Manual Review.

1. Per-trade approve / reject
2. Approve-all / reject-all
3. Approval records intent; live routing still needs cleared gates (D.12)

D.13 Day-to-day operation & reports

Activity	Where
Daily decisions / explainability	Decisions (at publish)
Pending authorizations	Allocate → pending trades (Manual Review)

Activity	Where
Connection / gates / kill switch	Allocate
Webhooks (at publish)	Settings → Webhooks
Forensics (sizing / narrative)	Decisions trace / Allocate forensics entry points when enabled

Pause anytime via kill switch or live-off. Re-enable only when you intend live routing again. Treat your broker statements as the custody record of truth.

D.14 Automation risks specific to live capital

Beyond Operator webhook self-execution risks (C.10), live Allocator automation adds:

Risk	Why it matters
Real capital	Fills debit/credit your broker account
Gate misconfiguration	Live-on + cleared kill switch + master gate = routing may proceed per your mode
Standing Instruction	Orders may route without per-trade clicks once conditions you set are met
Broker / API failures	Rejects, partial fills, outages — SagaHalla does not guarantee fills
Mandate drift	Stale limits or forgotten Standing Instruction after you change intent
Key scope mistakes	Never grant withdrawal permissions on API keys

You can pause or revoke at any time. If unsure, keep **Manual Review**, live submission **off**, and kill switch **engaged**.

Part E — Choosing a path & next steps

E.1 Path chooser

You want to…	Start with
Evaluate decisions and the explainability trace	**Validator** → Start Here “Evaluating”
Pipe daily decisions into Slack / Discord / your stack	**Operator** → Start Here “Integrating”
Route decisions on your own broker under explicit gates	**Allocator** → Start Here “Preparing Allocator”; begin Manual Review

Entitlements are additive: evaluate → integrate → deploy.

E.2 Where to go next

Resource	Where
This handbook	https://sagahalla.com/oracle-handbook
App	app.sagahalla.com
Webhook integration guide	Settings → Webhooks → Integration guide
Allocate (Alpaca or Coinbase / modes / gates)	App → **Allocate**
Support / feedback	Settings → Send Feedback · support@sagahalla.io

Appendices

X.1 Glossary

Term	Meaning
Closed bar	A completed daily bar the Oracle decides on — not a live tick
T+1	Prior closed bar relative to the latest publish
At publish	Latest closed bar, right after the daily pipeline publishes
Consensus	The Oracle’s combined model view for a closed bar, expressed as a decision
Decision	End-of-day consensus artifact (action context + explainability) for an instrument
Proof	Institutional paper validation surface
Fresh $100K Book	The Oracle’s fresh institutional paper validation book shown on Proof / Decisions — not a customer account
Open Proposals	Pending paper proposals (Operator and Allocator)

Term	Meaning
Fill Ledger	Institutional paper fills (Operator and Allocator)
Structural admissibility	Oracle integrity / pillar checks for a bar — not suitability for you
Pillars	Structural checks behind admissibility (analytics, not advice)
Mandate	Your versioned automation configuration (instruments, limits, mode)
Decision webhook	Signed outbound decision artifact to your URL / stack
Manual Review	Default mode — approve trades before routing
Standing Instruction	Advanced opt-in — route per your preconfigured permissions
Platform master live gate	Platform-wide switch required before any live order submits
Kill switch	Pauses Allocator automation without disconnecting
Platform license	One-time Allocator activation fee (amount in A.3)
Entitlement	Your plan tier and the surfaces it unlocks
Live-submission intent	Your preference to allow live order routing (default off)
Broker vault	Server-side AES-256-GCM storage (per-row DEK) for trading-scoped API keys; secrets never re-displayed; customer keys are vault-only (not `.env`)
BookSwitcher	UI to select Master Sim or a customer book; each customer book binds one venue
Master Sim	Institutional matrix / `dry_run` paper track — simulated; no customer vault keys

X.2 Version at a glance

Field	This handbook (v1.4 / 2026-07-25)
App	app.sagahalla.com
Webhook formats	`standard` and `tradingview` — see Settings → Webhooks → Integration guide
Webhook general acknowledgment	Required before creating / using decision webhooks
Webhook use-and-risk acknowledgment	Required before execution-ready / TradingView format
Standing Instruction acknowledgment	Required in Allocate before leaving Manual Review
Paper / sandbox disclosure	Shown on first paper sessions
Allocator	**Available** for entitled accounts (Alpaca or Coinbase per book); live routing only when all gates in D.12 clear