CHARTING · PLUGIN AUTHORS · OPERATORS

AXIS DOCUMENTATION

Complete AXIS manuals — architecture, plugins, UI, worker

2026-09-14 DIN A4 · HTTPS://HOOX.SH/DOCS OPEN SOURCE

CONTENTS

01 AXIS Documentation

02 End User

03 Installation

04 Quick start

05 Compose recipes

06 Manager and settings

07 PYNE Agent

08 Data Source Manager

09 On-Chain data

10 Script library

11 Strategy and results

12 Hyperparameter Optimisation

13 Drawings

14 Alerts

15 Interface gallery

16 Troubleshooting

17 Glossary

18 FAQ

19 Architecture

20 Architecture overview

21 Evaluation map

22 Architecture Decision Records

23 Topologies

24 State namespaces

25 Drawings and plot parity

26 Plugins

27 Plugin contracts

28 Plugin registry

29 Sources

30 Datasets

31 Streams

32 Engines

33 Storage

34 Dynamic plugin loader

35 PYNE Agent plugin

36 UI

37 Charting

38 Editor

39 Debugging

40 UI shell

41 Scripts (indicators & strategies)

42 Results and strategy

43 Store

44 Worker

45 Worker runtime

46 Durable Objects

47 Worker data plane

48 Worker auth
49 Worker bindings
50 DevOps
51 Local development
52 AXIS CLI
53 Build and serve
54 Desktop (Tauri)
55 Cloudflare deployment
56 VPS demo topology
57 CORS and origins
58 CI and testing
59 Feature atlas
60 Open capability gaps
61 Legacy shell
62 Testing reference
63 Plugin examples
64 Changelog

AXIS DOCUMENTATION

Open charting PWA — orthogonal sources, streams, engines, and storage. Own the axes; swap the engine.

AXIS DOCUMENTATION

AXIS is an installable charting PWA (v**2.5.0**) that treats **price**, **time**, and **calculation** as separable axes. Sources load history. Streams advance the present. Engines evaluate Pine Script™ via **PYNE**. Storage keeps your script library. Compose any lawful combination without a proprietary charting host.

AXIS workspace with BTCUSDT daily candles, watchlist, volume, RSI, and Pine editor

Pine Script™ and TradingView® are trademarks of TradingView, Inc. Cloudflare® is a registered trademark of Cloudflare, Inc. This project is independent and not affiliated with or endorsed by TradingView, Inc. or Cloudflare, Inc.

Abstract

AXIS is a charting PWA, not a language rewrite. Language fidelity lives in PYNE. The PWA contributes:

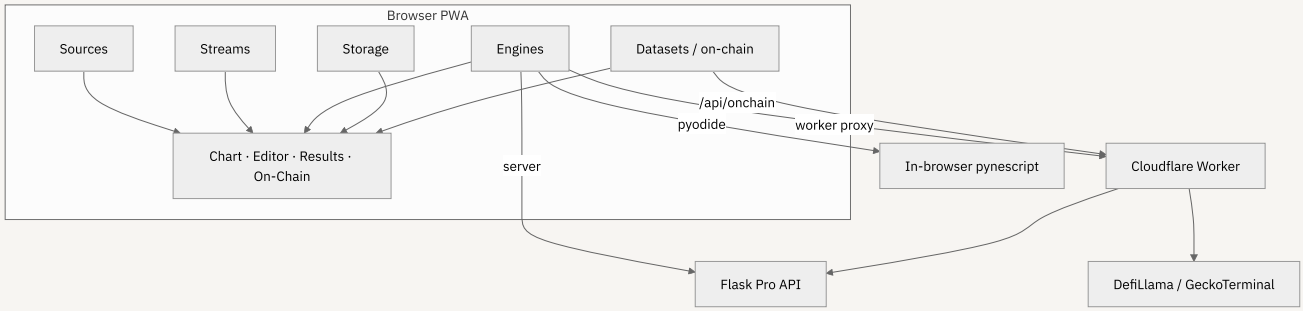
1. A **unified plugin registry** (`source | stream | engine | storage | dataset | component`)
2. A **Solid + Vite** product UI (chart, editor, results, manager, Workers Manager, on-chain panel) with a **mobile-first responsive shell** and **global keyboard shortcuts** (recordable, persisted)
3. Optional backends: local Flask, offline Pyodide, Cloudflare® Worker (+ DO/KV/D1/R2 + `/api/onchain` proxy)
4. **AXIS CLI** + optional **Tauri** desktop shell for operators

Invariant (AXIS ≠ engine): the shell never embeds a closed interpreter. Evaluation is always an engine plugin.

Product map

Track	Role	Start here
End User	Install, compose recipes, research workflows	End User hub
Architecture	ADRs, topologies, state namespaces	Architecture
Plugins	Contracts and built-in catalogs	Plugins
UI	Chart, editor, panels, store	UI
Worker	CF Pages + Worker data plane	Worker
DevOps	Build, CLI, VPS, CORS, CI	DevOps · AXIS CLI
Evaluation	Flask / Pyodide / Worker proxy — not PyneTS, not HOOX execution	Evaluation map
PYNE (separate)	Grammar & evaluator	PYNE docs

Conceptual model



Formal active set:

```
Active set = { source, stream, engine, storage }
Contract namespace: pynescript.axis.plugins.v1
App state: pynescript.axis.v1
On-chain datasets: kind dataset (+ source geckoterminal-ohlcv for pool candles)
```

Compose in one glance

Recipe	Source	Stream	Engine	Storage
Offline lab	`mock-walk`	`mock-poll`	`pyodide`	`local`
Desk research	`binance-rest`	`binance-ws`	`server` → Flask	`local`
AXIS at the edge	venue REST	venue WS / DO	`server` → Worker	`cloud`
On-chain TVL + CEX	`binance-rest` + DefiLlama dataset	venue WS	any	`local`
DEX pool candles	`geckoterminal-ohlcv`	`mock-poll`	any	`local`
Repo library	any	any	any	`git`

Full recipes: Compose recipes.

Infrastructure names (do not rename lightly)

Surface	Value
CF Pages project	`axis`
Pages URL	`https://axis.hoox.sh`
CF Worker script	`worker-axis`
Worker URL	`https://worker.axis.hoox.sh`
Health JSON service	`worker-axis`
Product brand	**AXIS**

Demo topologies

- Dev: Vite `:3000` + Flask `:5002` (`axis install` then `bun run dev`)
- Desktop: `bun run desktop:dev` (Tauri 2)
- Static: `bun run build` + `axis_pwa_server.py` `:8081`
- Worker: `axis dev worker` / wrangler on `:8787`
- Prod: `axis deploy` → Worker `worker-axis` (`https://worker.axis.hoox.sh`) + Pages `https://axis.hoox.sh`

See topologies, AXIS CLI, and local dev.

Offline & agent exports

Auto-generated like HOOX manuals (`bun run docs:exports`):

Kind	Path
Full-corpus LLM pack	`llm.txt`
LLM site map (llmstxt.org)	`llms.txt`
Track PDFs (A4)	`/exports/axis-*-manual.pdf`

See also

- Quick start
- Plugin contracts
- ADRs
- PYNE runtime

02 · ENDUSER

END USER

Install AXIS, compose source/stream/engine/storage recipes, and run research workflows without a proprietary chart host.

END USER

AXIS is an installable charting PWA: load history, stream the present, evaluate Pine Script™ through a pluggable engine, and keep a script library on disk, IndexedDB, cloud, or git. This track is for operators and researchers who use the AXIS—not for plugin authors or Worker deployers.

Language fidelity (grammar, builtins, runtime semantics) lives in PYNE. AXIS never reimplements the language; it **hosts** evaluation via engine plugins.

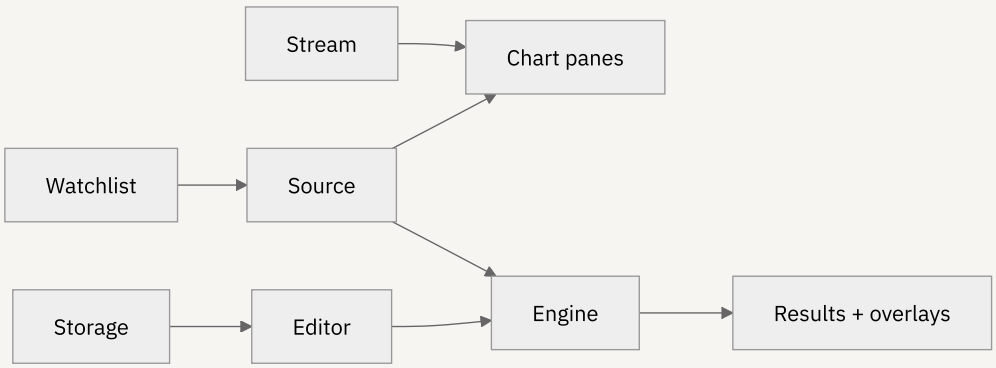
Abstract

You compose four axes:

Axis	Question it answers
Source	Where do historical OHLCV bars come from? (CEX venues or DEX pools)
Stream	How does the live bar advance?
Engine	Who evaluates the script?
Storage	Where do saved scripts live?
On-chain	Protocol TVL, DEX OHLCV, events — parallel plane (not a wallet)

Any lawful combination is valid. Offline lab, desk research against Binance + Flask, AXIS at the edge through a Cloudflare Worker, and a git-backed library are all first-class recipes—see Compose recipes.

Conceptual model



The shell (topbar, chart, editor, results, manager) is **AXIS**. Calculation is always `engine.run({ script, bars, config })`. Persistence of scripts is always a storage plugin.

Interface surface

Surface	Role
Topbar	Symbol, interval, source/stream/engine pickers, Load, Run, Live, On-Chain
Watchlist	Quick symbol switch + quote poll
Chart	lightweight-charts panes, drawings, trade markers, on-chain overlays, price-scale decimals
Editor	CodeMirror 6 Pine Script™ document (docked / popout)
Results	Events, Strategy, Optimise, Plots, Metrics, Raw + export
On-Chain	DefiLlama TVL, GeckoTerminal DEX, spike events, export / refresh
Manager	Plugin catalog, URL install (incl. **component** / PYNE Agent), Script Library
Workers Manager	Backend health probes, presets, install helpers
Settings	Endpoint, engine, storage, on-chain proxy notes
Desktop	Optional Tauri 2 shell (`bun run desktop:dev`)

Getting started

1. Installation — AXIS CLI, Vite dev, desktop, static `dist/`, or hosted PWA

2. Quick start — first load + first Run (`//@version=6`)
3. Compose recipes — offline, desk, multi-venue, CSV, edge, git

Guides

- Manager and settings — Plugins + **Workers Manager**
- PYNE Agent — natural language → scripts
- On-Chain data — TVL, DEX pools, events, Worker proxy
- Data Source Manager
- Script library — incl. v6 starter pack import
- Strategy and results
- Drawings
- Alerts — local price / Pine / on-chain; not HOOX orders
- Troubleshooting

Reference

- Glossary
- FAQ

Invariants (operator-facing)

1. **AXIS ≠ engine** — switching engine does not change the chart library; switching source does not change the language runtime.
2. **Active set is four-tuple** — source, stream, engine, storage are selected independently (with sensible defaults when source implies stream).
3. **API keys stay in the browser** — cloud storage and Pro keys are not uploaded by the shell except as request headers you configure.
4. **Chart OHLCV is ephemeral** — reload re-fetches for the topbar path; the **Data Source Manager** keeps a durable IDB `bars-cache` for deep history jobs.
5. Layout, script draft, and drawings survive in `pynescript.axis.v1`.

See also

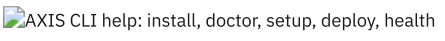
- Architecture — ADRs and topologies
- Evaluation map — Flask / Pyodide / Worker / not PyneTS
- AXIS CLI — install / doctor / deploy
- UI — chart, editor, store internals
- AXIS and HOOX
- Plugins — contracts and catalogs
- PYNE runtime — evaluation semantics

INSTALLATION

Install AXIS via the AXIS CLI, Vite+Solid dev, static dist PWA, desktop shell, or Cloudflare® Worker backends.

INSTALLATION

AXIS ships as the `axis` package (Solid + Vite), version **2.5.0**. Primary path is the product UI under `src/`. Legacy static shell (`main.js`, root `style.css`) is not the product UI—prefer `bun run dev` or built `dist/`.



Abstract

Operator modes:

Mode	Command surface	When
AXIS CLI (recommended ops)	`bun run axis` · `make axis-*`	Install / doctor / setup / deploy / health
Dev AXIS	`bun run dev` · `axis dev`	Day-to-day UI work
Desktop shell	`bun run desktop:dev`	Tauri 2 native window
Static PWA	`dist/` + `axis_pwa_server.py :8081`	Offline demo / VPS
AXIS at the edge	`axis deploy worker` (+ optional Pages)	Production data plane

Backend is optional when **Engine = Client-Side (Pyodide)** and sources/streams are mock or CSV.

Prerequisites

- **Bun ≥ 1.2** for package install, Vite, and the AXIS CLI
- **Python 3.10+** if you use Flask Pro API (`make run`) or `axis_pwa_server.py`
- Modern Chromium / Firefox / Safari (PWA install works best on Chromium)
- Optional: Cloudflare® auth (`CLOUDFLARE_API_TOKEN` or `wrangler login`) for Worker deploy
- Optional desktop: Rust + platform webview libs — Desktop (Tauri)

CLI-first bootstrap (recommended)

```
# From the AXIS monorepo root
bun install
cd packages/cli && bun install && cd ../..

axis install          # app + worker/ + CLI deps
axis doctor           # toolchain + optional CF auth; wrangler.toml warns until `axis setup`
axis setup            # ensure wrangler.toml + local D1 schema
# repo aliases (no global install): bun run axis:install / axis:doctor / axis:setup
# or: make axis-install && make axis-doctor && make axis-setup
```

Command	Repo alias (no global install)
`axis`	`bun run axis`
`axis install`	`bun run axis:install`
`axis doctor`	`bun run axis:doctor`
`axis setup`	`bun run axis:setup`
`axis deploy worker`	`bun run axis:deploy`
`axis health`	`bun run axis:health`
`axis …`	`make axis ARGS="…"` pass-through
`axis …`	`make axis-install` · `axis-doctor` · `axis-setup` · `axis-deploy` · `axis-health`

Full command surface: AXIS CLI.

Dev AXIS

```
# Terminal 1 — Pro API (server engine) from sister pyne repo
# Clone https://github.com/hoox-sh/pyne as ../pyne (local dir is sometimes still named pynescript)
make -C ../pyne run          # Flask :5002

# Terminal 2 — AXIS
bun run dev                  # Vite :3000
# or: bun run axis dev
```

Open `http://localhost:3000`. Default symbol `BTCUSDT`, engine often `server` with endpoint `http://localhost:5002` (or demo host `https://axis.hoox.sh`).

Desktop shell (optional)

```
bun run desktop:dev          # Tauri window + Vite HMR
bun run desktop:build        # native installers
```

See Desktop (Tauri).

Edge Worker (local)

```
bun run axis dev worker      # wrangler :8787
# or: cd worker && bun run dev
```

In AXIS: open **Workers Manager** (topbar activity icon) or Settings → set **Backend URL** to `http://127.0.0.1:8787`. Production Worker is `worker-axis` at `https://worker.axis.hoox.sh` — see topologies.

Deploy checklist (prod)

```
axis setup --github-client-id Ov23li… --remote-d1
axis secret put ADMIN_TOKEN
axis secret put EXTERNAL_BACKEND
axis deploy
axis health --oauth
```

Production build (static)

```
axis install
bun run build                # → dist/
python3 axis_pwa_server.py   # serves dist/ on :8081
```

Confirm:

- App shell loads; chart requests history
- DevTools → Application → Manifest + Service Worker
- Icons 192/512 from `public/assets/`

Pyodide assets and vendor wheels under `public/pyodide/` and `public/vendor/` must be present in `dist/` for offline engine—`bun run build` copies them via Vite `public/`.

Offline-first lab (no Flask)

1. Source → **Mock Walk**
2. Stream → **Mock Poll** (or **None**)
3. Engine → **Client-Side (Pyodide)**
4. Storage → **Local**

Disable network in DevTools; **Run** still executes. First Pyodide boot downloads/loads self-hosted runtime from the origin—allow that once while online, then go offline.

CORS when using server engine

Browser origin → Pro API must allow your AXIS origin. Flask uses `ALLOWED_ORIGINS`. Localhost regex is typically included; for a VPS demo host, set an explicit origin list. Symptom of failure: `POST /run` blocked in Network tab (no `Access-Control-Allow-Origin`). Worker CORS: CORS and origins.

PWA install

- Manifest: void theme `#0a0b10`, name AXIS
- Service Worker: cache-first shell; network-first `/api/*`; offline API returns structured failure so Pyodide path remains usable
- Chrome/Edge: install icon in the omnibox; AXIS also shows a modest **Install app** chip in the topbar when the browser fires `beforeinstallprompt` (right-click hides it for the session)

Verification checklist

Check	Expect
Load	Bars on chart for default symbol
Topbar	Source / Stream / Engine pickers populated
Run (server)	Flask or Worker responds; plots overlay
Run (pyodide)	Offline OK after warm-up
Manager	Catalog lists built-ins
Workers Manager	Health cards for local/prod backends
Theme	Dark/light toggle persists
`axis doctor`	Required toolchain green. Missing `worker/wrangler.toml` is a warning on a fresh clone until `axis setup` (copies `wrangler.toml.example`).

Failure modes

Symptom	Likely cause	Fix
Empty chart	Source network / CORS / wrong symbol	Load again; try mock-walk
Engine errors immediately	Endpoint down or wrong URL	Settings / Workers Manager → Probe
Pyodide “BadZipFile” / HTML	SPA fallback instead of wheels	Ensure `public/vendor` and pyodide in `dist/`
SW stale UI	Aggressive cache	Unregister SW or hard reload
`API_KEYS_REQUIRED` on scripts	D1 without KV	Bind `API_KEYS` or local `ALLOW_OPEN_KEYS=1` — Auth

Internals (repo paths)

Path	Role
`src/index.tsx`, `src/app.tsx`	Solid entry
`vite.config.ts`	Build
`axis_pwa_server.py`	Static host
`public/manifest.webmanifest`, SW	PWA
`worker/`	Cloudflare® Worker data plane
`packages/cli/`	AXIS CLI (`@hoox-sh/axis-cli`)
`src-tauri/`	Desktop shell

See also

- Quick start
- Compose recipes
- AXIS CLI
- DevOps local dev
- Desktop (Tauri)
- Worker

QUICK START

First historical load, first Pine Run, live stream, and reading the Results drawer in AXIS.

QUICK START

Fifteen-minute path from empty AXIS to plots, strategy stats, and optional live updates.

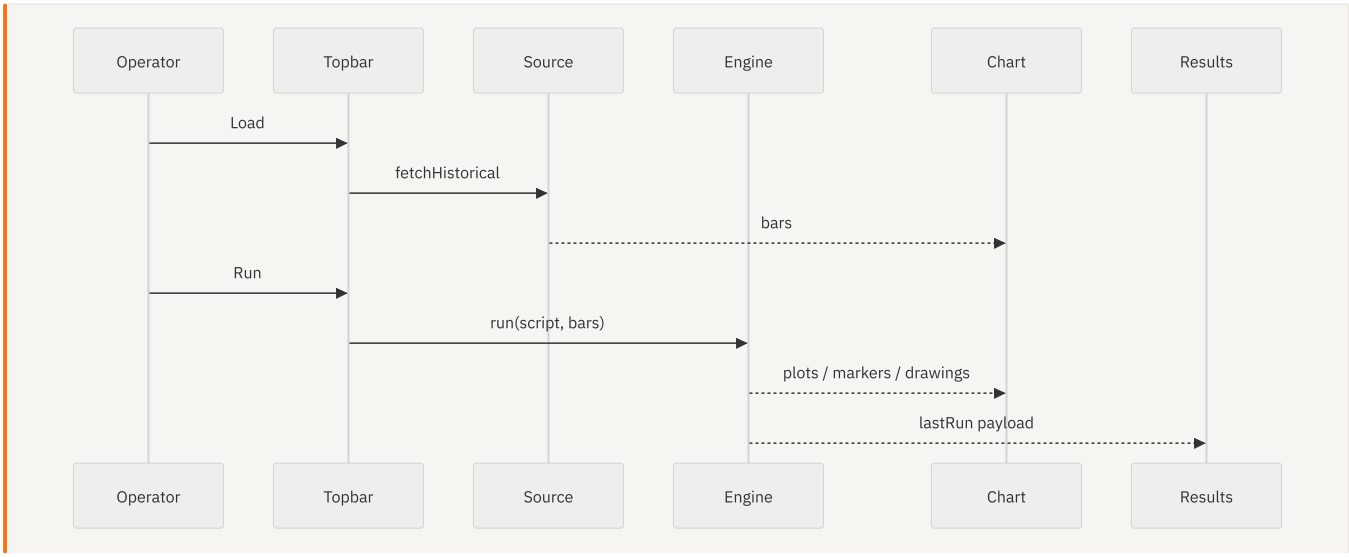
AXIS is a charting app you run in the browser (or install as a PWA / desktop app). You pick **where candles come from**, **whether they keep updating**, **who runs your Pine Script™**, and **where scripts are saved**. The status bar uses short chips for those four planes:

Chip	Means
SRC	**Source** — historical candles (Binance REST, CSV, mock, …)
STR	**Stream** — live ticks while **Live** is on (WebSocket or mock poll)
ENG	**Engine** — who calculates (this tab / a local server / a Worker)
STO	**Storage** — where scripts live (this browser, cloud, or git)

Abstract

1. Ensure AXIS is running (Installation).
2. Load bars for a symbol.
3. Paste or write a minimal indicator / strategy.
4. **Run** → inspect chart overlays + Results.
5. Optionally enable **Live**.

Conceptual model



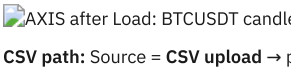
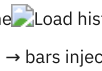
Interface surface

1. Load history

Topbar fields that matter:

Control	Default-ish	Notes
Symbol	`BTCUSDT`	Venue-specific for `binance-rest`
Interval	`1d`	Watchlist + chart share interval vocabulary
Source	`binance-rest`	Or mock / CSV
Stream	`binance-ws`	Paused until Live
Engine	`server` or `pyodide`	Settings for endpoint

Click **Load** (or pick a watchlist row). Status bar should move through loading → ready with bar count context.

 AXIS after Load: BTCUSDT candles, volume, and RSI pane  Load history then Run an RSI script

CSV path: Source = **CSV upload** → pick file when prompted → bars inject via upload store.

Deep history: for multi-page backfill to a past date (background, with gap validation), open the **Data** panel — see Data Source Manager.

2. Minimal script

Open the editor (docked right by default). Example indicator:

```
//@version=6
indicator("AXIS smoke", overlay=true)
plot(close, "close")
plot(ta.sma(close, 20), "sma20")
```

Strategy smoke (events for Results → Strategy):

```
//@version=6
strategy("AXIS strat smoke", overlay=true)
longCond = ta.crossover(ta.sma(close, 10), ta.sma(close, 30))
if longCond
    strategy.entry("L", strategy.long)
if ta.crossunder(ta.sma(close, 10), ta.sma(close, 30))
    strategy.close("L")
```

Language semantics: PYNE runtime.

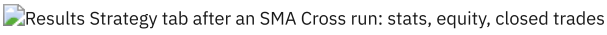
3. Run

Run (topbar or editor). Pipeline:

1. Active engine `run({ script, bars, config })`
2. `lastRun` stored in memory (not fully persisted)
3. Results drawer opens
4. Chart applies overlay lines, trade markers, Pine drawings

Success: status “Completed in Nms”; plots on price pane (or dedicated indicator pane if non-overlay).

4. Read Results

 Results Strategy tab after an SMA Cross run: stats, equity, closed trades

Tabs:

Tab	Content
Events	Normalized entry/exit/order stream
Strategy	Closed trades + win rate, PF, max DD
Plots	Series names, point counts, last value
Metrics	Runtime, engine, bar count, errors
Raw	Full JSON payload

Export: JSON run dump; trades CSV. Click a trade to scroll the chart to entry/exit time.

5. Live (optional)

Toggle **Live**. Stream plugin pushes bars; AXIS may re-run silently when `needsRerun` is set. Stream = **None** freezes time. Mock poll works offline after engine warm-up.

6. Save a script (optional)

Manager → **Script Library** → name → Save. Backend is active storage (`local` default = IndexedDB). See Script library. Editor **Pull / Push** uses the same storage when configured for git/cloud.

7. Power tools (optional)

Tool	Where	Purpose
⌘/Ctrl+K	Global	Command palette — panels, Run, layout, editor toggles
Workers Manager	Topbar activity · ⌘K	Backend health, presets, PYNE Agent install
Layouts	Topbar	Multi-chart 1 / 2H / 2V / 4 + recipes
Replay	Topbar	Scrub history (full bars, cursor on last)
Compare	Topbar	Second symbol overlay
Price decimals	Price-pane scale	Auto or 0–8 — Charting
Debug / Pins	Editor header	Line chips vs chart markers — Debugging
Layers / Alerts	Panel toggles	Drawings list; local price alerts
v6 starter pack	Script Library → Import	`examples/script-library-starter.json`

Worked example: offline lab

Setting	Value
Source	mock-walk
Stream	mock-poll
Engine	pyodide
Storage	local

Load → Run smoke indicator → toggle Live. No Flask required.

Worked example: desk + Flask

Setting	Value
Source	binance-rest
Stream	binance-ws
Engine	server
Endpoint	`http://localhost:5002`
Storage	local

`make run` in repo root; Vite or static AXIS as installed.

Invariants

- Empty editor **Run** is a no-op.
- Errors set status `error` and still populate Results Metrics/Raw when payload exists.
- Switching source auto-aligns default stream (e.g. mock-walk → mock-poll).

Failure modes

Symptom	Action
CORS on `/run`	Fix `ALLOWED_ORIGINS` / use Pyodide
No trades in Strategy	Need entry+exit pair events; open strategy script
Live no updates	Stream not `none`; network; check logs drawer
Pyodide first run slow	Expected warm-up; assets self-hosted under origin

See also

- Compose recipes
- Strategy and results
- Troubleshooting

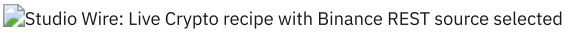
COMPOSE RECIPES

Lawful source × stream × engine × storage combinations: offline, desk, multi-venue, CSV, edge, and git library.

COMPOSE RECIPES

AXIS treats composition as the product. This page lists **recipes**—complete four-tuples with intent, constraints, and failure modes—not marketing tiers.

In the app: topbar **Studio → Wire** (or ⌘K → **Open Architecture**) opens the full-page compose studio. Start from a recipe, then swap or switch off any slot. Apply writes `source` / `live.streamId` / `engine` / `activePlugins` (and opens the on-chain panel when a dataset is selected). Empty slots link to **Plugins**; the active engine is configured on **Runtime**.


Studio Wire: Live Crypto recipe with Binance REST source selected

Abstract

Formal active set:

```
Active set = { source, stream, engine, storage }
```

Built-in ids (catalog may grow via URL plugins):

Kind	Built-ins
Source	`binance-rest` , `mock-walk` , `csv-upload` , (+ examples e.g. CoinGecko)
Stream	`binance-ws` , `mock-poll` , `none` , (+ CF DO example)
Engine	`server` , `pyodide`
Storage	`local` , `cloud` , `git`

Recipe matrix

Recipe	Source	Stream	Engine	Storage	Needs net	Needs backend
Offline lab	mock-walk	mock-poll	pyodide	local	first boot only	no
Desk research	binance-rest	binance-ws	server → Flask	local	yes	Flask
Multi-venue desk	binance-rest + URL sources	binance-ws / none	server	local	yes	Flask
CSV desk	csv-upload	none / mock-poll	server or pyodide	local	no*	optional
AXIS at the edge	venue REST	venue WS / DO	server → Worker	cloud	yes	Worker (+ optional Flask proxy)
Repo library	any	any	any	git	yes (save/list)	git host API

*CSV needs local file; engine may still call network if `server` .

Offline lab

Intent: airplane mode, demos, CI-less teaching, pure AXIS UX.

Axis	Id	Why
Source	`mock-walk`	Synthetic OHLCV in-process
Stream	`mock-poll`	Synthetic bar advance
Engine	`pyodide`	In-browser `pynescript` via Pyodide
Storage	`local`	IndexedDB script library

Setup

1. Install static or dev AXIS with `public/pyodide` + `public/vendor` present.
2. Select the four-tuple in topbar / settings.
3. Load → Run while online once (warm Pyodide).
4. DevTools → Offline; re-run.

Invariants: no `POST /run` ; no exchange traffic; drawings + library remain local.

Failure modes: missing wheels → HTML SPA fallback errors; cold start timeout on huge bar sets.

Desk research

Intent: real venue history + live klines + full PYNE server fidelity.

Axis	Id	Why
Source	`binance-rest`	Public REST klines
Stream	`binance-ws`	Public WS kline stream
Engine	`server`	Flask Pro API `POST {endpoint}/run`
Storage	`local`	Fast offline library

Setup

```
make run                # :5002
cd frontend && bun run dev
```

Settings → Backend URL `http://localhost:5002` → Test. Symbol like `BTCUSDT`, interval `1h` / `1d`.

CORS: AXIS origin must be allowed by Flask `ALLOWED_ORIGINS`.

Failure modes: rate limits; symbol not on venue; CORS; engine timeout on long history (AXIS scales timeout with bar count).

Multi-venue / multi-source

Intent: compare feeds without forking the AXIS.

Axis	Pattern
Source	Built-in `binance-rest` **or** Manager → Install URL plugin (e.g. CoinGecko example) → **Use**
Stream	Match venue when available; else `none` for historical-only
Engine	Usually `server` for parity
Storage	`local` or `git` for shared research

Workflow

1. Manager → Catalog → capability badges (offline / auth / network / proxy).
2. **Use** sets active source.
3. Load and Run as usual.

Dynamic plugins rehydrate from localStorage install list on boot (`restoreInstalledPlugins`).

Failure modes: plugin URL not CORS-readable; invalid export shape; built-ins cannot be unregistered without force flags.

CSV desk

Intent: proprietary or offline CSV/OHLCV files.

Axis	Id
Source	`csv-upload`
Stream	`none` (typical) or `mock-poll` for artificial advance
Engine	`pyodide` (airgap) or `server`
Storage	`local`

Workflow

1. Source → CSV upload (file picker opens if no file yet).
2. AXIS parses via `parseOhlcvFile` into upload store.
3. Load injects bars; symbol field less critical for synthetic series.
4. Run strategies against that tape.

Invariants: bars stay in memory/session upload store—not the long-lived app state blob (size).

Failure modes: bad headers/time units; empty parse; re-select same file requires clearing input (AXIS resets file input after pick).

AXIS at the edge

Intent: browser AXIS + Cloudflare Worker data plane (keys, usage, optional D1 scripts, DO fan-out).

Axis	Id	Why
Source	venue REST (built-in or proxy-aware)	History
Stream	venue WS **or** DO-backed example stream	Single upstream → N clients
Engine	`server` endpoint = Worker	Worker proxies `/api/run` → Flask or future in-worker runtime
Storage	`cloud`	`/api/scripts` + Bearer Pro key

Setup

1. `wrangler dev` / deploy Worker script **`worker-axis`** (`https://worker.axis.hoox.sh`).
2. Endpoint → Worker origin.
3. Storage cloud → API key from admin `/api/keys`.
4. Script Library uses cloud list/read/write; If-Match on concurrent writes.

Invariants: Pages project is `axis` (`https://axis.hoox.sh`); Worker script is `worker-axis` (`https://worker.axis.hoox.sh`); health JSON `service` is `worker-axis`.

Failure modes: missing Bearer key; D1 unbound → memory store (dev only); DO session query params wrong.

Git library

Intent: Pine sources as first-class repo files; Save = commit.

Axis	Id
Source / Stream / Engine	any research tuple
Storage	`git`

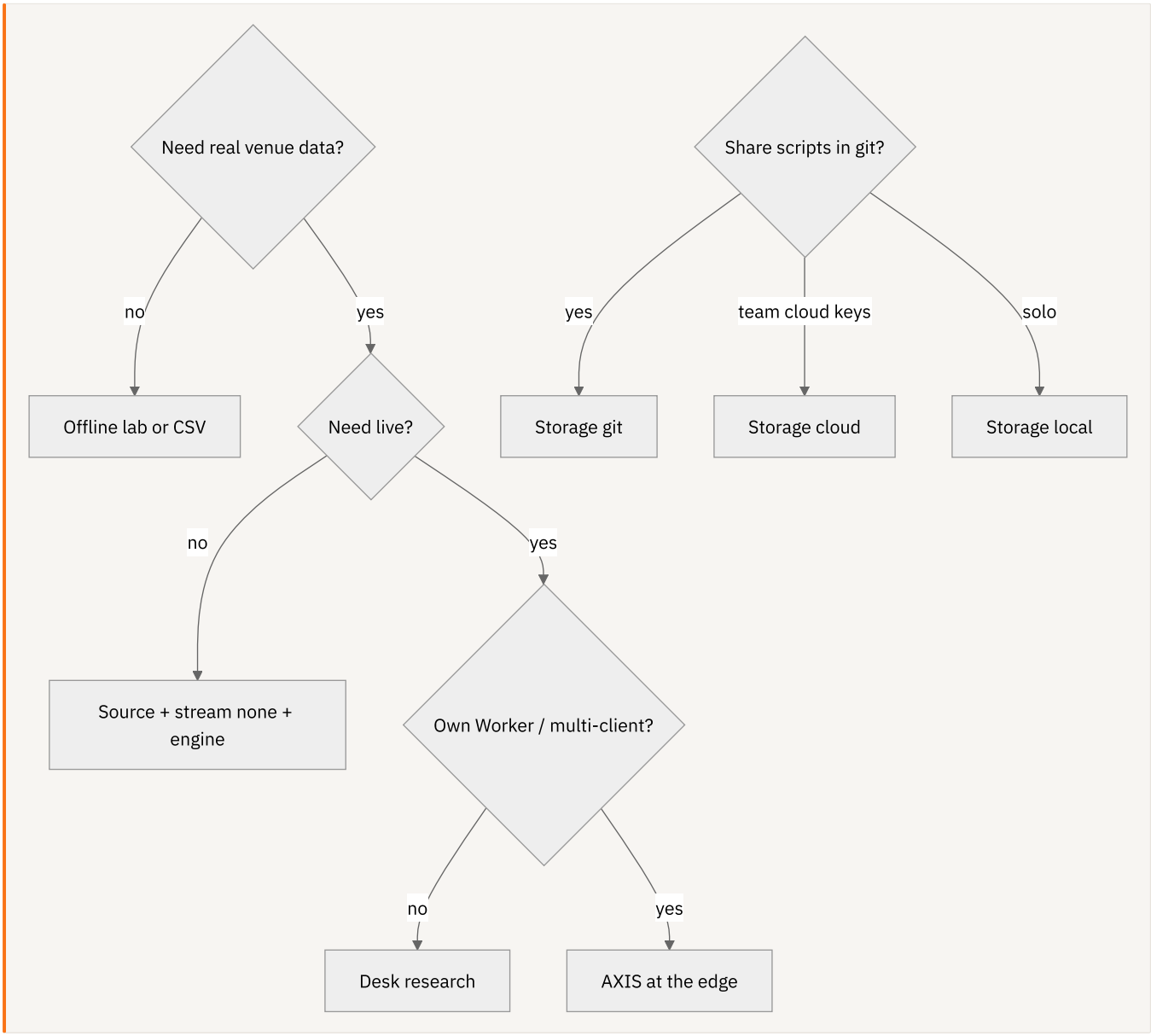
Config (Manager → Script Library or pluginsConfig)

Field	Notes
provider	`github` | `gitlab`
token	contents:write / api
owner, repo	GitHub path; GitLab projectId optional
branch	default `main`
basePath	default `pine-library`
apiBaseUrl	self-hosted forges

Invariants: drafts stay local; only explicit Save/remove hit the remote. Commit message template supports `{{name}}` / `{{iso}}`.

Failure modes: 401 token scope; wrong base path; GitLab path vs numeric id confusion.

Choosing a recipe



Internals

Concern	Path
Active selection	`frontend/src/store` `activePlugins`
Built-in catalogs	`sources/catalog.ts` , `streams/*` , `engines/catalog.ts` , `storage/catalog.ts`
URL plugins	`plugins/loader.ts`
Compose UI	`ui/Topbar.tsx` , `ui/PluginManager.tsx` , `ui/SettingsDialog.tsx`

See also

- Plugin contracts
- Manager and settings
- Topologies
- ADR-001 pluggable registry

MANAGER AND SETTINGS

Runtime, Workers, Plugins, Wire, and Settings — how AXIS chrome is split.

MANAGER AND SETTINGS

Five studio pages share one full-viewport overlay. Runtime is the active calculation context; Workers and Plugins are catalogs, not tabs inside it.

Studio Runtime: server engine, health, and backend endpoint

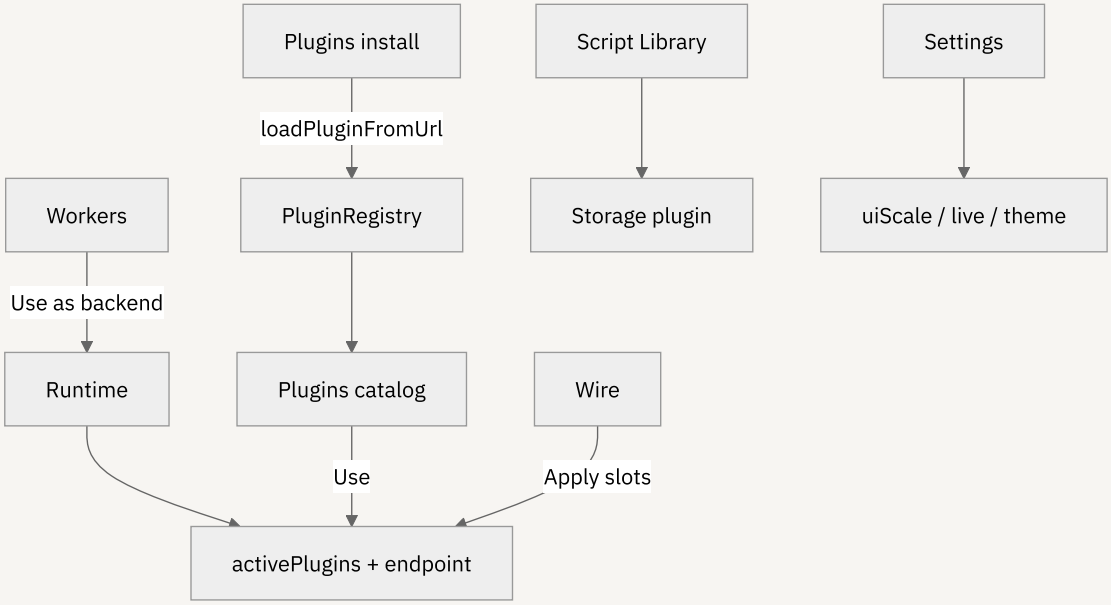
Studio rail switching Runtime, Wire, Settings, Workers, Plugins

Abstract

Surface	Opens from	Mutates
Runtime	Topbar **Studio** · studio rail · ⌘K “Open Runtime”	Active engine, endpoint, exec mode, probe of *that* backend
Workers	Studio rail · ⌘K “Open Workers” · Runtime card	Probe catalog; activate a backend (`endpoint` / engine)
Plugins	Studio rail · ⌘K “Open Plugins” · hidden topbar testid	Registry membership, Use, URL install, script library
Wire	Topbar **Studio** → Wire rail · ⌘K “Open Architecture”	Compose slots; Apply writes `activePlugins`
Settings	Topbar **Studio** → Settings rail · ⌘K “Open Settings”	Appearance, chart, live, session keys, editor intel, theme

Active plugins live in `store.activePlugins`; per-plugin fields in `store.pluginsConfig` keyed by `` `${'kind'}:${'id'}` ``.

Conceptual model



Plugin Manager

Catalog

Studio Plugins catalog: sources, streams, engines, storage with Use

Lists sources, streams, engines, storages from the unified registry.

Action	Effect
Use	`setActivePlugin(kind, id)` for that row’s kind
Capability badges	`offline` , `needsAuth` , `needsNetwork` , `needsProxy` from `plugin.capabilities`
Built-in flag	Built-ins resist casual unregister

Status bar shows active engine id and storage backend after changes.

Install (URL)

Studio Plugins Install: load an ES module URL or a same-origin example

Paste a module URL exporting a default plugin object (`kind` , `id` , `name` , contract methods). AXIS:

1. Fetches and validates export

2. Registers into the TypeScript registry
3. Persists install record so `restoreInstalledPlugins()` reloads next visit

Security notes (operator-level): only https/http schemes suitable for module load; treat third-party URLs as code execution in your browser origin. See security tests under `tests/security/`.

Supported URL kinds: **source**, **stream**, **engine**, **dataset**, **component**. Example plugins ship in `public/plugins/`:

- `example-coingecko-source.js`
- `example-tiny-pyne-engine.js`
- `example-cf-do-stream.js`

PYNE Agent (sister project — natural language → PYNE scripts) installs as a **component** from your deployed worker URL (or via Workers Manager → Install):

```
https://<pyne-agent-worker>/plugin/axis-pine-agent.js
```

See PYNE Agent and plugin reference. Works without the HOOX mesh.

Script Library tab

See Script library. The Plugins page hosts the panel so library and plugins share one studio surface.

Runtime vs Workers vs Plugins

One overlay (`AppPage` in `src/ui/studio/`) with a left rail. Switching rail pages does not close the overlay.

- **Runtime** — `src/ui/runtime/RuntimePage.tsx` — active engine, endpoint, exec mode.
- **Workers** — `src/ui/workers/WorkersPage.tsx` — backend inventory and probes.
- **Plugins** — `src/ui/plugins/PluginsPage.tsx` — catalog / install / library.

Workers

Studio Workers: backend inventory with pyne Pro API selected

Opens from the studio rail, ⌘K **Open Workers**, or a Runtime related card.

Sub-tab	Role
Overview	Health cards (distinct icon per worker) for pyne Pro API, AXIS Worker (prod + local wrangler), Pyodide, PWA service worker, optional PYNE Agent / pyne-worker
Detail	**Usage** (when to pick this worker), probe latency, feature flags, Use-as-backend / preload / install actions
Install	**When to use** + step-by-step setup with copyable commands (`make run` , `wrangler` , plugin URL, ...)
Configure	Paste Backend URL, presets (`:5002` , `:8787` , workers.dev, axis.hoox.sh), activate Pyodide, install agent plugin

Catalog items (usage)

Worker	Icon intent	When to use
pyne Pro API	server	Primary **Server** calculation backend (Flask `:5002`). Best for compile/Numba and long history.
AXIS Worker	zap	Production edge data plane (on-chain proxy, scripts D1, optional `/api/run`). Default for On-Chain.
AXIS Worker (local)	activity	Local wrangler `:8787` while developing the Worker.
Pyodide	cpu	In-browser offline calc — no Backend URL; first load ~20–30s.
Service Worker	wifi	PWA shell cache (auto in production; skipped in Vite dev). Not a calc engine.
PYNE Agent	download	Optional NL → Pine plugin; scripts still run via your engine.
pyne-worker	settings	Optional HOOX mesh edge evaluator; paste origin as Backend URL.

Probes

- Run once when the modal opens; **Refresh** re-runs.
- Each worker has a **hard wall-clock timeout** (~5s) so a hung host cannot leave “Probing...” forever.
- Pyodide probe uses **HEAD** (not a full `pyodide.js` download).
- HTTP probes hit `GET /health` (JSON markers). Implementation: `src/workers/*` , `src/ui/workers/WorkersPage.tsx` . Operator CLI twin: AXIS CLI.

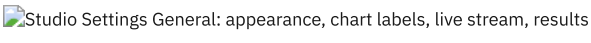
Hardened VPS note: production UFW typically allows only **22 + 443**. Pro API binds `127.0.0.1:5002` — public `:5002` probes correctly **time out**. On `https://axis.hoox.sh` , Workers Manager probes pyne Pro via **same-origin** `https://axis.hoox.sh/health` (nginx reverse-proxy), not loopback. See VPS demo.

Runtime (engine)

Engine, execution mode, Backend URL, Test, and Prefer WebSocket live on **Runtime**, not Settings. Storage is a **Wire** slot.

Field	Behavior
Engine	Registry engine list; labels via `engineOptionLabel`
Execution mode	When the active engine exposes `configSchema.mode` (server): `interpret` | `compile` | `auto`. Stored in `pluginsConfig.engine:<id>.mode` and sent on each run (WS and REST).
Prefer WebSocket run	Server engine: prefer `/ws/run` over REST when available
Backend URL	Shown when engine is `server` or has `configSchema.endpoint`
Test / Probe	`GET` health against endpoint; status message

Settings


Studio Settings General: appearance, chart labels, live stream, results

Product chrome only. Engine/endpoint/mode are on Runtime.

Field	Behavior
Default interval	Updates store; may reload bars for current symbol
Watchlist refresh	Clamp 5–120 seconds
Chart theme	Preset picker (void / classic / mono / boutique dark & light) — see `src/theme/presets.ts`
History bars	One-shot Load depth (venue may clamp further)

Deep multi-page history and gap repair live in the Data Source Manager, not Settings.

Execution mode maps to PYNE `Runtime.run(..., mode=…)`:

Mode	Behavior
`interpret`	Full AST interpreter
`compile`	Numba/numpy compiled path (first run JITs; Flask is not prewarmed)
`auto`	Default on Server / Pyodide: try compile, fall back to interpret on failure

The Connection HUD engine chip shows the selected mode. Save persists via Solid store `persist()` into `pynescript.axis.v1`. Escape closes; Ctrl/Cmd+Enter saves.

Pyodide engines hide endpoint—calculation is same-origin assets, not Flask.

Interface surface (UI map)

Control	Store field
Engine select	`activePlugins.engine`, flat `engine`
Storage select	`activePlugins.storage`
Endpoint	`endpoint`
Source/stream topbar	`activePlugins.*`, `source`, `live.streamId`
Theme toggle	`theme` + `data-theme` on `<html>`

Internals

Path	Role
`src/ui/studio/`	Full-page overlay kit (`ax-*`)
`src/ui/runtime/RuntimePage.tsx`	Active engine / endpoint / mode
`src/ui/plugins/PluginsPage.tsx`	Catalog / install / library
`src/ui/workers/WorkersPage.tsx`	Backend inventory / probes
`src/ui/settings/SettingsPage.tsx`	Product chrome
`src/ui/wire/WirePage.tsx`	Compose recipes
`src/ui/plugin-badges.tsx`	Capability badges
`src/plugins/registry.ts`	Unified registry
`src/plugins/loader.ts`	URL load + restore (incl. component)
`src/plugins/bootstrap.ts`	Built-ins

Invariants

1. Settings never write script bodies—only layout/config.
2. Catalog **Use** does not auto-Run; Load/Run remain explicit.

3. Source change may re-default stream (`defaultStreamForSource`).
4. API keys for cloud/git belong in `pluginsConfig` , not in git-committed defaults.

Worked examples

Point AXIS at local Worker

1. Settings → Engine Server-Side → Endpoint `http://127.0.0.1:8787`
2. Probe → OK
3. Save → Run

Install example source

1. Manager → Install → URL to `.../plugins/example-coingecko-source.js` (served origin)
2. Catalog → Source appears → Use → Load

Failure modes

Symptom	Fix
Empty catalog	Built-ins not registered—hard reload; check console
URL install fails	CORS on plugin host; bad export; mixed content
Probe fails	Backend down; wrong path; HTTPS mixed content
Settings not sticky	localStorage blocked / quota

See also

- Script library
- Compose recipes
- Plugins

PYNE AGENT

Write scripts from natural language (PYNE Agent) in AXIS using the optional pyne-agent-worker Cloudflare® plugin.

PYNE AGENT

What it does

Describe an indicator or strategy in plain language. The **PYNE Agent** (Cloudflare® Workers AI™) returns a full script you can insert into the AXIS editor, then **run** with your normal engine (local PYNE Pro API, edge worker, or Pyodide).

You do **not** need the full HOOX mesh. The agent worker is usable **standalone**.

Pine Script™ and TradingView® are trademarks of TradingView, Inc. Cloudflare® is a registered trademark of Cloudflare, Inc.

Prerequisites

1. A deployed pyne-agent-worker origin (or local `wrangler dev`).
2. AXIS open in the browser (dev or production PWA).

Optional: API key if the worker has `API_KEY` set.

Install the plugin

1. Open **Manager → Plugins → Install**.
2. Paste the **same-origin** module (avoids cross-origin module CORS):

```
/plugins/axis-pine-agent.js
```

Or the remote copy (needs CORS on the agent Worker):

```
https://pyne-agent-worker.cryptolinx.workers.dev/plugin/axis-pine-agent.js
```

3. Set **endpoint** to the agent Worker origin (no trailing slash), e.g. `https://pyne-agent-worker.cryptolinx.workers.dev`. Same-origin install still talks to this API host.
4. Set **API key** if required.
5. Prefer **v6** / **strategy** or **indicator** when you know the target style.

AXIS migrates a previously saved remote plugin URL to `/plugins/axis-pine-agent.js` on restore.

AXIS **loads component plugins from URL** (Manager → Install, or Workers Manager → install agent). You can also open the worker's built-in chat UI:

```
https://<your-agent-worker>/
```

or use the floating **PYNE Agent** button the plugin may inject after install.

Typical workflow

1. Ask for a script (example: "v6 RSI strategy with ATR trailing stop and clear plots").
2. Review the fenced Pine block.
3. **Insert into editor** (plugin action) or copy/paste.
4. Select your **engine** and **Run** as usual.
5. Iterate with follow-up messages ("use length input", "overlay false", …).

Tips

- Prefer specific version (`v5` / `v6`) and script kind in the request.
- The agent may use a private knowledge base (docs + open examples). Quality improves after the operator ingests docs — not required to start chatting.
- Validation against **pyne-worker** is optional on the agent side; AXIS evaluation is still your engine plugin.

Troubleshooting

Issue	Fix
Load error	Check URL HTTPS, CORS, and that the module is reachable
Unauthorized	Match plugin `apiKey` to worker secret
Weak answers	Ingest KB on the worker; still usable without it
Cannot insert	Use copy; ensure plugin `api.insertScript` is provided by host

See also

- Plugin deep dive
- Manager and settings
- Plugin examples

DATA SOURCE MANAGER

Background multi-page OHLCV backfill to a past date, dataset validation, and gap fills — without blocking the chart.

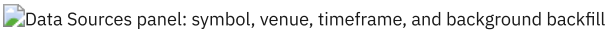
DATA SOURCE MANAGER

Deep history accumulation for a **symbol + source + timeframe**, down to a **past date**. Work runs fully in the **background**: chart, live streams, and the editor stay free.

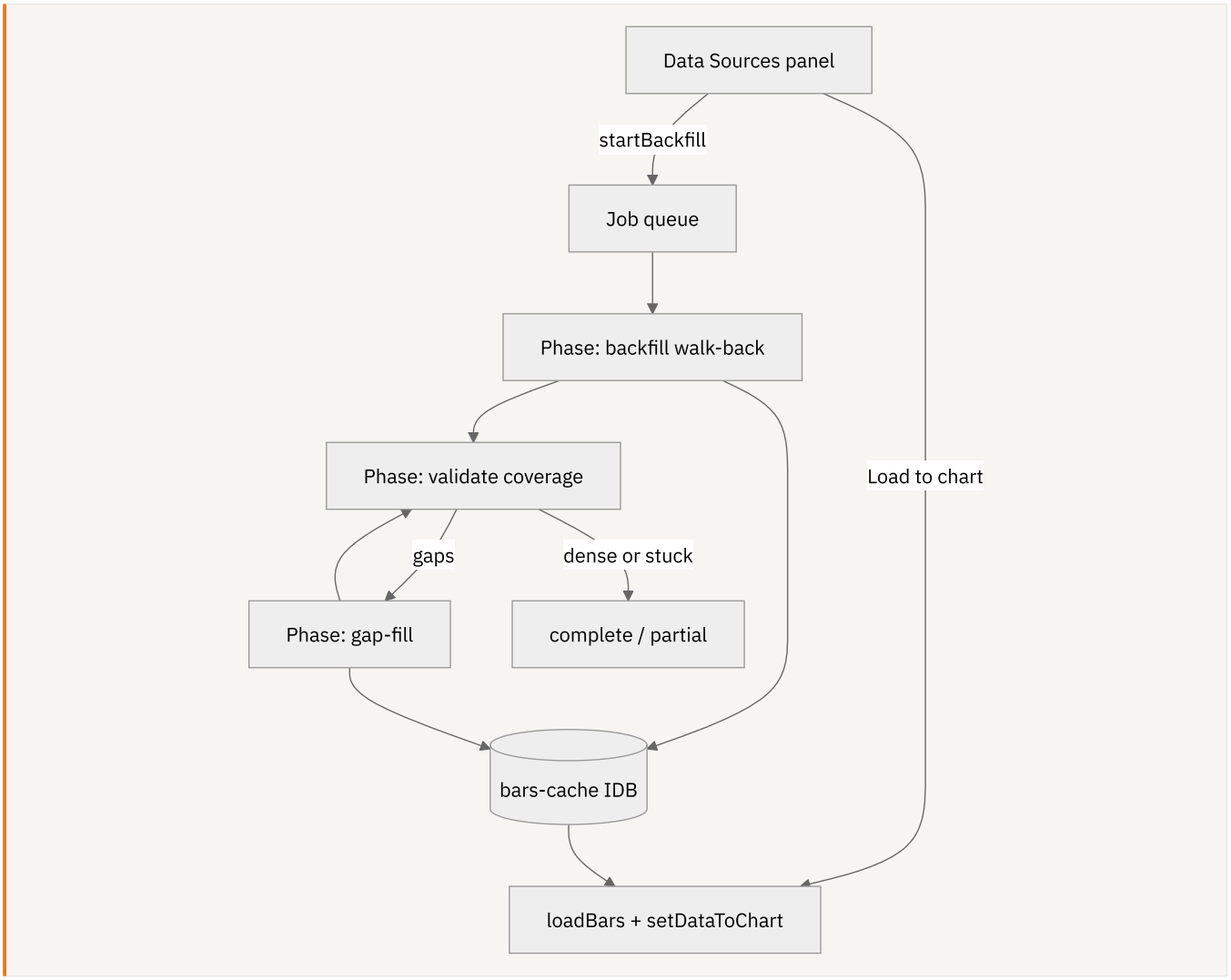
Abstract

Piece	Role
Topbar **Load**	One venue page (or `historyBars`) straight onto the chart
Data Sources panel	Queue backfill jobs, watch progress, load cache onto chart
Walk-back pages	Multi-page REST with `endTime` only (avoids start+end one-page traps)
Validate + gap-fill	After backfill, check density and re-download holes
IDB bars cache	Durable OHLCV per `source|symbol|interval`

Open from the topbar **Data** button, or command palette → **Toggle Data Source Manager**.


Data Sources panel: symbol, venue, timeframe, and background backfill

Conceptual model



How to use

1. Open **Data** (Data Sources panel).
2. Set **symbol**, **source / exchange**, **timeframe**, and **accumulate to** (UTC date).
3. Optionally check **Apply to chart when complete**.

4. Click **Start background backfill** — the job id appears immediately; work continues detached.
5. Watch phases: **Backfilling → Validating → Filling gaps → Complete** (or **Partial**).
6. Click **Load to chart** when ready (or rely on apply-when-complete).

Pause / resume / cancel apply between pages. Multiple jobs queue with low concurrency (rate-limit friendly).

Phases

Phase	Behavior
Backfill	Walk newest → older using `endTime` + page limit only. Never send `startTime` + `endTime` together to venue APIs that return the *first* N bars from start (would falsely complete in one page).
Validate	Measure coverage from target past date → job end time: leading, internal, and trailing holes via interval step + tolerance.
Gap-fill	For each gap window, walk-back download again; re-validate (several rounds).
Done	`datasetComplete` when dense; otherwise **Partial** with remaining gap count (venue may lack older history).

Job fields (UI)

Field	Meaning
Bars / Pages	Cached bar count and network pages this job
Oldest / Target	Current oldest open time vs past-date target
Gaps	Detected holes · how many fill attempts landed bars
Coverage	full / partial / in progress

Relation to Topbar Load

	Topbar Load	Data Source Manager
Blocks UI	Yes (status loading)	No (background jobs)
Depth	Single page / `historyBars`	Multi-page to past date
Chart	Always paints	Only on **Load to chart** or apply-when-complete
Cache	Ephemeral `store.bars`	Durable IDB `bars-cache`

Use Load for desk-speed symbol switches; use the manager when you need years of 1m/1h/1d history for research.

Internals

Path	Role
`src/data/data-source-manager.ts`	Job queue, walk-back, validate, gap-fill
`src/data/bars-cache.ts`	Memory-first + IDB durable OHLCV cache
`src/data/bars-gaps.ts`	Gap detection / coverage report
`src/ui/DataSourceManagerPanel.tsx`	Panel UI
`src/plugins/types.ts`	`SourceOpts` : `startTime?` , `endTime?` , `signal?`
`src/sources/catalog.ts`	Venue date params + `sourcePageLimit`

bars-cache performance (2.0.1+)

`bars-cache.ts` is **memory-first**:

API	Behavior
`getCachedBars`	Serve warm in-memory series immediately; cold miss hydrates from IndexedDB then fills memory
`getCachedBarCount`	Count without cloning the full series when warm
Range / window	`sliceBarsForLoad` + count helpers avoid full clones for progress UI
Soft caps	`BARS_CACHE_MAX` (100k bars/series), `BARS_CACHE_MAX_SERIES` (48 keys)

DSM gap-fill progress uses `getCachedBarCount` instead of loading full series each tick.

Limits & safety

- Max concurrent network jobs: **1** (queue the rest).
- Caps: pages per walk, bars per series (soft IDB cap), gap-fill rounds.
- Backfill forces `fallback: false` so synthetic walks do not fake venue history.
- Cancel uses `AbortController` on in-flight fetches (symbol / job switches abort mid-page).

See also

- Sources — built-in historical plugins
- Quick start — one-shot Load
- UI shell — topbar panels

ON-CHAIN DATA

DefiLlama protocol TVL, GeckoTerminal DEX pool OHLCV, and derived TVL spike events — not a wallet, not CEX-only price.

ON-CHAIN DATA

AXIS **on-chain data plane**: search public protocol and DEX metrics and plot them next to your chart. Built-in providers include **DefiLlama protocol TVL** (total value locked, USD), **GeckoTerminal DEX pool OHLCV**, and **derived TVL spike/drop events**.

On-Chain panel with Aave TVL attached beside the chart

Abstract

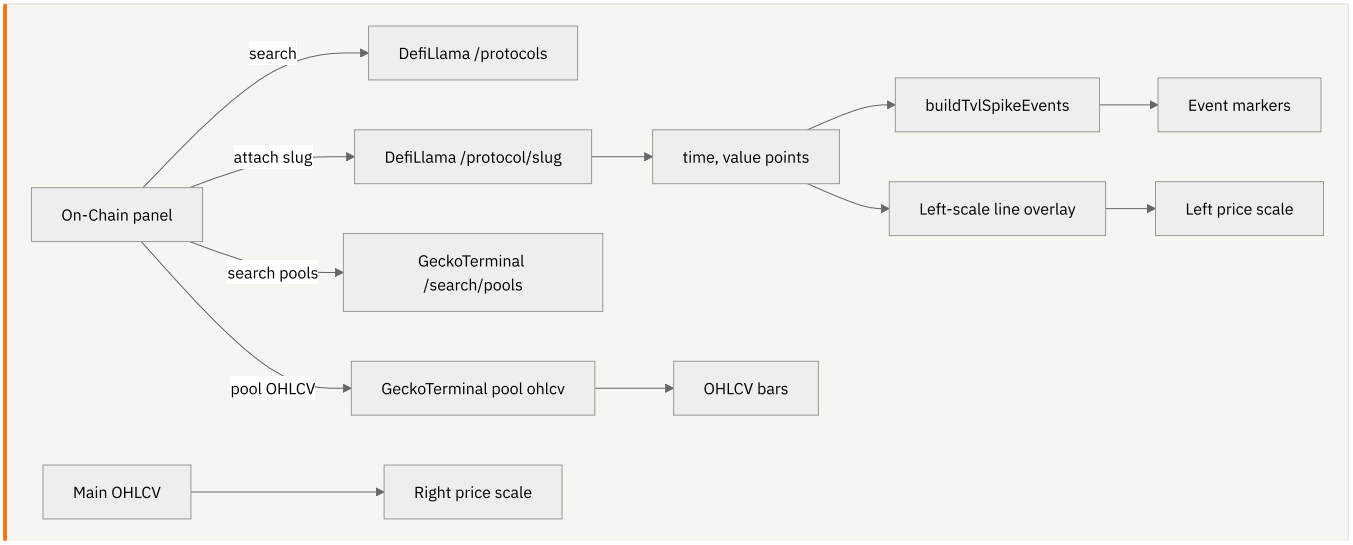
Piece	Role
On-Chain panel	Search protocols / pools, attach / hide / detach series
DefiLlama TVL	Daily USD liquidity history for a protocol slug
GeckoTerminal OHLCV	DEX pool candlesticks (minute / hour / day aggregates)
TVL spike events	Day-over-day |%| threshold events from attached TVL (`tvl_spike` / `tvl_drop`)
Left price scale	Independent metric axis so main OHLCV keeps the right scale
Provenance	Each series records provider + finality / snapshot notes

Open from the topbar **Panels** group (**On-Chain**), or command palette → toggle the On-Chain panel (panel id `onchain`).

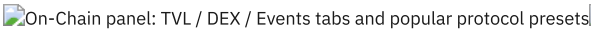
What this is not

- **Not a wallet.** AXIS does not connect to MetaMask, Ledger, or any signing wallet for this plane.
- **Not CEX price by default.** Protocol TVL is liquidity in USD, not the exchange candle series for the chart symbol. DEX pool OHLCV is **on-chain pool** price, which can diverge from centralized venues.
- **Not Pine.** These overlays are host-side dataset series; they are not produced by `indicator()` / `strategy()` plots.

Conceptual model



How to use (DefiLlama TVL)


On-Chain panel: TVL / DEX / Events tabs and popular protocol presets

Attaching Aave TVL from the On-Chain panel onto the chart

1. Load a market as usual (symbol + source + timeframe).
2. Open **On-Chain**.
3. Search a protocol (e.g. `aave` , `uniswap` , `lido`).
4. Click a result to **attach** its TVL history.
5. The curve appears on the **left scale** of the price pane (when overlay wiring is active).
6. Hide or remove series from the **Attached series** list; **Clear** removes all.

Multiple protocols can be attached (chart safety caps the number of on-chain lines).

Refresh, export, popular presets

Command palette (**⌘K** / **Ctrl+K**) exposes host jobs for attached TVL series — no Pine required:

Command id	Title	What it does
`onchain.refresh`	On-Chain: Refresh Attached TVL	Re-fetch every attached DefiLlama TVL series (`refreshAllAttachedTvl`)
`onchain.export.series`	On-Chain: Export Series CSV	Download **all** attached series as long CSV (`series,time,value`)
`onchain.attach.popular`	On-Chain: Attach Popular TVL	Attach the top **5** protocols by current TVL (`attachPopularTvl(5)`)

Related palette actions (panel / hygiene):

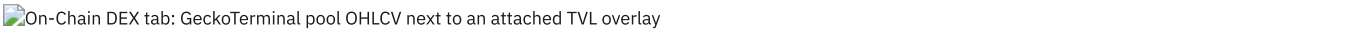
Command id	Role
`panel.onchain`	Toggle On-Chain panel
`onchain.mode.tvl`	Open panel (TVL mode entry)
`onchain.clear.series`	Detach all series
`onchain.clear.events`	Clear event markers plane
`onchain.health`	Probe Worker `/api/onchain/health` (Connection HUD)

Export format: one row per point — `series,time,value` — with `time` as unix seconds and `value` as USD TVL (or other scalar). Empty attachments still download a header-only file.

Popular preset: empty DefiLlama protocol browse (highest TVL first), capped by the chart series limit (`MAX_ONCHAIN_SERIES` , currently 8). Already-attached slugs refresh in place rather than duplicating.

Refresh: re-runs the DefiLlama protocol TVL fetch for each attachment so left-scale lines stay current after long sessions.

DEX pools (GeckoTerminal)


On-Chain DEX tab: GeckoTerminal pool OHLCV next to an attached TVL overlay

Phase 2 adds **DEX pool OHLCV** via GeckoTerminal public API (no API key). Use this when you want on-chain pool candles rather than protocol TVL:

Concept	Detail
Network	Gecko network id (`eth` , `solana` , `base` , `arbitrum` , `polygon_pos` , …)
Pool address	EVM `0x` + 40 hex, or Solana-style base58
Timeframe	`minute` / `hour` / `day` with aggregate (e.g. 1h → hour + aggregate 1)
Currency	Default USD when proxied with query pass-through
Page size	**Max 1000 candles per request** (`limit`); Gecko hard cap
Walk-back	`endTime` → Gecko `before_timestamp` (bars **strictly before** that unix second)

Client helpers live in `src/onchain/geckoterminal.ts` (`fetchGeckoPoolOhlcv` , `searchGeckoPools` , interval/network maps). Browsers should use the **Worker gecko proxy** (below) because public GeckoTerminal hosts often lack CORS for arbitrary origins.

Deep history (Data Sources + GeckoTerminal DEX)

Topbar **Load** fetches a **single page** (up to 1000 bars). For multi-page history down to a past date:

1. Set the chart source to **GeckoTerminal DEX** (`geckoterminal-ohlcv`).
2. Use a pool symbol such as `eth:0x…` (network + pool address).
3. Open **Data** (Data Source Manager) → set symbol / source / timeframe / **accumulate to**.
4. Start background backfill — AXIS walks older pages with `endTime` only (each page maps to `before_timestamp`). The source does **not** multi-page inside one `fetchHistorical` call; DSM does it page-by-page.

See Data Source Manager for job phases and cache behavior.

Events (TVL spikes)

From an attached TVL scalar series, AXIS can derive **day-over-day** percent-change events:

Field	Behavior
Type	`tvl_spike` (gain) or `tvl_drop` (loss)
Default threshold	10% absolute day-over-day change
Severity	`warn` below critical band; `critical` for large moves (default ~25% or 2.5× threshold)
Payload	`pctChange` , `prevValue` , `value` , thresholds

Pure helper: `buildTvlSpikeEvents` in `src/onchain/events.ts` . These are **derived** from daily snapshots — not mempool or block-final chain events.

DefiLlama **raises** and **token unlocks** require Pro API (`pro-api.llama.fi`) and are **not** fetched by the free Worker proxy; inject external events only if you already have that data.

Scale and chart behavior

Behavior	Detail
Scale	On-chain TVL lines use the built-in **left** price scale (`left`)
Main market	Primary OHLCV stays on the **right** scale
Compare	Compare overlays may share the left scale; on-chain keys are `onchain_*` , not `overlay_*`
Units	TVL values are **USD** liquidity totals from DefiLlama; DEX OHLCV is pool quote (often USD)

Settings

Settings → General → On-Chain is a short note only (no extra bindings):

- By default the PWA calls **DefiLlama / GeckoTerminal directly** (public CORS).
- Worker proxy (`{endpoint}/api/onchain/llama` / `…/gecko`) is used **only** when Backend URL is a real AXIS Worker (`worker.axis.hoox.sh` , `worker-axis` / `*.workers.dev` , or wrangler `:8787`).
- **Not a wallet** — public metrics only; no signing

Keep **Backend URL** as the Pro API for Pine (`https://axis.hoox.sh` or local `:5002`). That host is **not** an on-chain proxy.

Network path (CORS / Worker proxy)

Public llama.fi / geckoterminal APIs currently allow browser origins (`Access-Control-Allow-Origin: *`), so AXIS **defaults to direct fetch**.

Using `https://axis.hoox.sh/api/onchain/…` is wrong: nginx serves the **SPA HTML**, which produces *invalid JSON / got HTML*. The client now avoids that host for on-chain bases.

When Backend URL **is** an AXIS Worker, paths map as:

Client request	Worker route	Upstream
`{endpoint}/api/onchain/llama/protocols`	allowlisted GET	`https://api.llama.fi/protocols`
`{endpoint}/api/onchain/llama/protocol/{slug}`	allowlisted GET	`https://api.llama.fi/protocol/{slug}`
`{endpoint}/api/onchain/gecko/networks/{net}/pools/{addr}/ohlcv/{tf}`	allowlisted GET (+ query)	GeckoTerminal OHLCV
`{endpoint}/api/onchain/gecko/search/pools`	allowlisted GET (+ query)	GeckoTerminal search
`{endpoint}/api/onchain/health`	local	feature flags

- Local Worker: `http://127.0.0.1:8787` + `cd worker && bun run dev` if you want the allowlisted proxy.
- Override with plugin `config.baseUrl` when needed.

Isolate memory caches short-TTL responses (`X-Axis-Onchain-Cache: HIT|MISS`):

Route class	TTL (approx.)
Llama protocols list	10 min
Llama protocol detail	2 min
Gecko OHLCV	60 s
Gecko search	120 s

Provenance and finality

Each attached series carries a short provenance line, for example:

- Provider: **DefiLlama · protocol TVL (USD)**
- Provider: **GeckoTerminal · pool OHLCV**
- Finality: **Daily snapshot · not CEX price** (TVL) or pool candle finality notes (DEX)

Treat DefiLlama points as **daily** aggregates from the public API — not block-final on-chain events and not exchange marks. DEX candles reflect the **pool** at GeckoTerminal's sampling, not a CEX order book.

Network, CORS, and proxies

Situation	What to do
invalid JSON / got HTML for aave	Backend URL is a SPA host (e.g. `axis.hoox.sh`) — leave it for Pine; on-chain should hit llama.fi directly (reload after upgrade)
Local dev blocked by CORS	Point Backend URL at wrangler `:8787` Worker, or set plugin `baseUrl`
Self-hosted demo	Optional: reverse-proxy allowlisted `/api/onchain/*` to the Worker
Offline	Needs network; there is no offline synthetic TVL/DEX walk by default

See also CORS and origins for Worker origin allowlisting.

Dataset plugins (advanced)

On-chain providers implement the structural **dataset** plugin contract (`fetchDataset`). Built-ins register via the on-chain catalog; dynamic URL plugins may install additional datasets. End users normally only use the On-Chain panel — see Datasets for the contract sketch.

Related docs

- Data Source Manager — OHLCV backfill (separate from on-chain scalars)
- Sources — historical bar sources
- Worker overview — `/api/onchain/…` route table
- CORS and origins — browser cross-origin constraints

SCRIPT LIBRARY

Save, load, import/export Pine scripts via local IndexedDB, cloud Worker, or git (GitHub/GitLab).

SCRIPT LIBRARY

The script library is the operator-facing surface of **storage plugins**. The editor holds the working document; the library is durable, named documents with metadata.

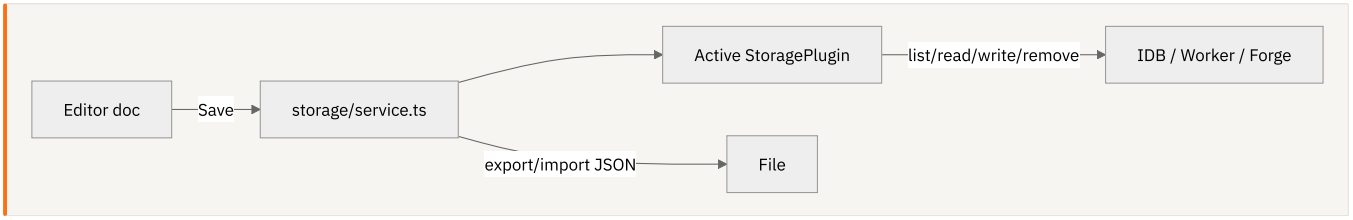
Script Library panel with local IndexedDB storage

Abstract

Backend	Id	Medium	Auth
Local	`local`	IndexedDB (`pynescript.axis.storage`), LS fallback	none
Cloud	`cloud`	Worker `/api/scripts` (D1 or memory)	Bearer Pro API key
Git	`git`	GitHub/GitLab Contents API	PAT

Active backend: `store.activePlugins.storage` (Settings or library picker).

Conceptual model



Service layer (`listScripts` , `readScript` , `writeScript` , ...) never talks to engines.

Interface surface

Manager → **Script Library**:

Control	Behavior
Backend select	Prompts to **copy** scripts onto the new engine (source is never deleted)
Refresh	`list()`
Name / description	Metadata for next write
Save	Write current editor doc
Load	`read` → inject editor (`loadLibraryDoc` / setDoc)
Delete	`remove`
Export JSON	Full library dump
Import JSON	Bulk write (`importLibraryJson`)
Cloud endpoint + key	Settings → Script storage **and** this panel. Saved under `pluginsConfig['storage:cloud']`
Version history	Every engine (local snapshots, Worker revisions, git commits). Open / Restore on a card
Git fields	provider, token, owner, repo, branch, basePath, ...

Status line: backend · remote · branch · connected · count.

Pine Script™ v6 starter pack

Repo ships a portable library dump of **v6** indicators, strategies, and helpers:

Path	Role
`examples/script-library-starter.json`	Importable library JSON (`//@version=6` scripts: RSI, MACD, ATR, MTF, sessions, math lib, ...)

Import: Manager → Script Library → **Import JSON** → pick the file (or any prior export). Uses `importLibraryJson` with new ids so it merges into the active backend (local / cloud / git).

Prefer `//@version=6` for new scripts; engines accept v5/v6 when the runtime supports them.

Local backend

- DB name `pynescript.axis.storage` v1 stores `scripts` + `kv`
- Older library keys migrate once

- Drafts: optional `saveDraft` / `loadDraft` without polluting remote backends
- Works fully offline

Invariant: browser profile wipe deletes local library—export JSON for backup.

Cloud backend

- `Authorization: Bearer <api_key>`
- Scripts partitioned by key hash on Worker
- Concurrent updates may use `If-Match` / revision → HTTP 409 on conflict
- Default endpoint falls back to `store.endpoint` or `http://127.0.0.1:8787`

Configure key in library panel or Settings-related config; keys stay in this browser's localStorage state blob.

Git backend

Invariant	Detail
Save commits	Each write/remove is a commit (+ push via API)
Drafts local	No remote commit for draft buffer
basePath	Default `pine-library/`
Providers	`github`, `gitlab` (+ self-hosted `apiBaseUrl`)

Token scopes: GitHub `contents:write`; GitLab `api` or `write_repository`.

Published libraries (import emulator)

`library("Name")` scripts can be **published** as versioned folders so other scripts can `import owner/Name/1 as alias`:

```
{basePath}/published/index.json
{basePath}/published/{namespace}/{Name}/{version}/lib.pyne
{basePath}/published/{namespace}/{Name}/{version}/manifest.json
```

- Library panel → **Publish library** (or a successful **Run** of a `library()` — auto, skipped if unchanged).
- Namespace defaults to the git owner, else `user`.
- Local cache always updates so Pyodide / offline runs resolve imports.
- The engine receives `{namespace, name, version, source}` and pyne registers them before `import` (interpret path).

Worked examples

Solo offline library

1. Storage = local
2. Write indicator in editor
3. Name `sma-ribbon` → Save
4. Reload page → Library → Load

Team cloud

1. Admin creates key on Worker
2. Storage = cloud; paste endpoint + key
3. Save shared strategies
4. Second browser: same key → list

Research monorepo

1. Storage = git; GitHub owner/repo; basePath `research/pine`
2. Save → commit message from template
3. PR review on GitHub UI

Internals

Path	Role
`src/ui/ScriptLibraryPanel.tsx`	UI
`src/storage/service.ts`	Facade
`src/storage/local.ts`	IDB
`src/storage/cloud.ts`	Worker API
`src/storage/git.ts`	Git plugin
`src/storage/git-github.ts`, `git-gitlab.ts`	Forges
`examples/script-library-starter.json`	Pine v6 starter pack

Contract: Plugin types `StoragePlugin`.

Failure modes

Symptom	Cause	Mitigation
Empty list after switch	Wrong backend / auth	Check status line + key
401 cloud	Bad/missing key	Regenerate; re-save config
409 cloud	Revision conflict	Re-read then write
Git 404	Wrong owner/repo/basePath	Fix config
Quota local	Huge scripts in LS fallback	Prefer IDB; export prune

See also

- Manager and settings
- Compose recipes — git / edge
- State namespaces

STRATEGY AND RESULTS

How AXIS turns engine events into trades, stats, chart markers, equity, and exportable Results tabs.

STRATEGY AND RESULTS

After **Run**, the engine returns a structured payload. AXIS **AXIS** normalizes events, pairs trades, paints markers, and exposes a Results drawer. Strategy math here is a **viewer**—not a brokerage.

Results overlay showing strategy stats, equity curve, and closed trades

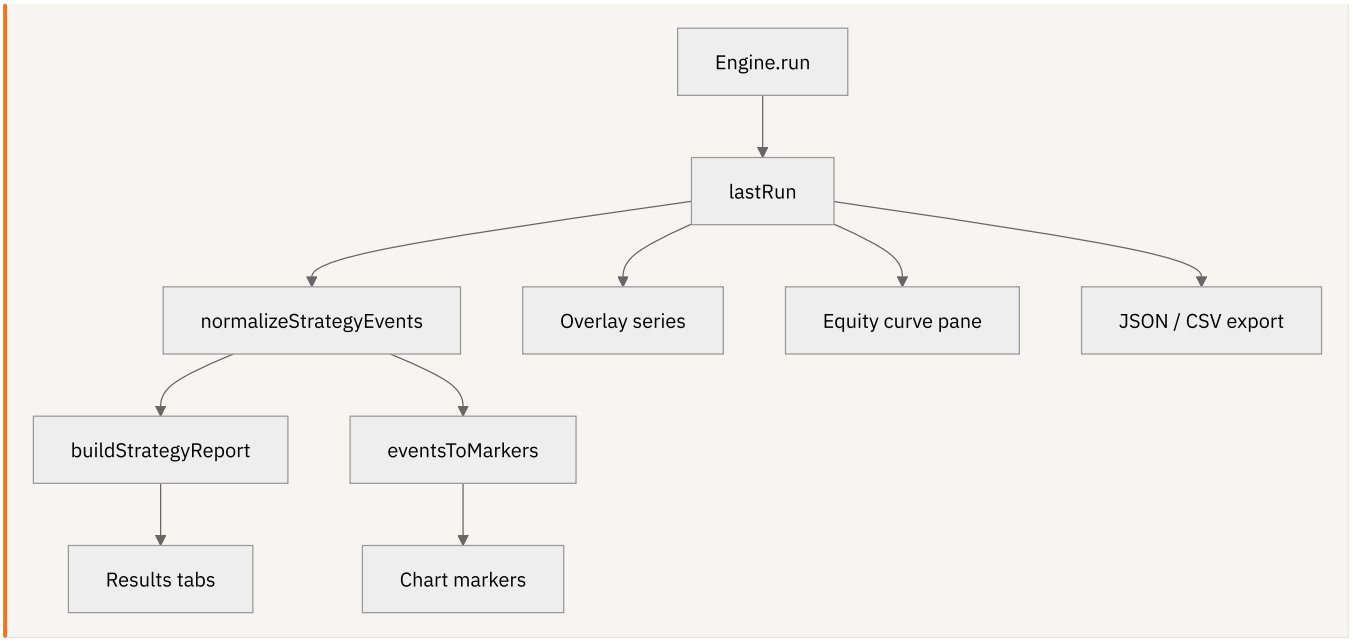
Abstract

Engine `RunResult` (simplified):

```
status: success | error
plots[] | series{}
events[]     — entries, exits, orders, custom
drawings[]   — Pine line/label/box objects
meta{}       — ms, overlay, script_name, plot_meta, …
error?
```

AXIS stores this as `store.lastRun` (in-memory; not fully rehydrated from localStorage).

Conceptual model



Results drawer tabs

Strategy tab: stats cards, equity curve, and closed-trades table

Tab	Operator value
Events	Chronological normalized stream (kind, time, price, id, dir) with a **Stream / Open⇄Close view switcher** — the Open⇄Close view groups events into pyramiding-aware position cycles (avg-entry parity with the pyne broker, per-fill P&L, jump-to-bar)
Strategy	Single column: stats → equity → closed trades below, with return %, bars-held, and fill-count detail
Optimise	Strategy-only `input.*` search — Hyperparameter Optimisation
Plots	Series inventory (points, last value)
Metrics	Runtime, status, engine, source, bars, high-level stats
Raw	Pretty-printed JSON for support / diffs

Saved runs persist a JSON-safe strategy stats snapshot, so script-library rows show win %, profit factor, return %, max drawdown, and average trade at a glance. A shared walker (`walkStrategyEvents` in `src/results/strategy.ts` , positions in `src/results/positions.ts`) feeds trades, stats, positions, and the events stream.

Strategy metrics

Computed by `buildStrategyReport` :

Stat	Definition (viewer)
totalPnl	Sum of closed trade PnL (price units)
winRate	% wins among closed trades
profitFactor	Gross profit / gross loss
avgTrade / avgWin / avgLoss	Means over closed set
maxDD	Peak-to-trough on cumulative trade PnL path
wins / losses / trades	Counts

Pairing rules: entry-like kinds open by `id`; exit/close kinds match id or sole open trade. Direction flips short PnL. Missing prices drop the event from pairing (with normalization attempting OHLC fill from bars when available).

Chart coupling

- Trade markers on the price pane (`eventsToMarkers`) show filled size as `Long N` / `Short N` (exits: order id + qty); missing engine qty keeps the id / `L` / `S` / `X` fallback
- Scroll-to-time on trade row click
- Equity curve series when strategy events warrant (`buildEquityCurve`)
- Status bar snippet: closed trade count · net PnL when available

Export

Export	Content
JSON	Full `lastRun`
CSV	Closed trades (`tradesToCsv`)
Clipboard	CSV of trades when supported

Filenames: `axis-run-<ts>.json`, `axis-trades-<ts>.csv`.

Interface surface

- Drawer height persisted (`resultsPanel.height`); open state toggled after runs (`openResults` default true for interactive runs)
- Silent live re-runs can skip auto-open
- Equity / indicator panes created on demand when non-overlay or strategy paths need them

Worked example

1. Load `BTCUSDT` daily history.
2. Run a crossover strategy (see Quick start).
3. Results → Strategy: inspect win rate.
4. Click a green trade → chart jumps to entry.
5. Export CSV for a notebook.

Internals

Path	Role
`frontend/src/ui/ResultsPanel.tsx`	Drawer UI
`frontend/src/results/strategy.ts`	Trade pairing + stats
`frontend/src/results/events.ts`	Normalize events, markers, equity
`frontend/src/indicators/runner.ts`	`runAndApply` orchestration
`frontend/src/chart/series-factory.ts`	Plot colors / series

Language-level strategy semantics: PYNE runtime.

Invariants

1. Results are a function of **last successful/error payload + current bars**—changing bars without re-run can desync prices used for fill.
2. Viewer PnL is not currency-normalized; treat as relative tape units.
3. `meta.overlay === false` routes plots to an indicator pane, not price.
4. Pine drawings (script) are distinct from user drawing tools (Drawings).

Failure modes

Symptom	Explanation
Events but 0 trades	Only entries, or unpaired ids
Empty plots	Series all null / name filtered (`__` prefix hidden)
Markers missing	Manager not mounted or events lack times
Huge Raw JSON	Many events—export file rather than copy

See also

- Quick start
- AXIS — results and strategy
- AXIS — indicators

HYPERPARAMETER OPTIMISATION

Search strategy input.* values over N trials (TPE / random / grid) with holdout or walk-forward. Strategies only.

HYPERPARAMETER OPTIMISATION

AXIS can search Pine Script™ `input.*` values on a **`strategy()`** script and apply the best assignment for this symbol and bar window.

The search engine is **pyne** (`POST /optimize` / `pynescript.optimize`). AXIS is the plugin UI. Pyodide and engines without `/optimize` fall back to an isolated local loop (random only).

This is not a live trading optimiser and not a Pine builtin. Best-on-sample is not a guarantee out of sample.

Optimise tab: search configuration and Fast/Slow input space

Holdout is the default for a reason. In-sample-only search overfits. Winners on one symbol or timeframe often fail on another.

Open it

```
Open a script that declares `strategy()`. Load history. A first **Run** is optional but helps engine-exported inputs appear.

Results drawer → **Optimise**, or command palette **Optimise strategy**. Indicators and libraries show a one-line disabled reason.

Tick the `input.int` / `float` / `bool` / `enum` fields to search. Numeric fields need **min** and **max** (`minval`/`maxval` from the script,

Set trials, sampler, objective, and validation. **Start** runs the study. **Cancel** aborts the in-flight engine call. Trials do not paint the

**Apply best** merges the winner into Script Inputs (other inputs stay). **Apply + re-run** also runs the winner so the Strategy tab matches. E
```

Study controls

Control	Default	Notes
Trials	30	Hard cap 200
Sampler	auto	Random if N<20, else TPE (TPE needs pyne `/optimize`)
Objective	composite	Also net PnL, profit factor, Calmar-like
Validation	holdout 30%	Walk-forward or in-sample (warned)
Min trades	5	Fewer closed trades → trial rejected

Grid is offered when the cartesian product fits the trial budget. On Pyodide (no `/optimize`) the UI falls back to random and says so.

Validation

Mode	What happens
Holdout (default)	Sampler sees the first ~70% of bars. Every trial is also scored on the last ~30%. Ranking prefers out-of-sample.
Walk-forward	Rolling train/test windows. Engine runs ≈ `N × folds` (capped at 400).
In-sample	One run on all bars. Warned in the UI.

Train and test are **bar slices**, not trade filters on a full-sample run.

Engines

Engine	Path
`server` (Pro API)	`POST {endpoint}/optimize` — parse-cached interpret loop, TPE / random / grid
Pyodide / missing `/optimize`	Isolated `runScript` — random only

Same-origin VPS / Docker must **proxy** **`/optimize`** to Flask (same as `/run`). Otherwise AXIS falls back to the slow client loop.

Overfitting

- Prefer holdout. Treat in-sample net profit as a search score, not a live forecast.
- Composite + min-trades rejects lottery-ticket parameter sets with 1–2 fills.
- Apply + re-run on a later date range or another symbol before trusting a winner.

See also

- Strategy and results
- pyne POST /optimize
- pyne CLI `optimize`

DRAWINGS

Interactive chart annotations in AXIS—tools, persistence, and how they differ from Pine script drawings.

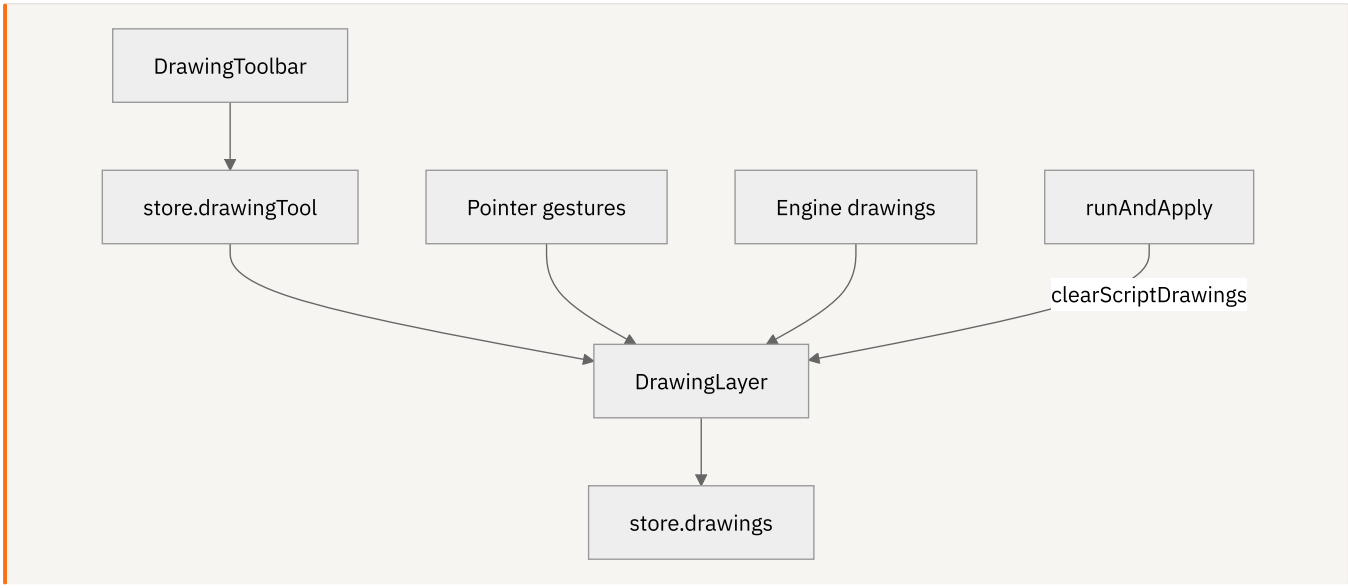
DRAWINGS

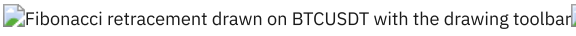
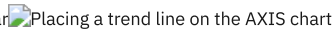
AXIS supports **user drawings** (interactive tools on the chart) and **script drawings** (objects returned by the engine). They share a canvas layer but different lifecycles.

Abstract

Class	Origin	Cleared when	Persisted
User drawings	Drawing toolbar	Explicit delete / clear	Yes (`store.drawings` in `pynescript.axis.v1`)
Script drawings	Engine `drawings[]`	Next Run (re-applied)	No (recomputed)

Conceptual model




Fibonacci retracement drawn on BTCUSDT with the drawing toolbar 
Placing a trend line on the AXIS chart

Tools

Left drawing tool rail: cursor, trend, shapes, fib, measure

Tool id	Kind	Geometry
`cursor`	—	Select / pan mode (not a drawable)
`hline`	hline	Horizontal price
`vline`	vline	Vertical time
`trend`	trend	Two points
`ray`	ray	Two points (extends)
`extend`	extend	Extended line
`rect`	rect	Two corners
`ellipse`	ellipse	Ellipse / oval
`arrow`	arrow	Directional segment
`fib`	fib	Two points; levels 0, 0.236, 0.382, 0.5, 0.618, 0.786, 1
`measure`	measure	Two points (delta readout)
`text`	text	Anchor + label

Colors: default accent `#939fff`, up/down/measure variants in `DRAWING_COLORS`.

Style bar and per-tool settings

The floating style bar (next to the rail) edits **the selected drawing**, or the **defaults for the active tool** when nothing is selected. Last-used extras are stored per tool (`drawingPrefs.byKind`).

Control	Where	Tools
Color / width / dash	Style bar	All place tools
Custom color	Style bar native picker	All place tools
Fill	Style bar slider	Shapes, ranges, fib, highlighter, positions
Extend L / R	Style bar arrows	Trend, ray, extend, info line, channel, fib
Show price	Style bar `$`	Horizontal/vertical lines, fib labels
Lock	Style bar (selection)	Selected drawing
Text / font size	Settings popover (gear); double-click on chart	Text, note, callout, flag, price label
Fib levels / reverse / %	Settings popover	Fibonacci family
Risk/reward	Settings popover	Long / short
Arrow caps	Settings popover	Trend, ray, arrow, forecast

Gear button: `axis-drawing-settings`. Popover: `axis-drawing-settings-popover`.

Layers & templates

- **Layers** panel lists user drawings (select, hide, delete)
- **Templates**: save/load drawing packs (`axis.drawingTemplates.v1`)
- **Duplicate / tag symbol** helpers for multi-chart workflows

Interface surface

- Toolbar on chart host (`DrawingToolbar`)
- Active tool in store; layer sync via `createEffect` on `store.drawingTool`
- Pane manager hosts price chart; drawings bind to time/price coordinates (not pixel-only)

Workflows

Annotate a level

1. Select Horizontal line.
2. Click price.
3. Reload page—line remains (persisted drawings array).

Fibonacci between swing points

1. Select Fib.
2. Click swing low → swing high (or reverse).
3. Levels render from `FIB_LEVELS`.

After Run with Pine lines/labels

1. Run script that emits drawings.
2. Layer clears previous script drawings, then applies new set.
3. User drawings remain.

Script drawings (engine → chart)

Engine `drawings[]` map through `pyne-drawings.ts` before paint:

Engine surface	Chart behavior
`line` / `box` / `label` / `polyline`	SVG geometry on the script's pane scale
`linefill` / `line_fill`	Filled quad between two line endpoints
`force_overlay` (line/box/label)	Paint on the **price** pane even when the script is `overlay=false`
`barcolor` series	Not SVG — per-bar candle body/wick tint via `plot-visuals`

Non-overlay scripts place ordinary geometry on the indicator pane Y-scale; user toolbar drawings always stay on price.

Internals

Path	Role
`frontend/src/chart/drawing-types.ts`	Tool ids, types, fib levels
`frontend/src/chart/drawing-layer.ts`	Interaction + render
`frontend/src/chart/DrawingToolbar.tsx`	UI
`frontend/src/chart/pyne-drawings.ts`	Map engine drawings → layer
`frontend/src/chart/ChartHost.tsx`	Mount / dispose

Invariants

1. User drawings ≠ strategy results; deleting drawings does not alter `lastRun`.
2. Time base is bar time (unix-compatible chart times).
3. `cursor` never creates geometry.
4. Persistence omits bars/logs but **includes** drawings—large freehand-less sets only.

Failure modes

Symptom	Fix
Clicks do nothing	Tool is cursor; chart empty (no bars/time scale)
Drawings vanish on Run	Those were script drawings—re-run or convert notes to user tools
Lost after wipe	Cleared site data / different browser profile

See also

- AXIS — charting
- Strategy and results

ALERTS

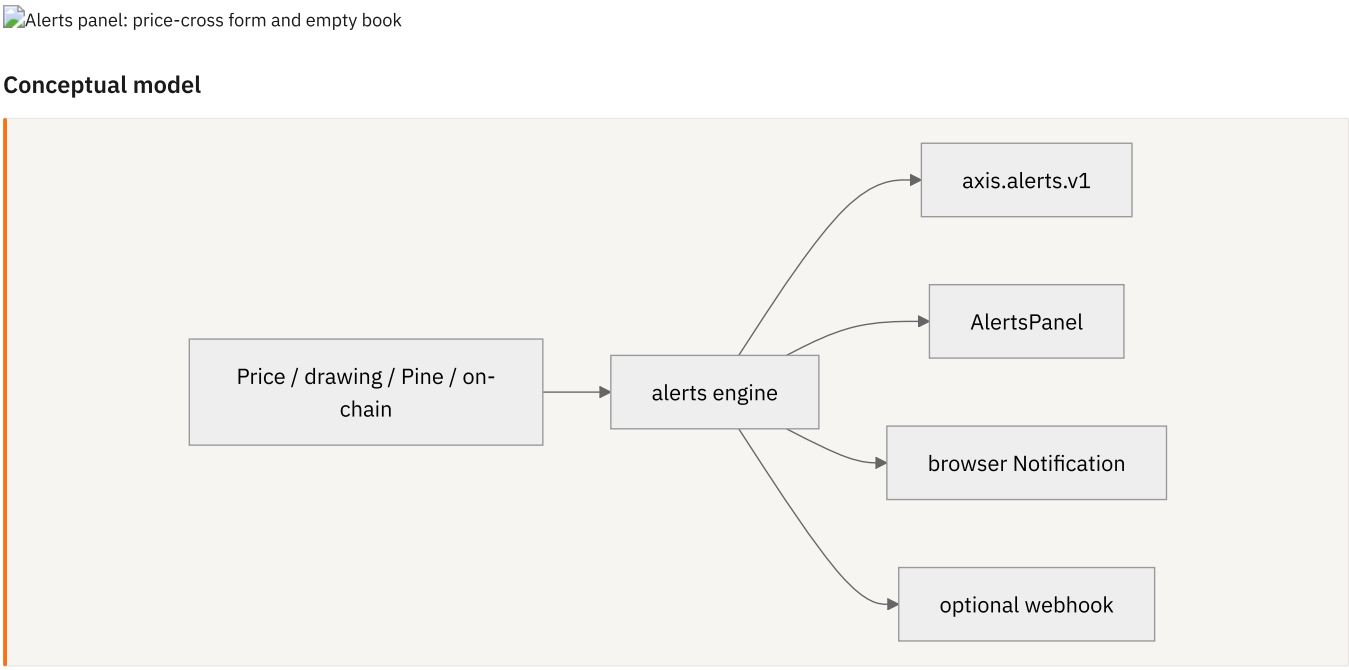
AXIS local alerts: price crosses, drawings, Pine conditions, on-chain spikes. Persist axis.alerts.v1. Not HOOX orders.

ALERTS

Abstract

AXIS ships a **client-side alert book** (`src/alerts/`) with an Alerts panel. Kinds cover price, percent change, drawing touch, indicator plot conditions, and on-chain TVL spikes. State persists in `axis.alerts.v1` . Delivery is local to the PWA: browser notifications, an optional HTTPS webhook, and an optional **L2 webhook** (richer JSON: `channel: "l2"` , message, params snapshot).

These alerts are **not** HOOX `trade-worker` orders. Pointing the L2 field at a PYNE/HOOX gateway is possible and **dangerous** — you are then inventing an execution path the PWA does not certify. Pine **`alert()`** and **`alertcondition()`** (and strategy `alert_message` on fills) fire as `pine_alert` rows when you arm them in the panel. Interpret runs export `alerts` / `alert_conditions` ; AXIS delivers only the **last bar** so a full backtest does not dump history. Plot-vs-threshold remains `pine_condition` .



Conceptual model

Interface surface

Kinds implemented in `src/alerts/` :

Kind	Fires when
`price_cross`	Price crosses a level
`price_above` / `price_below`	Price vs level
`pct_change`	Bar-to-bar (or window) percent move
`drawing_touch`	Price touches a user drawing's levels (hline, fib, trend anchors)
`pine_condition`	Applied indicator plot vs threshold (`>` , `<` , `cross` , …)
`pine_alert`	Pine `alert()` / `alertcondition()` (and strategy `alert_message`) on the last bar
`onchain_tvl_spike`	DefiLlama TVL jump (see On-Chain)
`onchain_event`	Catalogued on-chain event

UI: `src/ui/AlertsPanel.tsx` . Persistence key: **`axis.alerts.v1`** .

Delivery: enable **Notifications** in the panel (browser permission), optionally set a compact webhook URL, and/or an L2 webhook. Both URLs must be `https` (no loopback / private IPs). Live ticks from the chart stream evaluate price, drawing, and indicator kinds; on-chain kinds still fire from the on-chain events plane.

PYNE language `alert()` / `alertcondition()` on Flask / pyne-worker is documented at PYNE alerts.

Internals

Path	Role
`src/alerts/`	Engine + format + storage
`src/ui/AlertsPanel.tsx`	Panel
`src/onchain/`	TVL spike helpers (`buildTvlSpikeEvents`)

Invariants & edge cases

1. Alerts survive reload via local storage, not the Worker D1 script registry.
2. Closing the tab stops evaluation unless a stream + engine is still running.
3. On-chain kinds require the on-chain proxy (`/api/onchain`) to be reachable.
4. `lastRun` strategy results are **not** persisted; do not assume alert history equals a backtest.

Worked examples

1. Open Alerts panel.
2. Add `price_cross` on the active symbol / TF.
3. Enable browser notifications in the browser, then tick **Browser notification** on the form.
4. For an indicator: apply a script, pick its plot, set `>` / `<` / `cross` vs a threshold.
5. For a drawing: place a horizontal line (or fib), pick it in the form, optionally set a tolerance.
6. For Pine `alert()` / `alertcondition()` : run a strategy, open Alerts, kind **pine alert**, pick the script and `alert()` / `alertcondition()` / both. Optional title filter from the last run. In-browser compile records these in the compiled loop; the Flask/pyne-worker engine uses interpret for scripts that call them.

On-chain: attach a TVL series, then add `onchain_tvl_spike` (default ~10% day-over-day in `buildTvlSpikeEvents`).

Failure modes

Symptom	Cause	Fix
No rows	Engine not running / no stream	Start live or replay bars
Pine `alert()` never fires	No `pine_alert` row armed; or remote compile without AXIS fallback	Arm **pine alert**; AXIS uses interpret on Flask/pyne-worker when the script calls `alert()` / `alertcondition()` . In-browser compile records them in the compiled loop.
Pine kind never fires	Engine dropped alerts	Use Flask interpret; see PYNE alerts
On-chain silent	Proxy HTML / CORS	Use Worker `/api/onchain` , not the SPA origin
Webhook posted a trade	You pointed it at HOOX	Use pyne-worker instead

See also

- On-Chain
- Drawings
- Evaluation map
- PYNE alerts
- AXIS and HOOX

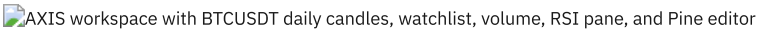
INTERFACE GALLERY

Stills and looping GIFs of the AXIS PWA, Studio, and CLI — captured from the public demo.

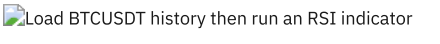
INTERFACE GALLERY

Product chrome as shipped. Stills live under `docs/images/`. GIFs are short looping tours, not live-exchange recordings.

Capture: `bun run capture:1080p` (Playwright, **1920×1080 @2x** → 3840×2160). A second series at **2560×1440 @2x** (5120×2880) lives under `docs/images/2560x1440/`.


AXIS workspace with BTCUSDT daily candles, watchlist, volume, RSI pane, and Pine editor

Workspace


Load BTCUSDT history then run an RSI indicator

Still	What
workspace.png	Default layout: watchlist, chart, editor
chart-panes.png	Price + volume + RSI panes
topbar.png	Market / data / compute / panel groups
watchlist.png	Symbol rail
statusbar.png	Connection HUD
command-palette.png	⌘K / Ctrl+K
layers.png	Panes, overlays, scripts, drawings
theme-panel.png	Settings → Theme presets
workspace-mobile.png	390×844 PWA

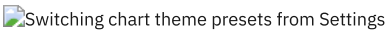
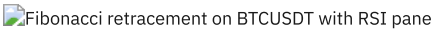
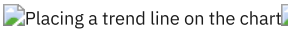
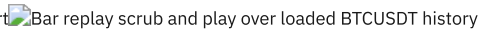

Switching chart theme presets from Settings

Chart & drawings


Fibonacci retracement on BTCUSDT with RSI pane

Still	What
drawings-toolbar.png	Left tool rail
drawings-fib.png	Fibonacci retracement
bar-replay.png	Replay strip on the chart
compare.png	Second-symbol compare
volume-profile.png	Volume profile overlay
chart-layout-2x2.png	2×2 multi-chart grid


Placing a trend line on the chart


Bar replay scrub and play over loaded BTCUSDT history

Editor & scripts

CodeMirror 6 Pine editor with a v6 RSI script

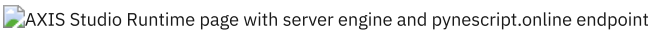
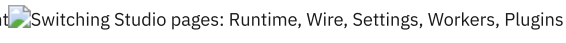
Still	What
editor.png	Docked Pine editor
scripts-panel.png	Applied scripts
script-library.png	Local library
scriptlogs.png	Script logs
run-rsi.png	Workspace after an RSI run

Research panels

Still	What
results-drawer.png	Strategy results overlay
results-tabs.png	Events / Strategy / Optimise / …
results-strategy.png	Closed trades + equity
hpo.png	Hyperparameter search
data-source-manager.png	Background backfill
onchain-tvl.png	DefiLlama TVL overlay
onchain-panel.png	On-Chain panel (TVL)
onchain-dex.png	DEX pool OHLCV tab
alerts.png	Local alerts

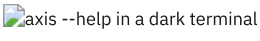
Attaching Aave TVL from the On-Chain panel

Studio


AXIS Studio Runtime page with server engine and pynescript.online endpoint

Switching Studio pages: Runtime, Wire, Settings, Workers, Plugins

Still	What
runtime.png	Active calculation
wire.png	Compose recipes
settings.png	Appearance / live / keys
workers.png	Backend inventory
plugins.png	Contract catalog
plugins-install.png	Install from URL

CLI


axis --help in a dark terminal

Still	What
help.png	`axis --help`
doctor.png	`axis doctor`

Landing crops

Finish-cut tiles used on hoox.sh/axis: `docs/images/landing/`.

Landing hero: AXIS workspace with BTCUSDT and RSI

Tile	File
Chart panes	axis-chart.png
Drawings	axis-drawings.png
Editor	axis-editor.png
On-chain	axis-onchain.png
Results	axis-results.png
Studio Wire	axis-studio.png
Mobile	axis-mobile.png
Open Graph	axis-og.png

TROUBLESHOOTING

Diagnose empty charts, CORS, engine failures, Pyodide assets, streams, storage, and PWA cache issues in AXIS.

TROUBLESHOOTING

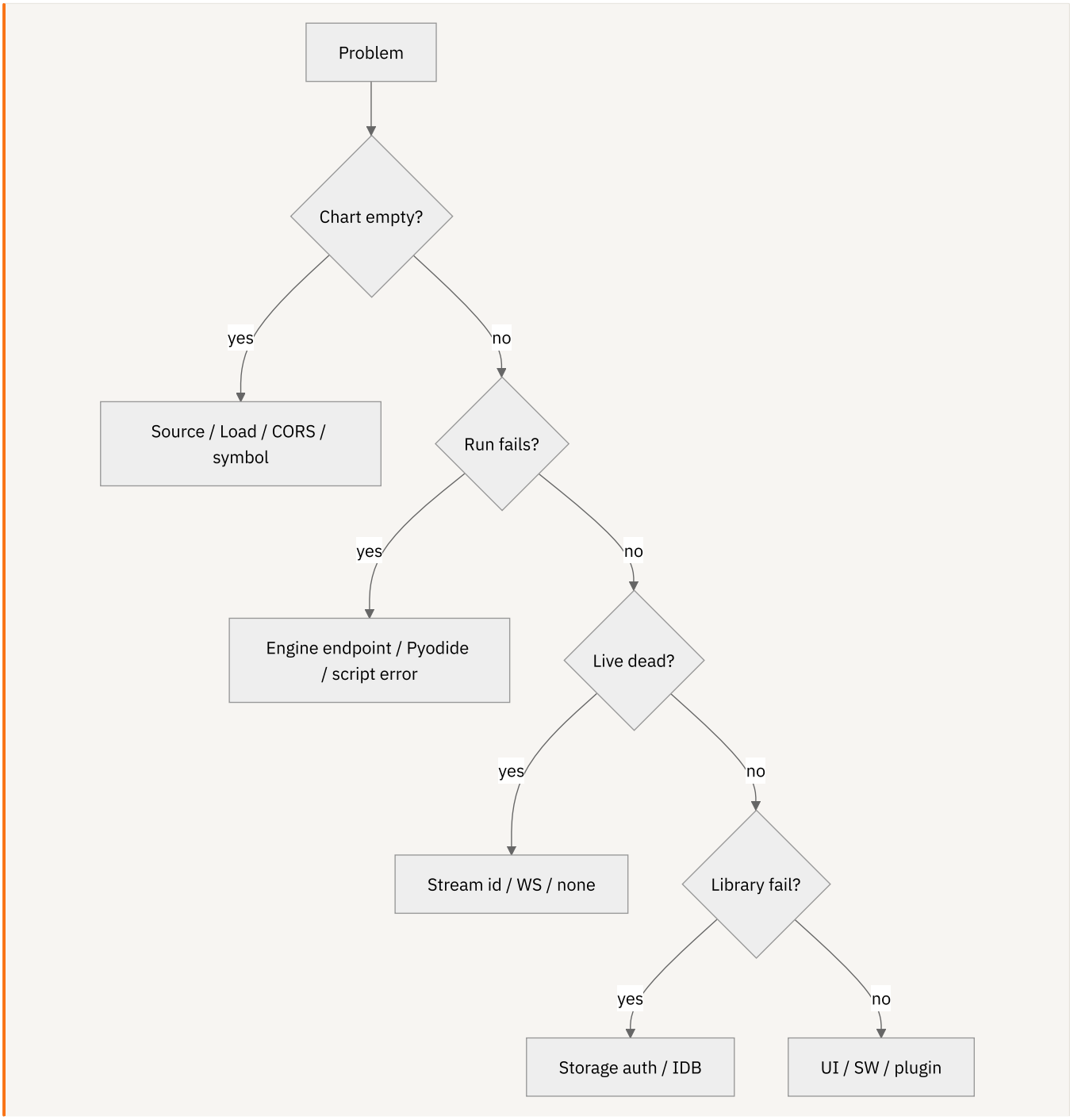
Systematic triage for AXIS. Prefer logs drawer + Network tab + status bar before reinstalling.

Abstract

Failure domains:

1. **Data plane** — source / stream
2. **Calc plane** — engine / endpoint / Pyodide assets
3. **Shell plane** — store, SW cache, plugins
4. **Storage plane** — library backends

Decision tree



Data plane

Symptom	Checks	Fix
Infinite loading	Network klines; status message	Correct symbol; try `mock-walk`
CORS on REST	Binance/public APIs usually OK; custom sources may fail	Proxy plugin or Worker
CSV no bars	Parse error in status	Fix columns/time; re-upload
Wrong interval bars	Store interval vs request	Settings interval + Load

Calc plane — server engine

Symptom	Checks	Fix
Failed fetch `/run`	Endpoint, CORS preflight	`ALLOWED_ORIGINS`; Settings probe
HTTP 4xx/5xx	Response JSON `message`	Fix script; backend logs
Timeout	Bar count × complexity	Shorter history; raise server resources
Plots missing	`status: success` but empty series	Script has no plots; check Raw

Probe: Settings → Test endpoint (health GET).

Calc plane — Pyodide

Symptom	Checks	Fix
HTML instead of wheel	Content-Type / ZIP magic	Deploy `public/vendor`, `public/pyodide` into `dist/`
Slow first run	Cold runtime	Wait; `preloadPyodide` on idle
Offline fail first visit	Assets never cached	One online warm-up
Interpreter error	Results Raw / logs	Language issue → PYNE

Live streams

Symptom	Checks	Fix
Live toggle no effect	Stream = `none`?	Select `binance-ws` or `mock-poll`
WS errors	Console; venue symbol format	Uppercase `BTCUSDT` style
Reconnect loop	Network flap	Logs drawer; stop/start Live
DO stream	Worker session query	Worker docs; example plugin

Storage / library

Symptom	Checks	Fix
Save no-op	Active storage; errors in panel	Read status line
Cloud 401	Bearer key	Settings/library key field
Git denied	Token scopes	PAT permissions
Lost local scripts	Cleared site data	Import JSON backup

Chart / editor (recent surfaces)

Symptom	Checks	Fix
Indicator blank / missing on chart	Oscillator on price scale; wrong pane id	Prefer `overlay=false`; re-run; see Indicators
Editor no lines / zero height	Dock height chain	Detach/reattach or toggle panel; editor should fill right strip
Editor covers Indicators	Side-by-side dock	Both open on right → Indicators left of Editor; resize each panel
Replay shows one candle only	Session start	Replay starts at **last bar** with full history; Play at end restarts from bar 0
No Debug chips / Pins	Toggles + log shape	Enable **Debug** / **Pins**; logs need `line` and/or `bar_index` — Debugging
Completions thin	Builtins corpus	Re-sync `pyne-builtins.json` from PYNE; optional remote LSP

Shell / PWA

Symptom	Checks	Fix
Old UI after deploy	Service Worker	Unregister SW; hard reload
State weird	`localStorage['pynescript.axis.v1']`	Export scripts; clear key; reload
Plugin vanished	URL install list	Re-install URL
Theme wrong	`data-theme`	Toggle theme once
Lost multi-chart slots	`chartLayout` in store	Named layouts menu; re-apply recipe

Logging

- **System logs** drawer: boot, pyodide ready, live re-run errors (`appendLog`)
- **Status bar**: ready / loading / running / error + short message

- **Results → Raw**: last engine payload

Security-related

- Poisoned localStorage: app should tolerate parse failures (see security tests)
- Plugin URL schemes restricted
- Never paste production PATs into shared screen recordings

Internals for deeper digs

Area	Path
Load history	`data/load-symbol.ts`
Run pipeline	`indicators/runner.ts`
Streams	`streams/multiplex.ts`
Engines	`engines/catalog.ts`
Store	`store/index.ts`

See also

- Installation
- FAQ
- DevOps
- Worker

GLOSSARY

AXIS end-user glossary: axes, AXIS, plugins, engines, storage keys, and result terms.

GLOSSARY

Terms as used in AXIS documentation and UI. Language runtime terms → PYNE.

Product

Term	Definition
AXIS	Charting PWA product : chart, editor, panels, and store—not the language engine.
UI track	Docs track for presentation surfaces (chart host, editor, shell, store) under `/axis/docs/ui`.
PYNE	Pine Script™ toolchain (parse/eval); engines embed or call it.
Active set	Tuple `{ source, stream, engine, storage }` currently selected.
Compose / recipe	A purposeful active set for a workflow (offline lab, desk, edge, …).

Plugin kinds

Term	Definition
Source	Historical OHLCV provider (`fetchHistorical`).
Stream	Live bar/tick push (`start` → dispose).
Engine	Calculator (`isReady` , `run`).
Storage	Script library backend (`list` / `read` / `write` / `remove`).
Component	Reserved UI slot plugins (phase 2).
Registry	In-memory catalog of plugins by kind (`PluginRegistry`).
configSchema	Declarative field schema for Settings / plugin config UI.
Capabilities	Flags: offline, needsAuth, needsNetwork, needsProxy.

Engines & backends

Term	Definition
server engine	`POST {endpoint}/run` with script + bars.
pyodide engine	In-browser Python + pynescript wheels.
Pro API / Flask	Local Python backend (`make run` , typically `:5002`).
Worker	Cloudflare Worker data plane for AXIS.
Endpoint	Base URL the server engine (and often cloud storage) calls.

Data & chart

Term	Definition
Bar	`{ time, open, high, low, close, volume? }`
Pane	Chart strip: price, volume, indicator, equity.
Overlay	Plot drawn on price pane (`meta.overlay !== false`).
Sub-pane	Non-overlay indicator strip; stable store id `indicator` .
User drawing	Interactive annotation tool geometry.
Script drawing	Engine-emitted line/label/box objects.
Marker	Trade/event marker on series.
Pane badge	Corner chip with script actions (settings / eye / re-run / remove).
Multi-chart	Grid layouts (1 / 2H / 2V / 4) with per-slot symbol/interval.
Bar Replay	Scrub/play over loaded OHLCV without live stream.
Compare	Second-symbol series overlaid on the price pane.
Volume profile	OHLCV volume-at-price histogram (Layers).
Watchlist	Side panel of symbols + quote refresh.
Scripts panel	Applied indicators/strategies list (UI title **Scripts**; internal id `indicators`).
Layers	Panel listing panes, scripts, and user drawings.
Data Source Manager	Background multi-page OHLCV backfill to a past date + gap validation (`Data` panel).
Bars cache	IndexedDB OHLCV store keyed by source/symbol/interval for the manager.
Chart theme preset	Named token set (void, classic, mono, obsidian, …) for chart + chrome accents.
Alert	Local price condition (+ optional webhook); Alerts panel.

Editor & debug

Term	Definition
Debug	Inline end-of-line chips from last-run logs with line refs.
Pins	Chart markers + □ gutter for logs with bar_index/time.
Problems	Clickable list of engine diagnostics for the last run.
Ruler	80-character column guide in the Pine editor.
Symbols	Editor catalog of TV-editor-safe arrows, box drawing, and chart emoji.
Command palette	⌘/Ctrl+K jump menu for panels, Run, toggles, git.
Git bar	Pull/Push via active storage plugin (local/cloud/git).

Results

Term	Definition
lastRun	In-memory last engine payload.
Event	Strategy/order/plot-adjacent event from engine.
Closed trade	Paired entry/exit with PnL in the viewer.
Equity curve	Cumulative PnL series derived from trades/events.
Scriptlogs	Floatable panel of Pine `log.*` from last run (not System Logs).
Optimise	Results tab that searches `strategy()` `input.*` values over N trials (pyne `/optimize` or isolated client fallback).
HPO	Hyperparameter optimisation — not a Pine builtin.

Persistence

Term	Definition
`pynescript.axis.v1`	Primary localStorage app state key.
`pynescript.axis.v2`	Older app-state key; migrated on read.
pluginsConfig	Per-plugin configuration map.
Library export	JSON dump of storage documents.

Infrastructure names

Term	Definition
`axis`	CF Pages project id. Canonical URL `https://axis.hoox.sh`.
`axis.hoox.sh`	Canonical AXIS Pages URL.
`worker-axis`	CF Worker script name and health JSON `service`.
`worker.axis.hoox.sh`	Canonical AXIS Worker URL.
void	Brand pack / dark chrome aesthetic for AXIS docs/UI.

See also

- FAQ
- State namespaces
- Architecture overview

FAQ

Frequently asked questions about AXIS, engines, offline use, TradingView relation, and data privacy.

FAQ

Product scope

Is AXIS TradingView?

No. AXIS is an independent charting PWA. Pine Script™ and TradingView® are trademarks of TradingView, Inc. This project is not affiliated with or endorsed by TradingView, Inc. Trademark notice appears on the AXIS docs home.

Does AXIS implement Pine Script?

Evaluation is performed by **engines** that call **PYNE** (or proxies that do). The AXIS does not embed a closed interpreter. See PYNE for language fidelity.

AXIS vs engine?

AXIS = UI + orchestration. Engine = `run(script, bars)`. You can swap engines without rewriting the chart host. That is **ADR-002** territory—ADRs.

Running & offline

Can I use AXIS fully offline?

Yes, with recipe **Offline lab**: `mock-walk` + `mock-poll` + `pyodide` + `local`, after one warm-up so Pyodide assets cache. Server engine and live venue streams require network.

Why is first Pyodide run slow?

Bootstraps Python + wheels in-browser. Subsequent runs reuse the runtime when still warm. Assets are self-hosted under the PWA origin when deployed correctly.

Default endpoint points at a VPS—is that required?

No. Change Settings → Backend URL to `http://localhost:5002` or your Worker. Defaults may reference a public demo API host for convenience.

Data & privacy

Where do API keys live?

In **this browser's** localStorage state / `pluginsConfig`. They are sent as HTTP headers only to endpoints you configure (cloud storage, etc.). They are not part of the Pine script body.

Are OHLCV bars uploaded?

Server engine sends script + bars to the configured endpoint for that run. **Pyodide** keeps bars local. Choose engine accordingly.

Are bars saved on disk?

App state persistence **omits** `bars`, `lastRun`, and `logs` to control size. History is re-fetched on Load.

Features

How do I share a setup?

Legacy hash state (`state-hash.js`) can encode symbol/interval/engine/source/stream (and short scripts). Prefer documenting the four-tuple + library export for durable sharing. URL-load plugins are re-fetched by install list.

Can I use my own data vendor?

Yes—implement a **source** plugin (and optional stream) and install via URL or built-in registration. See Plugins.

Why Strategy tab shows no trades?

Need paired entry/exit-style events with prices. Indicators that only `plot` will fill Plots/Metrics, not Strategy stats.

Drawings disappeared after Run

Script drawings are cleared and re-applied each run. User toolbar drawings persist across runs.

Popout editor out of sync?

Editor bridge uses shared document storage + message bridge between windows. Use **Reattach** on the main AXIS if the popout closed uncleanly.

What is Debug vs Pins?

	Debug	Pins
Shows	End-of-line chips in the editor	Markers on the chart + □ gutter
Needs	Line reference in logs/errors	`bar_index` and/or bar time
Toggle	Editor header **Debug**	**Pins** or **Alt+P**

Full workflow: Debugging.

Multi-chart, Compare, Replay, Alerts?

- **Layouts** topbar menu: 1 / 2H / 2V / 4 + named recipes
- **Compare**: second symbol overlay (% or absolute)
- **Replay**: scrub loaded history (starts with full history at last bar)
- **Alerts**: local price alerts panel (webhook optional)
- **⌘/Ctrl+K**: command palette (panels, Run, editor toggles, git)

UI detail: UI shell, Charting.

Why are Scripts next to the Editor?

Left/right docks with multiple open panels lay out **side-by-side** (row), so the **Scripts** panel can sit left of Editor on the right strip. Bottom dock still stacks. (Internal panel id is still `indicators` for saved layouts.)

Can AXIS tune strategy inputs?

Yes — Results → **Optimise**, or command palette **Optimise strategy**. Strategies only. pyne `POST /optimize` runs TPE / random / grid with holdout by default; Pyodide falls back to a local random loop. See Hyperparameter Optimisation.

How do I download years of history?

Topbar **Load** is one venue page. Open the **Data** panel (Data Source Manager) to backfill in the background to a past date, validate completeness, and fill gaps — see Data Source Manager.

Ops

What is `pynescript-axis`?

Former Cloudflare Pages / Worker id. Pages project is now **`axis`** (`https://axis.hoox.sh`); Worker script is **`worker-axis`** (`https://worker.axis.hoox.sh`). Brand is AXIS.

Legacy localStorage keys?

Migrated automatically into `pynescript.axis.v1` and library keys. See State namespaces.

Tests for the PWA?

```
cd frontend && bun run test:unit
bun run test:e2e:smoke
```

See `frontend/TESTING.md`.

See also

- Troubleshooting
- Glossary
- Compose recipes

ARCHITECTURE

AXIS system architecture: AXIS vs engine, plugin registry, topologies, ADRs, and state namespaces.

ARCHITECTURE

AXIS architecture is the discipline of **separable axes**—price history, live time, calculation, and library storage—coordinated by a Solid AXIS UI that never owns a closed interpreter.

Abstract

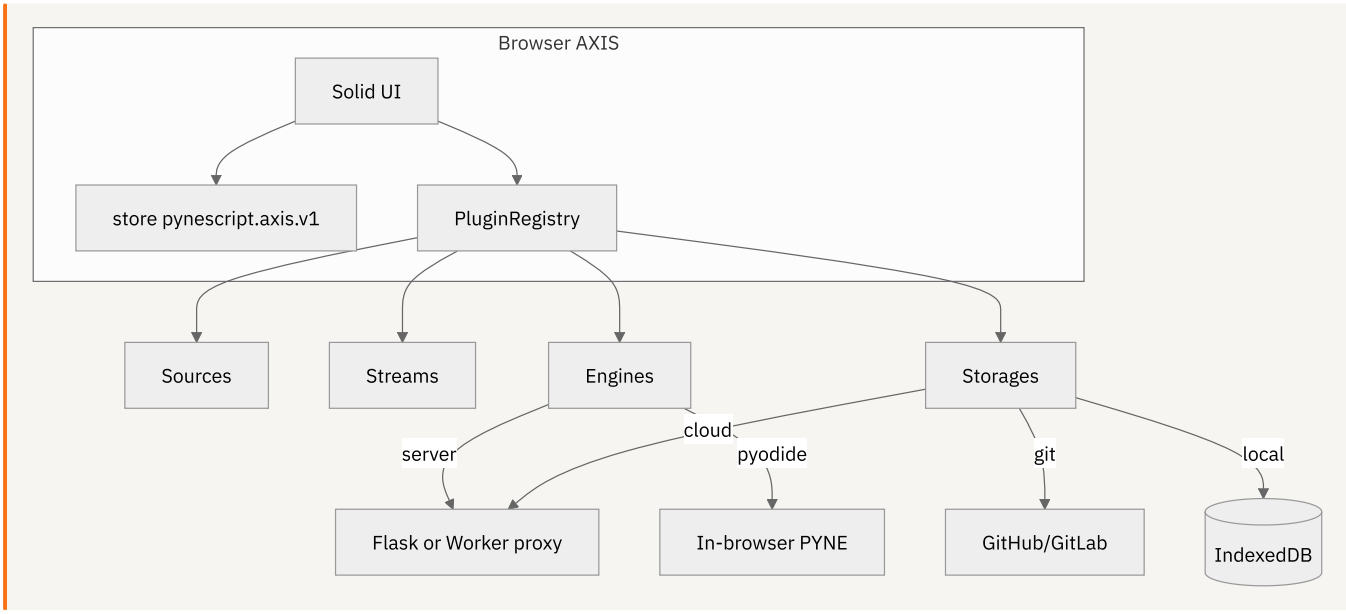
Layer	Responsibility
AXIS	UI, store, plugin orchestration, chart apply
Plugin contracts	`source | stream | engine | storage | dataset` (+ reserved component)
Engines	PYNE evaluation (browser Pyodide or remote `/run`)
On-chain plane	Parallel non-OHLCV datasets (TVL, DEX, events) via Worker allowlist proxy — ADR-015
Optional edge	Cloudflare Pages + Worker + DO/KV/D1/R2
Optional desk API	Flask Pro API

Invariant: AXIS ≠ engine. Evaluation always crosses an engine plugin boundary.

Track map

Page	Contents
Overview	End-to-end data/control flow
ADRs	ADR-001 … ADR-015
Topologies	Dev, static, edge deploy shapes
State namespaces	Keys, migration, hash, IDB
Drawings and plot parity	User tools vs Pine plots/drawings vs PYNE export

Conceptual model



Formal namespaces

```
Contract namespace:  pynescript.axis.plugins.v1
App state key:       pynescript.axis.v1
App state prefix:    pynescript.axis.*
Pages project id:    axis
Pages URL:           https://axis.hoox.sh
Worker script id:    worker-axis
Worker URL:          https://worker.axis.hoox.sh
Health service id:   worker-axis
```

Related tracks

- End User — operator workflows
- UI — UI subsystem design
- Plugins — contracts & catalogs
- Worker — edge data plane
- DevOps — build & CORS
- PYNE runtime — language evaluation

See also

- Root `README.md` — package-level map
- `LEGACY.md` — pre-Solid shell boundary
- AXIS CLI — operator install / deploy

ARCHITECTURE OVERVIEW

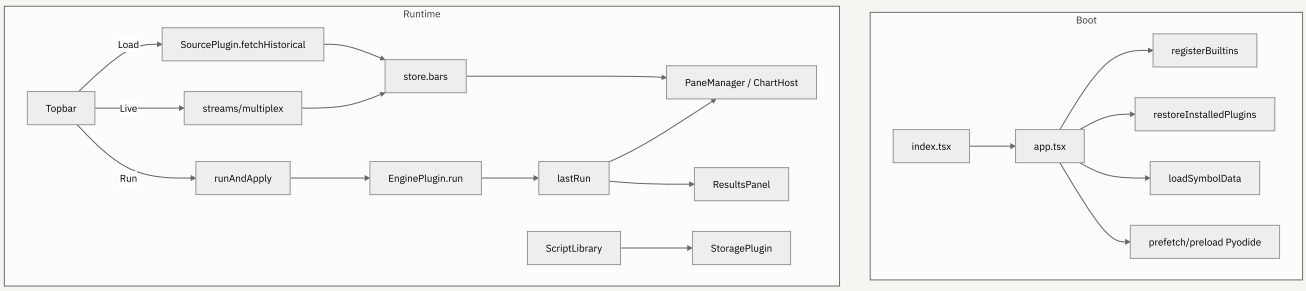
End-to-end AXIS architecture: Solid AXIS UI, unified registry, dual engines, chart apply pipeline, and optional edge plane.

ARCHITECTURE OVERVIEW

Abstract

AXIS is a **composition host**. The product thesis: own the axes (history, live, calc, library), swap the implementations. Shipping UI is Solid + Vite + lightweight-charts + CodeMirror 6; calculation is never inlined into UI components.

Conceptual model



Interface surface (module map)

Concern	Primary paths
Entry	`src/index.tsx` , `app.tsx`
State	`src/store/`
Plugins	`src/plugins/{registry,types,loader,bootstrap,active}.ts`
Catalogs	`sources/catalog.ts` , `streams/*` , `engines/catalog.ts` , `storage/*`
Chart	`chart/ChartHost.tsx` , `pane-manager.ts` , `drawing-layer.ts` , `price-precision.ts`
Editor	`editor/*`
Run apply	`indicators/runner.ts`
Results	`results/*` , `ui/ResultsPanel.tsx`
Studio pages	`ui/studio/` , `ui/runtime/` , `ui/wire/` , `ui/settings/` , `ui/workers/` , `ui/plugins/`
Workers catalog	`workers/*`
Worker	`worker/`
CLI	`packages/cli/`
Desktop	`src-tauri/` , `src/desktop/`

Legacy parallel: `state.js` , `registry.js` , `main.js` — not product path. See repo `LEGACY.md` .

Control plane vs data plane

Plane	What moves	Who owns it
Control	activePlugins, endpoint, theme, layout	Solid store + localStorage
Historical data	Bar arrays	Source plugins → `store.bars`
Live data	Bar updates	Stream plugins → multiplex → bars/chart
Calculation	script + bars → RunResult	Engine plugins
Library	ScriptDocument	Storage plugins
Telemetry	plane connectivity / latency / ticks	Ephemeral `store.telemetry` + Connection HUD

Transport preference (ADR-014)

Path	Preferred transport	Notes
Load (history)	REST / local	Venue kline HTTP, CSV, mock walk
Live	**WebSocket**	Venue WSS with reconnect; mock-poll local; DO = brokered
Engine server	HTTP	`POST /run` batch; latency in HUD
Engine pyodide	Local	Offline-capable after asset cache

Live pairs with history via `defaultStreamForSource`. `live.preferAfterLoad` auto-starts WS after Load and on boot when bars exist (default **on**).

Dual engines

Engine	Transport	Fidelity host
`server`	HTTP JSON `{ script, data }` → `/run`	Flask or Worker→Flask
`pyodide`	In-worker-thread-ish browser Python	Self-hosted wheels + `pynescript_runtime.py`

Timeout policy in runner scales with bar length (clamped). Live re-runs use shorter silent timeouts.

Chart apply pipeline

`runAndApply`:

1. `getActiveEngine().run(...)`
2. `setLastRun`
3. Optionally open results
4. **Sync** overlays in place (`syncOverlayLines`) — no blank destroy frame
5. Markers + strategy report side effects
6. Atomic Pine script drawings replace
7. Equity pane when appropriate (skip hide thrash on silent live re-runs)

History vs live data path: `loadBars` bumps `chartDataGen` → ChartHost full `setDataToChart` + fit. Live ticks only `manager.appendBar` (never full setData/fitContent).

PaneManager is imperative DOM under a Solid-owned outer shell—**Solid must not reconcile the pane root** (ChartHost invariant).

Plugin registry

`PluginRegistry` holds ordered maps per kind; events `registered` / `unregistered`. Built-ins register once (`ensureBuiltins` / catalog `ensure*Registered`). Dynamic plugins via `loadPluginFromUrl`. Active selection is **not** inside the registry—it is store state.

Contract namespace conceptually: `pynescript.axis.plugins.v1` (fields: `id`, `name`, `kind`, `configSchema`, `capabilities`, kind-specific methods).

Edge plane (optional)

Cloudflare split:

- Pages project **`axis`** (`https://axis.hoox.sh`): static `dist/`
- Worker script **`worker-axis`** (`https://worker.axis.hoox.sh`): `/api/run` proxy, keys/usage KV, scripts D1, stream DO fan-out
- AXIS still speaks engine/storage plugins—not raw CF APIs in UI components

Invariants

1. AXIS ≠ engine
2. Bars/logs/lastRun not fully persisted
3. Storage and engine endpoints may coincide (Worker) but remain separate plugin roles
4. Older keys migrate forward, never the reverse on write

Failure modes (architectural)

Mode	Manifestation	Mitigation
God-object UI	Logic in components	Keep runner/registry pure modules
Engine leakage	Fetch `/run` from random widgets	Only engine plugins perform calc I/O
Registry/store split brain	Flat `engine` ≠ `activePlugins.engine`	`setActivePlugin` keeps alignment
SPA asset fallback	Pyodide loads HTML	Assert ZIP magic on assets

See also

- ADRs

- Topologies
- State namespaces
- AXIS store

EVALUATION MAP

How AXIS runs Pine: Flask Pro API, Worker proxy, in-browser Pyodide. AXIS does not import PyneTS and does not place HOOX orders.

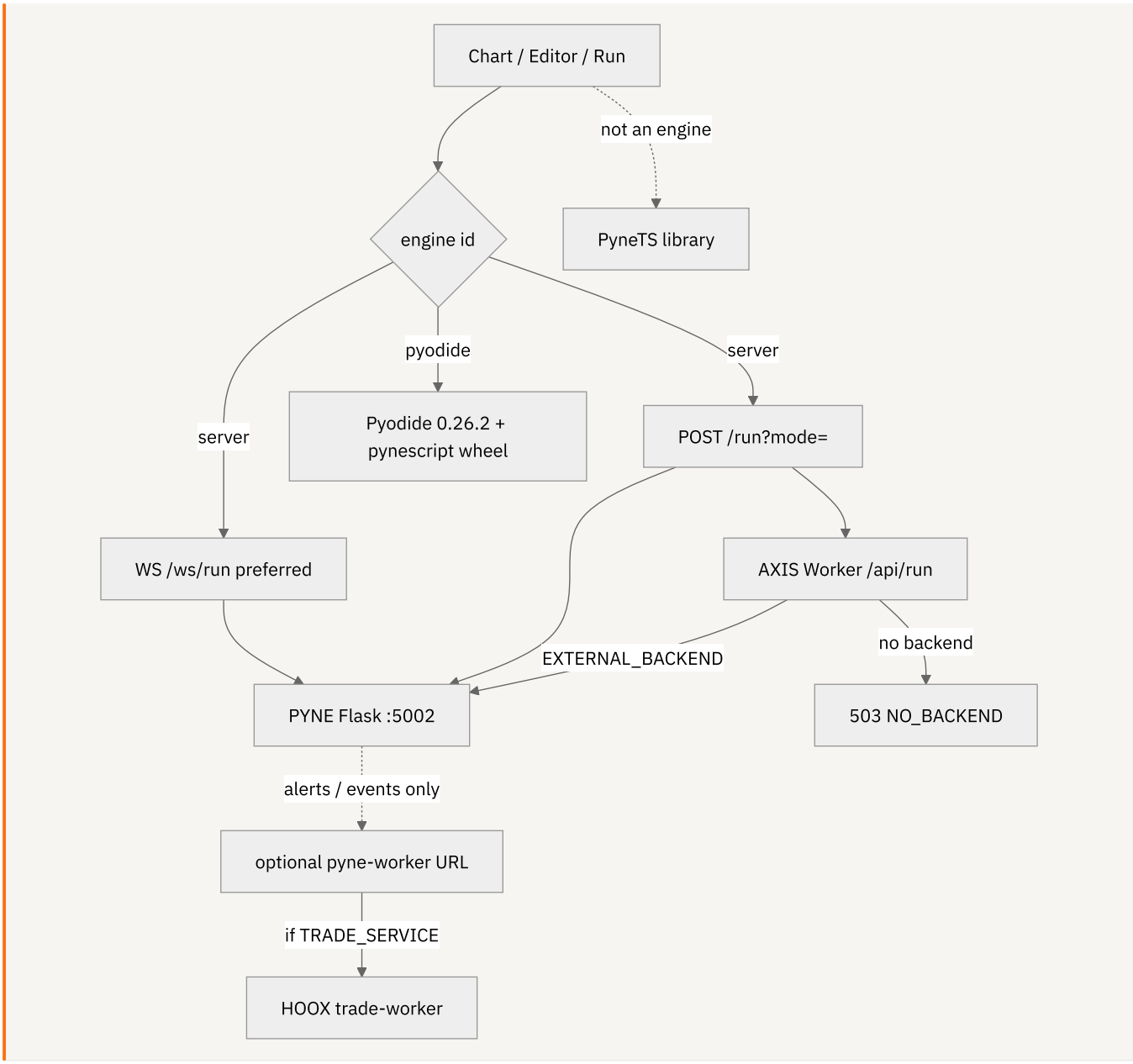
EVALUATION MAP

Abstract

AXIS is a charting PWA, not a language rewrite. Every Pine evaluation goes through an **engine plugin**. Built-ins are `server` (HTTP/WS to PYNE) and `pyodide` (in-browser Python wheel). Optional HOOX **pyne-worker** is just another Backend URL.

AXIS does not import `@hoox-sh/pynets`. AXIS does not call `trade-worker`. Drawing a strategy on the chart is research, not execution.

Conceptual model



Interface surface

Path	Implementation	Notes
Engine `server`	`src/engines/catalog.ts`	Prefer **WS** `ws(s)://{endpoint}/ws/run` (`preferWs` default true), else **POST** `{endpoint}/run?mode=` with `{ script, data, mode, inputs?, profiler?, libraries? }` . Modes `interpret | compile | auto` . Timeout scales with bar count (45–180s). Default endpoint `http://localhost:5002` . Strategy HPO uses **`POST {endpoint}/optimize`** .
Engine `pyodide`	same file	Self-hosted **Pyodide 0.26.2** (`/pyodide/v0.26.2/`), wheels `/vendor/pynescript-0.4.0-py3-none-any.whl` + antlr4, runtime `/pyodide/pynescript_runtime.py` . **Numba is not in Wasm**; `compile` / `auto` is object-mode. Cold load ~20–30s.
AXIS Worker `/api/run`	`worker/src/runtime.ts`	`PYODIDE_IN_WORKER=enabled` scaffold; else **proxy `EXTERNAL_BACKEND/run`** ; else 503 `NO_BACKEND` . Caps: 512 KiB script, 50k bars, 30 req/min/IP, 60s upstream.
HOOX **pyne-worker**	`src/workers/catalog.ts`	Not a built-in engine. Paste origin as Backend URL. Distinct from the AXIS Worker (`worker-axis`).
PyneTS	—	TypeScript library. PYNE docs. **Not referenced** in AXIS source.

Payloads follow the PYNE evaluate contract. Defaults: Flask `POST /run` uses `mode` **`auto`** . See modes.

Internals

Path	Role
`src/engines/catalog.ts`	`server` + `pyodide`
`src/workers/catalog.ts`	Workers page catalog entries
`worker/src/runtime.ts`	Edge `/api/run`
`public/pyodide/`	Wasm runtime + wheel
`scripts/sync-pyne-wheel.sh`	Refresh vendored wheel

Invariants & edge cases

1. **AXIS ≠ engine** (ADR-002). The shell never embeds a closed interpreter.
2. **AXIS ≠ execution.** Strategy events on the Results panel are not HOOX orders. See AXIS and HOOX.
3. **No `none` stream plugin** in `src/streams/catalog.ts` . Pausing live data is `stopLive()` , not a plugin id.
4. Pyodide wheel version can lag PyPI (`0.3.7` wheel vs `hoox-pyne` 0.3.9). Check `vendor/` .
5. CORS: Worker allows localhost, `*.hoox.sh` , `*.pynescript.online` , `*.axis.pages.dev` (plus legacy `*.pynescript-axis.pages.dev`) only.

Worked examples

Local desk

```
# terminal 1 — PYNE Pro API
cd pynescript && make run    # :5002

# terminal 2 — AXIS
cd axis && bun run dev
```

Manager → engine `server` , Backend URL `http://localhost:5002` .

Offline

Switch engine to `pyodide` . First run downloads/caches the self-hosted runtime. Do not expect Numba.

Edge proxy

Deploy AXIS Worker with `EXTERNAL_BACKEND` pointing at Flask or pyne-worker. Engine stays `server` ; endpoint is the Worker origin. Still no trade-worker hop unless **that** backend forwards events.

Failure modes

Symptom	Cause	Fix
503 `NO_BACKEND`	Worker has no `EXTERNAL_BACKEND` and Pyodide-in-worker off	Set backend or use browser Pyodide
Compile "slow / different" in Pyodide	No Numba	Use Flask for numeric compile
Expecting pynets in the PWA	Not wired	Import `@hoox-sh/pynets` in your own app
Strategy PnL on chart, no CEX fill	AXIS does not execute	pyne-worker live trading

See also

- Engines

- Worker runtime
- PYNE Pro API
- PyneTS
- HOOX pyne-worker
- AXIS and HOOX

ARCHITECTURE DECISION RECORDS

ADR-001 through ADR-015 for AXIS: registry, AXIS≠engine, Solid+Vite, dual engines, configSchema, URL load, storage, CF project id, proxies, DO fan-out, migration, CORS, results export, transport preference + telemetry, on-chain data plane.

ARCHITECTURE DECISION RECORDS

Formal ADRs for AXIS (product + `frontend/`). Status values: **Accepted** unless noted. Dates are conceptual product epochs, not git archaeology.

Index

ID	Title
ADR-001	Pluggable unified registry
ADR-002	AXIS ≠ engine
ADR-003	Solid + Vite product UI
ADR-004	Dual engines (server + Pyodide)
ADR-005	Declarative configSchema
ADR-006	URL-loadable plugins
ADR-007	Storage as a plugin
ADR-008	CF ids: Pages `axis`, Worker `worker-axis`
ADR-009	Worker proxies first
ADR-010	Durable Object stream fan-out
ADR-011	State namespace migration
ADR-012	CORS as deploy concern
ADR-013	Results export in AXIS
ADR-014	Transport preference + connection telemetry
ADR-015	On-chain data plane
ADR-016	Provider-locked market data; CCXT on the server

ADR-001: Pluggable unified registry

Context

Early AXIS registered sources/streams/engines in separate ad hoc maps. Operators and authors needed one mental model and one install path.

Decision

Adopt a **unified** **`PluginRegistry`** (`frontend/src/plugins/registry.ts`) with kinds:

`source | stream | engine | storage | component(reserved)`

Every plugin shares `PluginBase` (`id`, `name`, `kind`, `description`, `version`, `builtIn`, `configSchema`, `capabilities`, lifecycle hooks). Registration is ordered; listeners observe register/unregister.

Consequences

- Active selection lives in the Solid store, not the registry.
- Built-ins protected from casual unregister.
- Catalogs (`sources/catalog.ts`, …) become registration facades.
- Contract docs under Plugins.

ADR-002: AXIS ≠ engine

Context

Closed chart hosts couple rendering and language runtime. That blocks offline mode, alternate evaluators, and headless reuse of PYNE.

Decision

AXIS never embeds a closed interpreter. All evaluation goes through `EnginePlugin.run`. UI modules call `getActiveEngine()` / `runAndApply`, not Python bindings directly (except inside the pyodide engine module).

Consequences

- Language fidelity owned by PYNE.
- AXIS docs do not duplicate grammar theory.
- New engines (tiny demo, remote Worker, future WASM) are additive plugins.

ADR-003: Solid + Vite product UI

Context

Legacy vanilla JS shell (`main.js` , `style.css`) mixed DOM wiring with business logic and aged TV-blue tokens.

Decision

Product path is **SolidJS + Vite + Tailwind 4**, entry `src/index.tsx` / `app.tsx` . Imperative chart library (lightweight-charts) is isolated in `ChartHost` / `PaneManager` so Solid reconciliation does not thrash canvas DOM.

Consequences

- Legacy files retained for smoke/static tests only (`LEGACY.md`).
- Icons via `lucide-solid` .
- Deploy artifact is Vite `dist/` , not repo-root static tree.

ADR-004: Dual engines (server + Pyodide)

Context

Researchers need server fidelity and offline demos. A single transport cannot satisfy both without compromise.

Decision

Ship two built-in engines:

Id	Role
`server`	`POST {endpoint}/run?mode=` with `{ script, data: bars }`
`pyodide`	Browser Python + vendored pynescript / antlr wheels

Both implement the same `EnginePlugin` contract and return `RunResult` .

Consequences

- Endpoint field only for server-like engines.
- Pyodide requires correct static asset deployment (ZIP integrity checks).
- Timeouts unified at runner layer.

ADR-005: Declarative configSchema

Context

Per-plugin settings UIs do not scale; custom React/Solid forms per plugin forks the manager.

Decision

Plugins declare optional **`configSchema`** : map of field name → `{ type, default, label, options, min, max, ... }` . Settings and library panels resolve defaults + `pluginsConfig` overrides via shared resolve helpers.

Consequences

- New plugins get settings UI without shell changes (within field types).
- Types: `string | number | boolean | select` initially.
- Invalid user values remain a plugin responsibility at runtime.

ADR-006: URL-loadable plugins

Context

Third-party sources/engines should not require rebuild of the PWA for experiments.

Decision

Support **`loadPluginFromUrl`**: fetch ES module, validate shape, register, persist install metadata for restore on boot. Example plugins published under `public/plugins/`.

Consequences

- Browser origin trust model: URL plugins are code execution.
- Security tests constrain schemes and storage-via-URL footguns.
- Offline restore needs prior successful fetch/cache.

ADR-007: Storage as a plugin

Context

Script library began as localStorage-only; cloud and git were bolted differently.

Decision

Storage is a first-class plugin kind with `list/read/write/remove` (+ optional draft/sync/status). Built-ins: `local` (IDB), `cloud` (Worker `/api/scripts`), `git` (GitHub/GitLab Contents API). Active storage in `activePlugins.storage`.

Consequences

- Manager Script Library is backend-agnostic.
- Git commits only on explicit save; drafts local.
- Cloud uses Bearer Pro keys and optional optimistic concurrency.

ADR-008: Frozen CF project id

Context

Renaming Cloudflare Pages/Worker projects severs KV/D1/R2 bindings, custom domains, and CI secrets. Product brand evolved to AXIS. The original freeze used a single id `pynescript-axis` for both Pages and the Worker.

Decision

Pages project is `axis`. Canonical URL `https://axis.hoox.sh` (custom domain). Preview hashes live on `*.axis.pages.dev`. The previous Pages id `pynescript-axis` remains a CORS legacy alias for old preview hosts.

Worker script is `worker-axis`. Canonical URL `https://worker.axis.hoox.sh` (custom domain). Health JSON `service` is `worker-axis`. The pre-rename workers.dev host (`pynescript-axis.<account>.workers.dev`) is treated as a legacy alias in client host matching.

Brand in UI/manifest/docs is AXIS.

Consequences

- `wrangler.toml` `name` is `worker-axis`; Pages `--project-name` is `axis`.
- Canonical hosts: Pages `https://axis.hoox.sh`, Worker `https://worker.axis.hoox.sh`.
- Docs call out the split explicitly (this ADR).
- npm package name (`axis-worker`) may lag brand.

ADR-009: Worker proxies first

Context

In-Worker Pyodide is attractive but heavy; production needed a path immediately.

Decision

Worker **`/api/run` proxies to an external Python backend** first (`EXTERNAL_BACKEND`). In-Worker runtime remains scaffolded/gated. Keys, usage metering, scripts API, and stream DO can ship independently of full edge evaluation.

Consequences

- AXIS `server` engine talks to Worker URL transparently.
- Operational dependency on Flask (or equivalent) until edge runtime matures.
- Clear layering: Worker is data plane + auth, not necessarily the interpreter.

ADR-010: Durable Object stream fan-out

Context

N browser clients each opening venue WebSockets waste connections and risk rate limits.

Decision

`SessionDO` Durable Object: one upstream WS per session key; fan-out to N client sockets via `/api/stream`. Hibernation-friendly design on CF.

Consequences

- Stream plugins may target DO URLs (example plugin provided).
- Session/symbol/interval query params are part of the contract.
- Not a historical source—live only.

ADR-011: State namespace migration

Context

Hard-renaming storage namespaces would brick user layouts and libraries if keys changed hard.

Decision

Canonical key **`pynescript.axis.v1`**. On read, fall back to `pynescript.axis.v2` (and parallel library/draft keys), then **write forward**. Do not keep dual-write forever.

Consequences

- Seamless upgrade for existing browsers.
- Persistence strips bars/lastRun/logs.
- Tests cover migration (`tests/state.test.ts` patterns).

ADR-012: CORS as deploy concern

Context

Browser AXIS on origin A calling Pro API on origin B fails without CORS. Hardcoding origins in AXIS is wrong.

Decision

CORS is a backend/deploy configuration (`ALLOWED_ORIGINS` on Flask; Worker headers as deployed). AXIS documents required origins; ops sets explicit lists in production (avoid perpetual `*` outside demos).

Consequences

- Localhost regex for dev.
- Troubleshooting centers on preflight, not engine code.
- Static file server does not replace API CORS.

ADR-013: Results export in AXIS

Context

Researchers need artifacts (trades CSV, run JSON) without a separate analytics service.

Decision

Results panel owns export: download JSON of `lastRun`, CSV of closed trades, clipboard helpers. Strategy pairing and stats run client-side from engine events (`results/strategy.ts`, `results/events.ts`).

Consequences

- Export fidelity bounded by engine event quality.
- No server-side report store required for MVP.
- Viewer metrics ≠ broker statements (documented for operators).

ADR-014: Transport preference + connection telemetry

Context

Operators need to see **how** data and calculation move (WS vs REST vs local vs brokered), whether the live socket is actually open, and engine run latency. Naively "prefer WebSocket everywhere" is wrong: venues expose **history over REST** and **live klines over WSS**. Server engines batch-run over HTTP; Pyodide is local WASM.

Decision

1. **Transport policy**
   - History → REST (or local mock/CSV).
   - Live → **WebSocket first**, paired per source via `defaultStreamForSource` (`binance-rest` → `binance-ws`, …).
   - Engine → **prefer** `WS /ws/run` on the Pro API when available (`flask-sock`); fall back to HTTP `POST /run`. Pyodide remains local.
   - Brokered → CF Durable Object / `needsProxy` streams surface as transport class `broker`.
2. **Honest stream state** — `connecting` until `onStatus({ state: 'open' })`; reconnect uses `reconnecting` / degraded telemetry, not false green.
3. **Venue streams use reconnect with exponential backoff** (`streams/reconnect-ws.ts`).
4. **Connection HUD** — StatusBar second row (`ConnectionHud`) reads ephemeral `store.telemetry` (SRC/STR/ENG/STO chips, tick pulse, engine latency). Persist only `telemetry.hud` prefs + `live.preferAfterLoad` / `live.rerunOn`.
5. **Live re-run modes** — `every-tick` (default) or `bar-close` (venue `Bar.closed` or bar time advance).
6. **Chart stability** — full `setDataToChart` only on history load (`chartDataGen`); live uses `appendBar`; overlay/script drawings update in place to avoid hide/show flash.

Consequences

- UI never claims WS for historical load.
- Auto-live after Load is **on by default** (`preferAfterLoad`). Turn off in Settings → Live stream.
- Plugin capabilities may declare `transport`; otherwise HUD classifies from id/capabilities.
- See Streams, UI shell, Overview.

ADR-015: On-chain data plane

Context

AXIS began as an **OHLCV-first** chart host: `SourcePlugin` / `StreamPlugin` deliver `Bar[]` (open, high, low, close, volume) for CEX and similar venues, then engines run Pine against that series. Operators also need **non-price market structure**:

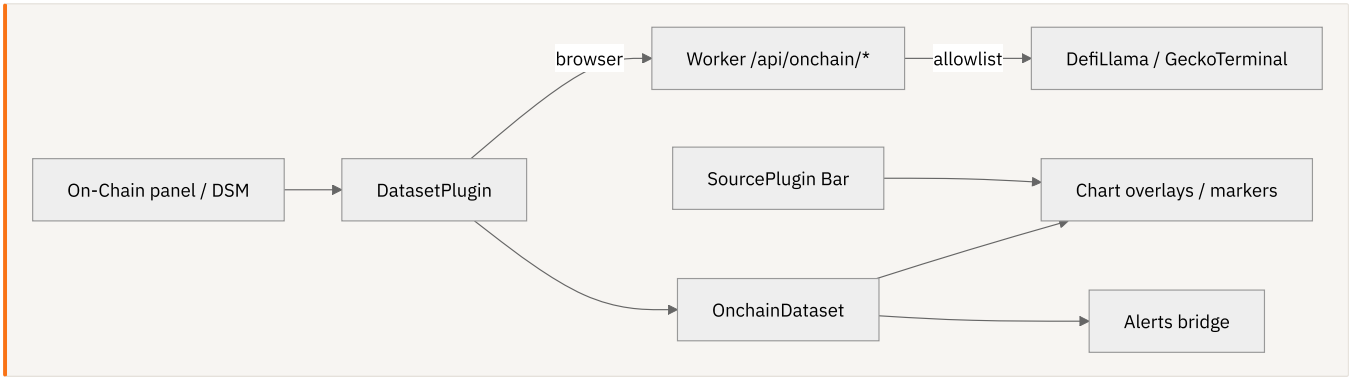
Need	Why `Bar` alone fails
Protocol TVL (USD)	Single scalar liquidity total, not OHLC
DEX pool candles	Same shape as bars, but different venue semantics, address/network keys, and page caps
Spike / unlock-style events	Discrete time markers with severity, not candles
Honest provenance	Provider, snapshot cadence, finality — not implied by "the chart symbol"

Forcing these into the primary `Bar` pipeline would corrupt scale semantics (TVL vs price), invent fake OHLC for scalar metrics, and couple wallet/signing UX to read-only analytics.

Decision

Ship a **parallel on-chain data plane** orthogonal to the OHLCV source/stream path:

1. **Plugin kind** `dataset` on the unified registry (`DatasetPlugin` in `src/plugins/types.ts`):
   - `fetchDataset(opts) → OnchainDataset`
   - optional `searchInstruments`
   - payload kinds: `ohlcv | scalar_series | multi_series | events | table`
   - required **provenance** + **finality** (`pending | safe | finalized | unknown`)
2. **Parallel plane modules** under `src/onchain/` (types, cache, keys, adapters, catalog, manager, events, alerts bridge) — not a new `Bar` field and not Pine output.
3. **Worker allowlist proxy** (`worker/src/onchain.ts`, routes under `/api/onchain/…`):
   - rewrite only known DefiLlama / GeckoTerminal paths
   - validate network ids and pool addresses
   - short-TTL isolate cache (`X-Axis-Onchain-Cache`)
   - client default bases from `store.endpoint` (`src/onchain/proxy.ts`)
4. **Chart UX**: attach as host-side overlays (left scale for scalars; separate series keys `onchain_*`); main market OHLCV stays on the right scale. Not wallet, not Pine plots.



Alternatives considered

Alternative	Rejected because
Force into `Bar` / active source	TVL is not OHLC; faking open=high=low=close lies to engines and the price scale. Multi-protocol attach becomes multi-symbol load thrash.
Piggyback Pine `request.security` only	Needs engine + script for every host overlay; no attach UI, no provider search, no finality UI.
Browser → provider direct	Public DefiLlama / GeckoTerminal hosts often lack CORS for arbitrary AXIS origins (ADR-012).
Open reverse proxy (any URL)	SSRF risk; Worker must **allowlist** path shape, not pass-through arbitrary upstream.
Wallet-connected RPC as the plane	Different product (signing, accounts, private RPC). On-chain here means **public analytics APIs**, not MetaMask.

Consequences

- **CORS / deploy** — browsers should use `{endpoint}/api/onchain/llama|gecko`; local Worker on `:8787` or deployed edge. Direct `baseUrl` overrides are for Node tests / trusted environments only.
- **Finality honesty** — DefiLlama TVL is typically a **daily snapshot**, not block-final chain state; Gecko OHLCV is **pool** sampling, not CEX marks. UI shows provenance lines; `synthetic` flags fabricated OHLC.
- **Not a wallet** — no signing, no injected providers, no private keys for this plane.
- **Not Pine** — overlays are host datasets; they do not replace `indicator()` / `strategy()` plots from the engine.
- **Engines unchanged** — active OHLCV source still feeds `EnginePlugin.run`; on-chain series do not silently become script `close`.
- **Cache** — client IDB dataset cache (`instrumentCacheKey`) plus Worker short-TTL memory; operators may see stale TVL briefly after large DeFi moves.
- **Security** — Worker rejects path traversal, invalid slugs/addresses, non-GET, and non-allowlisted Gecko paths.

Phases shipped

Phase	Capability	Primary surfaces
1 — TVL	DefiLlama protocol TVL scalar history + search	`defillama-tvl` dataset, `src/onchain/defillama.ts`, llama proxy
2 — DEX	GeckoTerminal pool OHLCV + pool search; optional `geckoterminal-ohlcv` source for DSM multi-page backfill	`geckoterminal` dataset, `src/onchain/geckoterminal.ts`, gecko proxy
3 — Events	Derived day-over-day TVL spike/drop events (`tvl_spike` / `tvl_drop`)	`src/onchain/events.ts` → chart markers
4 — Alerts	Bridge events into alerts engine (`onchain_tvl_spike` / `onchain_event`)	`src/onchain/alerts-bridge.ts`, `src/alerts/engine.ts`

Pro-only DefiLlama surfaces (raises, unlocks via `pro-api.llama.fi`) remain out of the free Worker allowlist.

Code map

Path	Role
`src/onchain/types.ts`	`OnchainDataset`, instruments, finality
`src/onchain/catalog.ts`	Built-in dataset registration
`src/onchain/manager.ts`	Attach / hide / detach store wiring
`src/onchain/proxy.ts`	Resolve Worker llama/gecko bases
`worker/src/onchain.ts`	Allowlisted proxy + health
`src/plugins/types.ts`	`DatasetPlugin` contract

Related docs

- Datasets — plugin contract
- On-Chain data (end user) — operator guide
- Worker — `/api/onchain/…` on the edge route table
- ADR-001, ADR-009, ADR-012

ADR-016: Provider-locked market data

Context

AXIS composes source × stream × engine, but **OHLCV aggregation is not composable**. Mixing Binance history with OKX live (or Kraken watchlist quotes on a Binance chart) silently poisons indicators, compare, watchlist, and cache. Prior to this ADR, source and stream were independently selectable, venue detection used plugin-id substrings, and the Binance synthetic fallback made network errors look like real data.

Key problems:

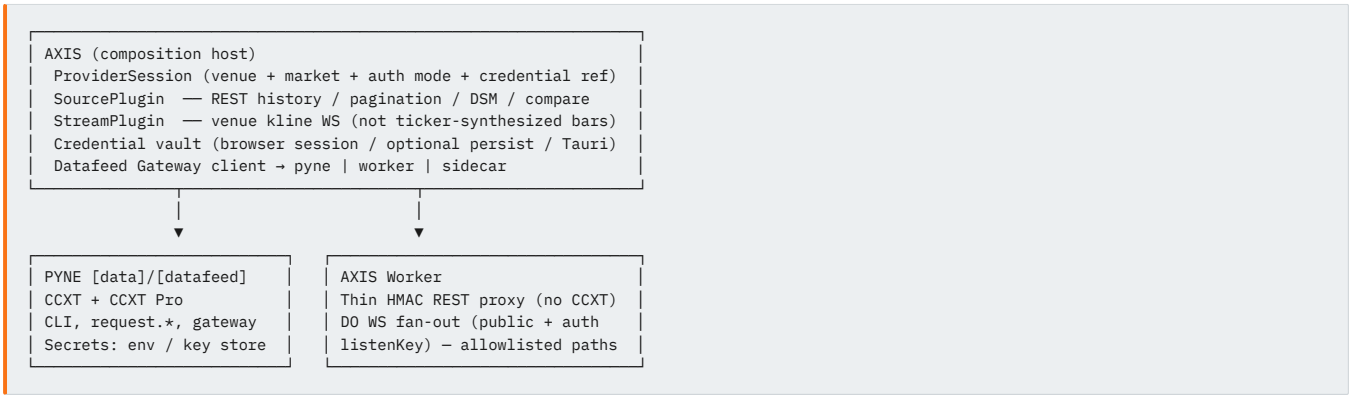
Issue	Risk
Source and stream independently selectable in Topbar	User loads Binance REST + OKX WS — mixed candles
DSM `sourceId` seeded once from `store.source`	Backfill targets a different venue than the chart
Watchlist Kraken → Binance tickers	Mixed-provider quotes
Coinbase WS ticker → synthetic bars	Not exchange candles; OHLCV ≠ REST history
Binance `fallback: true` synthesizes random walk	Chart looks "live" with fake prices

Decision

Introduce **`ProviderSession`** — a first-class locked market-data identity that all aggregators inherit:

1. **`ProviderSession`** (`src/data/provider.ts`): `{ id, sourceId, streamId, venue, market, authMode, credentialId, gateway }`. Persisted without secrets. Every aggregator reads `getActiveProvider()` instead of raw URLs or plugin-id substrings.
2. **Auto-pairing**: Source changes always re-pair the stream via `defaultStreamForSource`. Explicit mismatch is allowed but shows a HUD `⚠ pair · Fix` chip.
3. **Plugin capabilities**: `venue`, `market`, `klineStream` on `PluginCapabilities`. Venues with `klineStream: true` emit exchange candles (not ticker buckets).
4. **Coinbase WS**: switched from ticker buckets to Advanced Trade venue candles, folded into chart interval via `foldVenueCandle`.
5. **Binance fallback**: defaults **off** (`fallback: false`). Network errors must not look like real data.
6. **Watchlist lock**: no silent Binance fallback for Kraken, Gecko, or unknown venues. Each shows a clear "no live mux" message.
7. **Credential vault** (`src/data/credentials.ts`): in-memory session-only vault for API key/secret/passphrase. Secrets never reach `persist()`, `pluginsConfig`, error-share, or CHANGELOG dumps. `redactSecrets` and `stripSecretKeys` enforce this.
8. **Venue HMAC signers** (`src/data/venues/`): thin Binance / OKX / Bybit / Coinbase / Kraken / MEXC signers. No CCXT in the browser.
9. **Signed Binance klines**: vault key → HMAC direct fetch → Worker `/api/market/binance/signed/klines` (request-scoped `X-Exchange-Key` / `X-Exchange-Secret`). 401/403 stays thrown — bad keys must not silently mix public klines.
10. **Cache key**: includes `providerId|market|authMode|symbol|interval`. Authenticated and public candles do not share cache slots.

Ownership



Alternatives considered

Alternative	Rejected because
Bundle CCXT in browser	~1 MB+ bundle, Node APIs, isolate CPU, CORS issues
CCXT in Cloudflare Worker	Bundle size, 128 MB memory, shared rate-limit IP, not designed for 100-exchange routing
Keep venues as raw URL substrings	Silent venue mismatches poison aggregation correctness
Pluggable venue resolver	Over-engineered; first-party venues are a fixed set of 5

Consequences

- **Correctness** — aggregation (indicators, compare, watchlist, cache, DSM) inherits a locked venue. Mixed-provider bugs are structurally prevented.
- **Auth path** — public remains the default. Auth is a capability upgrade on the same provider, not a second source id.
- **Secret safety** — vault is RAM-only in v1. Tauri keychain / AES-GCM durable wrap is Phase 1.5. `pluginsConfig` and error-share are stripped of secret keys.
- **CCXT stays in PYNE** — server-side only. AXIS calls the gateway with session handles, never raw secrets in `/run`.

- **Long-tail exchanges** — AXIS `ccxt-rest` / `ccxt-ws` plugins that talk to a PYNE or Bun sidecar gateway. First-party venues stay thin native adapters.

Key files

Path	Role
`src/data/provider.ts`	`ProviderSession` type, `buildProviderSession`, `defaultStreamForSource`
`src/data/credentials.ts`	In-memory vault, `putCredential`, `redactSecrets`
`src/data/signed-fetch.ts`	Vault → HMAC → direct or Worker proxy
`src/data/venues/`	Per-venue HMAC signers (binance, okx, bybit, coinbase, kraken)
`src/plugins/active.ts`	`getActiveProvider()`
`src/plugins/types.ts`	`PluginCapabilities.venue`, `.market`, `.klineStream`
`src/store/index.ts`	`setActivePlugin` auto-pairing, `syncProviderSession`
`src/ui/ConnectionHud.tsx`	Venue label, pairing Fix button
`src/ui/SettingsDialog.tsx`	Data tab (exchange credentials)
`src/ui/error-share.ts`	`stripSecretKeys` redaction
`worker/src/market.ts`	Signed Binance klines proxy

Related docs

- Sources — source plugin catalog
- Streams — stream plugin catalog
- Data Source Manager — DSM guide
- Topologies — composition models
- ADR-001, ADR-009

ADR lifecycle

New ADRs should:

1. State context, decision, consequences.
2. Cross-link code paths.
3. Avoid re-litigating ADR-002 without supersession note.

See also

- Overview
- Topologies
- Plugins contracts
- Datasets

TOPOLOGIES

AXIS deploy and dev topologies: Vite+Flask, static PWA, Cloudflare Pages+Worker, offline Pyodide lab, and local Bun sidecar.

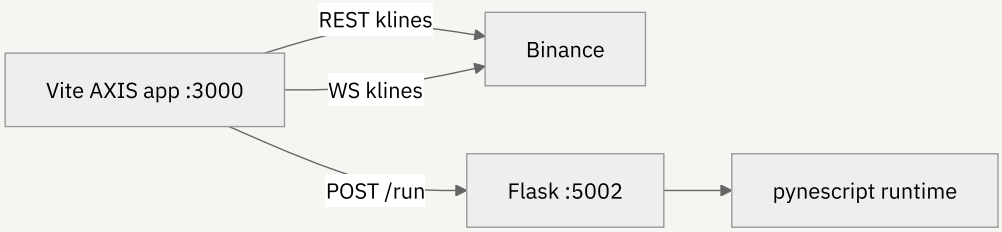
TOPOLOGIES

Concrete process and network shapes for AXIS. Choose a topology; then compose plugins inside it (recipes).

Abstract

Topology	AXIS	Calc	Live	Library
Dev desk	Vite `:3000`	Flask `:5002`	Binance WS direct	local
Static demo	`axis_pwa_server` `:8081`	Flask or remote	venue / mock	local
Edge	CF Pages	Worker → Flask	DO fan-out optional	cloud
Offline lab	any AXIS	Pyodide	mock-poll	local
Dev desk + sidecar	Vite `:3000`	Flask `:5002`	Bun sidecar `:5003`	local

Topology A — Dev desk

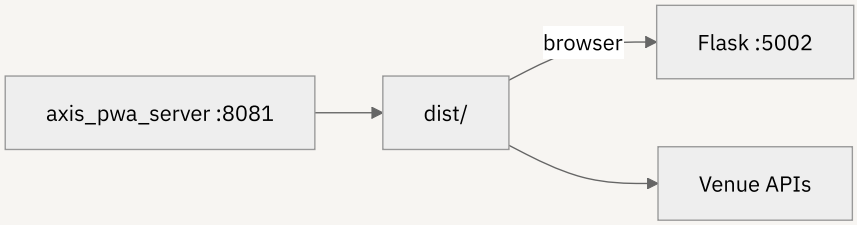


Bring-up

```
make run
cd frontend && bun install && bun run dev
```

Notes: Vite may proxy `/run`; Settings endpoint can still point absolute at `:5002`. CORS must allow Vite origin.

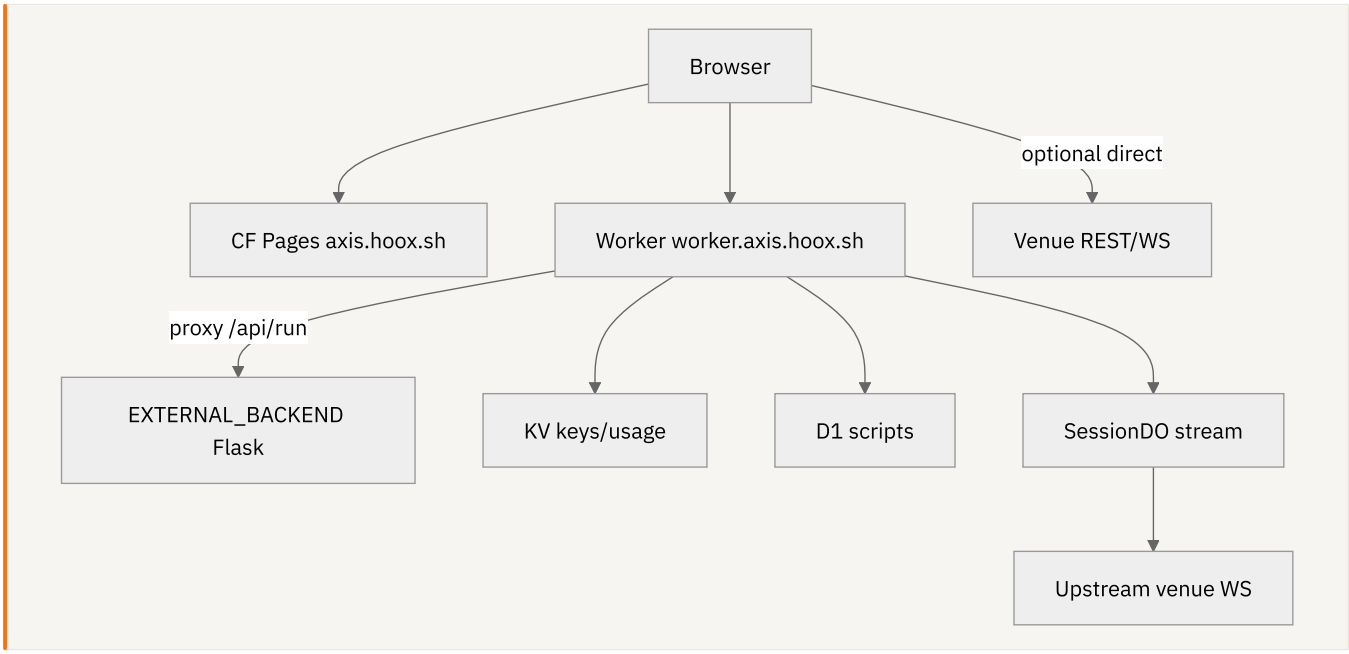
Topology B — Static PWA + Pro API




```
cd frontend && bun run build
python3 axis_pwa_server.py
make run   # separate process
```

VPS demo pattern: systemd units for `axis-pwa` and `pynescript-api`; `ALLOWED_ORIGINS` includes AXIS origin.

Topology C — Cloudflare AXIS at the edge



Deploy sketch

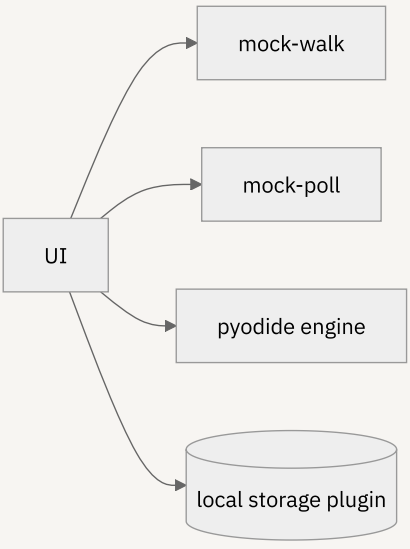
```
axis deploy
# or: cd worker && bun run deploy
cd frontend && bun run build
wrangler pages deploy dist --project-name=axis
```

Ids: Pages project `axis` at `https://axis.hoox.sh`; Worker script `worker-axis` at `https://worker.axis.hoox.sh` (ADR-008).

AXIS config:

- Engine `server`, endpoint = Worker origin
- Storage `cloud`, same origin + API key
- Stream: direct venue or DO-backed plugin

Topology D — Offline lab



No Flask, no venue. Requires vendored pyodide + wheels on origin once.

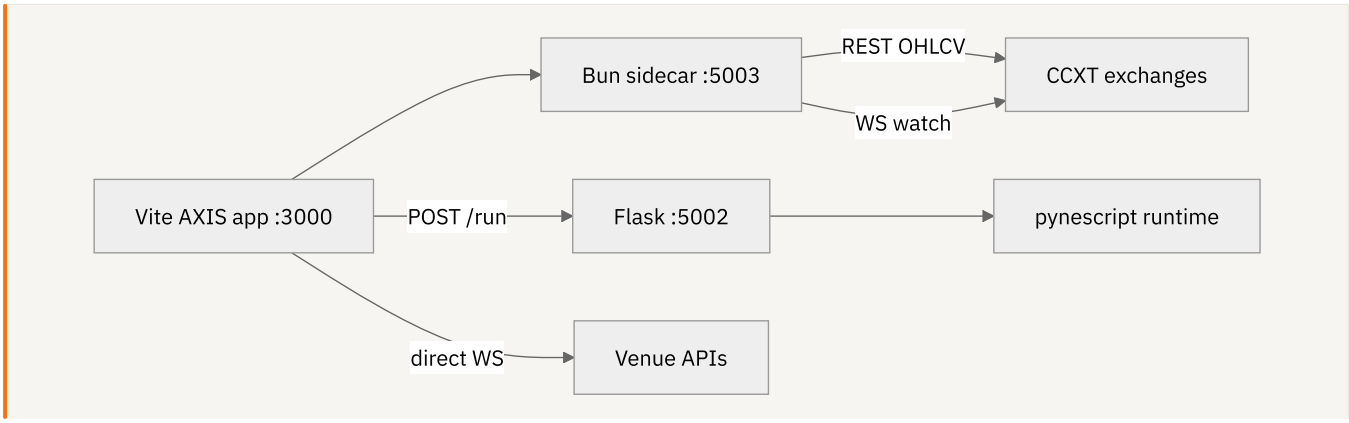
Topology E — Git-centric research

Any of A–C for calc/data; storage axis = `git`. Forges are orthogonal network peers:

```
AXIS --storage:git--> api.github.com | gitlab
AXIS --engine:server--> Flask/Worker
AXIS --source--> venue or CSV
```

Topology F — Dev desk + Bun sidecar (CCXT Pro)

For users who want authenticated WS feeds from any CCXT-supported exchange without running Flask. The sidecar (`packages/datafeed/`) mirrors the PYNE datafeed gateway contract.




```
bun run dev           # Vite :3000
bun run dev:datafeed  # Bun sidecar :5003
# in separate terminals, or combine in a Procfile
```

Gateway contract (same as PYNE datafeed):

Endpoint	Method	Description
`/datafeed/ohlcv?exchange=&symbol=&timeframe=`	GET	Historical OHLCV
`/datafeed/markets?exchange=`	GET	Available markets
`/datafeed/watch?exchange=&symbol=&timeframe=`	WS	Live bar stream
`/datafeed/session`	POST	Store credentials (RAM only)
`/health`	GET	Sidecar health

Notes: Credentials stay in RAM only (never persisted). The sidecar is optional — public venues work without it via direct browser WS.

Port map (defaults)

Service	Port
Vite dev	3000
Flask Pro API	5002
Bun datafeed sidecar	5003
Static PWA server	8081
Wrangler Worker	8787

Comparison

Concern	A Dev	B Static	C Edge	D Offline	F Sidecar
HMR	yes	no	no	n/a	yes
PWA SW realism	partial	full	full	full	partial
Multi-tenant keys	no	no	yes	no	local only
Ops complexity	low	medium	high	low	low
Fidelity	high (Flask)	high	high via proxy	engine-dependent	high (CCXT Pro)

Failure modes by topology

Topology	Typical break
A	Flask not running; CORS localhost vs 127.0.0.1 mismatch
B	Stale SW; missing `dist` pyodide assets
C	Unbound KV/D1; wrong project name; EXTERNAL_BACKEND down
D	First visit offline; SPA fallback for wheels

Internals

Path	Role
`vite.config.ts`	Dev/build
`axis_pwa_server.py`	Static host
`worker/wrangler.toml`	CF name + bindings
`packages/cli/`	AXIS CLI (setup / deploy / health)
`worker/README.md`	Edge ops

See also

- ADR-008, 009, 010, 012
- Installation
- Worker
- DevOps

STATE NAMESPACES

AXIS persistence keys, key migration, pluginsConfig, editor docs, library IDB, and URL hash state.

STATE NAMESPACES

AXIS persistence is multi-homed: **layout/config** in localStorage, **script library** in IndexedDB (or remotes), **ephemeral run data** in memory, optional **URL hash** for shareable slices.

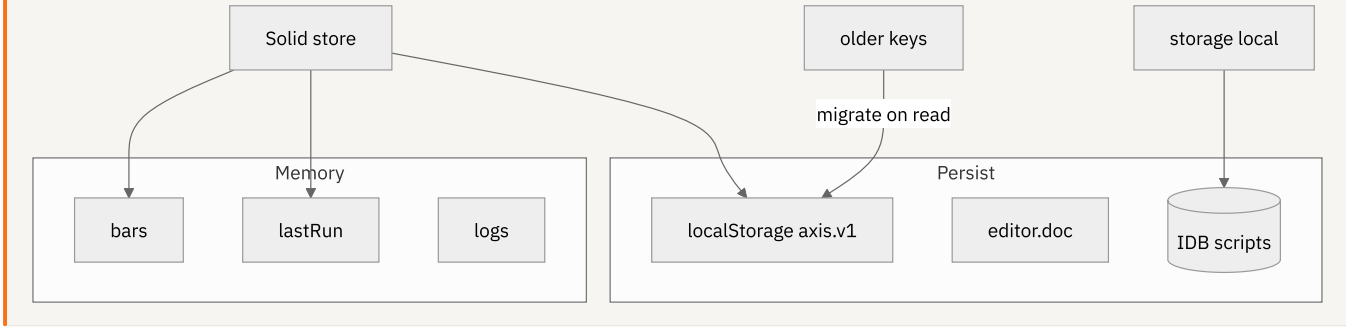
Abstract

Namespace	Medium	Purpose
`pynescript.axis.v1`	localStorage	App shell state
`pynescript.axis.editor.doc`	localStorage	Editor document backup
`pynescript.axis.storage`	IndexedDB	Local library
`pynescript.axis.library.v1`	localStorage fallback	Library if no IDB
`pynescript.axis.library.draft`	localStorage	Draft buffer
Plugin install list	localStorage	URL plugins restore
`store.bars` / `lastRun` / `logs`	memory	Session only

Contract / product identifiers (not storage keys):

```
pynescript.axis.plugins.v1   # conceptual plugin contract id
axis                         # CF Pages project id
axis.hoox.sh                 # Pages custom domain
worker-axis                  # CF Worker script id
worker.axis.hoox.sh          # Worker custom domain
```

Conceptual model



App state key `pynescript.axis.v1`

Solid store (`frontend/src/store/index.ts`):

Persisted (representative):

- Market: `symbol` , `interval` , `exchange` , `source` , `engine` , `endpoint`
- `activePlugins` `{ source, stream, engine, storage }`
- `pluginsConfig`
- Layout: `theme` , `editor` , `watchlist` , panel open/width/height
- `panes` , `scripts` (indicator list metadata)
- `live.streamId` (and related live flags carefully)
- `drawings` (user annotations)
- Flat mirrors: `source` / `engine` aligned via `setActivePlugin`

Explicitly omitted on persist:

- `bars`
- `lastRun`
- `logs`

Debounced write (~200ms).

Legacy migration

Read order:

1. `pynescript.axis.v1`
2. else `pynescript.axis.v2`

On older-key hit: copy forward into `pynescript.axis.v1`.

Legacy vanilla `state.js` uses the same STORAGE_KEY constant for the pre-Solid shell.

pluginsConfig

Map of configuration objects. Preferred keys:

> `` `${kind}:${id}` e.g. storage:cloud, storage:git, engine:server ``

Bare ids may still be read for compatibility (`cloud`, `git`). Values feed `configSchema` resolution inside plugins.

Sensitive fields: `apiKey`, git `token` — browser-local; treat profile as secret store.

Editor document

Key	Role
`pynescript.axis.editor.doc`	Draft / shared doc
Bridge messages	Popout synchronization

Popout mode uses `editor-bridge` + shared storage so detached windows share source without re-fetching library.

Local library IDB

Constant	Value
DB name	`pynescript.axis.storage`
Version	1
Stores	`scripts`, `kv`

Migration flags (`pynescript.axis.library.migrated`) prevent re-import loops from older library keys:

- `pynescript.axis.library.v1`
- `pynescript.axis.library.legacy`

Cloud / git (remote namespaces)

Not browser keys—server-side partitions:

Backend	Partitioning
cloud	Hash of API key on Worker; D1 table or memory
git	`owner/repo@branch` + `basePath`

Revisions: cloud may expose `revision` / If-Match; git uses blob SHAs via forge APIs.

URL hash state (legacy helper)

`frontend/src/state-hash.js` (legacy state path):

Feature	Detail
Keys	symbol, interval, engine, source, stream, timeRange
Script	base64 in hash if short
Cap	~2000 chars
API	`applyHashState`, `pushHashState`, `watchHashState`

Solid-first product may not wire this by default; treat as optional share mechanism. Prefer library export + recipe docs for durable shares.

Invariants

1. Migration is **read-repair** to AXIS keys.
2. Never persist full OHLCV in `axis.v1`.
3. `setActivePlugin` keeps flat fields coherent.
4. Logs panel open state forced closed on hydrate (noise control).
5. Drawing tool resets to `cursor` on hydrate; geometries restore.

Failure modes

Issue	Effect	Mitigation
QuotaExceeded	persist no-op	Export library; shrink drawings
Poisoned JSON	load defaults	Security tests; clear key
Split profiles	“missing” library	Same browser profile
Stale activePlugins id	missing plugin	Fall back / reselect built-in

Internals

Path	Role
`frontend/src/store/index.ts`	Solid persistence
`frontend/src/store/types.ts`	`AppState`
`frontend/src/state.js`	Legacy state
`frontend/src/state-hash.js`	Hash sync
`frontend/src/storage/local.ts`	IDB library
`frontend/src/plugins/loader.ts`	Installed plugin list

See also

- ADR-011
- AXIS store
- Script library

DRAWINGS AND PLOT PARITY

Plan to make user drawing tools and Pine plot/drawing outputs match PYNE and render with correct pane, color, geometry, and settings.

DRAWINGS AND PLOT PARITY

AXIS has two independent render paths:

Path	Source	Paint
User drawings	Toolbar tools	`DrawingLayer` user SVG group, editable
Pine output	PYNE `RunResult`	plots → LWC series; shapes/barcolor → candle markers; bgcolor → histogram; fill → SVG; line/label/box/polyline/linefill → script SVG; tables → HUD

This plan is the work list from a four-agent audit of AXIS (`src/chart` , `src/results` , `src/indicators/runner.ts`) against PYNE (`pynescript` `drawing.py` , `evaluator.py` , `host.py`).

Phases

P0 — AXIS correctness (this pass)

User tools and apply-pipeline bugs that do not wait on a PYNE export change.

1. **1-point body-move** — `shiftDrawing` only special-cased `text` / `priceLabel` . Shift any drawing with `p1` (and `points[]` when present): note, flag, crossline, anchoredText, arrowMarkUp/Down.
2. **p3 / extra vertex handles** — `hitTestHandle` / `resizeDrawing` only p1/p2. Add `p3` (and remaining `points[i]` as p3 for the third anchor). Paint extra handles after registry paint.
3. **Linefill GC** — `garbageCollectScriptDrawings` early-return ignored `skip.linefill` .
4. **Polyline / linefill overlay** — copy `force_overlay` and polyline `fill_color` → `bgcolor` .
5. **Stale SVG** — silent live re-run with `drawings: []` must clear script drawings.
6. **Pane routing** — `bgcolor` and `fill()` use the script pane when `overlay=false` (`syncBgcolorBands(paneId)` , `ensureScriptPaneLayer` for fills). `plotshape` / `plotchar` / `plotarrow` attach to a series on that pane when possible; otherwise stay on price (LWC markers need a host series).

P1 — User-tool settings (AXIS)

Done:

- Shared style bar: color (presets + custom), width, dash, fill for shapes / ranges / fib / highlighter / positions.
- Per-tool extras via `src/chart/drawings/tool-settings.ts` + settings popover (`axis-drawing-settings`).
- Last-used style per kind (`drawingPrefs.byKind`).
- `extendLeft` / `extendRight` UI + paint (trend, ray, extend, info line, channel, fib).
- Fib level editor, reverse, show % / price, fill between levels.
- Text re-edit (popover + double-click), font size.
- Per-drawing lock in the style bar (unlock bypasses `updateSelected` lock gate).
- Long/short risk-reward (`meta.rr`).
- Highlighter width honors `style.width` (create default 8).
- hray: 1-click (price only).
- xabcd / headShoulders: commit on 5th point.

P2 — PYNE export (pynescript repo)

AXIS cannot paint fields the engine never serializes.

Gap	Where	Needed on the wire
Table cell keys `(row,col)` vs `(col,row)`	`drawing.py` `table.cell` vs `cell_set_*`	One index order
Table style	`export_for_api`	halign, valign, width, tooltip, text_size, borders
`table.cell` kwargs	ctor	width, halign, valign, size
plotshape Y / `location.absolute`	`_coerce_plot_shape`	keep numeric series (or `{y, on}` cells)
plotbar / plotcandle	evaluator	open/high/low/close (or sibling titles in `plot_meta`)
`fill()` compile / untitled plots	host + evaluator	`plot1` / `plot2` series keys + per-bar fill color column
Positional `plot(..., linewidth)`	`evaluator.py` ~425	Pine order: series, title, color, linewidth, style
plotarrow	evaluator	`colorup` / `colordown` / minheight / maxheight
`offset` , `force_overlay` , `histbase`	`_pack_interpret_plot_columns`	remaining plot_meta fields (display is packed)
Polyline	export	`fill_color` , `curved` , `force_overlay`
Box / label extras	export	closed, extend, text_*, tooltip, border

P3 — Remaining Pine paint

Done this pass:

- Hide-drawings toggle includes script SVG + fill bands (all pane layers).
- Label `yloc` pad (8px off wick); `label.style_text_outline` as outlined text.
- First `addIndicator` re-owns plot fills / script drawings / plotshape (`EDITOR_RUN_KEY` → script id).
- Histogram `histbase` from `plot_meta` (default 0). PYNE still needs to serialize it for live values.
- Pine `display.*` (`none` / `pane` / `data_window` / `price_scale` / `status_line` / `all`) on plot, hline, shapes, fill, bgcolor, barcolor.

Still open:

- `location.top` / `bottom` = pane edges, not bar (LWC markers only have above/below/in bar).
- Per-bar `plot(..., color=expr)` , plotarrow length (`colorup` / `colordown` / `minheight` / `maxheight` wait on PYNE).

Invariants

- User drawings ≠ Pine drawings. Do not share hit-testing or magnet with script SVG (`pointer-events: none`).
- `overlay=false` plots, fills, bgcolor, and shapes belong on `ind_*` unless `force_overlay` .
- `barcolor()` stays on price candles.
- Color sanitize: `#rgb(a)` , `#rrggbb(aa)` , `rgb/rgba` . Unknown → VOID indigo for **user** tools; Pine should keep engine hex (including `#RRGGBBAA`).
- GC caps match Pine: 50 default, 500 lines/labels/boxes, 100 polylines. Linefill cap is AXIS-only until PYNE emits `max_lines_fills_count` .

Tests

- `shiftDrawing` / `resizeDrawing` for 1-pt and 3-pt kinds.
- Linefill GC overflow.
- Polyline `force_overlay` + `fill_color` .
- Runner: bgcolor/fill target pane when `overlay=false` (unit with a fake manager if needed).
- PYNE: table cell key, fill meta, plotshape Y (in the pynescript repo).

PLUGINS

AXIS unified plugin system: source, stream, engine, storage, dataset — contracts, registry, catalogs, and dynamic ES-module loading.

PLUGINS

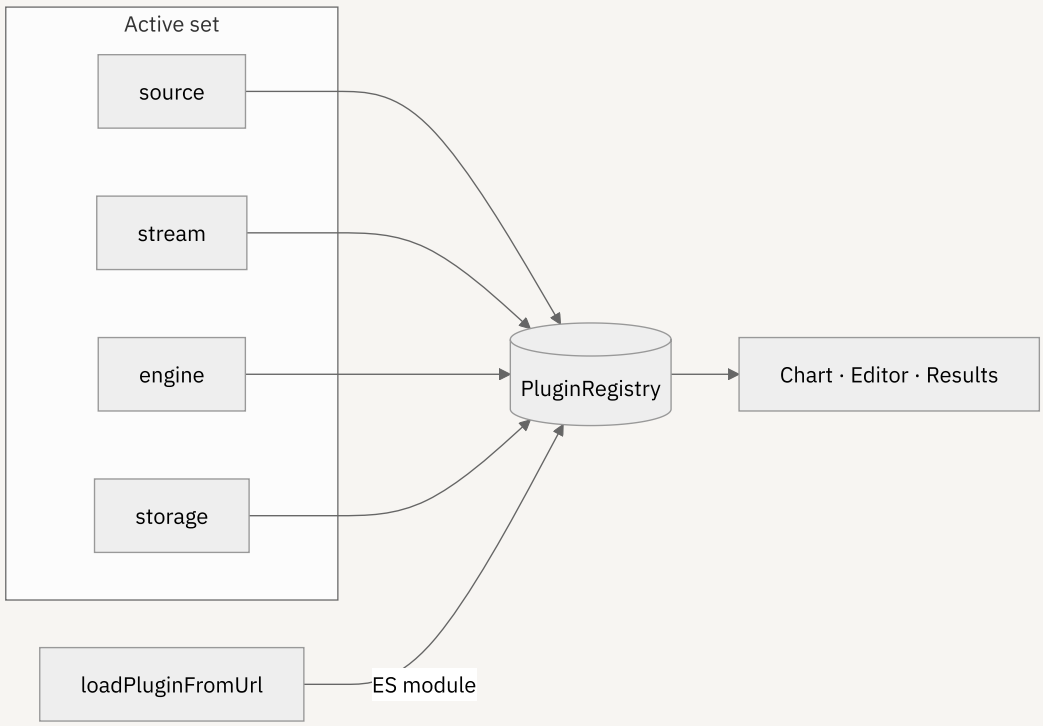
Abstract

AXIS is a **plugin-composed charting PWA**. Historical bars, live ticks, Pine evaluation, the script library, and **on-chain datasets** are not hard-coded product features — they are interchangeable modules that share one contract namespace:

> `pynescript.axis.plugins.v1`

The shell (AXIS) never embeds a closed interpreter. Evaluation is always an **engine** plugin. Market candles arrive through **source** and **stream**. Alternate series (protocol TVL, …) use **dataset**. Persistence goes through **storage**. Compose any lawful active set without a proprietary charting host.

Conceptual model



Kind	Role	Required surface
`source`	Historical OHLCV	`fetchHistorical(opts) → Bar[]`
`stream`	Live bar updates	`start(opts) → stop()`
`engine`	Pine / DSL evaluation	`isReady()` , `run(opts) → RunResult`
`storage`	Script library	`list` / `read` / `write` / `remove`
`dataset`	On-chain / alternate series	`fetchDataset(opts) → OnchainDataset` — Datasets
`component`	UI slots (phase 2 + URL agents)	`mount(slot, el, api)` — e.g. PYNE Agent

Built-ins register at bootstrap (`frontend/src/plugins/bootstrap.ts`). Dynamic plugins install from URL (`loader.ts`) and persist under `localStorage` key `pynescript.axis.plugins.v1` .

Interface surface

Module	Path
Contracts	`frontend/src/plugins/types.ts`
Registry	`frontend/src/plugins/registry.ts`
Bootstrap	`frontend/src/plugins/bootstrap.ts`
Active resolution	`frontend/src/plugins/active.ts`
Dynamic loader	`frontend/src/plugins/loader.ts`
Public barrel	`frontend/src/plugins/index.ts`
Source catalog	`frontend/src/sources/catalog.ts`
Stream catalog	`frontend/src/streams/catalog.ts`
Engine catalog	`frontend/src/engines/catalog.ts`
Storage catalog	`frontend/src/storage/catalog.ts`

Built-in inventory (quick)

Kind	Built-in IDs
Sources	`binance-rest`, `okx-rest`, `bybit-rest`, `coinbase-rest`, `mock-walk`, `csv-upload`
Streams	`binance-ws`, `okx-ws`, `bybit-ws`, `coinbase-ws`, `kraken-ws`, `mock-poll`
Engines	`server`, `pyodide`
Storage	`local`, `cloud`, `git`
Components	`hyperparameter-optimisation` (Results → Optimise)

Defaults when selection is missing (`active.ts`): source `binance-rest`, stream `binance-ws`, engine `server`, storage `local`.

Invariants

1. **AXIS ≠ engine** — the UI never ships a private Pine runtime; engines are plugins.
2. **Registry is the single source of truth** — catalogs write into `PluginRegistry`; UI lists come from `listSources()` / etc.
3. **Built-ins resist casual unregister** — `unregister*(id)` returns `false` for `builtIn: true` unless `allowBuiltIn: true`.
4. **Config keys** — prefer `pluginKey(kind, id)` → `` `${'${'}kind{'}'}:${'${'}id{'}'}` `` (e.g. `engine:server`).
5. **Storage via URL is rejected** — dynamic loader refuses `kind: 'storage'` until a hardened path exists.
6. **Install list persistence** — installed plugins are stored under `pynescript.axis.plugins.v1`.

Compose recipes (plugin view)

Recipe	Source	Stream	Engine	Storage
Offline lab	`mock-walk`	`mock-poll`	`pyodide`	`local`
Desk research	`binance-rest`	`binance-ws`	`server` → Flask	`local`
AXIS at the edge	venue REST	venue WS / DO	`server` → Worker	`cloud`
Repo library	any	any	any	`git`

Failure modes

Symptom	Likely cause
Empty source dropdown	`ensureBuiltins()` never ran at app entry
Dynamic plugin vanishes after reload	URL 404 or CORS; check System Logs (`plugins` channel)
Engine “not ready”	Flask/Worker down (`server`) or missing `/vendor/*.whl` / `/pyodide/` (`pyodide`)
Cloud library 401	Missing Bearer `pn_…` key or unbound `API_KEYS` KV in production

PYNE Agent (sister worker)

Natural-language script authoring is **not** an engine — it is an optional **component** plugin backed by pyne-agent-worker (Cloudflare® Workers AI™). Install from:

```
https://<pyne-agent-worker>/plugin/axis-pine-agent.js
```

Works **standalone** (no pyne-worker). Full guide: PYNE Agent plugin · End-user.

See also

- Contracts
- Registry
- Sources · Streams · Engines · Storage
- Dynamic loader
- Plugin examples
- Worker (cloud storage + stream relay)

PLUGIN CONTRACTS

Formal TypeScript interfaces for AXIS source, stream, engine, storage, and component plugins (pynescript.axis.plugins.v1).

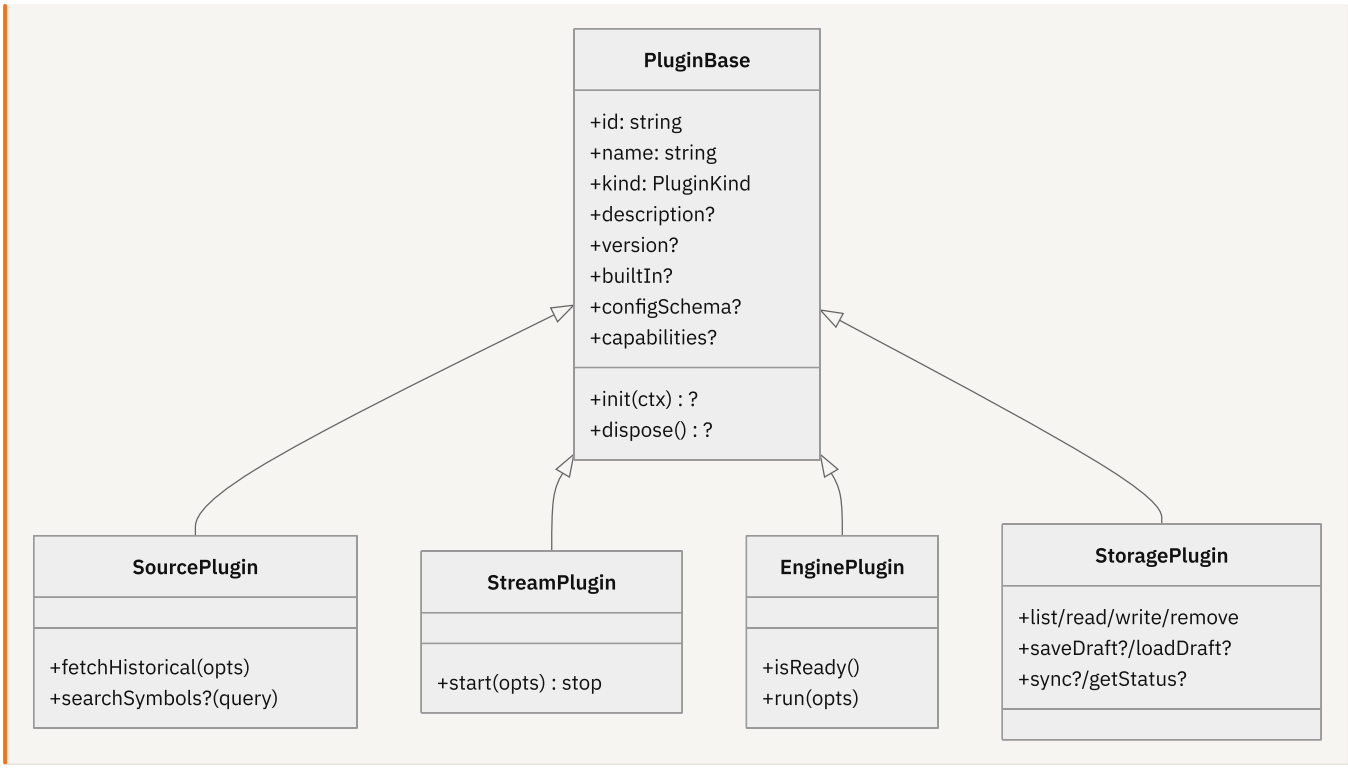
PLUGIN CONTRACTS

Abstract

All AXIS plugins share `PluginBase` and a discriminant `kind`. Contracts live in `frontend/src/plugins/types.ts`. Catalogs and the dynamic loader assert these shapes before registration; the UI never calls plugins that failed assertion.

Namespace (localStorage / config): **`pynescript.axis.plugins.v1`**.

Conceptual model



Interface surface

PluginKind

```
type PluginKind = 'source' | 'stream' | 'engine' | 'storage' | 'component';
```

PluginBase

Field	Type	Notes
`id`	`string`	Stable registry key within kind
`name`	`string`	UI label
`kind`	`PluginKind`	Discriminant
`description?`	`string`	Manager catalog
`version?`	`string`	Optional semver
`builtIn?`	`boolean`	Protects against unregister
`configSchema?`	`ConfigSchema`	Settings form
`capabilities?`	`PluginCapabilities`	Badges: offline / needsAuth / needsNetwork / needsProxy
`init?` / `dispose?`	lifecycle	Optional host hooks

ConfigSchema / FieldSchema

Each config field:

Field	Values
`type`	`'string' | 'number' | 'boolean' | 'select'`
`default?`	primitive
`label?`, `description?`, `placeholder?`	UI
`min?`, `max?`, `step?`	numbers
`options?`	select strings

PluginContext

Passed to `init` when used:

- `getConfig(): Record<string, unknown>`
- `setStatus(msg, level?)`
- `host.fetch?` — injectable fetch for tests / workers

pluginKey

```
pluginKey(kind, id) → `${kind}:${id}`
```

Used for `store.pluginsConfig` lookup (`active.ts`).

SourcePlugin

```
interface SourceOpts {
  symbol: string;
  interval: string;
  limit?: number;
  config?: Record<string, unknown>;
}

interface SourcePlugin extends PluginBase {
  kind: 'source';
  fetchHistorical(opts: SourceOpts): Promise<Bar[]>;
  searchSymbols?(query: string, config?: Record<string, unknown>): Promise<string[]>;
}
```

Bar shape (from `store/types`): `{ time, open, high, low, close, volume? }` with `time` in **unix seconds**.

Registry asserts: object, `id`, `name`, `kind === 'source'`, `fetchHistorical` is a function.

StreamPlugin

```
interface StreamOpts {
  symbol: string;
  interval: string;
  config?: Record<string, unknown>;
  lastBar?: Bar | null;
  onBar: (b: Bar) => void;
  onError: (e: Error) => void;
  onStatus: (s: {
    state: 'open' | 'closed' | 'reconnecting' | string;
    url?: string;
    detail?: string;
  }) => void;
}

interface StreamPlugin extends PluginBase {
  kind: 'stream';
  start(opts: StreamOpts): () => void; // stop
}
```

Invariant: `start` must return a **stop** function that closes sockets / clears timers. The AXIS calls stop on symbol change or unmount.

EnginePlugin

```
interface EngineOpts {
  script: string;
  bars: Bar[];
  config?: Record<string, unknown>;
  signal?: AbortSignal;
}

interface RunResult {
  status: 'success' | 'error';
  plots: (number | null)[];
  series?: Record<string, (number | null)[]>;
  events: Array<{ time: number; type: string; id?: string; price?: number; dir?: string; [k: string]: unknown }>;
  drawings?: Array<Record<string, unknown>>;
  error?: string;
  meta?: { mode?; script_id?; run_id?; ms?; overlay?; script_name?; [k: string]: unknown };
}

interface EnginePlugin extends PluginBase {
  kind: 'engine';
  isReady(): Promise<boolean>;
  run(opts: EngineOpts): Promise<RunResult>;
}
```

Notes:

- `plots` is the primary overlay series; multi-series engines also fill `series`.
- Strategy fills / orders appear in `events`.
- Pine line/label/box objects may appear in `drawings` from the interpret runtime.
- Prefer returning `status: 'error'` with `error` string over throwing for UI-friendly paths (built-in engines do both patterns carefully).

StoragePlugin

```
interface ScriptMeta {
  id: string;
  name: string;
  description?: string;
  path?: string;
  updatedAt: number;
  createdAt?: number;
  revision?: string;
  tags?: string[];
}

interface ScriptDocument extends ScriptMeta {
  content: string;
}

interface StoragePlugin extends PluginBase {
  kind: 'storage';
  list(opts?): Promise<ScriptMeta[]>;
  read(id, config?): Promise<ScriptDocument>;
  write(doc, config?): Promise<ScriptMeta>;
  remove(id, config?): Promise<void>;
  saveDraft?(doc, config?): Promise<void>;
  loadDraft?(config?): Promise<{ content: string; name?: string } | null>;
  sync?(direction: 'push' | 'pull' | 'both', config?): Promise<SyncResult>;
  getStatus?(config?): Promise<StorageStatus>;
}
```

Registry requires `list`, `read`, `write`, `remove`. Draft/sync/status are optional.

ComponentPlugin (reserved)

```
interface ComponentPlugin extends PluginBase {
  kind: 'component';
  slots: Array<'manager-tab' | 'results-tab' | 'topbar-action' | 'settings-section'>;
  mount(slot: string, el: HTMLElement, api: Record<string, unknown>): () => void;
}
```

Registered in the registry but not yet driven by the Manager UX. Do not rely on slots in production plugins.

Capabilities

```
interface PluginCapabilities {
  offline?: boolean;
  needsAuth?: boolean;
  needsNetwork?: boolean;
  needsProxy?: boolean;
}
```

Manager badges (`plugin-badges.tsx`) surface these for composition decisions (e.g. offline lab vs desk research).

Invariants & edge cases

1. **Export shape for dynamic modules** — default export, named `plugin` , or the module root object (`loader.asPlugin`).
2. **Config merge** — catalogs merge `configSchema` defaults with runtime `config` / `store.pluginsConfig` .
3. **AbortSignal** — `server` engine honors `signal` / adaptive timeouts; custom engines should too for long runs.
4. **Revision / If-Match** — cloud storage uses revision strings for optimistic concurrency (Worker `If-Match`).

Failure modes

Error	Contract violation
`source: fetchHistorical() required`	Missing method on register
`Unknown plugin kind`	Typo or future kind not supported by loader
`Custom storage plugins via URL are not supported yet`	Dynamic storage blocked by design

See also

- Registry
- Dynamic loader
- Plugin examples

PLUGIN REGISTRY

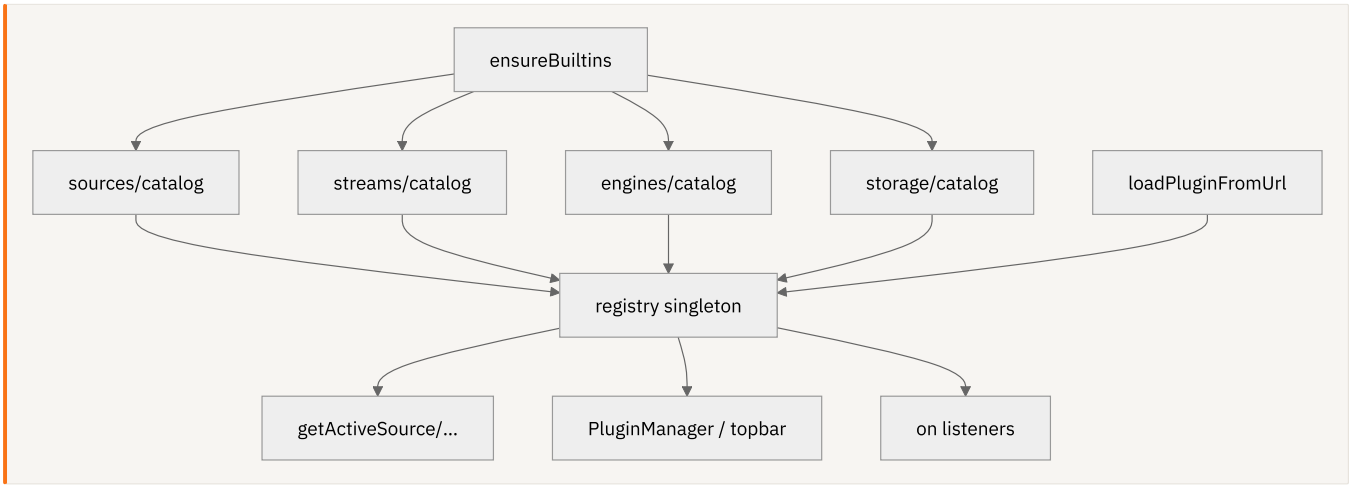
PluginRegistry singleton: ordered maps per kind, listeners, built-in protection, and bootstrap wiring.

PLUGIN REGISTRY

Abstract

`PluginRegistry` (`frontend/src/plugins/registry.ts`) is the **single in-memory source of truth** for every installed plugin of every kind. Catalogs, the dynamic loader, active resolution, and Manager UI all read/write through this class (or thin wrappers that call it).

Conceptual model



Interface surface

Singleton

```
// or from './plugins' barrel
```

Per-kind API (pattern)

Method	Behavior
`registerSource(p)`	Assert contract; set map; append order if new; emit `registered`
`getSource(id)`	Map lookup
`listSources()`	Registration **order**, not insertion-hash order
`unregisterSource(id, { allowBuiltIn? })`	Refuses `builtIn` unless opted in
Same pattern for stream / engine / storage	
`registerComponent` / `listComponents`	Phase-2 components (unordered Map values)

Bulk

Method	Behavior
`register(plugin: AnyPlugin)`	Dispatch on `kind`
`unregister(kind, id, opts?)`	Kind switch
`clear()`	Wipe all maps and order arrays (tests)
`summary()`	`{ sources, streams, engines, storages }` lightweight summaries
`on(listener)`	Subscribe to `{ type, kind, id }`; returns unsubscribe

Order preservation

Private arrays `_sourceOrder` , `_streamOrder` , `_engineOrder` , `_storageOrder` ensure UI dropdowns stay stable when plugins re-register (idempotent `set` without reordering).

Built-in protection

```
if (p.builtIn && !opts?.allowBuiltIn) return false;
```

Dynamic uninstall paths must never silently delete `binance-rest` / `server` / `local`.

Internals (repo paths)

Concern	Path
Class + singleton	`frontend/src/plugins/registry.ts`
Idempotent builtins	`frontend/src/plugins/bootstrap.ts` → `ensureBuiltins()`
Active set	`frontend/src/plugins/active.ts`
Dynamic register helpers	`sources/catalog.ts` `registerDynamicSource`, etc.

Bootstrap

```
// bootstrap.ts — once per page lifetime
ensureSourcesRegistered();
ensureStreamsRegistered();
ensureEnginesRegistered();
ensureStoragesRegistered();
```

Safe to call repeatedly: catalogs use a local `registered` flag; bootstrap uses `done`.

Active resolution defaults

Slot	Default id
source	`binance-rest`
stream	`binance-ws`
engine	`server`
storage	`local`

Fallbacks chain: `store.activePlugins.*` → legacy store fields → default → registry get with hard fallback id → throw if still missing (source/stream/engine). Storage returns `undefined` if nothing registered (should not happen after builtins).

Engine endpoint surface

`getActiveEngineConfig()` injects `store.endpoint` when the engine has an `endpoint` config field or id is `server` — keeps the topbar Endpoint field and engine config aligned.

Invariants & edge cases

1. **Re-register same id** — overwrites the plugin object; does **not** duplicate order entry.
2. **Listener errors are swallowed** — one bad subscriber cannot break registration.
3. **`clear()` does not reset bootstrap/catalog flags** — tests use `_resetBootstrapFlag` / `_reset*RegistrationFlag` helpers.
4. **Components** — no order array; `listComponents` is Map iteration order.

Worked examples

Register a test source

```
registry.registerSource({
  id: 'fixture',
  name: 'Fixture',
  kind: 'source',
  async fetchHistorical() {
    return [{ time: 1, open: 1, high: 1, low: 1, close: 1 }];
  },
});
```

Listen for dynamic installs

```
const off = registry.on((ev) => {
  if (ev.type === 'registered' && ev.kind === 'engine') {
    console.log('engine ready', ev.id);
  }
});
// later: off()
```

Failure modes

Symptom	Cause
`source: kind must be 'source'`	Wrong discriminant on register
Dropdown empty after `clear()` in test	Forgot re-`ensureBuiltins` / reset flags
Unregister returns false	Built-in protection

See also

- Contracts
- Dynamic loader
- Plugins hub

SOURCES

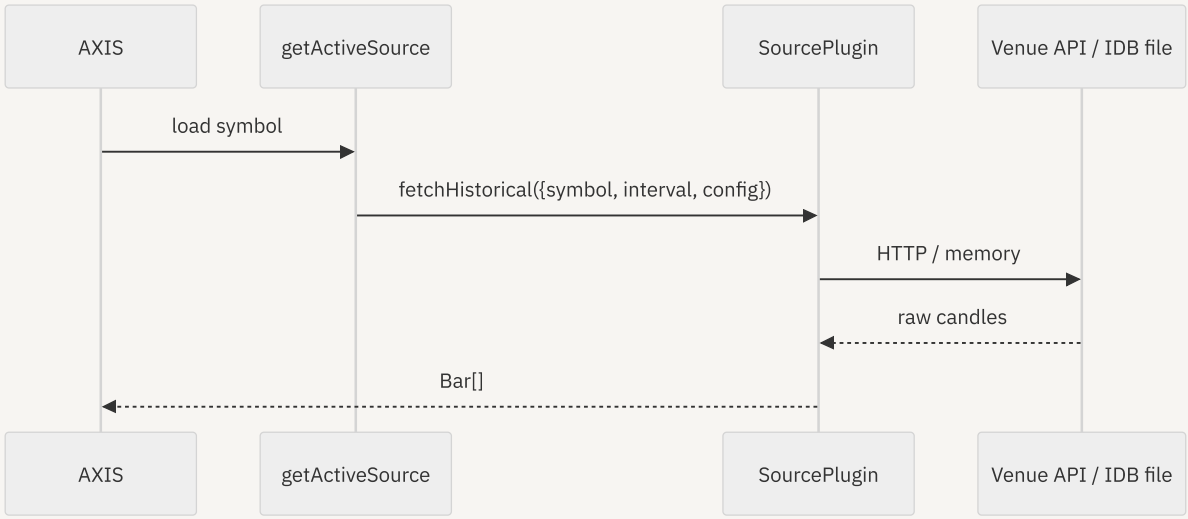
Built-in historical data sources: venue REST, mock walk, CSV upload — contracts, config, and fallbacks.

SOURCES

Abstract

A **source** loads a finite window of OHLCV bars for the chart and engine. Sources live in `src/sources/catalog.ts` and register into the unified registry via `ensureSourcesRegistered()`.

Conceptual model



Built-in catalog

id	Name	Network	Notes
`binance-rest`	Binance REST	yes	Public klines; **synthetic walk fallback** only when `fallback: true` (default **off**)
`okx-rest`	OKX REST	yes	Candles; symbol `BTCUSDT` → `BTC-USDT`; max 300 bars
`bybit-rest`	Bybit REST	yes	Spot v5 klines; newest-first reversed
`coinbase-rest`	Coinbase REST	yes	Exchange candles; `USDT` pair rewritten to `USD` product
`kraken-rest`	Kraken REST	yes	Public OHLC; pairs with `kraken-ws`
`mexc-rest`	MEXC REST	yes	Public spot klines (`api.mexc.com`); `1h` → `60m`; pairs with `mexc-ws`
`mock-walk`	Mock Walk	no	Offline random walk; optional Mulberry-like seed
`csv-upload`	CSV / JSON Upload	no	Reads last upload from `upload-store`

UI order is `BUILTIN_SOURCES` array order (Binance first).

Interface surface

Every source implements `SourcePlugin` (contracts):

```
fetchHistorical({
  symbol,
  interval,
  limit?,
  startTime?,   // unix seconds (optional)
  endTime?,     // unix seconds (optional) — walk-back pagination
  signal?,      // AbortSignal for background jobs
  config?,
}): Promise<Bar[]>
```

One-shot Load vs Data Source Manager

Path	Behavior
`src/data/load-symbol.ts`	Single `fetchHistorical` (limit from Settings `historyBars`) → chart
Data Source Manager	Multi-page walk-back with **`endTime` only** per page, then **validate + gap-fill**, durable IDB cache

Do not pass `startTime` + `endTime` together when paginating on Binance-style venues: they return the *first* N bars from `startTime`, which can falsely complete a multi-year job in one page.

Config highlights

binance-rest

Key	Default	Meaning
`baseUrl`	`https://api.binance.com`	Override for mirrors/proxies
`limit`	`500`	50–1000
`fallback`	`false`	On network error, `synthesizeWalk` (demo only)

When Settings → **Data** has a Binance key in the session vault, Load uses signed REST (higher weight) and falls back to the Worker `/api/market/binance/signed/klines` on CORS failure. Keys are never written to disk.

mock-walk

Key	Default	Meaning
`seed`	`0`	`0` = non-deterministic; non-zero = deterministic PRNG
`startPrice`	`100`	Walk origin
`limit`	`500`	Bar count

csv-upload

- No schema fields. User must Upload a CSV (`time,open,high,low,close[,volume]`) or JSON array first; otherwise throws a clear error.

Internals

Path	Role
`src/sources/catalog.ts`	Definitions + register/list/get + `sourcePageLimit`
`src/sources/upload-store.ts`	In-memory last upload for `csv-upload`
`src/data/load-symbol.ts`	One-shot Load pipeline
`src/data/data-source-manager.ts`	Background backfill + validate + gaps
`src/data/bars-cache.ts`	IDB OHLCV cache for manager
`src/data/bars-gaps.ts`	Coverage / gap detection
`src/data/parse-bars.ts`	CSV/JSON normalization

Interval → venue codes

Helpers map AXIS intervals (`1m`, `3m`, `5m`, `15m`, `30m`, `1h`, `2h`, `4h`, `6h`, `8h`, `12h`, `1d`, `3d`, `1w`, `1M`) to OKX `bar`, Bybit interval codes, Coinbase granularity seconds. Unmapped intervals fall back to daily-ish defaults — check venue docs if you add exotic TFs. Some venues omit a subset (MEXC has no `3m` / `2h` / `6h` / `8h` / `12h` / `3d`; Kraken has no `3m` / `2h`).

Dynamic registration

```
registerDynamicSource(plugin)  // from catalog / loader
unregisterDynamicSource(id)
listDynamicSourceIds()
```

Loader path: `kind === 'source'` + `fetchHistorical` required.

Invariants & edge cases

1. **Time unit** — always **seconds** unix (Binance ms ÷ 1000).
2. **Binance fallback** — offline demos “work” but are **not** real prices; disable `fallback` for strict desk research.
3. **CORS** — browser → public venue APIs must allow CORS; corporate proxies may require a Worker `/api/proxy` (not shipped as a general proxy today — plan carefully).
4. **Newest-first venues** (OKX, Bybit, Coinbase) — catalog reverses/sorts ascending by `time` for the chart.

Worked example

```
// Minimal custom source (ES module for loader)

  id: 'static-two',
  name: 'Two Bars',
  kind: 'source',
  async fetchHistorical() {
    return [
      { time: 1_700_000_000, open: 1, high: 2, low: 0.5, close: 1.5, volume: 10 },
      { time: 1_700_000_060, open: 1.5, high: 2, low: 1, close: 1.2, volume: 12 },
    ];
  },
};
```

Failure modes

Error / symptom	Fix
`No uploaded file…`	Use Upload before selecting `csv-upload`
Empty chart + Binance down	Expected synthetic walk if fallback on
OKX `code !== '0'`	Symbol format / region block
CORS blocked	Proxy or different source

See also

- Streams
- Contracts
- Plugin examples (CoinGecko)

DATASETS

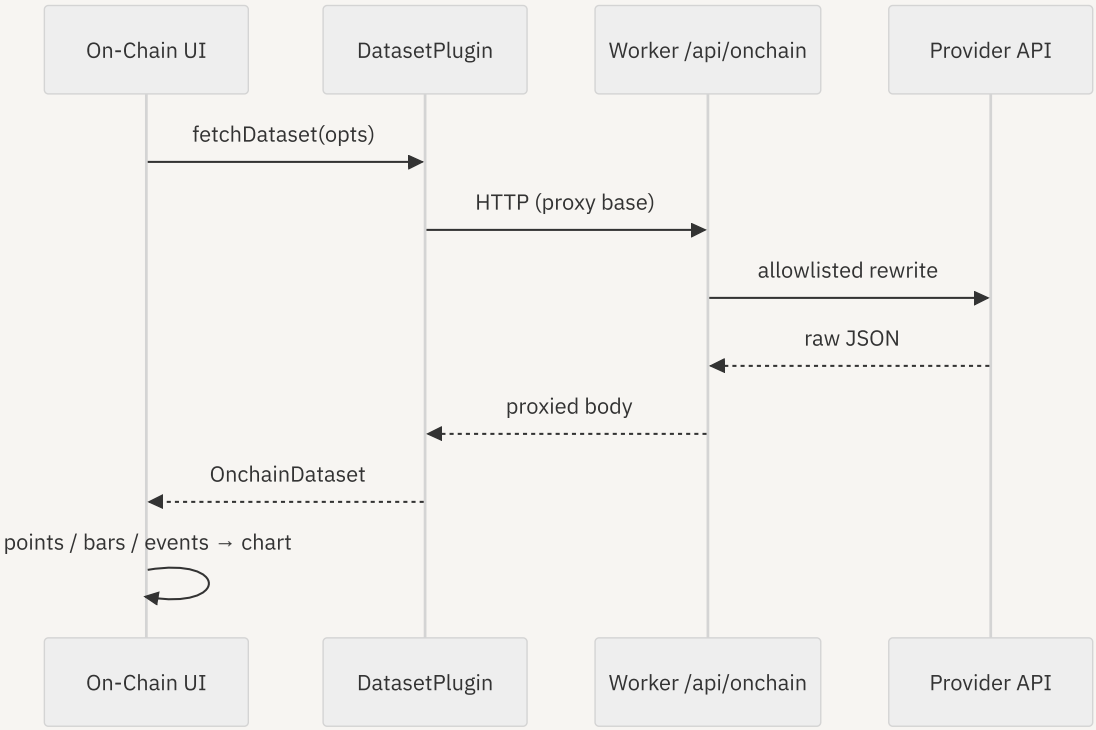
On-chain / alternate series dataset plugins — fetchDataset contract, DefiLlama TVL, GeckoTerminal DEX OHLCV.

DATASETS

Abstract

A **dataset** plugin loads non-OHLCV (or alternate OHLCV) series for the chart host (protocol TVL, DEX pool candles, funding, events). Kind is `dataset` on the unified registry. Phase 1 surfaces DefiLlama protocol TVL; Phase 2 adds GeckoTerminal pool OHLCV and derived TVL events. UI: On-Chain panel. Contract: `src/plugins/types.ts` (`DatasetPlugin`) and `src/onchain/types.ts` .

Conceptual model



Interface surface

```
interface DatasetPlugin {
  id: string;
  name: string;
  kind: 'dataset';
  fetchDataset(opts: DatasetFetchOpts): Promise<OnchainDataset>;
  searchInstruments?(query: string, config?: Record<string, unknown>): Promise<OnchainInstrument[]>;
  capabilities?: { needsNetwork?: boolean; needsProxy?: boolean; /* … */ };
}
```

`DatasetFetchOpts`

Field	Notes
`instrument`	`chainId` , optional `protocolId` / `address` , `metric` , optional `facet` , `symbol` label
`resolution`	e.g. `1d` , `1h`
`startTime` / `endTime`	unix seconds (optional window)
`limit`	optional max points
`signal`	`AbortSignal`
`config`	plugin config bag (`baseUrl` , …)

`OnchainDataset`

Field	Notes
`kind`	`ohlcv` | `scalar_series` | `multi_series` | `events` | `table`
`points` / `series` / `bars` / `events`	payload shape depends on kind
`provenance`	`{ provider, queryId?, url? }` — required for UI honesty
`finality`	`pending` | `safe` | `finalized` | `unknown`
`synthetic`	true when OHLC was synthesized from marks

Providers

id	Name	Network	Notes
`defillama-tvl`	DefiLlama TVL	yes	Protocol TVL history; browser uses Worker llama proxy by default
`geckoterminal`	GeckoTerminal	yes	DEX pool OHLCV + pool search; browser uses Worker gecko proxy by default

Helpers (not only plugins):

Module	Role
`src/onchain/defillama.ts`	Protocol TVL fetch + history parse
`src/onchain/geckoterminal.ts`	Pool OHLCV parse/map, search, interval/network maps
`src/onchain/events.ts`	`buildTvlSpikeEvents` from scalar TVL points
`src/onchain/proxy.ts`	`resolveDefiLlamaBaseUrl` / `resolveGeckoTerminalBaseUrl`

Worker proxy (CORS)

Default bases (when plugin `config.baseUrl` is empty):

```
{store.endpoint}/api/onchain/llama
{store.endpoint}/api/onchain/gecko
```

DefiLlama

Client path	Worker	Upstream
`/protocols`	`GET /api/onchain/llama/protocols`	`https://api.llama.fi/protocols`
`/protocol/:slug`	`GET /api/onchain/llama/protocol/:slug`	`https://api.llama.fi/protocol/:slug`

GeckoTerminal

Client path	Worker	Upstream
`/networks/:net/pools/:addr/ohlcv/:tf`	`GET /api/onchain/gecko/networks/.../ohlcv/...`	`https://api.geckoterminal.com/api/v2/networks/.../ohlcv/...`
`/search/pools`	`GET /api/onchain/gecko/search/pools`	`https://api.geckoterminal.com/api/v2/search/pools`

Allowlisted query keys:

- OHLCV: `aggregate`, `limit`, `currency`, `before_timestamp`
- Search: `query`, `network`, `page`, `include`

Address validation on the Worker: EVM `0x[a-fA-F0-9]{40}` or Solana-style base58 `^[1-9A-HJ-NP-Za-km-z]{32,48}$`. Network ids: `^[a-z0-9_]+$`.

Resolver: `src/onchain/proxy.ts`. Handler: `worker/src/onchain.ts`.

Cache keys (IDB dataset cache):
`provider|chainId|protocol|address|metric|facet|resolution` via `instrumentCacheKey`.

Related

- On-Chain data (end user)
- Worker overview — `/api/onchain/…` route table
- Sources — OHLCV (separate plane)
- Contracts — shared plugin base

STREAMS

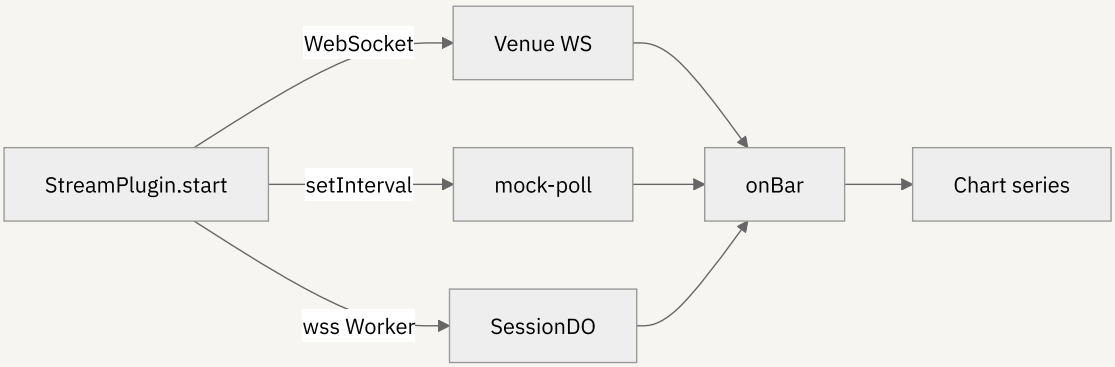
Live bar plugins: venue WebSockets, mock poll, and Cloudflare Durable Object relay example.

STREAMS

Abstract

A **stream** advances the chart's present. It opens a connection (or timer), emits `onBar` updates, and returns a **stop** function. Built-ins live in `frontend/src/streams/catalog.ts`. An optional Cloudflare Durable Object relay lives as a loadable example (`example-cf-do-stream.js`) plus Worker `SessionDO`.

Conceptual model



Built-in catalog

id	Name	Offline	Upstream
`binance-ws`	Binance WebSocket	no	`wss://stream.binance.com:9443/ws/{symbol}@kline_{interval}`
`okx-ws`	OKX WebSocket	no	`wss://ws.okx.com:8443/ws/v5/business` candle channels
`bybit-ws`	Bybit WebSocket	no	Public linear/spot kline topics
`coinbase-ws`	Coinbase WebSocket	no	Advanced Trade **candles** (venue OHLC, not ticker buckets)
`kraken-ws`	Kraken WebSocket	no	Public OHLC channel
`mexc-ws`	MEXC WebSocket	no	JSON kline (`wss://wbs.mexc.com/ws`); 15s PING
`mock-poll`	Mock Poll	**yes**	1s synthetic updates from `lastBar`

UI order: venues first, `mock-poll` last (`BUILTIN_STREAMS`).

Interface surface

```
start(opts: StreamOpts): () => void
```

Callbacks:

Callback	Use
`onBar`	OHLCV update (seconds)
`onError`	Hard failure (connect / parse)
`onStatus`	`{ state: 'open' | 'closed' | …, url?, detail? }`

Config is usually empty for venue streams; custom streams may declare `configSchema` (e.g. DO endpoint).

Internals

Path	Role
`frontend/src/streams/catalog.ts`	Built-ins + register helpers
`frontend/src/streams/ws-venues.ts`	Shared venue WS URL + subscribe builders
`frontend/src/streams/binance.ts`	Shared Binance helpers (if split)
`frontend/src/streams/multiplex.ts`	Multi-symbol helpers
`frontend/worker/src/durable-objects/session.ts`	`SessionDO` multi-venue fan-out
`frontend/public/plugins/example-cf-do-stream.js`	Loadable DO client stream

Binance kline mapping

```
onBar({
  time: Math.floor(k.t / 1000),
  open: +k.o, high: +k.h, low: +k.l, close: +k.c, volume: +k.v,
});
```

Open bars may update repeatedly for the same `time` until the kline closes — chart series should treat same-time updates as replace, not append-only.

mock-poll behavior

- Aligns bar `time` to interval slots.
- Updates high/low/close within the open slot; advances on slot change.
- Immediate first tick so live mode feels responsive offline.

Cloudflare DO stream (example plugin)

Not built-in. After load:

1. Config `endpoint` = Worker origin (`https://…workers.dev` or local `http://127.0.0.1:8787`).
2. Client opens `wss://…/api/stream?session=…&venue=binance&symbol=BTCUSDT&interval=1m` .
3. Worker routes to `SessionDO` ; DO opens one upstream WS for the venue and broadcasts raw kline JSON.

The DO supports the built-in CEX venues (Binance, OKX, Bybit, Coinbase, Kraken, MEXC) via the `venue` query param. Each venue's WS URL and subscribe message is built from `src/streams/ws-venues.ts` — the same shared helper used by browser-side StreamPlugins.

Requires `SESSIONS` Durable Object binding (often commented in `wrangler.toml` until provisioned).

Reconnect & status honesty

Built-in venue streams use `openReconnectableWs` (`frontend/src/streams/reconnect-ws.ts`):

- Exponential backoff (1s base, 30s cap, 8 attempts default).
- `onStatus({ state: 'reconnecting', detail })` between attempts.
- Hard `onError` only on construct failure or reconnect exhausted.
- Multiplex keeps stream green **only** after `state: 'open'` ; starts as `connecting` .

Optional `Bar.closed` (Binance `k.x` , OKX confirm, Bybit `confirm`) feeds live re-run mode `bar-close` (settings). Multiplex also treats **bar time advance** as a close for venues without a flag.

Invariants & edge cases

1. **Always return stop** — stop cancels reconnect timers and closes the socket.
2. **Symbol casing** — Binance lowercases stream symbols; OKX uses `BTC-USDT` inst ids.
3. **Browser WS limits** — many tabs × many symbols hit connection caps; DO relay amortizes upstream sockets.
4. **Reconnect is built-in for venue WS** — mock-poll does not reconnect (timer only).

Failure modes

Symptom	Cause
Stream never opens	CORS N/A for WS; check firewall / venue geo blocks
DO stream 503 `NO_DO`	`SESSIONS` binding missing
Silent no bars	Message shape mismatch (non-kline payloads)

See also

- Sources
- Worker durable objects
- Plugin examples

ENGINES

Calculation engines: server (Flask/Worker) and client Pyodide — run contracts, assets, and readiness.

ENGINES

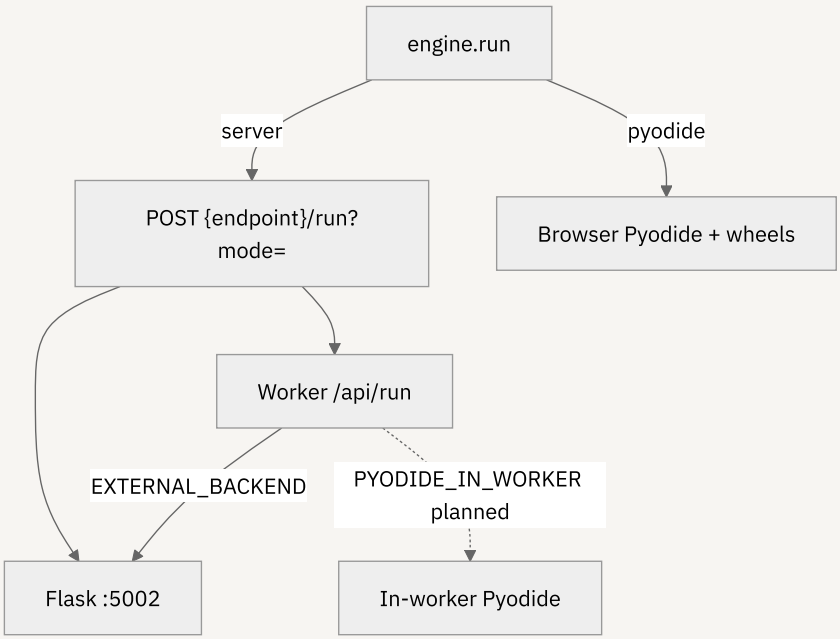
Abstract

An **engine** evaluates a script against bars and returns plots, series, events, and optional drawings. AXIS ships three built-ins in `src/engines/catalog.ts`:

id	Name	Where Python runs
`server`	Server-Side	External Flask Pro API **or** AXIS Worker `/api/run` proxy
`pyne-worker`	pyne-worker (edge)	HOOX **pyne-worker** Cloudflare® edge evaluator (`POST /run`)
`pyodide`	Client-Side (Pyodide)	Browser WebAssembly via self-hosted Pyodide

AXIS Worker in-process Pyodide (`PYODIDE_IN_WORKER`) remains a separate, feature-gated edge path — see Worker runtime.

Conceptual model



Interface surface

```
isReady(): Promise<boolean>
run({ script, bars, config?, signal? }): Promise<RunResult>
```

`RunResult` is defined in contracts.

server engine

Config schema

Key	Default	Notes
`endpoint`	`http://localhost:5002`	Overridden by topbar / `store.endpoint`
`mode`	`auto`	`interpret` | `compile` | `auto` (try compile, fall back to interpret)

HTTP contract

- `POST ${endpoint}/run?mode=…`
- Body: `{ script, data: bars }`
- Success JSON: `plots` , `series?` , `events?` , `drawings?` , `meta?` , `mode` , `script_id` , `run_id`
- Error: non-OK status or `status: 'error'` → mapped to `RunResult` with `error`

WebSocket contract (preferred when available)

- URL: `ws(s)://host/ws/run` (derived from `endpoint` ; requires `flask-sock` on the Pro API)

- Client → `{ type: "run", id, script, data, mode? }`
- Server → `{ type: "result", id, status: "success", plots, series, … }` or `{ type: "error", id, message }`
- AXIS server engine tries WS first (`preferWs`, default true), then REST. Health `GET /` reports `"websocket": true` when the route is mounted.
- `meta.transport` is `"ws"` or `"rest"` so the Connection HUD can show the path used.

Readiness: `GET ${endpoint}/` within 8s.

Timeouts: adaptive `min(180s, max(60s, 30s + bars×80ms))`.

Note on Worker path: the PWA's `server` engine posts to `/run` on whatever endpoint you set. The AXIS data-plane Worker exposes **`POST /api/run`**. When pointing the PWA at that Worker, either put a reverse-proxy path rewrite in front, or set the endpoint so the engine path matches your deployment. Worker `handleRun` validates `{ script, data, mode? }` and **today prefers proxying to `EXTERNAL_BACKEND`** (Flask). In-worker Pyodide only if `PYODIDE_IN_WORKER=enabled` and the wheel pipeline works.

pyne-worker engine

Dedicated HOOX **pyne-worker** edge host (not `worker-axis`).

Key	Default	Notes
`endpoint`	`https://pyne-worker.cryptolinx.workers.dev`	Production workers.dev
`mode`	`interpret`	`interpret` | `compile` | `auto` (see `/health` `features.modes`)
`preferWs`	`false`	Edge deploys are usually REST-only
`apiKey`	`''`	Sent as `X-API-Key` + Bearer when the Worker secret `API_KEY` is set

Contract: same evaluate body as Flask (`POST /run`). Health: `GET /health` → `{ status: "ok", worker: "pyne-worker", … }`.

UI: Topbar engine list, Settings preset **pyne-worker edge**, Workers Manager → **Use as calculation backend** / **Use pyne-worker engine**.

Browser CORS: the pyne-worker origin must allow the AXIS page Origin (or use a same-origin reverse proxy). Auth failures return HTTP 401 without a key.

pyodide engine

Config

Key	Default
`indexUrl`	`/pyodide/v0.26.2/` (self-hosted)

Boot pipeline (`_ensure`)

1. Prefetch assets (`prefetchPyodideAssets`) — wasm, stdlib zip, micropip wheels.
2. `loadPyodide({ indexURL })`.
3. `micropip.install` same-origin `/vendor/pynescript-0.2.0-py3-none-any.whl` and antlr runtime wheel.
4. Fetch `/pyodide/pynescript_runtime.py` and `runPythonAsync` it.

Refreshing the pyne wheel (after pyne compiler/runtime changes):

```
# from axis repo — builds sibling ../pyne (or PYNE_ROOT; dir sometimes still named pynescript) and vendors the wheel
./scripts/sync-pyne-wheel.sh
bun run build
```

Keep the hard-coded wheel filename in `src/engines/catalog.ts` / `index.js` in sync if the package version changes. The browser bridge is interpret-first; `mode=compile` / `auto` uses the wheel's `pynescript.compiler` when NumPy loads (object-mode works without Numba; pure-numeric compile still needs server/Numba). 5. `run` calls `run_script(script, bars)` in Python and `JSON.parse`s the result.

Guards: `assertZipAsset` rejects HTML SPA fallbacks (classic deploy footgun when `public/vendor` is missing from `dist/`).

Capabilities: `{ offline: true, needsNetwork: false }` after assets are cached same-origin.

Helpers: `preloadPyodide()`, `LOCAL_PYODIDE_VERSION = '0.26.2'`.

Internals

Path	Role
`frontend/src/engines/catalog.ts`	`serverEngine`, `pyodideEngine`, registration
`frontend/src/indicators/runner.ts`	UI run orchestration
`frontend/public/vendor/*.whl`	Shipped wheels
`frontend/public/pyodide/`	Self-hosted Pyodide + runtime py
`frontend/worker/src/runtime.ts`	Edge `/api/run`
`frontend/worker/RUNTIME.md`	In-worker Python plan

Invariants & edge cases

1. **AXIS never imports pynescript Python** except through engines.

2. **Abort** — pass `signal` from UI cancel; server engine respects it.
3. **Error as data** — both engines often return `status: 'error'` instead of throwing so the Results panel can show messages.
4. **Pyodide size** — ~14MB self-hosted index; first ready can take seconds; preload on idle.

Worked examples

Desk research (Flask)

- Active engine: `server`
- Endpoint: `http://127.0.0.1:5002`
- Backend: `make run` (Flask Pro API)

Offline lab

- Source `mock-walk`, stream `mock-poll`, engine `pyodide`, storage `local`
- Requires `dist/` (or Vite public) to serve `/pyodide/**` and `/vendor/**`

Tiny non-Python engine

See plugin examples — `example-tiny-pyne-engine.js` implements a JS DSL with `sma` / `ema` / `rsi` for demos without PYNE.

Failure modes

Symptom	Cause
`pynescript wheel returned HTML`	SPA fallback; deploy vendor into dist; static server must not rewrite `.whl`
`loadPyodide not available`	Missing pyodide.js / blocked script
HTTP error from server	Flask down; wrong endpoint; CORS
Worker 503 `NO_BACKEND`	`EXTERNAL_BACKEND` empty and Pyodide path disabled/failed

See also

- Evaluation map — Flask vs Pyodide vs Worker vs (not) PyneTS
- Worker runtime
- Build and serve
- PYNE runtime
- PyneTS — TS library; AXIS does not import it
- AXIS and HOOX

STORAGE

Script library backends: local (IndexedDB), cloud (Worker /api/scripts), and git (GitHub/GitLab).

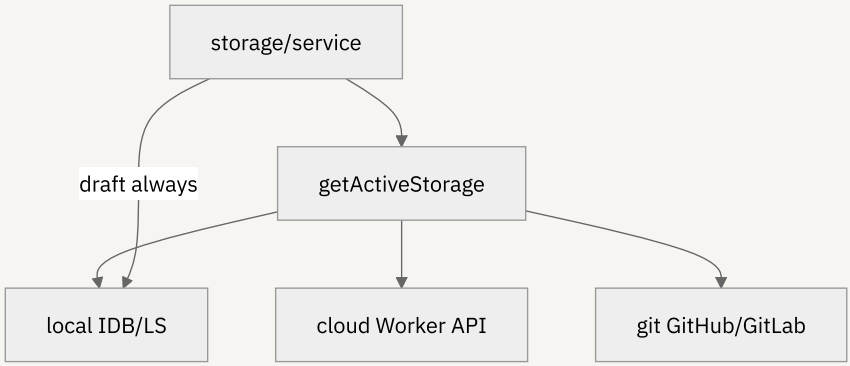
STORAGE

Abstract

Storage plugins persist the user's Pine library and drafts. Built-ins: `local`, `cloud`, `git` — registered from `frontend/src/storage/catalog.ts`. High-level UI/editor APIs go through `frontend/src/storage/service.ts`, which always dual-writes drafts to **local** for crash recovery.

Dynamic `kind: 'storage'` install via URL is **not supported** (`loader.ts` throws).

Conceptual model



Built-in plugins

local (`frontend/src/storage/local.ts`)

Concern	Detail
Primary	IndexedDB `pynescript.axis.storage` — stores `scripts`, `kv`
Fallback	`localStorage` keys `pynescript.axis.library.v1`, draft keys
Migration	Older library keys (`pynescript.axis.library.v1`, editor docs)
Memory	In-process maps when neither IDB nor LS exist (tests/SSR)
Capabilities	Offline-friendly; file versioning (snapshots on each save)

cloud (`frontend/src/storage/cloud.ts`)

Concern	Detail
Transport	`fetch` to Worker `/api/scripts`
Auth	`Authorization: Bearer <api_key>` (`pn_…`)
Config	`endpoint` (Worker URL — never the Pine engine host), `apiKey` (`pn_…`). Set in **Settings → Script storage** or Script Library.
Concurrency	`If-Match` revision headers on write
Partition	Worker hashes key → `userId`; rows scoped per user

git (`frontend/src/storage/git.ts`)

Concern	Detail
Providers	GitHub Contents API / GitLab repository files
Config	`provider`, `token`, `owner`, `repo`, `branch`, `basePath` (default `pine-library`), commit template
Commit boundary	**Explicit Save only** — not every keystroke
Drafts	`saveDraft` / `loadDraft` are no-ops; local dual-write handles drafts
Version history	`listVersions` / `readAtRevision` — GitHub/GitLab commits for the script path; local IndexedDB/localStorage snapshots; Worker `script_versions`
Restore	`restoreScriptVersion` writes historical content as a **new** tip commit
Capabilities	`needsNetwork`, `needsAuth`

Interface surface

Contract: StoragePlugin.

service.ts API (UI-facing)

Function	Behavior
`listScripts(prefix?)`	Active backend list
`readScript` / `writeScript` / `removeScript`	CRUD + log
`saveDraft` / `loadDraft`	Local first; also active if supported
`exportLibraryJson` / `importLibraryJson`	Portable backup
`getStorageStatus`	Connected / dirty / remote hints
`supportsScriptVersioning`	True when active storage implements version history (local, cloud, git)
`listScriptVersions(id)`	Version history for a script
`readScriptVersion(id, rev)`	Content at a revision
`restoreScriptVersion(id, rev)`	Write historical content as the new tip
`copyScriptsBetweenStorages(from, to)`	Copy library between engines — **never deletes the source**

Catalog helpers

```
ensureStoragesRegistered()
getStorage(id)
listStorages()
registerDynamicStorage(plugin)  // in-process only (tests / future)
unregisterDynamicStorage(id)
```

Internals

Path	Role
`storage/catalog.ts`	Registration
`storage/local.ts`	IDB + LS
`storage/idb.ts`	Promise wrappers for IDB
`storage/cloud.ts`	Worker client
`storage/git.ts`	Orchestration
`storage/git-github.ts` / `git-gitlab.ts`	Provider APIs
`storage/git-config.ts`	Config normalize
`storage/service.ts`	Facade
`worker/src/scripts.ts`	Cloud API implementation
`worker/schemas/scripts.sql`	D1 schema

D1 schema (cloud)

```
PRIMARY KEY (user_id, id)
-- scripts + script_drafts tables
```

Without D1, Worker keeps per-isolate in-memory maps (fine for `wrangler dev`).

Invariants & edge cases

1. **Draft recovery** — never rely solely on remote drafts; service always hits local.
2. **Git PAT scope** — GitHub `contents:write`; GitLab `api` / `write_repository`.
3. **Revisions** — local uses `local-${timestamp}`; cloud/git use remote revisions for conflict detection.
4. **Active default** — `getActiveStorageId()` → `local` when unset.
5. **Engine switch is a copy** — switching storage copies scripts onto the target and **never removes** them from the source.
6. **Cloud credentials** — Worker URL + API key live in Settings (and Script Library). `store.endpoint` is the Pine engine and is never used as the scripts Worker base.

Worked examples

Export / import

```
const docs = await exportLibraryJson();
// …move machine…
await importLibraryJson(docs, { forceNewIds: true });
```

Point cloud at local Worker

1. `axis setup --prod` then `axis secret put ADMIN_TOKEN` then `axis deploy all` then `axis keys create`
2. `axis keys create` — paste the `pn_…` key into Settings → Script storage
3. Local demo: `axis dev worker` (`:8787`) + Settings **Generate demo key** when `ALLOW_OPEN_KEYS=1`

Failure modes

Error	Cause
`Cloud storage requires an API key`	Empty `apiKey` — set it in Settings → Script storage
401 `NO_KEY` / `INVALID_KEY`	Auth path; see Worker auth
409 on write	`If-Match` revision conflict
Git 401/404	Token, owner/repo, or basePath wrong
Quota errors on localStorage	Large libraries should prefer IDB (automatic when available)

See also

- Worker data plane
- Worker auth
- Contracts

DYNAMIC PLUGIN LOADER

Install ES-module plugins from URL (source, stream, engine, dataset, component), persist, restore, safety rules.

DYNAMIC PLUGIN LOADER

Abstract

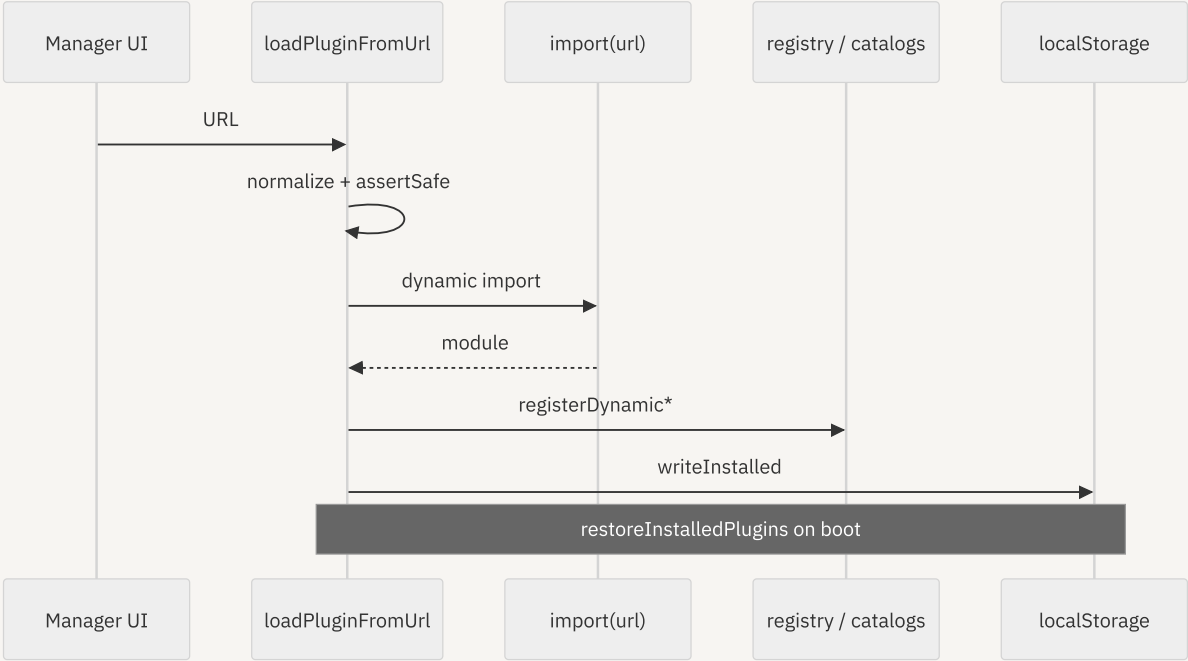
The loader (`src/plugins/loader.ts`) lets users install **third-party or example** plugins at runtime via dynamic `import()` . Supported kinds: **`source`** , **`stream`** , **`engine`** , **`dataset`** , **`component`** (e.g. PYNE Agent). Installed URLs persist in `localStorage` under:

Key	Role
`pynescript.axis.plugins.v1`	Installed plugin list

Studio Plugins Install tab: ES module URL field and same-origin examples

On boot, `restoreInstalledPlugins()` re-imports every saved URL.

Conceptual model



Interface surface

```
loadPluginFromUrl(url: string): Promise<InstalledPlugin>
removePlugin(id: string, kind?: string): void
getInstalledPlugins(): InstalledPlugin[]
restoreInstalledPlugins(): Promise<void>
normalizePluginUrl(url: string): string
assertSafePluginUrl(href: string): void
PLUGINS_KEY // 'pynescript.axis.plugins.v1'
```

InstalledPlugin

```
{ url, id, name, kind, description? }
```

Module export shapes accepted

1. `export default plugin`
2. `export const plugin = …`
3. Module namespace object that *is* the plugin

Must include `id` and `kind` . Required methods / contracts:

kind	Requirement
`source`	`fetchHistorical`
`stream`	`start`
`engine`	`run`
`dataset`	dataset fetch contract (`onchain` catalog)
`component`	`ComponentPlugin` — `slots` / `mount` ; optional `init` / `dispose` (registered on the unified registry)
`storage`	**Rejected** via URL install — use built-in local/cloud/git

Internals

URL normalization

```
// /src/plugins/foo.js → /plugins/foo.js
href.replace(/(^|\/)src\/plugins\//, '$1plugins/');
```

Dev-only Vite paths never ship in `dist/` ; production examples live under `public/plugins/` → `/plugins/…` .

Safety

`assertSafePluginUrl` rejects:

- `javascript:`
- `vbscript:`
- `data:text/html…`

This is **not** a full sandbox. Dynamic plugins run with the page's privileges (network, DOM). Treat plugin URLs like executable code supply chain.

Dedup on install

List filters out same URL **or** same `(kind, id)` before append — reloading an updated module replaces the entry.

Unregister

`removePlugin` calls `unregisterDynamic*` for the resolved kind (or all kinds if unknown) and rewrites the installed list.

Bootstrap coupling

`loadPluginFromUrl` / `restoreInstalledPlugins` call `ensureBuiltins()` first so catalogs exist before dynamic ids land.

Manager UX (product)

From `src/plugins/README.md` :

1. **Manager → Plugins → Load from URL**
2. Example: `http://localhost:8081/plugins/example-coingecko-source.js`
3. **Export installed** / **Import…** for machine migration
4. Auto-activate patterns for source/stream/engine after install (Manager catalog **Use**)

Shipped examples (also under `public/plugins/`):

File	Kind
`example-coingecko-source.js`	source
`example-tiny-pyne-engine.js`	engine
`example-cf-do-stream.js`	stream

External component example: PYNE Agent — `https://pyne-agent-worker.cryptolinx.workers.dev/plugin/axis-pine-agent.js`

Invariants & edge cases

1. **CORS on module URL** — the JS file origin must allow module import (same-origin is easiest).
2. **CORS on plugin's own API** — separate problem; may need proxy (see plugins README).
3. **Partial restore** — one bad URL logs error and continues others.
4. **No code signing** — trust the URL owner.

Worked example

```
# After bun run build + axis_pwa_server.py
# Load:
http://127.0.0.1:8081/plugins/example-coingecko-source.js
```

Then select **CoinGecko** in the source picker (id `coingecko`).

Failure modes

Error	Meaning
`URL required`	Empty input
`Plugin URL scheme not allowed`	Dangerous scheme
`Module did not export a plugin object`	Wrong export shape
`Plugin needs id and kind`	Incomplete object
`Source plugin needs fetchHistorical()`	Incomplete source
`Unknown plugin kind`	Typo or unsupported kind (not in source/stream/engine/dataset/component)
Restore log errors	404, network, syntax error in remote JS

See also

- Plugin examples
- PYNE Agent plugin
- Registry
- CORS and origins

PYNE AGENT PLUGIN

Install the pyne-agent-worker AXIS component plugin — natural-language PYNE script chat via Cloudflare® Workers AI™ (standalone or HOOX-enhanced).

PYNE AGENT PLUGIN

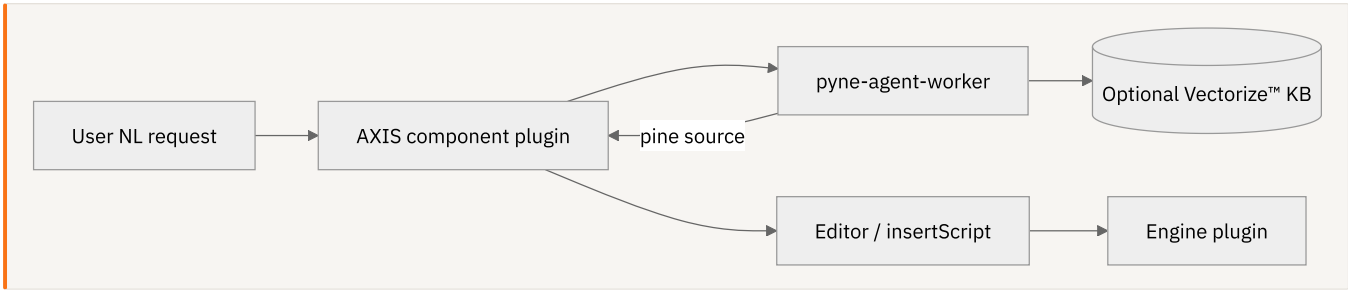
Abstract

pyne-agent-worker is a sister Cloudflare® Worker that turns natural language into PYNE-compatible scripts. AXIS consumes it as a **component** plugin (`kind: 'component'`) loaded from URL. It does **not** replace engines — evaluation still goes through your active engine (`server` , `pyodide` , …). The agent only **writes** script source into the editor / chat UI.

Repo	hoox-sh/pyne-agent-worker
Plugin module	AXIS ships `/plugins/axis-pine-agent.js` ; remote `GET /plugin/axis-pine-agent.js` on the agent Worker
API	`POST /v1/chat`
Standalone	Workers AI™ only — **no** pyne-worker / HOOX mesh required
Optional	pyne-worker validate→retry when configured

Pine Script™ and TradingView® are trademarks of TradingView, Inc. Cloudflare® is a registered trademark of Cloudflare, Inc. Independent project — not affiliated with TradingView, Inc. or Cloudflare, Inc.

Conceptual model



Install in AXIS

1. Deploy pyne-agent-worker (standalone is enough) for the **API** endpoint.
2. AXIS → **Manager → Plugins → Install from URL** — prefer the **same-origin** module:

```
/plugins/axis-pine-agent.js
```

Remote (needs CORS on the agent Worker):

```
https://pyne-agent-worker.cryptolinx.workers.dev/plugin/axis-pine-agent.js
```

3. Configure plugin fields:

Field	Meaning
`endpoint`	Agent Worker origin (no trailing slash) — not the AXIS page origin
`apiKey`	Worker `API_KEY` (optional in open local mode)
`pineVersion`	`auto` | `v5` | `v6`
`style`	`auto` | `indicator` | `strategy` | `library`

4. Open the agent from the **manager tab** / **topbar** action, **Workers Manager → Install/Configure**, or the floating **PYNE Agent** button when mounted.

Component URL load is enabled (AXIS 2.0.x): `loadPluginFromUrl` accepts `kind: 'component'` , registers via `registry.registerComponent` , and runs optional `init({ getConfig })` .

Contract namespace remains `pynescript.axis.plugins.v1` . Export shape: default ES module with `kind: 'component'` , `slots` , `mount` , optional `init` / `dispose` .

See also: Dynamic loader, Plugin examples, end-user guide PYNE Agent, Workers Manager.

Standalone vs HOOX

Mode	Config	Behavior
Standalone (default)	No `PYNE_SERVICE` / `PYNE_WORKER_URL` on the worker	Chat + optional RAG; `validation.skipped`
HOOX-enhanced	Optional pyne-worker binding or URL	generate → `POST /run` → fix retries

AXIS users who only want natural-language script authoring never need to deploy pyne-worker.

API surface (worker)

Method	Path	Role
`GET`	`/health`	`mode: standalone | hoox`
`POST`	`/v1/chat`	NL → `reply` + extracted `pine`
`GET`	`/plugin/axis-pine-agent.js`	This plugin module
`GET`	`/`	Built-in chat shell (iframe / demo)

Auth: `X-API-Key` or `Authorization: Bearer` when the worker secret is set.

Legal / content

- The agent repo **never ships** TradingView® built-in indicator sources.
- Knowledge (docs v5/v6, open corpus ≤ 1000, operator builtin refs) is ingested privately into R2 + Vectorize™.
- Always treat marks: Pine Script™, TradingView®, Cloudflare®.

Failure modes

Symptom	Likely cause
Plugin load fails (`kind: component`)	CORS / mixed content on module URL; verify HTTPS and that AXIS is current (component URL load is supported)
401 on chat	Missing/wrong `apiKey` vs worker `API_KEY`
Empty RAG quality	KB not ingested; chat still works with model knowledge only
`validation.available: false`	Expected in standalone — not an error

See also

- Manager and settings
- End-user: PYNE Agent
- pyne-agent-worker README
- PYNE docs

UI

AXIS UI subsystem: chart host, editor, UI shell, indicators, results, and Solid store—presentation without a closed engine.

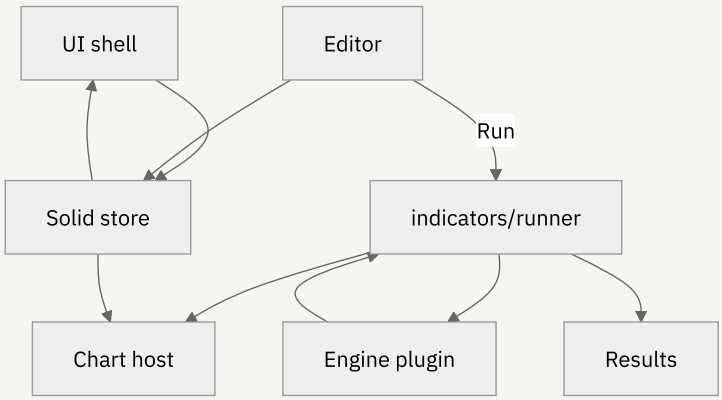
UI

UI is the AXIS browser presentation layer: everything the operator sees and clicks that is *not* language evaluation. Engines are plugins; AXIS applies their outputs to panes, markers, and drawers.

Abstract

Subsystem	Responsibility
Charting	Panes, multi-chart, replay, compare, drawings, badges
Editor	CM6, diagnostics, ruler, git bar, stats, dock/popout
Debugging	Scriptlogs, Profiler, Debug chips, Pins, Problems
UI shell	Topbar groups, docks side-by-side, palette, shortcuts, mobile shell, alerts
Scripts	Run pipeline, Scripts panel, overlay/sub-pane routing
Results & strategy	Event normalize, trades, equity, export
Store	Solid store, persistence, active plugins

Conceptual model



Invariant (AXIS ≠ engine): no Solid component imports PYNE directly; only engine modules talk to Python/HTTP runtimes.

Interface surface (visual map)

```
┌─ Topbar (grouped: market · data · compute · layout · panels · system) ─┐
├─ left docks ─┬─ ChartHost (1–4 slots) + drawings ─┬─ right docks ──────┤
│  watchlist   │  multi-pane + badges + replay       │  indicators|editor │
│  layers …    │                                     │  (side-by-side)    │
├─ Results / scriptlogs (bottom) ──────────────────────────────────────────┤
├─ System logs ─────────────────────────────────────────────────────────────┤
└─ StatusBar (connection HUD) ──────────────────────────────────────────────┘
   + Studio (Runtime · Wire · Settings) · Command palette (⌘K)
```

Implemented in `src/app.tsx` (repo root product UI).

AXIS UI: watchlist, multi-pane chart, and editor

Stack

Concern	Choice
Framework	SolidJS 1.9
Bundler	Vite 6 + `vite-plugin-solid`
Charts	lightweight-charts 5
Editor	CodeMirror 6
Icons	lucide-solid
Style	Tailwind 4 + void tokens (`data-theme`)

Relation to other tracks

- Operators: End User
- Structure: Architecture
- Contracts: Plugins
- Language: PYNE runtime

See also

- `LEGACY.md` — what is *not* the AXIS product path
- `TESTING.md` — unit/e2e around UI modules

CHARTING

AXIS chart: ChartHost, PaneManager, series factory, drawings, crosshair sync, and trade markers.

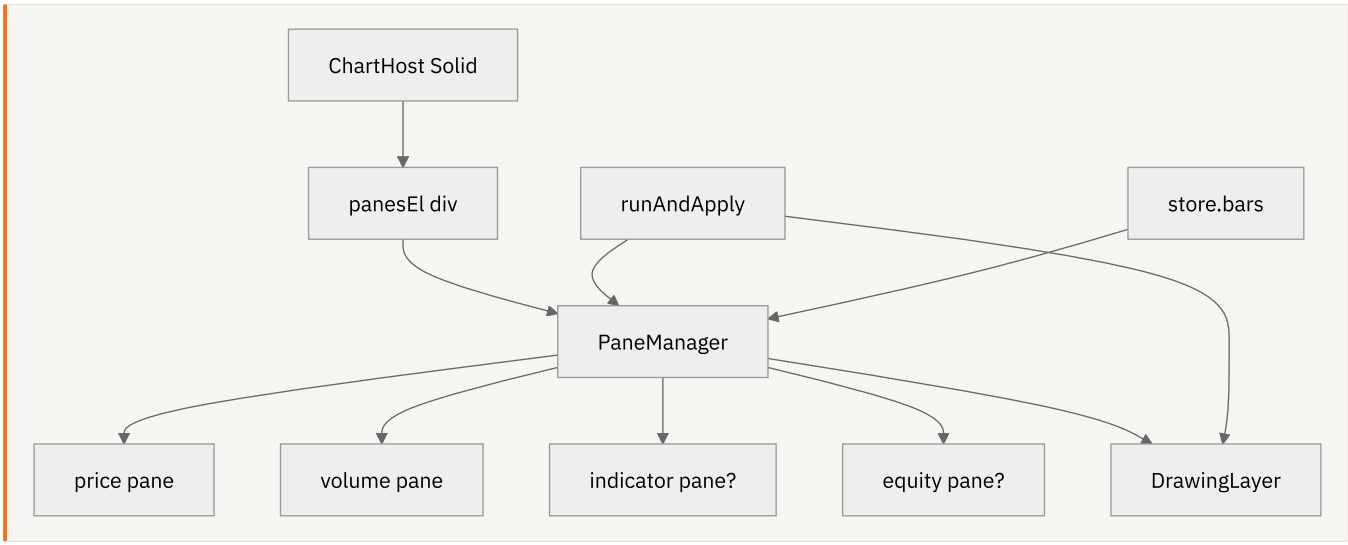
CHARTING

Charting is the spatial surface for OHLCV, overlays, and annotations. Implementation is **imperative** under a Solid shell so lightweight-charts owns its DOM subtree.

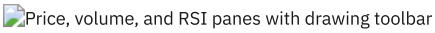
Abstract

Module	Role
`ChartHost.tsx` / `ChartWorkspace.tsx`	Solid mount; multi-slot grid; empty-state
`pane-manager.ts`	Multi-pane LWC charts, time sync, overlay apply
`pane-badge.ts`	Corner name + script action icons
`series-factory.ts`	Candle/volume/line series helpers + palette
`chart-type.ts`	Price style transforms (HA, hollow, …)
`bar-replay.ts` + `BarReplayControls`	History scrub / play
`layout.ts` / `layout-recipes.ts` / `chart-registry.ts`	Multi-chart layouts + presets
`compare-overlay.ts`	Second-symbol compare series
`volume-profile.ts`	OHLCV volume-at-price overlay
`drawing-layer.ts` + `drawings/*`	User + script drawings, templates, sync helpers
`price-precision.ts`	Price-scale decimals (`auto` or 0–8) from symbol + bars
`results/plot-visuals.ts`	Pine `plot.style_*` → LWC series kind (line, stepline, columns, area, …)
`crosshair-sync.ts`	Cross-pane crosshair
`pyne-drawings.ts`	Engine drawing objects → layer
`manager-access.ts`	Process-wide manager/layer getters

Conceptual model



Interface surface


Price, volume, and RSI panes with drawing toolbar

Panes

`store.panes` : `{ id, type, height, order, visible, label }`.

type	Typical series
`price`	Price series (style below) + overlay lines + markers + drawings
`volume`	Histogram
`indicator`	Non-overlay plots
`equity`	Strategy equity curve

Defaults: price + volume. Indicator/equity created on demand.

Price chart styles (`store.chartType`)

Topbar **Type** select (persisted). Implementation: `chart/chart-type.ts` + `createPriceSeries` .

chartType	Series	Notes
`candles`	Candlestick	Japanese candles (default)
`hollow`	Candlestick	Hollow body on up bars
`bars`	Bar	Classic OHLC bars
`heikinashi`	Candlestick	Heikin-Ashi transform of OHLCV
`line`	Line	Close only
`area`	Area	Close with fill
`baseline`	Baseline	Close vs first-bar base

Series still live under pane key `candle` so markers, hlines, and the drawing layer keep a stable host. Changing type swaps the LWC series and rebinds data without a full history reload.

Empty / status overlay

When `bars.length === 0` , ChartHost shows contextual title/sub from `store.status` (loading, error, running, or “Load data to begin”).

Price-scale decimals

Price pane labels use `chart/price-precision.ts` :

Mode	Behavior
`auto` (default)	Merge symbol heuristics (e.g. BTC→2, SHIB→8) with recent OHLCV significant digits
`0` – `8`	Fixed display decimals + matching `minMove`

UI control cycles **A → 0 → … → 8 → A** on the price-pane scale chrome. Store field: `priceScaleDecimals` (persisted).

Plot style parity

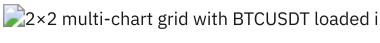
Engine plot series honor Pine Script™ `plot.style_*` via `mapPlotStyleToSeriesKind` (`results/plot-visuals.ts` → pane-manager series factory):

Pine style (examples)	Chart series
`plot.style_line` / default	Line
`plot.style_stepline`	Stepline (LWC WithSteps)
`plot.style_stepline_diamond`	Stepline + vertex point markers
`plot.style_histogram`	Histogram (base 0)
`plot.style_columns`	Columns → LWC Histogram (base 0; no column-width/gap API)
`plot.style_area`	Area
`plot.style_circles`	Circular point markers + faint hairline
`plot.style_cross`	Larger point markers, connector hidden
`plot.style_linebr` / `steplinebr` / `areabr`	Broken line / stepline / area — `na` ends a segment (LWC would otherwise connect through whitespace)

LWC limits: Histogram series has no bar-width or gap option, so `columns` and `histogram` share the same bar geometry (both zero-based). Point markers are always circular — `cross` is approximated by discrete markers without a connector; `stepline_diamond` uses stepline + vertex markers. Sparse `plotshape` styles map diamond/cross to square markers (optional `+` / `×` glyphs when text is omitted).

`fill(plot1, plot2, color=…)` renders as SVG bands between the two series when both resolve on the same pane.

Multi-chart layouts


2×2 multi-chart grid with BTCUSDT loaded in the active slot

- Modes: **1 / 2H / 2V / 4** (`store.chartLayout`)
- Per-slot symbol/interval via registry; active slot drives topbar + Run
- **Named layouts** save/load (optional chrome snapshot)
- **Recipes** in Layouts menu: Scalp 1m+5m, Swing HTF, Quad, BTC majors (`layout-recipes.ts`)

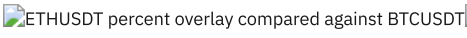
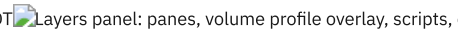
Bar Replay

Bar replay strip on BTCUSDT: scrub, play, and speed Scrubbing and playing bar replay over loaded history

Topbar **Replay** → session over loaded OHLCV (`bar-replay.ts`):

- Enters with **full history visible** (cursor on last bar)
- Scrub / step / play; **Play at end restarts from bar 0**
- Mutually exclusive with Live

Compare & volume profile

 ETHUSDT percent overlay compared against BTCUSDT  Layers panel: panes, volume profile overlay, scripts, drawings

- **Compare** control: second symbol as % or absolute overlay (`compare-overlay.ts`)
- **Volume profile** toggle in Layers → OHLCV approximation histogram (`volume-profile.ts`)

Pane badges

Each pane's top-left chip (`pane-badge.ts`): label + script actions when scripts are applied:

Icon	Action
Settings	Script inputs modal
Eye	Show / hide series
Refresh	Re-run script
Trash	Remove from chart

Volume/equity without scripts: name + hide.

Drawings toolbar

See Drawings. Tools include cursor, hline, vline, trend, ray, extend, rect, ellipse, arrow, fib, measure, text (plus fib/shape/annotation/trading packs). The style bar edits color/width/dash plus per-tool extras (extend, fib levels, RR, text); last-used style is stored per kind. Templates and duplicate helpers live under Layers / drawings modules.

Crosshair & time

`syncTimeScales` / `syncCrosshair` keep multi-pane navigation coherent. `scrollToTime` / `jumpToDebugPin` used from Results and debug pins.

Internals

Solid vs imperative boundary

```
// ChartHost — Solid owns outer shell
// PaneManager only mutates panesEl — never put Solid children inside panesEl
```

On cleanup: destroy drawing layer, dispose manager, clear accessors.

Data path

`setDataToChart(bars)` applies OHLCV; `createEffect` on `store.bars` re-applies if bars change outside load helper.

Run apply (chart side)

From `runAndApply` (indicators):

- `syncOverlayLines` — update existing overlay series in place (avoids destroy→blank→add flash on live re-runs)
- Markers from normalized events (set in place)
- Script drawings atomic replace via DrawingLayer (`setScriptDrawings` without empty frame); `linefill` quads and `force_overlay` geometry route with other script drawings; `barcolor` tints candles via plot-visuals (not the SVG layer)
- Equity series when strategy report warrants (silent live runs skip hide thrash)

History vs live bars

- Full loads: `loadBars` → `chartDataGen++` → `setDataToChart` + fitContent
- Live: `multiplex` → `appendBar` + `manager.appendBar` only (no fitContent, no full setData)
- **Performance (2.0.x):** live tick path is O(1) append + conflation under load; heavy history paint is batched; load requests use **AbortController** so symbol switches cancel in-flight venue fetches
- Overlay re-runs prefer in-place series updates (`syncOverlayLines`) to avoid destroy→blank flash

series-factory

Centralizes series construction options, plot-style mapping, and color palette so Results plot summary and chart stay chromatically related.

Invariants

1. Solid reconciliation must not rewrite pane roots.
2. Script drawings ephemeral per run; user drawings store-backed.
3. Overlay routing: engine `meta.overlay` + script type; **oscillator-scale** series forced to sub-pane when they would vanish on price (see Indicators).
4. Shared non-overlay sub-pane id is stable: **`indicator`**.
5. Manager may be null before mount—runner no-ops chart apply safely.

Failure modes

Symptom	Cause
Blank panes after hot reload	Manager disposed; full remount
Overlays stack forever	Old path skipped removeOverlays
Fib/tools ignore clicks	No bars / time scale; wrong tool
Markers at t=0	Event times not bar-aligned

Worked example (dev)

```
getManager()?.scrollToTime(barTime);
```

See also

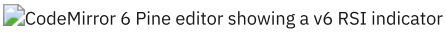
- Indicators
- Results and strategy
- End-user drawings

EDITOR

AXIS Pine editor: CodeMirror 6, diagnostics, 80-col ruler, git sync, stats, docked vs popout, editor bridge.

EDITOR

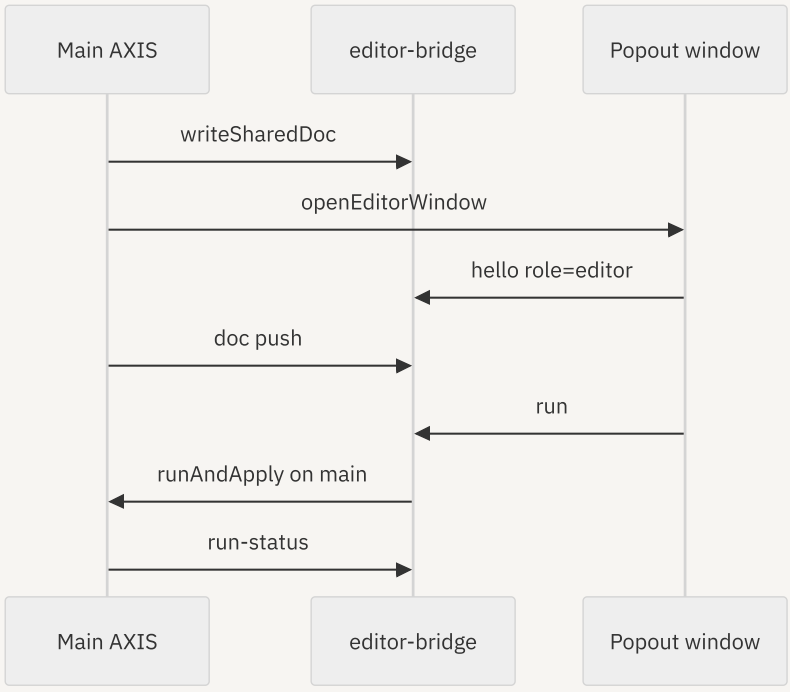
The editor is the **source axis of intent**: Pine text the operator authorizes the engine to run. Completion and hover can use local builtins metadata and optional remote LSP; **evaluation** is always an engine plugin.



Abstract

Module	Role
`EditorPane.tsx`	Docked/standalone chrome, header tools, detach
`tabbed-editor.tsx`	Multi-tab docs, stats strip, Problems, Colors, Symbols, git bar
`PyneEditor.tsx`	CM6 mount (extensions + view APIs)
`pyne-language.ts`	StreamLanguage / highlighting (namespaces, types, multiline strings)
`pine-symbols.ts` + `SymbolEmojiManager.tsx`	TV-editor-safe glyph catalog + insert panel
`pyne-lsp.ts` + `data/pyne-builtins.json`	Offline completion/hover corpus
`column-ruler.ts`	80-character guide line
`diagnostics.ts`	Underlines, gutter, jump-to from last run
`editor-intel.ts`	Settings bag for lint / hover / complete / marks / timings
`inline-debug.ts`	EOL chips, pin gutter, line flash
`git-sync.ts` + `ui/EditorGitBar.tsx`	Pull / push / storage status
`ui/EditorProblems.tsx`	Collapsible problems list
`editor-bridge.ts`	Cross-window doc + run messages
`cm-void.ts`	Void-theme CM styling

Conceptual model



Why run on main? Chart, bars, and pane manager live in the main window. Popout is a pure editing surface.

Interface surface

Docked mode

- Right dock by default (`panelChrome.editor`); may sit **side-by-side** with other right panels (e.g. Indicators left of Editor)
- FloatableShell: dock left/right/bottom, float, new window
- Detach → popup or tab (`openEditorWindow`)

Popout mode

- Main shows “Editor detached” chip + **Reattach**
- Shared doc key + bridge messages: `hello` , `doc` , `run` , `run-status` , `reattach` , `popout-opened` , `popout-closed`

Run entry points

- Editor / topbar **Run** → `runAndApply(getDoc())`
- Popout Run → bridge → main executes
- Command palette: save library, git push/pull, jump to line, toggles

Header tools (`axis-editor-tools`)

Icon-only (labels in tooltips / `aria-label`). **Open in new tab** is in the left hamburger menu with dock options.

Control	Effect
Run	Run script against loaded bars
Scriptlogs	Open last-run `log.*` panel
Profiler	`profilerEnabled` + optional line % gutter; re-run when enabling
Debug	Inline end-of-line log/error chips (`inlineDebugEnabled`)
Pins	Chart markers + □ gutter (`debugPinsEnabled`); **Alt+P**
Ruler	80-column guide (`editorRulerEnabled`)
Hamburger → Open in new tab	Editor in a full browser tab (vs **New window** popup)

Details: Debugging.

Stats strip

Bottom of tabbed editor (`data-testid="axis-editor-stats"`):

Field	Meaning
Pos L:C	Cursor line:column (1-based)
Ln	Total lines in document
Words / Chars	Document stats
Diagnostics badge	Error/warning count from last run — click jumps to first
Problems	Expand/collapse problems list
Colors	Color chips / picker / converter for hex and `color.*`
Symbols	Insert TV-editor-safe arrows, box drawing, marks, spaces, and chart emoji (raw, quoted, or `plotchar`)
wrap	Soft wrap toggle (`editorWrapEnabled` , default on)

Symbols & emoji

Status-bar **Symbols** (`data-testid="axis-editor-symbols"`) opens a searchable catalog of glyphs used in Pine Script™ `plotchar` / `label.new` / table text.

- **Mono-safe** — ~1 cell in IBM Plex Mono / the TV editor (box drawing, `▲▼●◆✓★` , arrows).
- **Wide / chart-only** — official `□□` and emoji (`□□□`). Filter with **Mono-safe**.
- Click inserts at the cursor (replaces the selection). Right-click copies.
- Insert mode: raw glyph, quoted string, or a `plotchar(cond, "mark", "…", location.belowbar)` snippet.

Sources: TradingView *Exploring Unicode*, Text and shapes, box-drawing U+2500, PineCoders conventions.

Highlighting

`pyne-language.ts` is a stateful stream tokenizer: `//@` annotations, `//` and `/* */` (including across lines), quoted and `"""` / `'''` strings with `\\n` escapes, `#RRGGBB(AA)` colors, namespaces (`ta.` , `label.new`), types (`int` / `float` / `string` , `series` / `simple` in qualifier position, and declared UDTs / enums as `typeName` , **bold** in the void theme), control vs definition keywords, and function names before `(` . Format leaves continuation lines of an open string or block comment untouched.

Four information surfaces

Surface	When	Answers
Hover	Rest on a symbol	Keyword, type, series, namespace, builtin, user input/function, or hex color
Param checklist	Inside a call	Used / current / unused args with type + default
Pre-eval	While typing / Save	What is wrong *before* Run
Debug chips	After Run, **Debug** on	Last-run `log.*` / errors on that source line

Hover + param checklist are writing-time docs — not a TradingView® debug API. Pre-eval marks the buffer; chips still read **last run**. Details: Debugging.

Diagnostics & Problems

Pre-eval (as you type) — `schedulePreeval` waits the **idle delay** after the last keystroke (default `PREEVAL_IDLE_MS` = 1000 ms; tab switch uses a shorter beat). Marks clear while typing unless **Clear marks while typing** is off. Local parse/lint (`localPreevaluate`) plus `POST /lsp/diagnostics` when engine=server and remote lint is on. Severity **error** disables Run when **Block Run on errors** is on (topbar, editor, Mod-Enter, command palette). Warnings and typos do not block.

Every lint / hover / complete / underline / chip option lives in **Settings → Editor** (command palette: *Open Editor Settings*). Defaults match the previous hardcoded behavior.

Local checks: unclosed `(` / `[` / `{` and strings, missing entry point, `study()` → `indicator()` / `strategy()`, bare `security()` → `request.security`, duplicate `indicator()` / `strategy()` / `library()`, unknown builtin members (`strategy.etry` , `plt` → “did you mean …?”). User functions and import aliases are not flagged.

Merge: remote style-noise C001–C004 is dropped; remote syntax errors win the same span; local unknown-member marks still show when remote is down. Hover tooltips use the diagnostic message (including the suggestion).

After a run, `diagnosticsFromLastRun(lastRun, doc)` still builds ranges from engine `error` , `meta.errors` / `diagnostics` , and error logs with line refs (`line 12:` , `L12:` , …). Live pre-eval for the current buffer is listed first; last-run engine messages are kept when they add detail (dropped only when the buffer no longer matches the stamped run source).

- CodeMirror: wavy underlines, severity gutter, hover tooltips
- **Problems** panel: clickable rows → `scrollToLine` / `jumpToDiagnostic` ; origin **`pre-eval`** vs **`run`** when the diagnostic has a source
- Empty / pending pre-eval stays **No problems** (no flicker)
- Unit tests: `tests/editor-diagnostics.test.ts` , `tests/editor-problems.test.ts` , `tests/preevaluate.test.ts`

80-column ruler

`columnRulerExtension` paints a dashed guide at character column **80** (measured via CM coords). Toggle **Ruler** or command palette “Toggle Editor Ruler”.

Git sync bar

Tab toolbar **Pull / Push / status** (`EditorGitBar`):

- Uses active **storage** plugin (`local` | `cloud` | `git`)
- Push → `writeScript` (git commits when storage is git)
- Pull → optional plugin `sync('pull')` + list/reload bound library id
- Config stays in Settings / Script Library (no second credentials UI)

Completion corpus

`src/editor/data/pyne-builtins.json` is synced from PYNE (`scripts/sync-pyne-builtins.sh`). Offline completion/hover fall back to this catalog; optional remote LSP when the Pro API is available.

Library load

`loadLibraryDoc(doc, name)` on `editorRef` injects storage reads into the active tab.

Internals

editorRef pattern

App holds `{ getDoc, setDoc?, loadLibraryDoc?, scrollToLine?, jumpToDiagnostic?, getCursor?, insertAtCursor? }` populated by CM mount—avoids full-doc store writes every keystroke (doc also mirrored to `EDITOR_DOC_KEY`).

Persistence

Key / field	Role
`store.editor` + `panelChrome.editor`	Dock geometry, open, mode
`pynescript.axis.editor.doc`	Draft text
`profilerEnabled` / `inlineDebugEnabled` / `debugPinsEnabled` / `editorRulerEnabled` / `editorWrapEnabled`	Editor tools

Invariants

1. Empty doc Run is a no-op at topbar/editor handlers.
2. Popout does not own PaneManager.

3. Reattach restores shared doc into docked CM.
4. AXIS ≠ engine: editor never fully evaluates Pine itself; pre-eval is parse/lint only.
5. Debug/Pins still read **last run**; static diagnostics also come from pre-eval.

Failure modes

Symptom	Mitigation
Two editors diverge	Prefer bridge doc events; reattach
Detached but no window	Popup blocked; allow popups; reattach
Lost text on crash	editor.doc localStorage; library Save / git push
Blank CM (no lines)	Dock column height chain; editor must fill strip
No chips/pins	Enable Debug/Pins; logs need line and/or bar_index

See also

- Debugging — Debug vs Pins, Scriptlogs, Profiler
- UI shell — Topbar, docks, command palette
- Indicators
- Script library

DEBUGGING

Scriptlogs, Profiler, Debug chips, chart Pins, diagnostics, and System Logs.

DEBUGGING

AXIS separates **script-authored diagnostics** (Pine `log.*`, engine errors, line timing) from **shell telemetry** (load, stream, engine connection).

Script Logs panel for last-run Pine log output

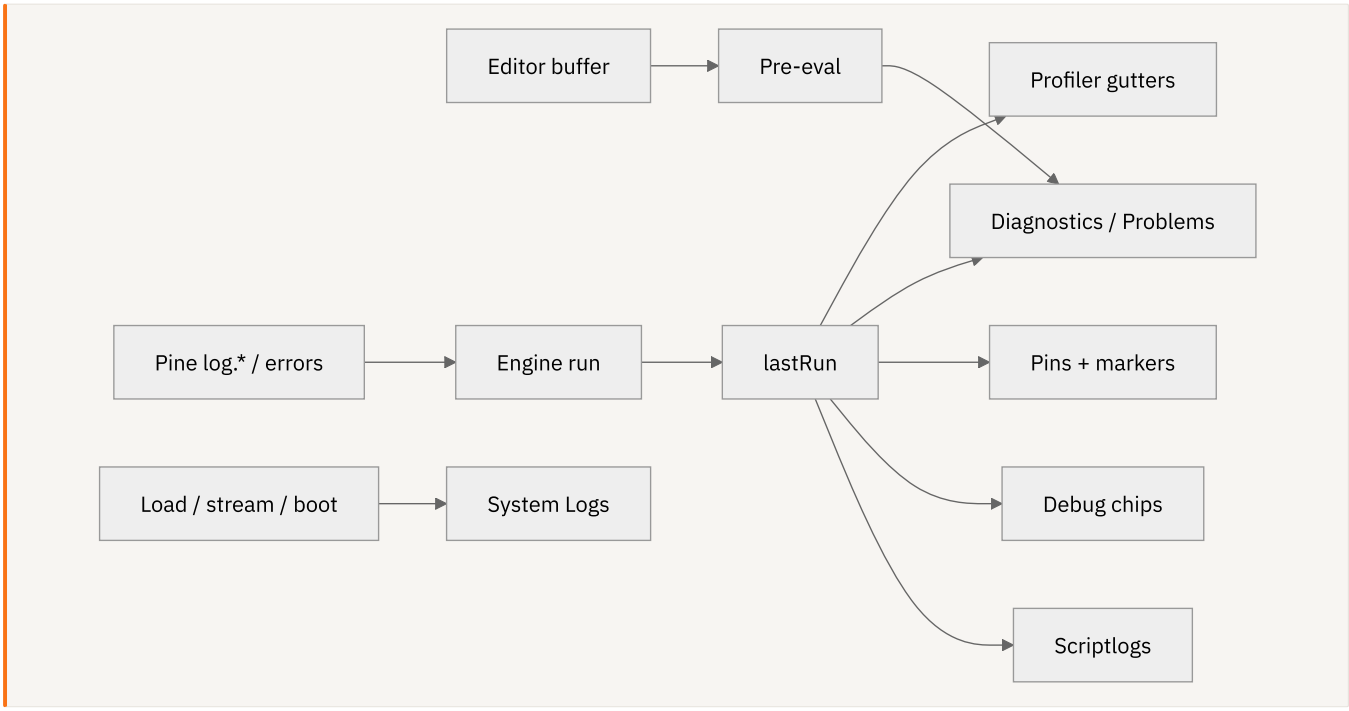
Writing-time and last-run info share four surfaces — hover, the param checklist, pre-eval, and debug chips — so each answer stays on its own chrome.

Abstract

Surface	Source of truth	Open / toggle
Hover	Builtin / identifier docs (local corpus or LSP)	Rest cursor on a symbol
Param checklist	Call signature + remaining named args	Type inside `plot(` / `ta.sma(` / …
Pre-eval / Problems	Parse/lint ∪ last-run engine errors	Underlines + Problems (`pre-eval` vs `run`)
Scriptlogs	`store.lastRun` → `normalizePyneLogs`	Editor header **Scriptlogs**
Profiler	`profilerEnabled` + `profile.lines`	Editor header **Profiler**
Debug chips	`inlineDebugEnabled` + last-run logs with **line**	Editor header **Debug**
Pins	`debugPinsEnabled` + logs with **bar_index / time**	Editor header **Pins** · **Alt+P**
System Logs	`store.logs` ring buffer	Status bar / logs strip

Module	Role
`ui/ScriptLogsPanel.tsx`	Floatable last-run `log.*` list
`results/pyne-logs.ts`	Normalize / filter / TSV
`results/profiler.ts`	Line % profile
`results/inline-debug.ts`	Annotations + pinable helpers
`results/debug-pins.ts`	Bar pins, markers, `countDebugPins`
`editor/inline-debug.ts`	CM chips, pin gutter, line flash
`editor/diagnostics.ts`	Error underlines / gutter / jump
`ui/EditorProblems.tsx`	Problems list UI
`chart/manager-access.ts`	`applyDebugPinsToChart`, `jumpToDebugPin`

Conceptual model



Invariant: Scriptlogs / Debug / Pins never write into `store.logs`.

Four information surfaces

Surface	When	Answers
Hover	Idle on a symbol	What is this builtin / identifier?
Param checklist	Inside a call	Which arguments remain?
Pre-eval	While typing / Save	What is wrong *before* Run?
Debug chips	After Run, **Debug** on	What did this **source line** log or error?

Hover and the param checklist never invent TradingView® host APIs — they read PYNE builtins / local docs. Pre-eval marks the buffer; Debug chips still read **last run** only.

Debug vs Pins

	Debug	Pins
Question	What happened on this **source line**?	Which **bar** was that on the chart?
Requires	Line ref (`line 12:` , structured `line` , ...)	`bar_index` and/or `time` / `bar_time`
UI	End-of-line chips + optional level gutter	Chart circle markers + □ editor gutter
Click	Pin-able chips jump if bar info present	Chip / □ / Scriptlogs → chart jump + line flash
Store	`inlineDebugEnabled`	`debugPinsEnabled`

Workflow

1. Add `log.info` / `log.warning` / `log.error` (or rely on engine errors) with **line** and ideally **bar_index**.
2. **Run**.
3. Toggle **Debug** to see chips; **Pins** to mark bars (count shows on the Pins button).
4. Click a pin-able chip or □ → `jumpToDebugPin` (crosshair + scroll); source line flashes (`cm-debug-pin-flash`).

Both can be on at once. Pins without Debug still show □ gutter and chart markers.

Scriptlogs

Panel for **script** logging from the last run.

Pine call	Panel level
`log.info`	info
`log.warning`	warning
`log.error`	error

Open: Editor → **Scriptlogs** (`axis-btn-scriptlogs`). Pin-able rows jump to the chart when bar time/index is known.

Editor Profiler

Toggle **Profiler** (`axis-btn-profiler`):

- Persists `profilerEnabled`
- May send `profiler: true` to the engine
- Header shows last-run ms
- Gutter shows line `%` when `profile.lines` is present

Diagnostics & Problems

Independent of Debug chips: structural **error reporting** from pre-eval and the run payload.

- Pre-eval (`preeval*`) marks parse / lint issues as you type; last-run adds engine `error` , `meta.errors` , `diagnostics` , error-level logs
- CodeMirror underlines + severity gutter (larger hit target than line numbers)
- Stats strip badge (`axis-editor-diag-count`) and **Problems** list (`axis-editor-problems`)
- Each row shows origin **`pre-eval`** or **`run`** when the diagnostic has a source
- Click → jump to line/range
- Empty / in-flight pre-eval stays **No problems** (no checking flicker)

Scriptlogs vs System Logs

	Scriptlogs	System Logs
Source	Script `log.*` on last run	AXIS shell events
Store	`lastRun`	`store.logs`
UI	Floatable Scriptlogs	Strip above status bar
Control	Editor header	Status bar

Persistence & test ids

Key	Role
`panelChrome.scriptlogs` / `editor`	Panel chrome
`profilerEnabled`	Profiler
`inlineDebugEnabled`	Debug chips
`debugPinsEnabled`	Chart + pin gutter

testid	Role
`axis-btn-scriptlogs`	Open Scriptlogs
`axis-btn-profiler`	Profiler
`axis-btn-inline-debug`	Debug
`axis-btn-debug-pins`	Pins
`axis-debug-pin-count`	Pin count
`axis-editor-diag-count`	Diagnostics badge
`axis-editor-problems`	Problems panel

Unit coverage: `tests/pyne-logs.test.ts`, `tests/profiler.test.ts`, `tests/inline-debug.test.ts`, `tests/debug-pins.test.ts`, `tests/editor-diagnostics.test.ts`.

See also

- Editor
- Charting — markers and jump
- UI shell

UI SHELL

AXIS chrome: topbar, watchlist, status bar, settings, plugin manager, logs, and layout chrome around the chart.

UI SHELL

The shell is the non-canvas chrome: navigation of axes, market selection, modals, and status. It binds human gestures to store + plugins without owning calculation.

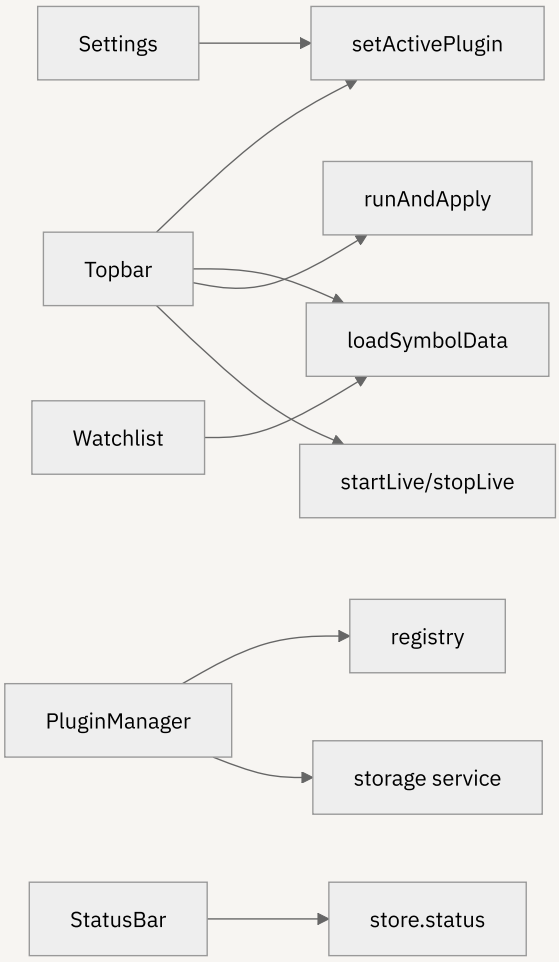
AXIS topbar groups: market, venue, engine, panels, Studio

Command palette filtering theme commands

Abstract

Component	Path	Role
Topbar	`ui/Topbar.tsx`	Axes pickers, Load/Run/Live, theme, modals
Watchlist	`ui/Watchlist.tsx`	Symbols, quotes, click-to-load
StatusBar	`ui/StatusBar.tsx`	Status, engine/storage badges, trade snippet
StudioHost	`ui/studio/StudioHost.tsx`	Full-page Runtime / Wire / Settings / Workers / Plugins
SettingsPage	`ui/settings/SettingsPage.tsx`	Appearance, chart, live, keys, editor, theme
PluginsPage	`ui/plugins/PluginsPage.tsx`	Catalog / install / library
WirePage	`ui/wire/WirePage.tsx`	Compose-recipe wiring (source × stream × engine × storage × dataset)
SystemLogs	`ui/SystemLogs.tsx`	Ring buffer of `store.logs`
ResultsPanel	`ui/ResultsPanel.tsx`	Bottom drawer (see results doc)
Icons	`ui/icons.tsx`	Lucide wrappers
ResizeHandle	`ui/ResizeHandle.tsx`	Panel widths/heights

Conceptual model



Topbar responsibilities

Grouped left→right (`axis-tb-group`, `data-tb-group`):

Group	Contents
brand	Logo + AXIS
market	Symbol, Interval, Type (`TopbarField`), Compare
data	Source (+ CSV upload), Load, Reload (icon-only)
compute	Engine, Stream, **Run**, Live, Replay
layout	Multi-chart layouts + recipes
panels	List, Editor, Library, **Data** (Source Manager), **Scripts**, Layers, Alerts, Data window, Inputs, Results
system	Studio (Runtime / Wire / Settings / Workers / Plugins), Theme (`margin-left: auto`)

Control	Effect
Symbol / interval	Store + Load (Enter / blur reload)
Source / Stream / Engine	`setActivePlugin` + side effects
Chart type	`setChartType` (candles, HA, …)
Compare	Second-symbol overlay
Load / Reload	One-shot historical fetch → chart
Data panel	Data Source Manager — background deep history
Scripts panel	Applied indicators/strategies list
Run	Editor doc → `runAndApply`; **accent only while `status === 'running'`**
Live	multiplex start/stop (stops Replay)

Chart themes

Settings / command palette expose **ten** curated presets (void dark/light, classic, mono, obsidian, graphite, pacific, dusk, porcelain, parchment). High-end soft surfaces — not neon high-contrast. See `src/theme/presets.ts`. | Replay | Bar replay over loaded bars | | Panel toggles | `isPanelOpen` / dual-write chrome | | Settings / Plugins | Modal open | | Detach editor | Bridge + popout |

Integrated labels: `ui/TopbarField.tsx` (`.axis-tb-field*`).
`catalogTick` forces memo re-list when plugins install/remove.

Command palette

Ctrl/Cmd+K — `CommandPalette` + `command-registry.ts`: panels, theme, layouts, Run, symbol focus, editor toggles (ruler, debug, pins, profiler), jump to line, git push/pull, save library. Palette entries render any recorded shortcut chords.

Keyboard shortcuts

Global chord dispatch hub (`src/ui/shortcuts/`): capture-phase handling with a **dialog-skip guard** so shortcuts never fire while a modal is open. Wired to three scopes:

- **App-level** — palette, Run, panel toggles, replay, layouts
- **Chart** — keymap routed through the dispatch hub (cancel draft, layouts, replay)
- **Editor** — `src/editor/cm-line-ops.ts` `Prec.high` line ops: duplicate line, toggle comment, indent/unindent, delete line

Recorder and discovery:

- **Recorder**: Settings → Keyboard → record a chord; per-user overrides persist on the app store
- **Shortcuts modal**: lists default bindings with live rendering
- **Conflict flags**: scope-aware (`app.escape` vs `chart.cancel-draft` layer by design — no false conflict)

Alerts panel

Dockable **Alerts** (`src/alerts/*` + `AlertsPanel.tsx`): local-first book for **price**, **% change**, **drawing touch**, **indicator plots**, **Pine `alert()` / `alertcondition()`**, and on-chain kinds. Delivery: browser Notification, optional HTTPS webhook, optional L2 webhook. Live ticks evaluate price/drawing/plot kinds; strategy/indicator runs evaluate Pine `alert()` events (last bar). Continuous server-side arming is not required for session use.

Workspace snapshots

`storage/workspace-snapshot.ts` + optional chrome menus capture layout/plugin prefs for restore (distinct from OHLCV — bars still re-fetch on Load).

Panel docks (side-by-side)

Left/right docks with **2+ open panels** lay out **horizontally** (row), not stacked under each other:

- Example: **Indicators** left of **Editor** on the right strip
- Column width = **sum** of panel widths (capped)
- Each panel keeps its own width (resize handle on chart-facing edge)

- Bottom dock still stacks vertically

See `ui/panels/dock-layout.ts`, `FloatableShell.tsx`.

Mobile shell (phones / tablets)

`src/ui/mobile/MobileShell.tsx` + `src/ui/responsive.ts` — reactive `phone | tablet | desktop` mode from viewport + `pointer: coarse` (desktop ≥1024px is unchanged):

- **Phones (≤767px)**: compact header (brand · venue · symbol · interval) + bottom tab bar (**Chart · Panels · Editor · Studio · More**) replace the Topbar/StatusBar. Panels and More open as **nav overlays**.
- **Panel sheets**: docked/floating panels render as full-viewport sheets — one at a time, dismiss via **swipe-down** on the title bar. Desktop panel geometry is bypassed, never overwritten, so phone↔desktop switches preserve layouts. Phones boot chart-first.
- **Force-single chart**: 2H/2V/4 layouts collapse to the active slot with a slot-switcher chip row; other slots stay preserved in `store.chartLayout`.
- **Tablets (768–1023px)**: desktop shell with a horizontally scrollable topbar.
- Dialogs become **bottom sheets**; safe-area insets, `100dvh` root, ≥44px touch targets.

Watchlist

Watchlist rail with BTCUSDT selected and live 24h quotes

- Default universe includes majors (`BTCUSDT`, …)
- `refreshSec` clamped 5–120
- Source-aware labels (e.g. CSV mode)
- Click sets symbol and loads through active source

Status bar

Connection HUD: LIVE, tick, engine, and source/stream/storage chips

Single row:

- **Left — Connection HUD** (`ConnectionHud`, `data-testid="axis-connection-hud"`)
  - LIVE badge (off / connecting / live / err)
  - Tick pulse (fixed-width price + age; ping does not resize layout)
  - Engine chip (id · interpret/compile · latency)
  - Plane chips SRC / STR / ENG / STO with transport badges **WS / REST / LOCAL / BROKER**
  - Source↔stream pairing warning when mismatched
- **Right** — `status` / `statusMessage`, strategy snippet, log toggle, bar/ind/pane counts

Telemetry is ephemeral (`store.telemetry`); HUD compact mode and live prefs persist via settings.

Live settings (Settings dialog)

Setting	Default	Effect
Enable live stream	on	`live.preferAfterLoad` → multiplex after Load and on boot when bars exist
Indicator re-run	every-tick	or `bar-close` only
Compact connection HUD	off	hide plane chips

Logs

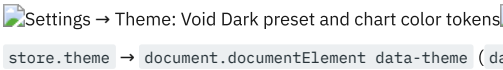
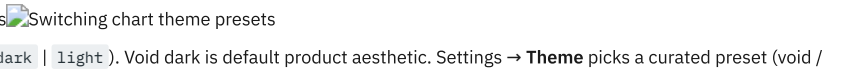
System Logs strip

- `appendLog(level, message, source)`
- Cap `MAX_LOGS` (500)
- Not persisted
- Boot messages, pyodide ready, live errors

Scriptlogs & Profiler

Script `log.*` output and editor line-timing live on a separate path — see Debugging. Controls sit on the **editor header** (not the topbar) and do not write into `store.logs`.

Theming


Settings → Theme: Void Dark preset and chart color tokens 
Switching chart theme presets

`store.theme` → `document.documentElement data-theme` (`dark` | `light`). Void dark is default product aesthetic. Settings → **Theme** picks a curated preset (void / classic / mono / boutique) and edits chart tokens live.

Accessibility / test hooks

E2E selectors use `data-testid` on critical controls (`axis-btn-load`, `axis-manager`, …) per `TESTING.md`.

Invariants

1. Shell never calls Python.
2. Plugin lists always come from registry facades (`listSources`, …).
3. Modal settings persist only on Save (not on every keystroke of endpoint field until save).
4. Live off leaves historical bars intact.

Failure modes

Symptom	Shell-side check
Pickers empty	Builtins not registered at boot
Live ignores click	Stream `none` or multiplex error in logs
Settings probe fails	Network / CORS—not topbar bug

See also

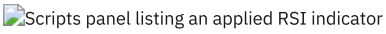
- Manager and settings
- Store
- Architecture overview

SCRIPTS (INDICATORS & STRATEGIES)

AXIS applied-scripts UI: runAndApply pipeline, Scripts panel, overlay routing, and engine binding.

SCRIPTS (INDICATORS & STRATEGIES)

In AXIS, **scripts applied to the chart** may be Pine `indicator()` or `strategy()` definitions. The product panel is labeled **Scripts** (panel id remains `indicators` for workspace compatibility). The module boundary is `indicators/runner.ts` plus list UI—not the PYNE evaluator.

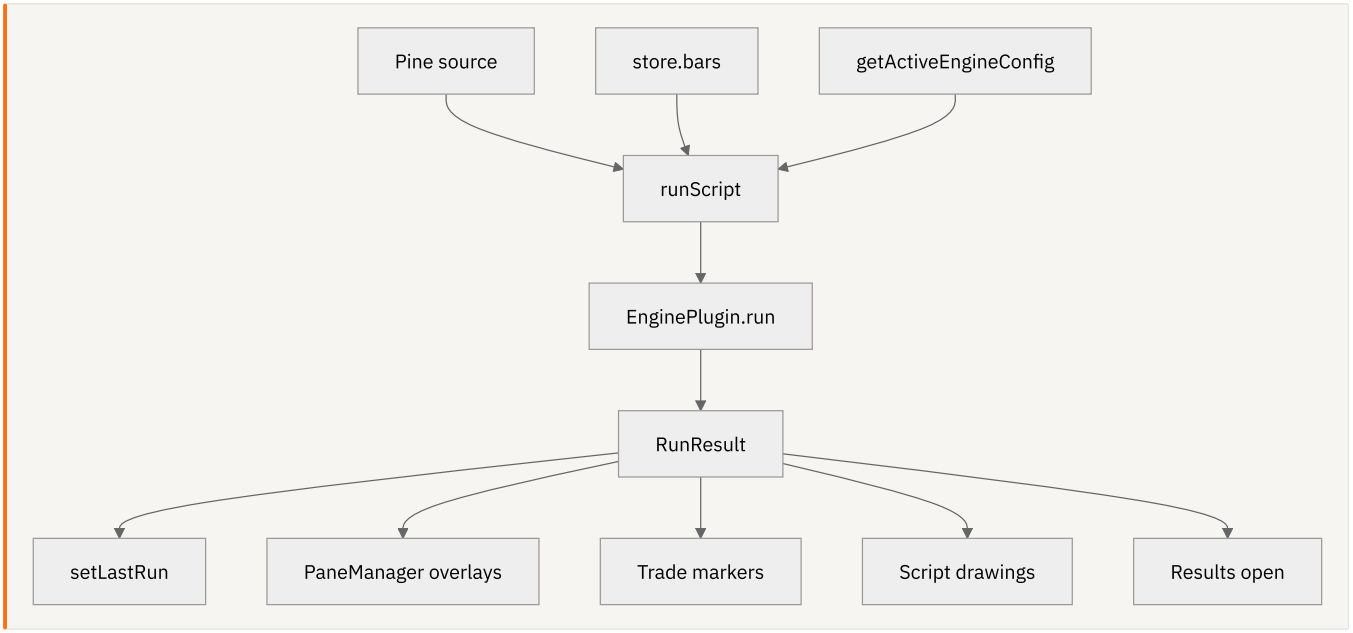


Abstract

Piece	Role
`runScript`	Engine invocation + status/timeouts
`runAndApply`	Chart + lastRun + results open; may `addIndicator`
`probeEndpoint`	Settings health check helper
Scripts panel (`IndicatorPanel`)	List / toggle / colors for `store.scripts`
`IndicatorCard`	Per-script UI chrome

Language semantics: PYNE runtime.

Conceptual model



Interface surface

runScript options

Option	Default	Meaning
`silent`	false	Quiet status; live re-run logging
timeout	derived	Interactive: scales with bars (60–180s); silent 45s

runAndApply options

Option	Default	Meaning
`openResults`	`!silent`	Open results drawer
`indicatorId`	optional	Update existing store indicator

Overlay routing

```
overlay = resolveOverlayFlag(meta.overlay, script_type)
  // explicit false → sub-pane
  // explicit true  → price
  // missing: indicator → sub-pane; strategy → price

if overlay && seriesWouldHideOnPrice(plots, bars):
  overlay = false   // RSI / scaled oscillators on BTC scale
paneId = overlay ? 'price' : `ind_<scriptId>`   // one sub-pane per script
```

Each non-overlay script gets its own sub-pane (`ind_<id>`), not a shared `indicator` host. Re-runs keep the script's private pane; legacy shared `indicator` panes migrate on re-run. Removing a script clears only that owner's series and destroys the sub-pane only when empty.

Persist + reopen

Applied scripts (`store.scripts` with full `code` , `paneId` , colors, inputs, strategy props) are written to localStorage with the rest of durable UI state. On boot they hydrate via `sanitizePersistedScripts` . After each successful history paint (`load-symbol` , DSM cache apply), `reapplyChartScripts` silently re-runs every visible script in plot-dependency order so the chart matches the saved set without manual re-run.

Detach path: `detachIndicatorFromChart` (Scripts panel, Layers, pane badges) — owner-scoped overlay clear, orphan pane sweep, no sibling wipe.

Prefer `indicator(..., overlay=false)` and raw `plot(rsi)` for oscillators — not `strategy(..., overlay=true)` with `rsi * 0.01` on price.

Series selection

Prefer `result.series` entries whose keys do not start with `_` / `__` . Fallback: single `plots[]` array as one line named from `script_name` .

Colors: `plot_meta[k].color` or `PLOT_PALETTE[i]` . Pine Script™ `plot.style_*` maps to LWC series kinds (line, stepline, histogram/columns, area, circles) — see Charting — plot style parity.

Indicator panel & pane badges

- Side list of `store.scripts` (`Indicator` : id, name, code, paneId, visible, plots)
- Visibility toggles affect display; re-run is still engine-driven
- Chart pane chips mirror settings / eye / re-run / remove (see Charting)

Active engine binding

```
// plugins/active.ts pattern
getActiveEngine()        // registry lookup by activePlugins.engine
getActiveEngineConfig()  // pluginsConfig + store.endpoint merge
```

Switching engines mid-session does not auto-rerun; operator hits Run.

Live re-run coupling

Streams may set `live.needsRerun` ; multiplex/runner path can call `runAndApply` with `silent: true` so the status bar is not spammed. Failures go to logs.

Internals

Path	Role
`src/indicators/runner.ts`	Pipeline
`src/indicators/IndicatorPanel.tsx`	List UI
`src/plugins/active.ts`	Active engine/source helpers
`src/engines/catalog.ts`	Built-in engines
`src/results/plot-visuals.ts`	Plot style → series kind

Invariants

1. Runner is the **only** chart-facing calc orchestrator (avoid duplicate apply logic in components).
2. Errors still `setLastRun` with `status: 'error'` when possible.
3. Clearing overlays happens before re-adding—no unbounded series leak.
4. AXIS does not interpret Pine; it only maps `RunResult` .

Failure modes

Symptom	Layer
Engine exception	Results error + status
Success empty plots	Script/meta—not runner
Manager null	Chart not mounted; run still sets lastRun
AbortError	Timeout—reduce bars or optimize script

See also

- Charting
- Results and strategy
- ADR-002 / ADR-004

RESULTS AND STRATEGY

UI-side event normalization, trade pairing, metrics, equity curve, and ResultsPanel export pipeline.

RESULTS AND STRATEGY

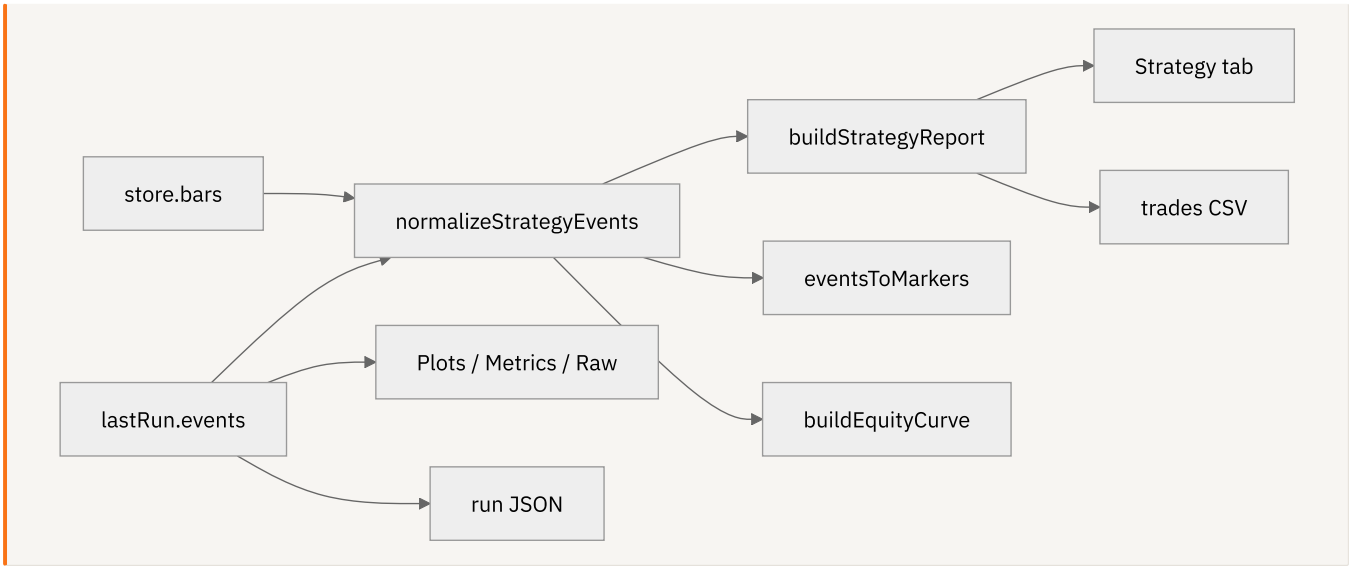
Results UI turns opaque engine payloads into **operator-consumable tables, metrics, and files**. It is a pure-ish viewer layer over `lastRun` + `bars`.

Results overlay with strategy stats, equity curve, and closed trades

Abstract

Module	Role
`results/events.ts`	Normalize heterogeneous event shapes; markers; equity series
`results/strategy.ts`	Shared walker `walkStrategyEvents` → trades, stats, positions, stream; CSV
`results/positions.ts`	Pyramiding-aware position cycles (avg-entry parity with the pyne broker, per-fill P&L)
`ui/ResultsModal.tsx`	Tabs, Events Stream/Open⇄Close switcher, export, jump-to-time
`ui/StrategyReport.tsx`	Stats cards (return %, bars-held, fills), equity curve, trades table, CSV
`ui/StatusBar.tsx`	Compact trade summary

Conceptual model



Event normalization

Engines may emit Pro API parity fields (`kind`, `bar_time`, `direction`, `ohlc`) or legacy UI fields (`type`, `time`, `dir`, `price`). Normalizer:

- Unifies time/price/dir/id
- Optionally fills price from bars when OHLC present/empty
- Can include order-like events for the Events tab

Trade pairing algorithm

Sketch (`buildStrategyReport`):

1. Sort by time.
2. On entry-like kind → open map by `id`.
3. On exit/close-like → match id, else sole open trade.
4. PnL = Δprice × long/short sign.
5. Aggregate win rate, profit factor, max DD on cumulative PnL.

Not modeled: fees, funding, partial fills, portfolio multi-symbol—viewer honesty.

ResultsPanel tabs

Tab	Data source
Events	`normalizedEvents` — **Stream** (chronological) or **Open⇄Close** (position cycles via `results/positions.ts`) view
Strategy	`report.trades` + `report.stats` (single column: stats → equity → trades)
Plots	`series` / `plots` summary
Metrics	meta + context + stats subset
Raw	`JSON.stringify(lastRun)`

Interactions: copy, export, `scrollToTime` on trade rows. Scriptlogs / Debug **Pins** also jump the chart (`jumpToDebugPin`) when bar time/index is present — see Debugging.

Equity

`buildEquityCurve` feeds chart equity pane when runner/strategy path requests it—derived series, not engine-native unless also provided.

Export (ADR-013)

Artifact	Function
`axis-run-*.json`	Full payload
`axis-trades-*.csv`	`tradesToCsv`
Clipboard	CSV when permitted

Invariants

1. Results recompute from `lastRun` reactively; they do not re-query engines.
2. Changing bars without re-run can change normalization fills—re-run after Load.
3. `lastRun` not rehydrated from localStorage (session artifact).
4. Infinite profit factor when no losses but profits exist—display must tolerate non-finite.

Failure modes

Symptom	AXIS explanation
Events > 0, trades = 0	Unpaired or missing prices
CSV disabled	No closed trades
Jump no-ops	Manager unmounted / invalid time

Internals cross-links

Operator narrative: Strategy and results.
Architecture: ADR-013.

See also

- Indicators
- Charting

STORE

AXIS Solid store: AppState shape, activePlugins, persistence rules, logging, and alignment helpers.

STORE

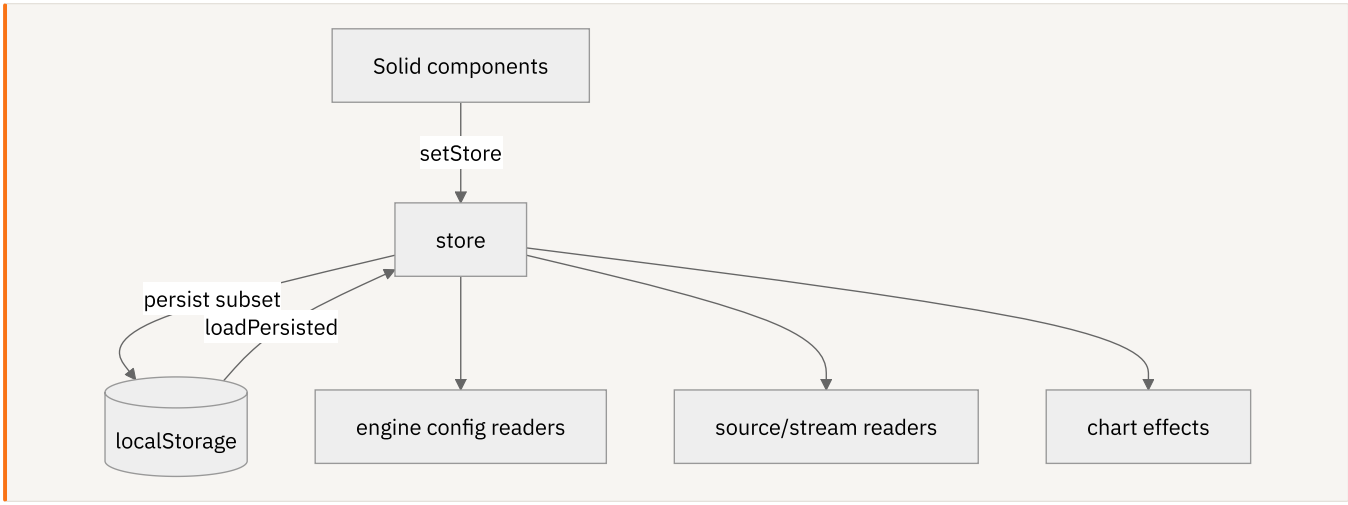
The Solid store is the **control-plane nexus** of AXIS. Plugins read configuration from it; UI binds reactively; persistence snapshots a safe subset to `pynescript.axis.v1`.

Abstract

Export	Role
`store` / `setStore`	`createStore<AppState>`
`persist`	Debounced localStorage write
`setActivePlugin`	Kinded selection + flat field mirrors
`appendLog` / status helpers	Operator-visible telemetry
`STORAGE_KEY`	`pynescript.axis.v1`

Types: `frontend/src/store/types.ts`.

Conceptual model



AppState map

Field	Persistence	Notes
`bars`	no	Session history
`symbol` , `interval` , `exchange`	yes	Market context
`source` , `engine`	yes	Flat mirrors
`endpoint`	yes	Server engine base URL
`activePlugins`	yes	Canonical four-tuple
`pluginsConfig`	yes	Per-plugin fields
`scripts` , `panes`	yes	Indicator list + layout
`live`	partial	streamId important
`theme`	yes	dark/light
`editor` , `watchlist` , panels	yes	Chrome geometry
`stream.status`	yes/volatile	Connection hint
`status` , `statusMessage` , `lastRunMs`	yes*	UX; ok to restore
`lastRun`	**no**	Forced null on hydrate
`logs`	**no**	Forced empty
`profilerEnabled`	yes	Editor profiler (line % when engine provides profile)
`inlineDebugEnabled`	yes	End-of-line Debug chips from last-run logs
`debugPinsEnabled`	yes	Chart pin markers + editor pin gutter
`editorRulerEnabled`	yes	80-col guide (default **on**)
`editorWrapEnabled`	yes	Soft line wrap (default **on**; stats strip **wrap** toggle)
`panelChrome` / `alertsPanel` / `layerPanel` / `dataViewPanel`	yes	Dock chrome (side-by-side left/right)
`chartLayout` / `savedLayouts`	yes	Multi-chart grid + named layouts
`compare`	prefs yes	Second-symbol overlay (bars ephemeral)
`drawingTool`	reset cursor	On hydrate
`drawings` / `drawingPrefs` / `drawingUi`	yes	User annotations + toolbar prefs
`selectedDrawingId`	**no**	Ephemeral selection

*Status fields may restore but boot also appends fresh logs.

Defaults (product)

Representative defaults from store:

Key	Default
symbol	`BTCUSDT`
interval	`1d`
source / active source	`binance-rest`
stream	`binance-ws`
engine	`server`
storage	`local`
endpoint	demo/API host or localhost depending on build era
editor width	460 docked open
watchlist	majors, 15s refresh

Treat demo endpoints as **overridable**—operators should set local URLs in desk topology.

setActivePlugin

```
setActivePlugin(kind, id):
  activePlugins[kind] = id
  if source → source = id
  if engine → engine = id
  if stream → live.streamId = id
  persist()
```

Prevents registry/store split brain (architecture overview).

Logging

```
appendLog(level, message, source?)
levels: info | ok | warn | error
MAX_LOGS = 500
```

`setStatus` maps app status to log levels for important transitions.

Legacy dual path

Vanilla `state.js` remains for legacy shell with the same storage key family. Product AXIS uses Solid store only—do not dual-write new features to `state.js` unless maintaining legacy.

Invariants

1. Hydrate never restores `lastRun` or `logs`.
2. Persist strips `bars`.
3. Key migration is read-repair (state namespaces).
4. `pluginsConfig` keys prefer `` `${kind}:${id}` ``.
5. Store is not a plugin registry—ids must exist in registry to function at runtime.

Failure modes

Mode	Behavior
JSON parse fail	Defaults
localStorage throws	No-op persist
Unknown plugin id	UI may show id; run/load fails at getActive*
Oversized drawings	Quota risk—export/clear

Testing notes

- Import `tests/setup.ts` for localStorage stubs
- Unit coverage includes store + migration patterns
- Do not require real IDB in pure store tests

See also

- State namespaces
- ADR-011
- UI shell

WORKER

Cloudflare Worker for AXIS: /api/run proxy, keys, scripts, usage, and Durable Object stream relay.

WORKER

Abstract

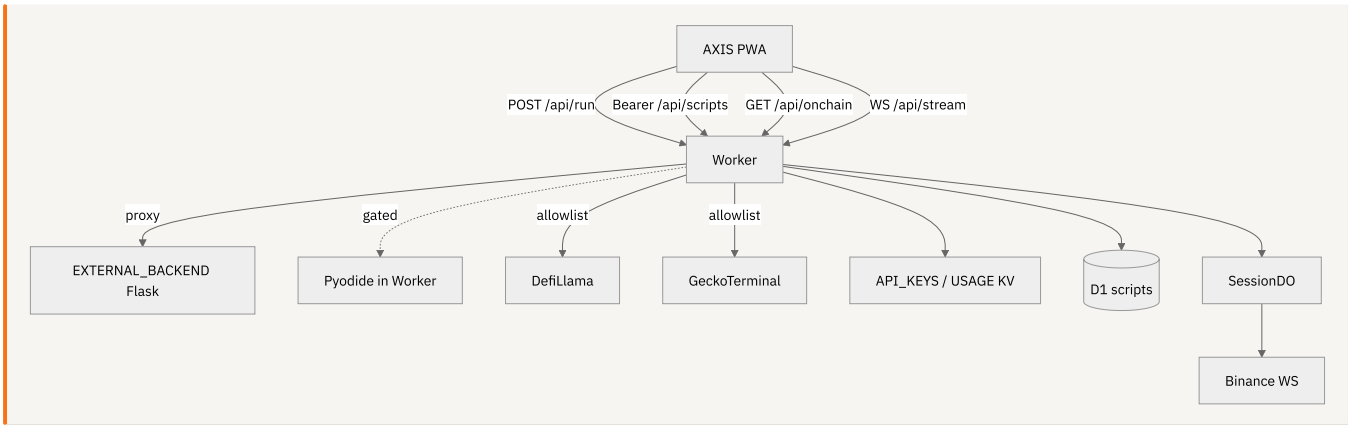
The AXIS Cloudflare® Worker (`worker/`) is the **edge data plane** for the PWA: JSON APIs, optional script library (D1), API keys (KV), usage meters, on-chain proxy, and a WebSocket session relay (Durable Objects). It is **not** a full Pine Script™ interpreter in production today.

Honest runtime fact: `POST /api/run` **proxies to** `EXTERNAL_BACKEND` (typically Flask) unless `PYODIDE_IN_WORKER=enabled` and the in-worker path succeeds. In-worker Pyodide is **planned / feature-gated** — see Runtime and `worker/RUNTIME.md` .

Security (2.0.1+): gated `/api/run` (auth when `API_KEYS` / `REQUIRE_RUN_AUTH`), rate limits + body caps, fail-closed Worker auth when D1 is bound without `API_KEYS` KV, product-scoped CORS (not open `*.pages.dev`), OAuth env client ids over body `clientId` . See Auth and CORS.

Operator path: AXIS CLI — `axis setup` · `axis deploy worker` · `axis health` .

Conceptual model



Infrastructure names (frozen)

Surface	Value
Wrangler Worker script	**`worker-axis`** — canonical `https://worker.axis.hoox.sh`
Pages project	**`axis`** — canonical `https://axis.hoox.sh`
npm package	`axis-worker`
Health JSON `service`	`worker-axis`
Product brand	**AXIS**

Pages `--project-name` is `axis` ; custom domain is `axis.hoox.sh` . Worker URL is `https://worker.axis.hoox.sh` .

Interface surface

Path	Method	Role
`/` , `/health`	GET	Health + feature flags (`scripts` , `d1` , `keys` , `onchain`)
`/api/run`	POST	Run script (proxy / gated Pyodide)
`/api/keys`	GET/POST	Validate / create keys (`X-Admin-Token` for create)
`/api/usage`	GET	Usage stub / KV-backed counters
`/api/scripts` …	CRUD	Script library (Bearer)
`/api/onchain/…`	GET	Public on-chain proxy (DefiLlama + GeckoTerminal allowlist; see below)
`/api/stream`	WS	Session DO relay

On-chain proxy routes (`worker/src/onchain.ts`)

Allowlisted **GET** only (public, no API key). Other `/api/onchain/*` paths return 404.

Worker path	Upstream	Cache TTL
`/api/onchain/health`	local (providers + cache size)	—
`/api/onchain/llama/protocols`	`https://api.llama.fi/protocols`	~10 min
`/api/onchain/llama/protocol/:slug`	`https://api.llama.fi/protocol/:slug`	~2 min
`/api/onchain/gecko/networks/:network/pools/:address/ohlcv/:timeframe`	`https://api.geckoterminal.com/api/v2/networks/.../ohlcv/...`	~60 s
`/api/onchain/gecko/search/pools`	`https://api.geckoterminal.com/api/v2/search/pools`	~120 s

Gecko validation: `network` `^[a-z0-9_]+$` ; `address` EVM `0x` +40 hex **or** Solana-style base58 32–48; `timeframe` `day` | `hour` | `minute` .
OHLCV query pass-through: `aggregate` , `limit` , `currency` , `before_timestamp` . Search: `query` , `network` , `page` , `include` .

Responses may include `X-Axis-Onchain-Cache: HIT|MISS` . Product guide: On-Chain data.

Entry: `worker/src/index.ts` .

Local entrypoints

```
cd frontend/worker
npm install   # or bun install
npm run dev   # wrangler dev → http://127.0.0.1:8787
```

Makefile: `make worker-dev` , `make worker-deploy` .

Implementation status

Feature	Status
`/api/run` → `EXTERNAL_BACKEND`	**Works today**
Admin `/api/keys`	Works (KV or shape-check)
Usage KV increment on run	Works when `USAGE` bound
`SessionDO` + `/api/stream`	Implemented; binding often commented until provisioned
`/api/scripts` + D1 schema	Implemented; memory fallback without D1
Pyodide scaffold	`pyodide_runtime.ts` + flag; **wheel pipeline incomplete**
Response cache for `/api/run`	Deferred

Page map

Page	Topic
Runtime	`/api/run` , proxy, Pyodide plan
Durable Objects	Stream fan-out
Data plane	Scripts, D1, drafts
Auth	Bearer keys, admin token
Bindings	wrangler.toml, vars, provision

See also

- Cloudflare deploy
- Engines
- Worker README: `frontend/worker/README.md`

WORKER RUNTIME

POST /api/run — auth gate, rate limits, size caps, EXTERNAL_BACKEND proxy, feature-gated in-worker Pyodide.

WORKER RUNTIME

Abstract

`worker/src/runtime.ts` implements `handleRun` for **`POST /api/run`**. Execution preference is documented in-file:

1. **`PYODIDE_IN_WORKER=enabled`** → `tryRunInWorker` (`pyodide_runtime.ts`)
2. Else **`EXTERNAL_BACKEND`** set → HTTP proxy to `${EXTERNAL_BACKEND}/run`
3. Else **503** `NO_BACKEND` with a pointer at env vars and `worker/RUNTIME.md`

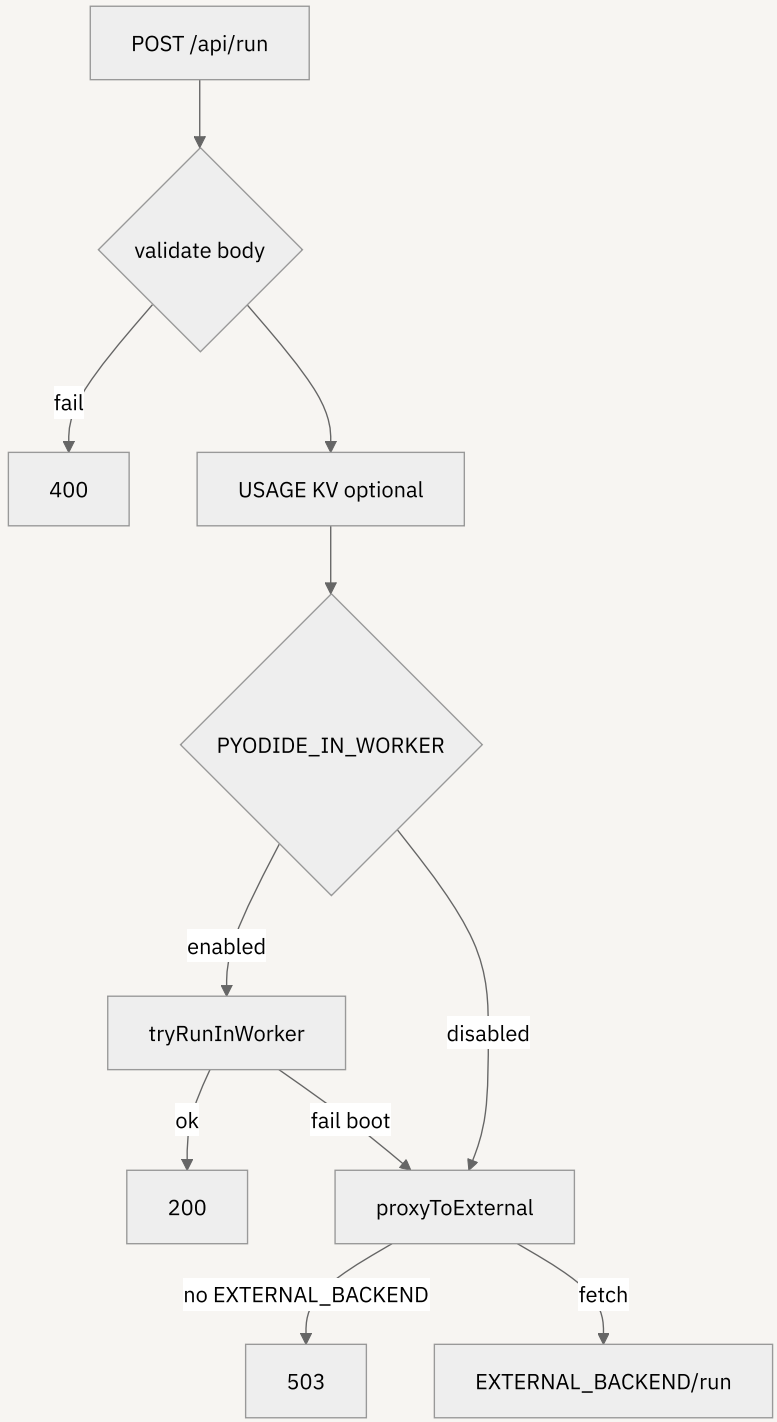
Production reality: ships as a **proxy** to Flask (or another Python host). In-worker Pyodide is a **scaffold**, not a complete wheel-upload pipeline.

Auth & abuse controls (2.0.1+)

Control	Behavior
Auth gate	When `API_KEYS` KV is bound **or** `REQUIRE_RUN_AUTH=1`, `requireApiKey` is mandatory
Rate limit	30 requests / minute per IP (and per key when authenticated)
Script cap	Max **512 KiB** source chars
Bars cap	Max **50 000** OHLCV rows per run
Upstream timeout	60s proxy abort to `EXTERNAL_BACKEND`

Optional `Authorization: Bearer` still increments `USAGE` KV when bound (`usage:<key>`, 30-day TTL).

Conceptual model



Interface surface

Request body

```
{
  script: string;           // required non-empty
  data: Bar[];              // required non-empty OHLCV array
  mode?: 'interpret' | 'compile';
}
```

Invalid bodies → `400` `{ status: 'error', code: 'BAD_REQUEST', message }`.

Success / upstream

Proxy returns upstream status and body verbatim (JSON content-type, CORS origin header). Flask Pro API is the usual upstream.

Error codes (Worker-originated)

Code	HTTP	Meaning
`BAD_REQUEST`	400	Validation / size caps
`NO_KEY` / `INVALID_KEY`	401	Auth required and missing/unknown key
`RATE_LIMIT`	429	>30 runs/min for IP or key
`NO_BACKEND`	503	No external backend and Pyodide path off/failed path to proxy
(Pyodide)	200 body error	`tryRunInWorker` may return `{ status: 'error', error }` JSON

Usage metering

If `Authorization: Bearer …` and `USAGE` KV bound:

```
usage:{key} → count++, TTL 30 days
```

Internals

Path	Role
`worker/src/runtime.ts`	auth gate, rate limit, validate, meter, dispatch
`worker/src/pyodide_runtime.ts`	gated boot + `run_script` stub path
`worker/RUNTIME.md`	architecture plan for wheels / R2 / cold start
`worker/wrangler.toml` `[vars]`	`EXTERNAL_BACKEND`, `PYODIDE_IN_WORKER`, `REQUIRE_RUN_AUTH`

pyodide_runtime scaffold

When enabled:

- Lazy-loads Pyodide from jsDelivr `v0.26.2` (dev fallback).
- Production intent: R2 wheels + module-level cache (`RUNTIME.md`).
- Calls `run_script(script, bars)` via `runPythonAsync` — requires runtime bootstrap that is **not fully wired** like the browser engine's `pynescript_runtime.py` + micropip install.

Do not enable in production without completing the wheel pipeline and load testing CPU/memory limits.

Constraints (from RUNTIME.md)

Resource	Budget
CPU (paid)	30s limit; typical 500 bars ≪ 1s target
Memory	128 MB; Pyodide ~30 MB + wheel ~5 MB
Cold start	~3s first Pyodide boot (plan)

Roll-out flag

`RUNTIME.md` also mentions `RUNTIME_MODE=in-worker` as product language; **code checks** `PYODIDE_IN_WORKER === 'enabled'`. Prefer the code flag when configuring.

Invariants & edge cases

1. **Body stream** — request body is consumed once; proxy re-serializes the already-parsed object.
2. **CORS** — run responses set `Access-Control-Allow-Origin` from `pickOrigin` (see CORS).
3. **PWA path mismatch** — browser `server` engine posts to `{endpoint}/run`; Worker route is `/api/run`. Align via gateway rewrite or endpoint base path convention.
4. **No response caching yet** — README lists cache for identical `(script, data_hash)` as deferred.

Worked examples

Local proxy to Flask

```
# wrangler.toml [vars]
EXTERNAL_BACKEND = "http://127.0.0.1:5002"
PYODIDE_IN_WORKER = "disabled"
```

```
curl -sS -X POST http://127.0.0.1:8787/api/run \
  -H 'content-type: application/json' \
  -d '{"script":"//@version=6\nindicator(\"t\")\nplot(close)","data":[{"time":1,"open":1,"high":1,"low":1,"close":1}]}'
```

Force 503 for missing backend

Unset `EXTERNAL_BACKEND`, keep Pyodide disabled → expect `NO_BACKEND` message referencing `RUNTIME.md`.

Failure modes

Symptom	Fix
503 `NO_BACKEND`	Set `EXTERNAL_BACKEND` or finish Pyodide path
Proxy 502/connection refused	Flask not listening; wrong URL from Worker (use tunnel/public URL in CF, not localhost)
Pyodide enabled but errors	Wheel missing; CDN blocked; incomplete `run_script` bootstrap

See also

- Bindings
- Engines
- Auth (Bearer when gated; metering)
- CORS and origins
- AXIS CLI — `axis deploy worker` · `axis health`

DURABLE OBJECTS

SessionDO WebSocket relay: one Binance upstream per session, fan-out to browser clients.

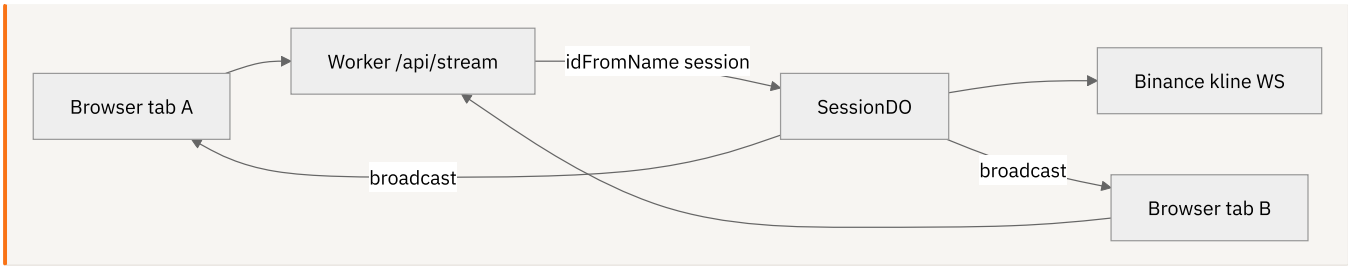
DURABLE OBJECTS

Abstract

`SessionDO` (`frontend/worker/src/durable-objects/session.ts`) is a **per-session WebSocket relay**. Browsers connect to the Worker at `/api/stream` ; the Worker pins a Durable Object by session name and the DO opens **one** upstream Binance kline socket, broadcasting payloads to connected clients.

Goal: survive browser per-origin WebSocket connection limits and share upstream subscriptions.

Conceptual model



Interface surface

Client → Worker

```
wss://<worker>/api/stream?session=<name>&symbol=BTCUSDT&interval=1m
```

`frontend/worker/src/index.ts` :

- Requires `env.SESSIONS` binding; else `503` `NO_DO` .
- `idFromName(session || 'default')` .
- Rewrites request to DO path `/ws?…` and returns upgrade response.

DO HTTP surface

Path	Role
`/ws`	Only path; requires `Upgrade: websocket` (else 426)

Query: `symbol` (default `BTCUSDT`), `interval` (default `1m`).

Client → DO messages (JSON)

Message	Effect
`{ "action": "subscribe", "symbol", "interval" }`	Rebind upstream if changed
`{ "action": "ping" }`	`{ action: "pong", t }`

DO → client

- Raw Binance kline JSON (same as direct venue WS).
- Status: `{ type: 'status', state: 'open'|'closed', url? }`
- Errors: `{ type: 'error', message }`

Example plugin

`frontend/public/plugins/example-cf-do-stream.js` maps kline fields to `Bar` and builds the Worker URL from config `endpoint` .

Internals

Topic	Detail
Upstream URL	`wss://stream.binance.com:9443/ws/{symbol}@kline_{interval}`
Client lifecycle	`acceptWebSocket` ; on last client close → `closeUpstream`
Hibernation	Comment intent: hibernate when empty; implementation uses explicit close
Fan-out	`broadcast` to all `clients`

wrangler binding (often commented until ready)

```
# [[durable_objects.bindings]]
# name = "SESSIONS"
# class_name = "SessionDO"

# [[migrations]]
# tag = "v1"
# new_sqlite_classes = ["SessionDO"]
```

`SessionDO` is re-exported from `src/index.ts` for the Workers runtime class registry.

Invariants & edge cases

1. **Session key** — example plugin uses `symbol@interval` as session name so like-minded tabs share one DO.
2. **No auth on stream today** — treat as public relay of public market data; do not put secrets in query strings.
3. **Upstream hard-coded Binance** — not a generic multiprovider DO yet.
4. **Binding optional in local** — without `SESSIONS`, stream API fails closed with `NO_DO`.

Failure modes

Symptom	Cause
503 `NO_DO`	Binding commented / not deployed
426 expected websocket	Non-upgrade GET
No bars	Upstream block or wrong interval string
Stale symbol	Client did not send `subscribe` after change

See also

- Streams
- Bindings
- Plugin examples

WORKER DATA PLANE

Script library API: /api/scripts, D1 schema, drafts, revisions, and in-memory fallback.

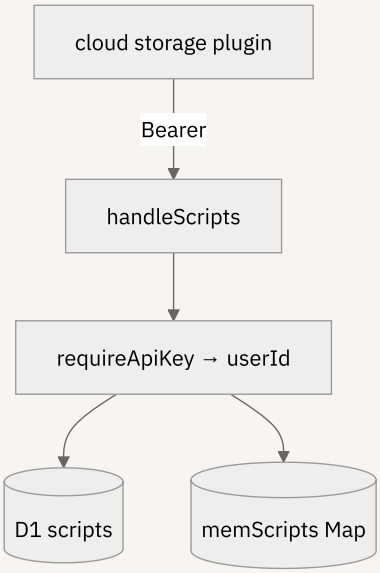
WORKER DATA PLANE

Abstract

The Worker **data plane** for user scripts is `frontend/worker/src/scripts.ts` — REST under `/api/scripts` with Bearer auth (auth). Persistence prefers **D1** (`env.DB`); without D1, per-isolate **in-memory** maps support local `wrangler dev`.

The PWA **cloud** storage plugin is the primary client.

Conceptual model



Interface surface

All routes require valid API key unless noted.

Path	Methods	Behavior
`/api/scripts`	GET	List meta for `userId`
`/api/scripts`	POST	Create script
`/api/scripts/:id`	GET	Read full document
`/api/scripts/:id`	PUT	Upsert; honors `If-Match` revision
`/api/scripts/:id`	DELETE	Remove
`/api/scripts/_draft`	GET/PUT	Per-user draft buffer

Document fields

Field	Notes
`id`	Client or server assigned
`name`, `description`, `path`	Meta
`content`	Pine source
`revision`	Opaque string (`rev_…`) for concurrency
`created_at` / `updated_at`	ms epochs (API may camelCase in JSON)

Concurrency

PUT with `If-Match: <revision>` rejects mismatched revisions (409 path covered in security tests). Fresh writes mint `newRevision()`.

Internals

D1 schema (`schemas/scripts.sql`)

```
CREATE TABLE scripts (
  user_id TEXT NOT NULL,
  id TEXT NOT NULL,
  name TEXT NOT NULL,
  description TEXT,
  path TEXT,
  content TEXT NOT NULL,
  revision TEXT NOT NULL,
  created_at INTEGER NOT NULL,
  updated_at INTEGER NOT NULL,
  PRIMARY KEY (user_id, id)
);

CREATE TABLE script_drafts (
  user_id TEXT PRIMARY KEY,
  content TEXT NOT NULL,
  name TEXT,
  updated_at INTEGER NOT NULL
);
```

Apply:

```
wrangler d1 execute pynescript --remote --file=schemas/scripts.sql
wrangler d1 execute pynescript --local  --file=schemas/scripts.sql
```

Binding name in `wrangler.toml` **must** be `DB` (code uses `env.DB`). Database name can be `pynescript`.

Partitioning

`userId = SHA-256(api_key).hex.slice(0, 32)` — raw keys are not stored as D1 partition ids.

Health feature flags

```
{
  "status": "healthy",
  "service": "worker-axis",
  "features": { "scripts": true, "d1": true/false, "keys": true/false }
}
```

Optional R2

`BUNDLES` R2 is reserved for indicator / pynescript wheels (`RUNTIME.md`). Not required for scripts CRUD.

Invariants & edge cases

1. **Memory store is not multi-isolate durable** — production needs D1.
2. **Drafts ≠ commits** — drafts are crash buffers; library list is separate.
3. **CORS** — scripts handler sets allow headers including `If-Match`.
4. **Cloud plugin dual-write** — browser still saves drafts to local storage.

Failure modes

Code / symptom	Meaning
401	Missing/invalid Bearer
404	Unknown script id for user
409	Revision conflict
Empty list after deploy	Schema not applied / wrong DB id

See also

- Storage plugins
- Auth
- Bindings

WORKER AUTH

API keys (pn_...), fail-closed D1 without KV, admin token, Bearer, ALLOW_OPEN_KEYS, and /api/run gating.

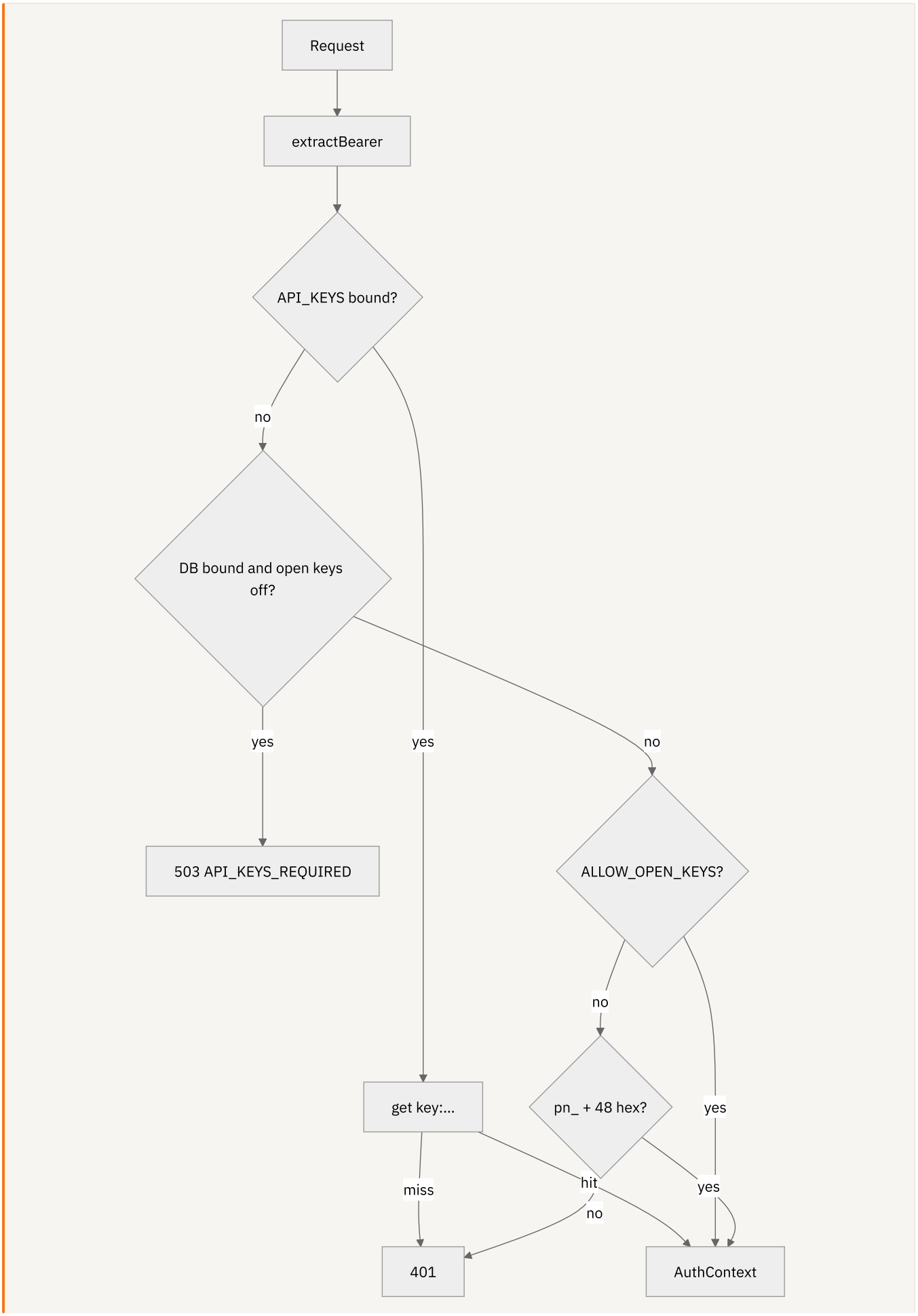
WORKER AUTH

Abstract

AXIS Worker auth is **API-key based**, not session cookies. Keys are created by admins (`X-Admin-Token`), validated via KV when bound, and used as `Authorization: Bearer` on the script library and (when gated) on `/api/run` .

Core helpers: `worker/src/auth.ts` . Key CRUD: `worker/src/keys.ts` .

Conceptual model


Request
extractBearer
API_KEYS bound?
no
DB bound and open keys off?
yes
yes
no
503 API_KEYS_REQUIRED
ALLOW_OPEN_KEYS?
no
get key:...
pn_ + 48 hex?
yes
miss
hit
yes
no
401
AuthContext

Interface surface

AuthContext

```
{ key: string; userId: string; tier: string }
```

`userId` is a truncated SHA-256 of the key (stable partition id).

extractBearer

1. `Authorization: Bearer <token>`
2. Else query `?key=`

requireApiKey

Returns either `{ ok: true, ctx }` or `{ ok: false, status, code, message }`.

Code	HTTP	When
`NO_KEY`	401	Empty Bearer / `?key=`
`INVALID_KEY`	401	Unknown in KV / malformed shape
`API_KEYS_REQUIRED`	503	**Fail-closed:** D1 (`DB`) bound without `API_KEYS` KV and `ALLOW_OPEN_KEYS` off

Fail-closed with durable storage (2.0.1+)

When D1 is active but `API_KEYS` KV is **not** bound, inventable shape-only keys would partition real script data by attacker-chosen tokens with no mint/revoke path. In that configuration:

1. Only explicit `ALLOW_OPEN_KEYS=1` accepts any non-empty Bearer (local demos).
2. Otherwise respond **503** `API_KEYS_REQUIRED` — bind `API_KEYS` and mint via `/api/keys`.

`/api/run` auth gate

From `worker/src/runtime.ts`:

Condition	Auth on `POST /api/run`
`API_KEYS` bound	**Required**
`REQUIRE_RUN_AUTH=1` (or `true` / `yes`)	**Required**
Neither	Optional (Bearer still meters when present)

Always rate-limited (30/min) regardless of auth — see Runtime.

Admin keys API (`/api/keys`)

Action	Auth	Behavior
create (`POST` or `?action=create`)	`X-Admin-Token` === `env.ADMIN_TOKEN`	Mint `pn_` + 48 hex, store `key:{key}` in KV (1y TTL) with tier
validate (`GET` or `?action=validate`)	Bearer or `?key=`	Tier + created_at if known

Tiers: `free` | `hobby` | `pro` | `team` | `enterprise`.

Without `ADMIN_TOKEN` set, **create always fails** (`isAdmin` false).

Do **not** leave `ADMIN_TOKEN = ""` in `[vars]`. That empty assignment is still a Worker binding, and Cloudflare error **10053** rejects `wrangler secret put ADMIN_TOKEN` (`Binding name already in use`). `axis secret put ADMIN_TOKEN` comments the stub, deploys to drop it, then sets the secret.

CLI:

```
axis setup kv                # create + bind API_KEYS
axis secret put ADMIN_TOKEN  # comments empty [vars] stub if present
axis keys create --tier hobby # prompts, or --admin-token / AXIS_ADMIN_TOKEN
axis keys validate --key pn_…
```

Without KV on validate: accepts well-formed `pn_[a-f0-9]{48}` as hobby (dev).

ALLOW_OPEN_KEYS

`wrangler.toml` defaults local demos to `"1"`:

> accept any non-empty Bearer key for `/api/scripts`

Never leave open in production. Prefer bound `API_KEYS` + real admin-minted keys.

Internals

Path	Role
`worker/src/auth.ts`	Bearer, hash, requireApiKey (fail-closed)
`worker/src/keys.ts`	Admin create / validate
`worker/src/runtime.ts`	Run gate + rate limit + usage meter
`worker/src/scripts.ts`	requireApiKey on library CRUD
`worker/src/git-oauth.ts`	Env `GITHUB_OAUTH_CLIENT_ID` / `GITLAB_OAUTH_CLIENT_ID` wins over body `clientId`

Key format

```
'pn_' + 24 random bytes as hex  // 48 hex chars
```

Invariants & edge cases

1. **Admin token is a shared secret** — rotate via wrangler secrets in production, not committed vars.
2. **KV record shape** — JSON `{ key, tier, createdAt }` under `key:${apiKey}`.
3. **No OAuth / cookies** — PWA stores the key in plugin config (`storage:cloud`).
4. **Stream DO unauthenticated** — public market data only.

Worked examples

Create a key (admin)

```
curl -sS -X POST 'http://127.0.0.1:8787/api/keys?action=create' \
  -H "X-Admin-Token: $ADMIN_TOKEN" \
  -H 'content-type: application/json' \
  -d '{"tier":"hobby"}'
```

Validate

```
curl -sS 'http://127.0.0.1:8787/api/keys?action=validate' \
  -H "Authorization: Bearer pn_…"
```

Failure modes

Symptom	Cause
403 create	Wrong/missing admin token
401 scripts	Open keys off + no KV + bad shape
503 `API_KEYS_REQUIRED`	D1 without `API_KEYS` KV and open keys off — bind KV + mint keys
401 on `/api/run`	`API_KEYS` or `REQUIRE_RUN_AUTH` set; missing Bearer
OAuth start uses attacker client	Env client id not set — set `GITHUB_OAUTH_CLIENT_ID` (env always wins over body)
Cross-user leakage tests fail	Partition hash regression

See also

- Data plane
- Bindings
- Runtime
- Security tests
- AXIS CLI — `axis secret put` · `axis setup oauth`

WORKER BINDINGS

wrangler.toml vars, KV, D1, R2, Durable Objects — provision once, paste IDs. Worker script worker-axis.

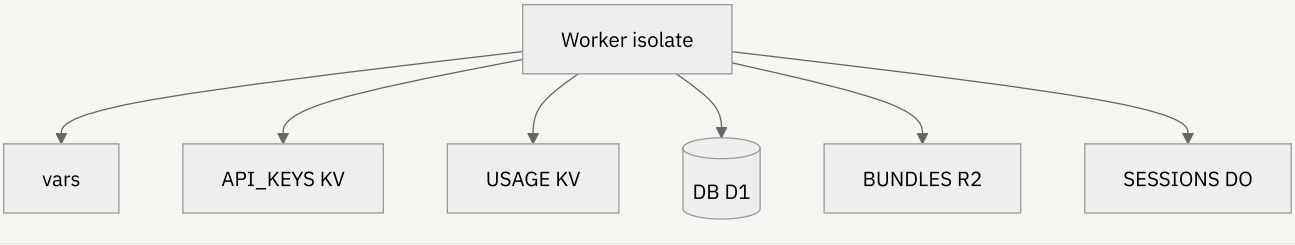
WORKER BINDINGS

Abstract

Bindings and vars for the AXIS Worker are declared in `worker/wrangler.toml` (copy from `wrangler.toml.example` via `axis setup worker`). Many production bindings ship **commented** so clone-and-dev does not require Cloudflare® resources; `wrangler dev` synthesizes local substitutes where possible.

Worker script name: `name = "worker-axis"`. Canonical host `worker.axis.hoox.sh`. Pages project is `axis`; canonical Pages host is `axis.hoox.sh`.

Conceptual model



Interface surface — Env

From `src/index.ts`:

```
interface Env {
  API_KEYS?: KVNamespace;
  USAGE?: KVNamespace;
  DB?: D1Database;
  BUNDLES?: R2Bucket;
  SESSIONS?: DurableObjectNamespace;

  EXTERNAL_BACKEND?: string;
  ALLOWED_ORIGIN?: string;
  ADMIN_TOKEN?: string;
  PYODIDE_IN_WORKER?: string;
  ALLOW_OPEN_KEYS?: string;
  REQUIRE_RUN_AUTH?: string;
  GITHUB_OAUTH_CLIENT_ID?: string;
  GITLAB_OAUTH_CLIENT_ID?: string;
}
```

Vars (`[vars]`)

Var	Default (repo)	Role
`EXTERNAL_BACKEND`	`""`	Flask (or other) base URL for `/api/run` proxy
`ALLOWED_ORIGIN`	`https://pynescript.online`	Extra CORS allowlist (comma-separated); product hosts built-in
`ADMIN_TOKEN`	`""`	Shared secret for key minting
`PYODIDE_IN_WORKER`	`"disabled"`	Gate in-worker Python
`ALLOW_OPEN_KEYS`	`"1"`	Local demos; set `"0"` in prod
`REQUIRE_RUN_AUTH`	unset	`"1"` forces Bearer on `/api/run`
`GITHUB_OAUTH_CLIENT_ID`	unset	Preferred OAuth App id (wins over body)
`GITLAB_OAUTH_CLIENT_ID`	unset	Same for GitLab

Prefer **secrets** for `ADMIN_TOKEN` and production backends:

```
# CLI-first
axis secret put ADMIN_TOKEN
axis secret put EXTERNAL_BACKEND
# or: wrangler secret put …
```

Prod invariant: if `DB` (D1) is bound, also bind `API_KEYS` KV — otherwise script routes fail closed with `API_KEYS_REQUIRED`.

Provision recipes

KV

```
wrangler kv namespace create API_KEYS
wrangler kv namespace create USAGE
# paste ids into wrangler.toml [[kv_namespaces]]
```

D1

```
wrangler d1 create pynescript
# [[d1_databases]] binding = "DB"  (name MUST be DB)
wrangler d1 execute pynescript --remote --file=schemas/scripts.sql
```

Repo may already contain a `database_id` for the project's D1 — treat IDs as environment-specific when forking.

R2 (optional)

```
wrangler r2 bucket create indicator-bundles
# binding BUNDLES
```

Durable Objects

Uncomment bindings + migrations in `wrangler.toml`, then:

```
wrangler deploy
```

Class: `SessionDO` exported from `src/index.ts`.

Other wrangler settings

Setting	Value
`main`	`src/index.ts`
`compatibility_date`	`2024-11-01`
`compatibility_flags`	`nodejs_compat`
`[observability]`	`enabled = true`

Static PWA assets are **not** served by this Worker in the documented split: **Pages** serves `dist/` at `https://axis.hoox.sh`, Worker serves API/WS. Project name for Pages deploy is `axis`.

Deploy commands

```
cd frontend/worker
wrangler deploy

# Pages (from frontend/)
bun run build
wrangler pages deploy dist --project-name=axis
# or npm run deploy:pages from worker package
```

Makefile: `make worker-deploy`, `make pages-deploy` (note: some Makefile targets still point at legacy tree — prefer `dist/` from Vite build; see build and serve).

Invariants & edge cases

1. **Do not rename** CF project without migrating all binding IDs, custom domains, and CI.
2. **Local wrangler** auto-provisions mock KV/D1; production needs real IDs.
3. **`EXTERNAL_BACKEND=localhost` on CF** cannot reach your laptop — use a public tunnel or co-located VPS.
4. **Health** reports which optional bindings are present.

Failure modes

Deploy / runtime issue	Fix
Deploy fails placeholder KV	Uncomment only after create, or leave commented
Scripts empty	Apply SQL schema
Stream 503	Enable DO binding + migration
CORS fails prod origin	Set `ALLOWED_ORIGIN` to exact Pages origin

See also

- Cloudflare DevOps
- Runtime
- CORS

DEVOPS

Build, serve, Cloudflare, VPS demo, CORS, and CI for the AXIS PWA and Worker.

DEVOPS

Abstract

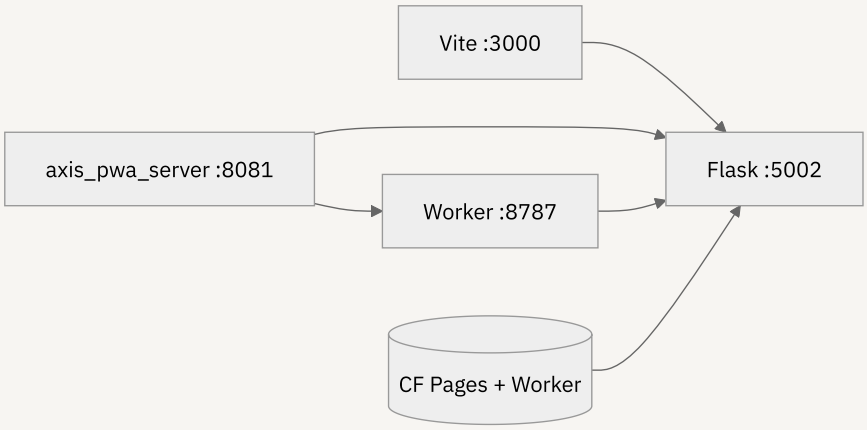
AXIS DevOps spans five runnable surfaces:

1. **Vite + Solid PWA** — product UI (`bun run dev`)
2. **Static server** (`axis_pwa_server.py` or Vite preview) — production-like assets
3. **Cloudflare Worker** (`worker/`) — edge API / WS / OAuth / on-chain proxy
4. **Desktop (Tauri 2)** — native shell around the same UI (`bun run desktop:dev`)
5. **AXIS CLI** (`packages/cli`) — install, doctor, setup, secrets, deploy, health

See Desktop (Tauri) and AXIS CLI for operator entry points.

Python evaluation for desk workflows still often uses the **Flask Pro API** (`make run` on `:5002`). The Worker proxies there when `EXTERNAL_BACKEND` is set.

Conceptual model



Topology cheat sheet

Mode	Command sketch	Ports
Dev AXIS	`bun run dev` / `bun run axis dev`	Vite (often `:3000`)
Dev + Flask	+ `make -C ../pyne run` (checkout sometimes named `pynescript`)	`:5002`
Static dist	`bun run build` + `python axis_pwa_server.py`	`:8081`
Worker local	`bun run axis dev worker` / `cd worker && bun run dev`	`:8787`
CF Worker prod	`axis deploy`	HTTPS workers.dev / custom
Operator	`axis doctor` · `axis setup` · `axis health`	—

Page map

Page	Topic
Local dev	Day-to-day loop
AXIS CLI	Install, doctor, setup, secrets, deploy, health
Build and serve	Vite build, SPA server, assets
Cloudflare®	Pages + Worker deploy
Desktop (Tauri)	Native shell
VPS demo	Single-box static + Flask
CORS and origins	Allowed origins matrix
CI and testing	GitHub Actions, coverage, e2e

Frozen names

Surface	Value
Pages project	`axis`
Pages URL	`https://axis.hoox.sh`
Worker script	`worker-axis`
Worker URL	`https://worker.axis.hoox.sh`
Health service	`worker-axis`

See also

- Worker
- Testing reference
- AXIS hub

LOCAL DEVELOPMENT

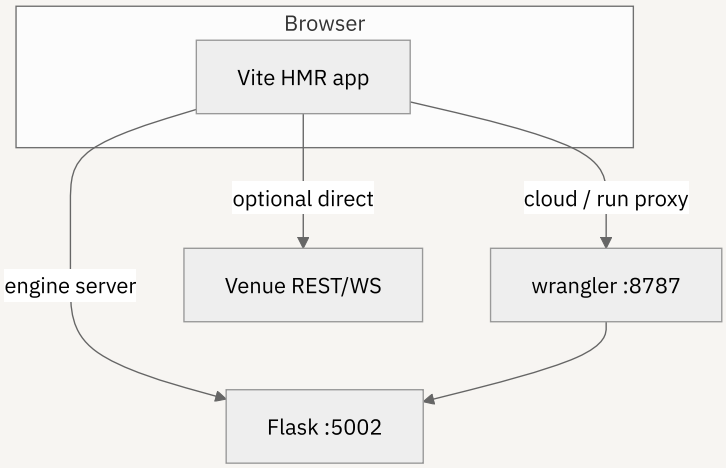
Day-to-day AXIS loop: Vite, Flask, Worker wrangler, endpoints, and plugin examples.

LOCAL DEVELOPMENT

Abstract

Local AXIS development is a **multi-process** topology. AXIS is a Solid app under Vite; evaluation usually hits Flask; optional cloud/storage/stream features hit the Worker.

Conceptual model



Interface surface — commands

Bootstrap (CLI-first)

```
axis install   # app + worker + CLI
axis doctor    # missing worker/wrangler.toml is a warning until `axis setup`
```

Frontend (preferred product path)

```
bun run dev            # Vite + Solid → typically http://127.0.0.1:3000
# or: bun run axis dev
```

Desktop (optional Tauri shell)

```
bun run desktop:dev    # Vite + native window (see devops/desktop)
# or: bun run axis dev desktop
```

Flask Pro API (engine backend)

From the pyne checkout (sister repo; clone hoox-sh/pyne as `../pyne` — the local directory is sometimes still named `pynescript`):

```
make -C ../pyne run   # Flask → :5002
```

Set PWA endpoint to `http://127.0.0.1:5002` for the **server** engine. Workers Manager and Settings can probe this URL.

Worker

```
bun run axis dev worker   # http://127.0.0.1:8787
# or: cd worker && bun run dev
```

Point cloud storage / Worker endpoint at `http://127.0.0.1:8787`. For `/api/run` proxy, set Worker `EXTERNAL_BACKEND=http://127.0.0.1:5002` (works on local wrangler; not on Cloudflare® edge).

AXIS CLI

Preferred operator entry for install / doctor / setup / deploy — see AXIS CLI:

```
bun run axis install
bun run axis doctor
bun run axis setup
bun run axis dev worker
```

Makefile helpers

Target	Intent
`make run-frontend`	Historical Bun static server (`frontend/server.ts`) on `:8081` — **legacy shell path**
`make worker-dev`	Wrangler dev
`make worker-install`	Install worker deps
`make test-frontend`	Bun unit tests under `frontend/tests/`

For the **Solid** product, prefer `bun run dev` / `bun run build` over the legacy `server.ts` unless you are debugging the old tree (legacy shell).

Internals — useful paths

Path	Role
`frontend/src/index.tsx`	App entry
`frontend/vite.config.*`	Vite / Solid plugin
`frontend/src/engines/catalog.ts`	Default engine endpoint
`frontend/worker/wrangler.toml`	Local vars
`frontend/public/plugins/`	Example modules served at `/plugins/`

Worked loop

1. Terminal A: `make run` (Flask).
2. Terminal B: `cd frontend && bun run dev` .
3. Open Vite URL; set engine **Server-Side**, endpoint `http://127.0.0.1:5002` .
4. Load `mock-walk` if venues are blocked.
5. Optional Terminal C: Worker with `ALLOW_OPEN_KEYS=1` for cloud library experiments.

Load an example plugin

With Vite or static server:

```
http://127.0.0.1:3000/plugins/example-coingecko-source.js
```

Manager → Plugins → Load from URL.

Invariants & edge cases

1. **CORS** — Flask must allow the Vite origin; Worker `pickOrigin` allows `:3000` and `:8081` .
2. **Pyodide local** — needs vendor wheels under `public/` (copied to dist on build).
3. **Two endpoints** — topbar endpoint for engine vs cloud storage plugin config can differ (Flask vs Worker).
4. **Legacy Makefile message** still mentions AXIS and `:8081` Bun server.

Failure modes

Issue	Fix
Engine not ready	Flask down / wrong port
CORS errors	Align origins; see CORS
Plugin 404	File only under `src/plugins/` not `public/plugins/`
Worker scripts 401	Set open keys or mint `pn_` key

See also

- Build and serve
- Worker local
- Plugin examples

AXIS CLI

packages/cli — install, doctor, Worker setup (D1/OAuth), secrets, deploy, health — CLI-first operator path.

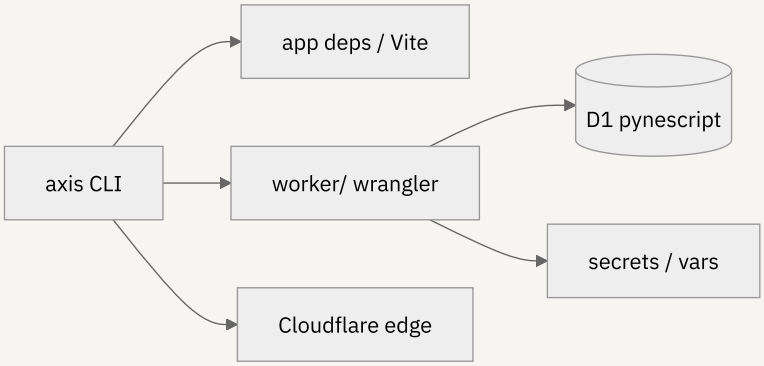
AXIS CLI

Abstract

The **AXIS CLI** (`@hoox-sh/axis-cli` under `packages/cli`) is the **CLI-first** operator entry for local install, Worker bootstrap, Cloudflare® secrets, deploy, and health checks. It wraps Bun + Wrangler with AXIS-specific paths and defaults (Worker name `worker-axis` , D1 `pynescript` , OAuth device flow).

Requires **Node ≥ 20** to run the published binary (`npm i -g @hoox-sh/axis-cli`) and **Bun ≥ 1.2** for install/dev/build commands that shell out to Bun. Product version follows root `package.json` ; the CLI package (`@hoox-sh/axis-cli`) is versioned independently and publishes on `v*` tags via `.github/workflows/release.yml` .

Conceptual model



Install

```
# npm
npm install -g @hoox-sh/axis-cli
axis --help

# from monorepo root
bun install
cd packages/cli && bun install && bun run build && cd ../..

bun run axis --help
# Make pass-through
make axis ARGS="--help"
# Optional global link
cd packages/cli && bun link && axis --help
```

Repo aliases (bun scripts & Make targets)

`axis` is the primary interface. These aliases exist for working inside the repo checkout without a global install:

Alias	Runs
`bun run axis -- <args>`	`packages/cli/bin/axis.js`
`bun run axis:install`	`axis install`
`bun run axis:doctor`	`axis doctor`
`bun run axis:setup`	`axis setup`
`bun run axis:deploy`	`axis deploy` (default worker)
`bun run axis:health`	`axis health`
`make axis ARGS="doctor --remote"`	same bin with args
`make axis-install` · `axis-doctor` · `axis-setup` · `axis-deploy` · `axis-health`	fixed wrappers

When using `bun run axis setup -- --flag` , keep the `--` so Bun forwards flags to the CLI.

Command surface

Command	Role
`axis install`	`bun install` root + `worker/` + CLI
`axis doctor` [`--remote`]	Toolchain, CF auth, optional live `/health`. Missing `worker/wrangler.toml` is a warning until `axis setup` (expected on a fresh clone).
`axis setup`	Bootstrap: install → ensure toml → local D1
`axis setup --prod`	Production bootstrap: remote D1 schema + `API_KEYS` KV
`axis setup worker`	Copy `wrangler.toml.example` → `wrangler.toml` if missing
`axis setup d1 --local|--remote` [`--create`]	Apply `worker/schemas/scripts.sql` (scripts + version history)
`axis setup kv [--usage]`	Create and bind `API_KEYS` KV (required when D1 is on)
`axis setup oauth --github-client-id …`	Set public OAuth App id in `[vars]` (or `--secret`)
`axis keys create [--tier hobby]`	Mint a `pn_…` key via `POST /api/keys` (`ADMIN_TOKEN`)
`axis keys validate --key pn_…`	Validate a key against the Worker
`axis secret put|list|delete`	Wrangler secrets (`ADMIN_TOKEN`, `EXTERNAL_BACKEND`, …)
`axis deploy` / `deploy worker`	Apply remote D1 schema, then deploy Worker `worker-axis`
`axis deploy pages`	Vite build + Pages project
`axis deploy all`	Schema + Worker + Pages
`axis health [--oauth] [--scripts] [--url …]`	Probe `/health`, optional OAuth start, optional `/api/scripts`
`axis whoami`	Cloudflare® account
`axis dev`	Vite product UI
`axis dev worker`	Local wrangler Worker
`axis dev desktop`	Tauri desktop shell

Global flags: `--json`, `--quiet`, `-y` / `--yes`.

Installation self-check

Every `axis` command starts with a lightweight, local-only self-check (skipped for `--version` / `--help`, silenced by `--quiet` / `--json`). It never blocks a command — it only prints stderr warnings when something is off:

- **Runtime compliance** vs `engines` (Node ≥ 20, Bun ≥ 1.2)
- **Install context** — npm install, npx cache, or repo checkout
- **Version drift** — a globally installed CLI older than the repo checkout it is operating on (`npm i -g @hoox-sh/axis-cli` to update)

The same check appears as the optional **CLI installation** row in `axis doctor` (and in `doctor --json` as `cli-install`).

Production checklist

```
axis install
axis doctor
axis setup --prod --github-client-id Ov23li…
axis secret put ADMIN_TOKEN   # comments empty [vars] ADMIN_TOKEN, deploys, then sets secret
axis secret put EXTERNAL_BACKEND
axis deploy all
axis keys create              # --admin-token / AXIS_ADMIN_TOKEN / prompt
axis health --scripts --oauth
# Paste Worker URL + pn_… into Settings → Script storage
```

Item	Guidance
`GITHUB_OAUTH_CLIENT_ID`	Public OAuth App id; Device Flow enabled on GitHub; **env wins over body `clientId`**
`GITLAB_OAUTH_CLIENT_ID`	Same for GitLab when using git storage
`ADMIN_TOKEN` / `EXTERNAL_BACKEND`	Prefer `axis secret put`. An empty `[vars]` stub collides with secrets (Cloudflare 10053)
`API_KEYS` KV	`axis setup kv` then `axis deploy worker`. D1 without KV fails closed (`API_KEYS_REQUIRED`)
`ALLOW_OPEN_KEYS`	`"0"` in production
`REQUIRE_RUN_AUTH`	`"1"` to force Bearer on `/api/run` even without KV
Worker script	`worker-axis` (`https://worker.axis.hoox.sh`)
Pages project	`axis` (`https://axis.hoox.sh`)

Env overrides

Variable	Role
`AXIS_ROOT`	Force monorepo root
`AXIS_WORKER_URL`	Default health / deploy probe URL
`AXIS_ADMIN_TOKEN`	Admin token for `axis keys create`
`AXIS_API_KEY`	Default key for `axis keys validate`
`CLOUDFLARE_API_TOKEN`	Non-interactive Wrangler auth
`AXIS_CLI_SRC=1`	Load `src/` instead of `dist/`

Internals

Path	Role
`packages/cli/bin/axis.js`	Bin entry (Bun)
`packages/cli/src/commands/*`	Commander handlers (`install` , `doctor` , `setup` , `keys` , `deploy` , `secrets` , `health` , `dev` , `whoami`)
`packages/cli/src/services/wrangler-toml.ts`	Toml vars, D1 id, KV binding upsert
`packages/cli/src/services/health.ts`	`/health` + OAuth probes
Root `package.json` `axis` / `axis:*` scripts	Convenience wrappers
Root `Makefile` `axis` / `axis-*`	Make wrappers

See also: Installation, Cloudflare® deployment, Worker bindings, Worker auth, packages/cli/README.md.

BUILD AND SERVE

Vite production build, dist layout, axis_pwa_server.py SPA rules, and asset caching.

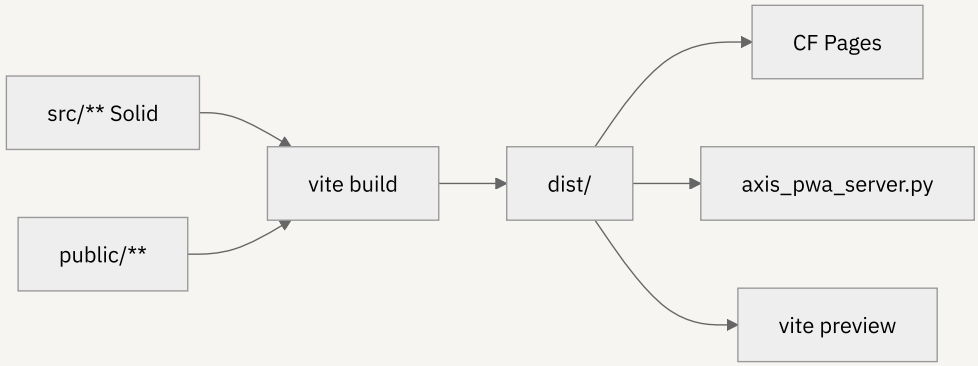
BUILD AND SERVE

Abstract

Production AXIS assets are a **static site** produced by Vite (`bun run build` → `frontend/dist`). Serve them with any static host, Cloudflare Pages, or the repo's `axis_pwa_server.py` SPA-aware server.

Critical requirement: **real binary assets** for Pyodide (`.wasm` , `.whl` , `.zip`) must not be rewritten to `index.html` .

Conceptual model



Interface surface

package.json scripts

```
"dev": "vite",
"build": "vite build",
"preview": "vite preview",
"test": "bun test tests/ worker/tests/",
"test:coverage:gate": "bun run test:coverage && bun scripts/check-coverage.mjs 95",
"test:security": "bun test tests/security/",
"test:e2e:smoke": "playwright test --grep @smoke",
"test:all": "bun run test:coverage:gate && bun run test:security"
```

axis_pwa_server.py

```
cd frontend
bun run build
python axis_pwa_server.py
# HOST / PORT env; default 0.0.0.0:8081 → dist/
```

Behavior (`frontend/axis_pwa_server.py`):

Rule	Detail
Root	Serves `dist/`
Cache	`/assets/*` → `immutable` long cache; else `no-cache`
`GET /health` (`/healthz`)	`200` `text/plain` `ok` — never SPA-fallback to `index.html`
`OPTIONS`	`204` with CORS headers (not 501)
`/favicon.ico`	Serves `assets/icon-192.png` when present
SPA fallback	Unknown paths → `/index.html`
Never rewrite	`/health` , `/favicon.ico` , `/assets/` , `/plugins/` , `/vendor/` , `/pyodide/` , or extensions including `.js/.css/.whl/.wasm/.zip/.py/.json…`

This SPA exception list exists because Pyodide + micropip die with opaque `BadZipFile` when HTML is returned for a wheel URL. The engine's `assertZipAsset` detects that class of failure early.

Expected public assets

Path (public → dist)	Purpose
`/plugins/*.js`	Example dynamic plugins
`/vendor/*.whl`	pynescript + antlr wheels
`/pyodide/v0.26.2/**`	Self-hosted Pyodide
`/pyodide/pynescript_runtime.py`	Browser run bridge

Internals

Path	Role
`frontend/package.json`	Build scripts
`frontend/axis_pwa_server.py`	Threading HTTP SPA server
`frontend/src/engines/catalog.ts`	Asset URL expectations
`frontend/LEGACY.md`	Product path vs old shell

Makefile note

`make pages-deploy` may historically deploy the wrong tree — **prefer**:

```
cd frontend && bun run build
wrangler pages deploy dist --project-name=axis
```

Worked example — static desk demo

```
cd frontend
bun install
bun run build
PORT=8081 python axis_pwa_server.py
# another terminal
make run   # Flask :5002
```

Open `http://127.0.0.1:8081`, engine endpoint `http://127.0.0.1:5002`.

Invariants & edge cases

1. **Service workers / offline** — product path uses Vite PWA assets; legacy root `sw.js` is not the shipping SW.
2. **Relative pyodide index** — resolves against `location.origin`.
3. **Do not hand-edit dist** — regenerate from build.

Failure modes

Symptom	Cause
Blank routes on refresh	Server lacks SPA fallback
Pyodide HTML wheel error	Fallback rewriting `.whl` or missing vendor
Missing plugins	`public/plugins` not copied (check Vite publicDir)

See also

- Local dev
- Cloudflare
- Engines

DESKTOP (TAURI)

Run and package AXIS as a native desktop app with Tauri 2.

DESKTOP (TAURI)

Abstract

AXIS ships as a browser PWA and as an optional **desktop shell** built with Tauri 2. The Solid/Vite UI is unchanged; Tauri embeds it in a system webview (WebKitGTK on Linux, WKWebView on macOS, WebView2 on Windows).

Commands

From the repo root (after `bun install`):

```
bun run desktop:dev     # Vite :3000 + Tauri window (hot reload)
bun run desktop:build   # Vite production build + native installers
bun run desktop:info    # toolchain / webview diagnostics
```

Script	What it does
`desktop:dev`	Runs `tauri dev` → `beforeDevCommand` starts Vite, opens a native window on `http://127.0.0.1:3000`
`desktop:build`	Runs `tauri build` → `bun run build` then packages `dist/`
`desktop:info`	Prints Rust, system libraries, and webview status

Installers land under `src-tauri/target/release/bundle/` (AppImage, deb, rpm, msi, dmg — depending on host OS and `bundle.targets`).

CI (GitHub Actions)

Every **push to `main`** (and PRs that touch desktop/frontend paths) runs `.github/workflows/desktop.yml`:

Trigger	Result
Push `main`	Matrix build: Linux, macOS arm64, macOS x64, Windows → **workflow artifacts** (14-day retention)
Tag `v*` / `desktop-v*`	Same matrix + **GitHub Release** assets attached to the tag
`workflow_dispatch`	Manual re-run

Artifacts are named `axis-desktop-{linux\|macos-arm64\|macos-x64\|windows}-{sha}`. Concurrency cancels superseded runs on the same ref so each push rebuilds cleanly.

Prerequisites

Always

- Bun (or another JS package manager that can run the scripts)
- Rust (`rustc` / `cargo`) — see `src-tauri/Cargo.toml` `rust-version`
- System webview libraries for your platform (Tauri prerequisites)

Linux (Arch / CachyOS example)

```
sudo pacman -S --needed webkit2gtk-4.1 base-devel curl wget file \
  openssl appmenu-gtk-module libappindicator-gtk3 librsvg patchelf
```

Debian/Ubuntu equivalents use `libwebkit2gtk-4.1-dev` and related `-dev` packages (see the Tauri docs for the current list).

Engine backends

Desktop AXIS is the same product as the PWA. For Pine evaluation you still need one of:

Engine	Notes
Local **pyne** Pro API	Clone hoox-sh/pyne as `../pyne` (`make run` → `:5002`; Vite proxies `/run` in dev). Local checkouts are sometimes still named `pynescript`.
Cloudflare Worker	Configure worker URL in Manager
Pyodide (in-webview)	Fully offline path; largest first load

Layout

Path	Role
`src-tauri/`	Rust host, `tauri.conf.json`, icons, capabilities
`src-tauri/tauri.conf.json`	App id `sh.hoox.axis`, window size, build hooks
`src/`	Unchanged Solid product UI
`dist/`	Vite output consumed by `frontendDist`

Behaviour notes

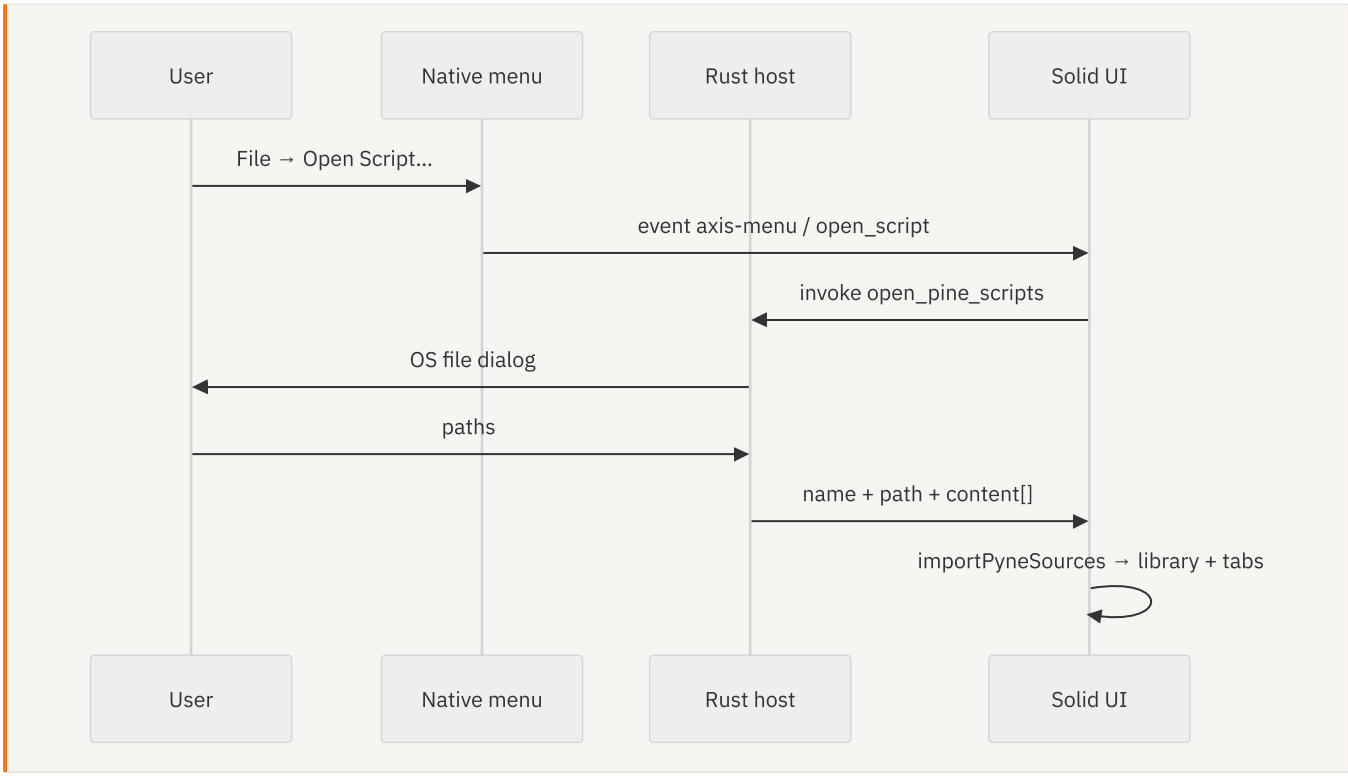
- **Service worker** is skipped in the Tauri shell (`src/pwa/register-sw.ts`). Offline install is a PWA concern; the desktop app is already installed as a native binary.
- **Window** defaults: 1440×900, min 960×640, centered.
- **CSP** is left open (`null`) so external venues, Pyodide assets, and worker APIs work the same as in the browser. Tighten later if you ship a locked-down build.
- Icons are generated from `public/assets/icon-512.png` via `bunx tauri icon`.

Native menu & open from disk

The Tauri host builds a native app menu:

Menu	Item	Action
File	**Open Script…** (`⌘/Ctrl+O`)	System multi-file dialog → library + editor tabs
File	**Quit AXIS**	Quit (platform predefined)
Help	**About AXIS**	Info dialog (version / license / site)

Flow

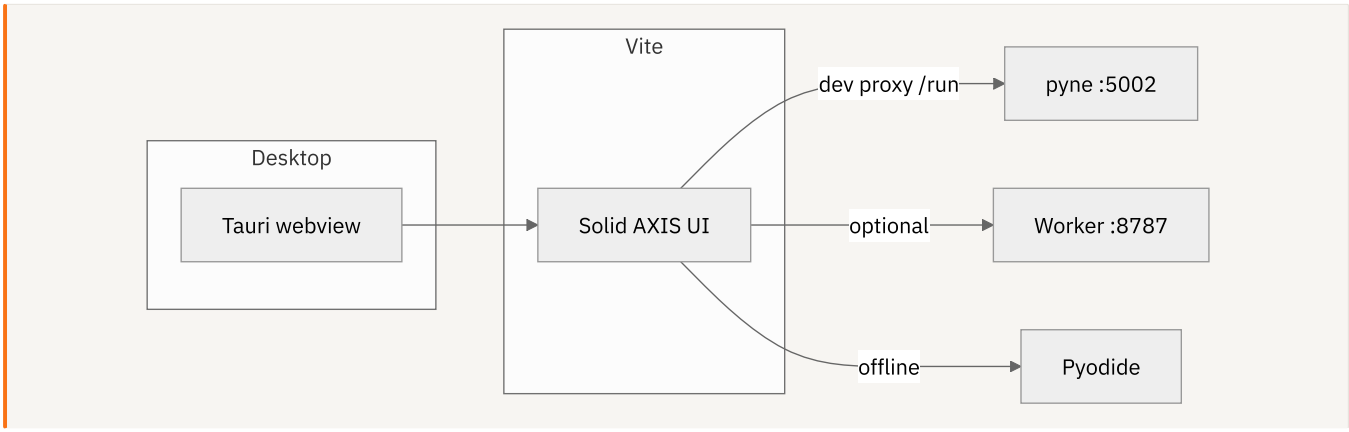


Path	Role
`src-tauri/src/lib.rs`	Menu, `open_pine_scripts` command, dialog + disk read
`src/desktop/`	`isTauriShell`, menu listen, open + About wiring
`src/storage/import-pyne-open.ts`	Shared status/logs/editor open after import
`src/storage/import-pyne-files.ts`	`importPyneSources` (text) + `importPyneFiles` (browser `File`)

Accepted extensions match the PWA drop path: `.pyne`, `.pine`, `.pinescript`, `.pinev5`, `.pinev6`. Per-file size cap on the host: **8 MiB**.

Drag-and-drop of the same extensions still works in the desktop webview.

Conceptual model



CLOUDFLARE DEPLOYMENT

Pages + Worker for AXIS: Pages axis.hoox.sh (project axis), Worker worker.axis.hoox.sh, CLI deploy, bindings, security checklist.

CLOUDFLARE® DEPLOYMENT

Abstract

Production AXIS on Cloudflare® is a **split deploy**:

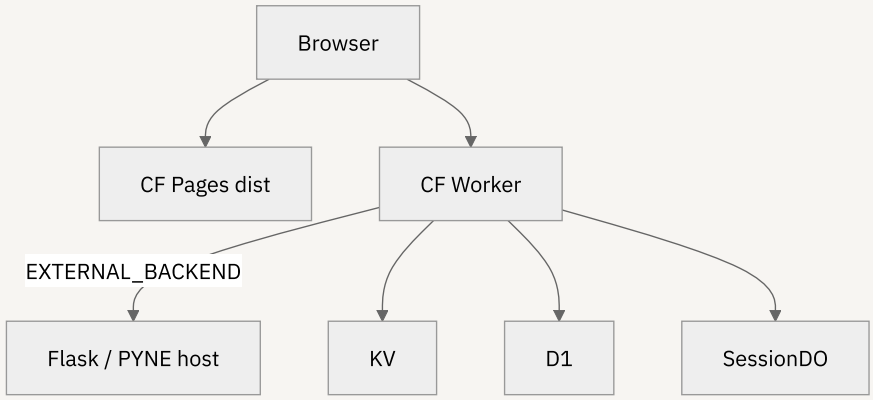
Piece	Role
Pages	Static PWA (`dist/`)
Worker	JSON API + WebSocket (`worker/`)

Pages: project `axis` , canonical `https://axis.hoox.sh` . **Worker:** script `worker-axis` at `https://worker.axis.hoox.sh` .

Evaluation truth: Worker `/api/run` **proxies to** `EXTERNAL_BACKEND` unless the gated Pyodide path is fully implemented and enabled. You still need a Python host (VPS Flask, container, etc.) for real Pine Script™ fidelity today.

Prefer the **AXIS CLI** for setup and deploy (`axis deploy`).

Conceptual model



Deploy procedure

1. Provision bindings (once)

See Worker bindings:

- KV `API_KEYS` , `USAGE`
- D1 `pynescript` + schema
- Optional R2, Durable Objects

2. Configure secrets / vars

Production checklist:

Setting	Production guidance
`EXTERNAL_BACKEND`	Public HTTPS URL of Flask/PYNE API (`axis secret put`)
`ALLOWED_ORIGIN`	Extra exact origins (comma-separated); product hosts + `*.axis.pages.dev` are built-in — CORS
`ADMIN_TOKEN`	Secret, strong
`API_KEYS` KV	**Bind in prod** — D1 without KV fails closed (`API_KEYS_REQUIRED`)
`ALLOW_OPEN_KEYS`	`"0"` or unset
`REQUIRE_RUN_AUTH`	`"1"` optional force Bearer on `/api/run`
`GITHUB_OAUTH_CLIENT_ID`	Env preferred; never rely on body `clientId` alone
`PYODIDE_IN_WORKER`	`"disabled"` until wheel pipeline ready

3. Deploy Worker

```
# Preferred: AXIS CLI
axis deploy
# or: make axis-deploy

# Schema first (remote)
axis setup d1 --remote

# Equivalent raw wrangler
cd worker
bun install
bun run deploy
```

4. Build & deploy Pages

```
bun run axis deploy pages
# or
bun run build
bunx wrangler pages deploy dist --project-name=axis
# Make: make pages-deploy
```

5. Point the PWA

- Engine endpoint: Worker URL (if path-mapped to `/run`) **or** Flask URL directly
- Cloud storage endpoint: Worker origin
- Ensure CORS allows Pages origin

Internals

Path	Role
`worker/wrangler.toml` / `.example`	Worker name + bindings
`worker/README.md`	Ops narrative
`worker/package.json`	deploy scripts
`packages/cli/`	`axis deploy` / `axis health`
Makefile `axis-deploy` / `pages-deploy` / `worker-deploy`	Convenience targets

Health check

```
axis health
# or
curl -sS https://<worker>/health
# service: worker-axis
```

Invariants & edge cases

1. **Pages ≠ Worker host** — CORS required unless same-origin routing via custom domain routes.
2. **Localhost EXTERNAL_BACKEND** is useless from the edge.
3. **DO migrations** apply on deploy when uncommented.
4. **Observability** enabled in wrangler for log tails (`wrangler tail`).

Failure modes

Issue	Fix
503 NO_BACKEND	Set reachable EXTERNAL_BACKEND
CORS blocked	ALLOWED_ORIGIN mismatch
SPA HTML for wheels	Pages must serve `/vendor` and `/pyodide` as static files
Stream NO_DO	Enable SessionDO bindings

See also

- Worker
- CORS
- VPS demo (backend co-host)

VPS DEMO TOPOLOGY

Single-box demo: static AXIS dist + Flask Pro API (+ optional reverse proxy) without Cloudflare.

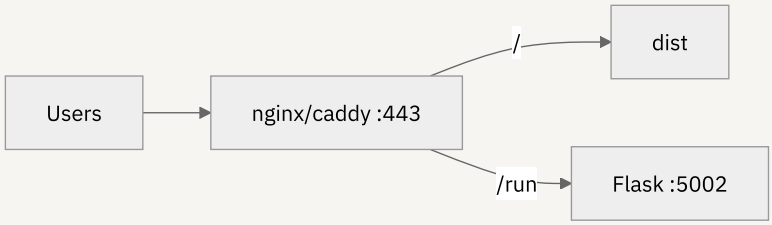
VPS DEMO TOPOLOGY

Abstract

A **VPS demo** is the lowest-surprise production-like setup for Pine evaluation: one machine runs Flask (PYNE) and serves the static AXIS build. Optional nginx/caddy terminates TLS and reverse-proxies `/run` same-origin to avoid CORS.

No Worker required. Cloud storage and DO streams will not work without the Worker.

Conceptual model



Interface surface

Processes

Process	Command	Port
Build (CI or once)	`cd frontend && bun run build`	—
Static	`PORT=8081 python axis_pwa_server.py`	8081
API	`make run` / `python -m backend.app`	5002

Same-origin proxy sketch (Caddy)

```
axis.example.com {
  handle /run* {
    reverse_proxy 127.0.0.1:5002
  }
  handle /optimize* {
    reverse_proxy 127.0.0.1:5002
  }
  handle {
    reverse_proxy 127.0.0.1:8081
  }
}
```

Then PWA endpoint = `https://axis.example.com` and the **server** engine posts to `/run` on the same host.

Production (split: Pages PWA + Hetzner API)

- **PWA** — Cloudflare Pages custom domain `https://axis.hoox.sh`
- **PYNE Pro API** — Hetzner VPS `https://pynescript.online` (Cloudflare orange-cloud → origin `204.168.138.51`). nginx `:443` reverse-proxies `/run`, `/optimize`, `/ws/`, `/health`, `/lsp/`, `/datafeed/`, … to gunicorn `127.0.0.1:5002`
- PWA **Backend URL** = `https://pynescript.online` (cross-origin; product CORS regex)
- Cloudflare Pages previews (`*.axis.pages.dev`) use the same Backend URL — pyne must allow those Origins on `/health` and `/run` (always-on product regex; see CORS)

SSH alias: `ssh pynescript` (`204.168.138.51`). The previous combined VPS (`ssh axis` / `162.254.38.194`) stays until decommission.

Update deploy (from a machine that can SSH to the VPS):

```
# 1. Ship latest main
git push origin main

# 2. On VPS (paths may vary — often /root/axis + rsync dist → PWA WorkingDirectory)
# Pro API (from pynescript checkout)
./scripts/deploy_vps.sh
# or:
ssh pynescript 'cd /root/pynescript && git pull --ff-only && systemctl restart pynescript-api'
# PWA: `bun run build` + wrangler pages deploy (axis.hoox.sh)
```

Local operator CLI for the **Worker** edge (separate from VPS static): `axis deploy` — see AXIS CLI.

Hardened network (UFW) + health checks

Production VPS pattern (verified on `pynescript.online`):

Surface	Bind / allow	Health URL
UFW	`deny incoming` default; **allow 22/tcp, 80/tcp, 443/tcp**	—
nginx	`0.0.0.0:443` TLS (+ `:80` ACME)	`https://pynescript.online/health` → proxy `127.0.0.1:5002/health`
Pro API (gunicorn)	**`127.0.0.1:5002` only**	loopback; **not** public
PWA static	Cloudflare Pages (`axis.hoox.sh`)	not on this VPS
fail2ban	`sshd` jail	—

Do not open UFW for `5002`, `80`, or `8787` on a public VPS. Direct probes to `http://<vps-ip>:5002/health` **must time out** if hardening is correct.

Workers Manager browser probes:

- On **product API host** (`pynescript.online`), pyne Pro is probed via **same-origin** `/health` (nginx), not `http://127.0.0.1:5002` (that would hit the *client* PC and always look “down”).
- From **CF Pages** (`axis.hoox.sh` / `*.axis.pages.dev`) catalog `publicEndpoint` (`https://pynescript.online`) is used — never client loopback.
- CF Worker health stays on `https://worker.axis.hoox.sh/health` (edge; independent of VPS UFW).

Minimal without TLS

- Static `:8081`, Flask `:5002`
- Prefer a hostname over raw IP when the page is HTTPS
- Open firewall; accept CORS from static origin (configure Flask accordingly)

Internals

Concern	Notes
Pyodide offline	Ship full `dist` including `vendor` + `pyodide`
Process supervisor	systemd units for flask + axis_pwa_server
Updates	Rebuild dist artifact; restart static only
Secrets	Flask admin / pro keys separate from Worker

Worked example — systemd sketch

```
# /etc/systemd/system/axis-pwa.service
[Service]
WorkingDirectory=/opt/axis/frontend
Environment=PORT=8081
ExecStart=/usr/bin/python3 axis_pwa_server.py
Restart=on-failure
```

Pair with an existing `backend` unit for Flask.

Invariants & edge cases

1. **CPU/RAM** — Pyodide is browser-side; VPS load is Flask evaluate concurrency.
2. **Do not expose open admin tokens**.
3. **Disk** — dist + wheels are tens of MB.
4. Hybrid: VPS Flask as `EXTERNAL_BACKEND` for a CF Worker still works if VPS has a public URL.

Failure modes

Issue	Fix
Mixed content	HTTPS page calling HTTP Flask blocked
CORS	Prefer same-origin proxy
502 on /run	Flask crashed; check journalctl

See also

- Build and serve
- Cloudflare
- CORS

CORS AND ORIGINS

Worker pickOrigin: local-dev, product hosts, project-scoped Pages previews, ALLOWED_ORIGIN list, Flask constraints.

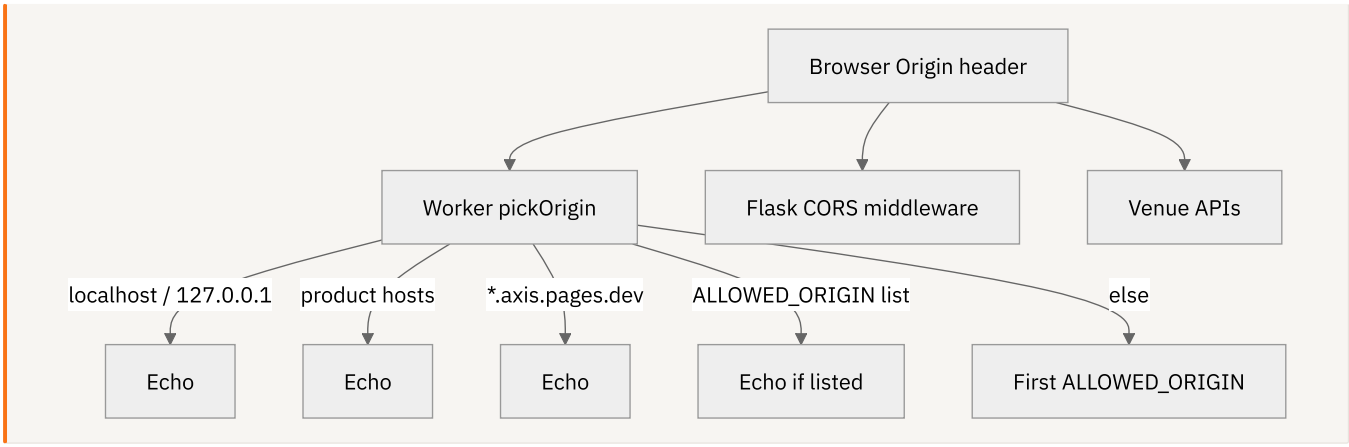
CORS AND ORIGINS

Abstract

AXIS is a browser product that talks to **cross-origin** backends (Flask, Worker, venues). CORS misconfiguration is the #1 local-deploy footgun after missing wheels.

The Worker implements an explicit origin picker (`pickOrigin` in `worker/src/index.ts`); Flask and venue APIs have their own rules.

Conceptual model



Worker behavior

Resolution order for `Access-Control-Allow-Origin`:

1. Local-dev (always echoed)

Any `http://` or `https://` origin on **`localhost`** or **`127.0.0.1`**, optional port (e.g. Vite `:3000`, `axis_pwa_server` `:8081`, Playwright ephemeral ports).

Not allowed (and not needed): `http://0.0.0.0:…` — browsers do not use `0.0.0.0` as a page origin; binding with `HOST=0.0.0.0` only means "listen on all interfaces."

Localhost/127 allowlisting is **safe for CORS**: only pages actually served from those hosts present that `Origin`. A public site cannot forge `Origin: http://localhost:3000` in a real browser.

2. Product hosts (2.0.1+)

Exact product / HOOX / legacy PYNE hosts under:

- `*.hoox.sh` / `hoox.sh`
- `*.pynescript.online` / `pynescript.online`

Examples: `https://axis.hoox.sh` (canonical Pages), `https://hoox.sh`, `https://pynescript.online`.

3. Project-scoped Cloudflare® Pages (not open `*.pages.dev`)

Canonical Pages is **`https://axis.hoox.sh`** (product host above). Preview hashes on **`*.axis.pages.dev`** (and the apex `axis.pages.dev`) are also echoed. Legacy **`*.pynescript-axis.pages.dev`** still matches. Arbitrary third-party Pages projects (`evil.pages.dev`, other apps) fall through to the allowlist fallback — they are **not** reflected.

4. `ALLOWED_ORIGIN` allowlist

Comma-separated exact origins in `env.ALLOWED_ORIGIN` (e.g. `https://a.example,https://b.example`). If the request Origin is listed, echo it; otherwise return the **first** entry (default `https://pynescript.online`).

CORS headers applied:

```
Access-Control-Allow-Origin: <picked>
Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS
Access-Control-Allow-Headers: Content-Type, Authorization, X-Admin-Token, If-Match
Access-Control-Max-Age: 86400
Vary: Origin
```

`OPTIONS` → `204` preflight.

Production implication

For a custom domain already under product regex, no extra var is required. For one-off preview hosts outside the project, list them:

```
ALLOWED_ORIGIN = "https://my-preview.example.com,https://axis.example.com"
```

Flask / server engine

The **server** engine posts from the browser to the configured endpoint. Flask Pro API must:

1. Answer `OPTIONS` preflight if cross-origin.
2. Reflect or allow the PWA origin.
3. Accept `Content-Type: application/json`.

Pyne always appends the same product Origin regex as Worker `pickOrigin` (including `*.axis.pages.dev`) even when systemd `ALLOWED_ORIGINS` is a short list. `GET /health` (AXIS Settings probe) and `POST /run` are **free CORS paths** — they echo any `Origin` so a Pages preview can reach `https://axis.hoox.sh` without a per-preview allowlist entry.

Same-origin reverse proxy (VPS demo) eliminates CORS when the PWA and Pro API share `https://axis.hoox.sh`. Cross-origin Pages → VPS still needs those headers (they are now default).

Venue sources/streams

Public Binance/OKX/etc. REST must send CORS headers usable by browsers. Firefox often reports **"CORS request did not succeed / status (null)"** when the TCP/TLS connection fails entirely (geo block, firewall, extension) — that is **not** a missing `Access-Control-Allow-Origin` header.

AXIS resilience for Binance:

1. Direct hosts: `api.binance.com` then `data-api.binance.vision` (`src/data/binance-http.ts`).
2. Worker allowlisted proxy: `GET /api/market/binance/klines|ticker/24hr|exchangeInfo` on the AXIS Worker (`worker/src/market.ts`).
3. Live WS rotates `:9443` → `:443` → `data-stream.binance.vision` (port 9443 is frequently firewalled).
4. Offline last resort: `mock-walk` / `mock-poll`, or warm bars-cache.

Prefer the DO relay (`cf-do` stream / `/api/stream`) when many tabs share one upstream WS — still requires the Worker edge to reach Binance.

AXIS resilience for MEXC:

1. Worker allowlisted proxy first: `GET /api/market/mexc/{klines,ticker/24hr,exchangeInfo}` (`src/data/mexc-http.ts`). Public `api.mexc.com` does **not** send `Access-Control-Allow-Origin`.
2. Direct `api.mexc.com` only as last-ditch fallback (offline lab, tests, or Worker 5xx). Worker 4xx is not retried against the CORS-blocked host.

Dynamic plugins

- **Module import** of plugin JS: needs CORS on the **script origin**. Prefer same-origin `/plugins/*` (PYNE Agent ships as `/plugins/axis-pine-agent.js`).
- **Remote** modules (e.g. legacy `pyne-agent-worker…/plugin/axis-pine-agent.js`) need ACAO + CSP `script-src` allow when a CSP is present.
- **fetch inside plugin**: subject to third-party CORS independently (agent API still uses the pyne-agent Worker origin).

Invariants & edge cases

1. **Credentials** — Worker CORS does not set `Allow-Credentials: true`; use Bearer headers, not cookies.
2. **Multiple prod domains** — product regex + comma-separated `ALLOWED_ORIGIN` cover multi-host; unknown sites never get open reflection.
3. **Vite vs 8081** — both covered for Worker; Flask config must match whichever you use.
4. **Health checks** from curl omit Origin — still work; browser calls need correct ACAO.
5. **No open `*.pages.dev`** — only the Pages project `axis` previews (`*.axis.pages.dev`) are product-scoped.

Failure modes

Browser console	Meaning
blocked by CORS policy	Origin not allowlisted
preflight 404	Server lacks OPTIONS
No ACAO on error body	Some error paths forgot headers (Worker try/catch mostly covered)

See also

- Worker bindings
- Local dev
- Dynamic loader

CI AND TESTING

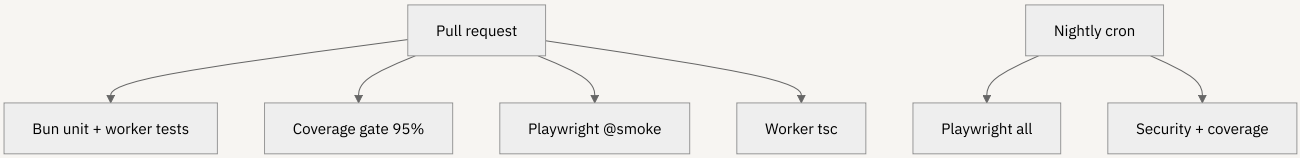
GitHub Actions for AXIS: unit coverage gate, Playwright smoke, worker typecheck, nightly full e2e.

CI AND TESTING

Abstract

AXIS quality gates live primarily under `frontend/` (Bun tests) and `frontend/worker/tests/`, orchestrated by GitHub Actions (`.github/workflows/ci.yml`, `axis-nightly.yml`). Deep test conventions are documented in Testing reference and `frontend/TESTING.md`.

Conceptual model



Interface surface — local commands

```
cd frontend
bun run test                # tests/ + worker/tests/
bun run test:unit
bun run test:coverage:gate  # lcov + scripts/check-coverage.mjs 95
bun run test:security
bun run test:e2e:smoke
bun run test:e2e:critical
bun run test:e2e
bun run test:all            # gate + security
```

Worker:

```
cd frontend/worker
npm run typecheck
```

Repo Makefile:

```
make test-frontend
make worker-typecheck
```

CI workflows

ci.yml (excerpted jobs)

Job (names vary)	What
Frontend tests / coverage	Bun install, coverage, artifact `axis-coverage-lcov`
`axis-e2e`	Playwright `@smoke` on PR
Worker typecheck	`frontend/worker` tsc
Related	Python suite remains for pynescript core

axis-nightly.yml

Job	What
Playwright full	all e2e specs
Security + coverage gate	`test:coverage:gate` + `test:security`

Schedule: `0 4 * * *` UTC + `workflow_dispatch`.

Internals

Path	Role
`frontend/TESTING.md`	Authoritative testing guide
`frontend/scripts/check-coverage.mjs`	Scoped line gate
`frontend/playwright.config.ts`	webServer build+preview `:4173`
`frontend/e2e/`	Specs with `@smoke` / `@critical` tags
`frontend/worker/tests/`	auth, keys, scripts, runtime
`.github/workflows/ci.yml`	PR gates
`.github/workflows/axis-nightly.yml`	Nightly

Coverage policy (summary)

- **Gate:** 95% lines on a **scoped** core (plugins, storage minus idb, store, results, sources, streams, data, selected chart helpers, worker auth/keys/runtime/scripts).
- **Excluded from gate:** heavy chart UI, pyodide boot, legacy JS, Solid `.tsx` surfaces (unit-tested elsewhere or deferred).

E2E design

- Builds production preview; baseURL `http://127.0.0.1:4173`.
- Mocks `/run` and Binance where possible so smoke does not need live Pro API.
- Selectors use `data-testid` (`axis-btn-load`, `axis-manager`, ...).

Invariants & edge cases

1. **No real network in unit tests** — mock `fetch` / WebSocket.
2. **Import `./setup` first** when touching store/plugins.
3. **CI Bun version** pinned in workflows (e.g. 1.3).
4. **Playwright browsers** installed with `--with-deps` on CI.

Failure modes

CI red	Likely
Coverage gate	New untested code in scoped paths
Smoke timeout	Build slow; webServer 180s
Worker tsc	Env typing / DO exports
Flaky e2e	Missing testid; race without mock

See also

- Testing reference
- Local dev

FEATURE ATLAS

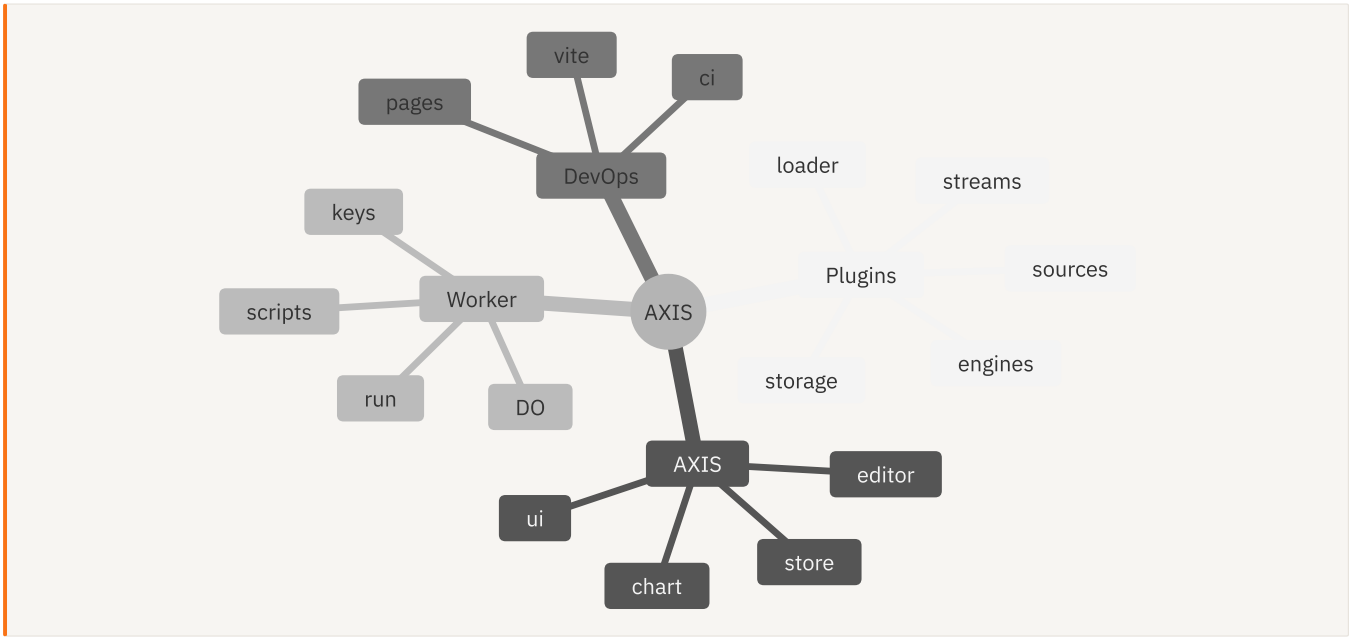
Map of AXIS product surfaces to repository paths: plugins, AXIS, worker, storage, test, legacy.

FEATURE ATLAS

Abstract

This atlas is a **path index** for AXIS. Use it to jump from a product feature to the code that implements it. Prefer Solid product paths over legacy static shell files.

Conceptual model



Atlas tables

Plugin system

Feature	Primary paths
Contracts	`src/plugins/types.ts`
Registry	`src/plugins/registry.ts`
Bootstrap	`src/plugins/bootstrap.ts`
Active set	`src/plugins/active.ts`
Dynamic install	`src/plugins/loader.ts` (source/stream/engine/dataset/**component**)
Barrel	`src/plugins/index.ts`
Manager UI	`src/ui/PluginManager.tsx`, `plugin-badges.tsx`
Docs (in-tree)	`src/plugins/README.md`

Data plugins

Feature	Primary paths
Sources	`src/sources/catalog.ts`, `upload-store.ts`
Streams	`src/streams/catalog.ts`, `multiplex.ts`
Engines	`src/engines/catalog.ts`
Load symbol (one-shot)	`src/data/load-symbol.ts`, `parse-bars.ts`
Data Source Manager	`src/data/data-source-manager.ts`, `bars-cache.ts`, `bars-gaps.ts`, `ui/DataSourceManagerPanel.tsx`
Script run	`src/indicators/runner.ts`
Scripts panel	`src/indicators/IndicatorPanel.tsx`, `IndicatorCard.tsx`
Chart themes	`src/theme/presets.ts`, `catalog.ts`, `manager.ts`, `ui/ThemePanel.tsx`

Storage / library

Feature	Primary paths
Catalog	`src/storage/catalog.ts`
Local IDB	`src/storage/local.ts`, `idb.ts`
Cloud	`src/storage/cloud.ts`
Git	`src/storage/git.ts`, `git-github.ts`, `git-gitlab.ts`
Service façade	`src/storage/service.ts`
Library panel	`src/ui/ScriptLibraryPanel.tsx`
v6 starter pack	`examples/script-library-starter.json`

UI (UI)

Feature	Primary paths
Entry	`src/index.tsx`, `src/app.tsx`
Chart host	`src/chart/ChartHost.tsx`, `ChartWorkspace.tsx`, `pane-manager.ts`, `pane-badge.ts`, `series-factory.ts`
Price-scale decimals	`src/chart/price-precision.ts`
Plot style parity	`src/results/plot-visuals.ts` → pane-manager
Multi-chart / recipes	`src/chart/layout.ts`, `layout-recipes.ts`, `chart-registry.ts`, `ui/ChartLayoutMenu.tsx`
Bar replay	`src/chart/bar-replay.ts`, `ui/BarReplayControls.tsx`
Compare / volume profile	`src/chart/compare-overlay.ts`, `volume-profile.ts`
Drawings	`src/chart/drawing-layer.ts`, `drawings/*`, `pyne-drawings.ts`, `DrawingToolbar.tsx`
Editor	`src/editor/*` (CM6, diagnostics, ruler, git-sync, inline-debug, symbols)
Editor chrome	`ui/EditorGitBar.tsx`, `ui/EditorProblems.tsx`
Alerts	`src/alerts/*`, `ui/AlertsPanel.tsx`
Command palette	`ui/CommandPalette.tsx`, `command-registry.ts`
Docks	`ui/panels/*` (side-by-side left/right)
Results	`src/results/*`, `ui/ResultsPanel.tsx`, `StrategyReport.tsx`
Workers Manager	`src/ui/WorkersManager.tsx`, `src/workers/*`
Store	`src/store/index.ts`, `types.ts`
Topbar / settings	`ui/Topbar.tsx`, `TopbarField.tsx`, `SettingsDialog.tsx`
Workspace snapshot	`src/storage/workspace-snapshot.ts`
Desktop shell bridge	`src/desktop/*`, `src-tauri/`

Worker

Feature	Primary paths
Router / CORS	`worker/src/index.ts` (`pickOrigin`)
Run (auth + rate limit)	`worker/src/runtime.ts`, `pyodide_runtime.ts`
Auth / keys	`worker/src/auth.ts`, `keys.ts`
Scripts	`worker/src/scripts.ts`, `schemas/scripts.sql`
Git OAuth	`worker/src/git-oauth.ts`
Session DO	`worker/src/durable-objects/session.ts`
Config	`worker/wrangler.toml.example`, `RUNTIME.md`, `README.md`

Build / ops

Feature	Primary paths
Package scripts	root `package.json` (`dev` , `build` , `test:*` , `axis:*` , `desktop:*`)
AXIS CLI	`packages/cli/` (`@hoox-sh/axis-cli`)
Static server	`axis_pwa_server.py`
Public assets	`public/plugins` , `public/vendor` , `public/pyodide`
Makefile	`axis` / `axis-*` , `pages-deploy` , `docker-*`
Desktop CI	`.github/workflows/desktop.yml`
CI	`.github/workflows/ci.yml` (coverage + e2e smoke)
Coverage gate	`scripts/check-coverage.mjs` (scoped core ≥95%)
Harden/perf audit	`docs/devops/harden-perf-audit-2026-08-11.md`

Examples & tests

Feature	Primary paths
Example plugins	`public/plugins/*.js`
Pine v6 library starter	`examples/script-library-starter.json`
Unit tests	`tests/**`
Worker tests	`worker/tests/**`
E2E	`e2e/**` , `playwright.config.ts`
Testing guide	`TESTING.md`
Legacy notes	`LEGACY.md`

Namespaces & keys

Key / namespace	Use
`pynescript.axis.plugins.v1`	Installed dynamic plugins
`pynescript.axis.library.v1`	Local library LS fallback
`pynescript.axis.storage`	IDB database name
`pynescript.axis.v1`	App state
`pynescript.axis.v2`	Older app-state key (write-forward)
Contract ns	`pynescript.axis.plugins.v1` (conceptual)

Infrastructure constants

Name	Value
Pages project	`axis`
Pages URL	`https://axis.hoox.sh`
Worker script	`worker-axis`
Worker URL	`https://worker.axis.hoox.sh`
Health service	`worker-axis`
Pyodide version	`0.26.2` (self-hosted path)

See also

- Plugins
- Worker
- Legacy shell

OPEN CAPABILITY GAPS

Research synthesis of high-demand charting, strategy, and script-host capabilities that remain scarce or incomplete in mainstream platforms—framed for AXIS product direction.

OPEN CAPABILITY GAPS

Abstract

This page records a **product research synthesis**: roughly thirty concrete capabilities that traders, quant authors, and chart power-users still treat as missing or incomplete on mainstream charting and strategy hosts. It is not a shipped-feature checklist for AXIS and not a competitive teardown of any named vendor.

Use it when prioritizing AXIS surfaces (sources, streams, engines, strategy results, debugging, data plane). Items below are durable demand themes from community threads and documented hard limits—not a public vendor roadmap with delivery status.

How to read this list

Term	Meaning here
Still open	Capability is absent, plan-gated to a partial substitute, or still described as unreliable for professional use
Not a ranking poll	No single global leaderboard exists; order is thematic, then by how often demand shows up as trade-blocking
Status changes	Individual venues or UX affordances can ship later; verify against current product notes before treating any row as forever

Highest-pressure themes overall: true tick and order-flow data, portfolio-grade strategy tooling, live multi-bracket and automated execution, higher cloud/screener ceilings, and a real interactive debugger.

Order flow, volume, and tick data

#	Capability	Why it stays open
1	**True tick volume** under volume profiles (including reliable delta)	Profiles over the same range can disagree on buy/sell aggression; many treat minute-aggregated volume as unsuitable for professional volume work
2	**Real volume footprint** with aggressor-tagged prints	Synthetic or approximate footprints are still not trusted next to tick-native tools
3	**Transaction-level order-flow viewing**	Charts that are primarily bar/minute-based cannot show trade-by-trade flow without true tick history

Live trading, brackets, and execution

#	Capability	Why it stays open
4	**Multi take-profit / multi-bracket management on live broker routes**	Multi-exit levels often exist in paper/simulators while live multi-exit brackets remain incomplete or broker-dependent
5	**Direct automated live order placement from strategy scripts**	Strategy engines commonly fill via a broker *emulator*; real exchange orders need external webhooks or bridges
6	**Unified routing** of strategy orders through the same trading panel path as manual/paper orders	Script strategies and discretionary trading panels are often separate stacks

Strategy model: portfolio, hedge, and fill realism

#	Capability	Why it stays open
7	**Native multi-symbol / portfolio strategy backtesting**	One script run is typically bound to one chart dataset; multi-asset P&L is export-and-compare, not a portfolio engine
8	**Simultaneous long + short (hedge) on the same symbol**	Position models are often single-direction at a time
9	**Positions in symbols other than the chart asset**	Cross-asset entries from one strategy remain unsupported on many hosts
10	**Full tick-level historical fill realism**	Fills usually walk chart OHLC with assumed open→high→low→close (or open→low→high→close) paths; higher-resolution path reconstruction is partial and often tier-gated
11	**Large trade histories without silent trim**	Non–deep backtests often cap order history (older trades dropped from testers); deep modes raise the ceiling but remain product-tier features

Script runtime, quotas, and tooling

#	Capability	Why it stays open
12	**Higher unique multi-series / remote-series request ceilings**	Cloud hosts hard-cap distinct series fetches per script (plan tiers only raise the number)
13	**Higher total script wall-time budgets**	Entire-run timeouts remain fixed per plan class
14	**Higher per-bar loop budgets**	Tight loop-per-bar limits reject heavy iterative logic
15	**Higher data, memory, and compiled-size caps**	Large libraries and data-heavy studies hit opaque resource walls
16	**Large multi-symbol scanning without burning unique-request quotas**	Dynamic loops still count each distinct symbol/timeframe against the unique ceiling
17	**Runtime profilers** (wall time, memory, compiled size) before failure	Authors discover limits by rejection or runtime error, not by measurement tools
18	**Interactive step-through debugger with breakpoints**	Official debugging remains logs, plots, drawings, and chart colors—not a classic debugger

Script-driven screener constraints

#	Capability	Why it stays open
19	**Multiple custom indicators per screener pass**	Many screeners allow only one user script per screen
20	**Indicator-on-indicator** composition inside the screener	Nested studies are unsupported
21	**Custom timeframes** in scripted screens	Non-standard bars often unavailable
22	**Higher remote-series call counts** inside screener scripts	Typical caps are very small (single-digit)
23	**Lookback beyond a short trailing window** (e.g. hundreds of bars only)	Screen math is truncated to recent history
24	**Full input-type coverage** for screener-hosted scripts	Several input kinds are unsupported in the screener surface

Chart UX and market coverage

#	Capability	Why it stays open
25	**Stable, discoverable watchlist gestures** (e.g. context-menu add)	Removal or relocation of common chart chrome produces sustained restore demand
26	**Timely exchange / venue chart coverage** when a market is popular	Coverage gaps surface as high-engagement “still missing” demand until a data feed lands
27	**Sector / industry / theme heatmaps** and similar multi-name overview boards	Recurring UX requests outside pure charting

Open script host vs general-purpose stacks

#	Capability	Why it stays open
28	**External data sources, HTTP, and host APIs from inside scripts**	Cloud sandboxes intentionally block outbound and arbitrary I/O
29	**Databases, file I/O, and machine-learning libraries inside the script runtime**	Language surface is chart-bound, not a general quant stack
30	**True custom UI and host programming** beyond chart plots and fixed panels	Outgrow path is usually a multi-asset engine with live broker APIs outside the chart host

---

Themes for AXIS

These gaps map cleanly onto AXIS axes without implying any third-party product parity:

Demand theme	AXIS lever
Tick / order-flow fidelity	Pluggable **sources** and **streams** with higher-resolution bars or trades when data exists
Portfolio multi-symbol strategies	Multi-source strategy evaluation and results beyond single-chart P&L
Live multi-exit and automation	Optional execution adapters / webhooks (explicit, user-owned—not silent live routing)
Runtime ceilings and profilers	Local and Worker engines with measurable limits you control
Interactive debugger	Step and pin tooling against the active engine (see UI debugging surfaces)
Screener scale	Offline or self-hosted scan loops unbound by a single vendor’s cloud quota
External data / ML / DB	Host bridges and plugins—not inventing closed chart-host APIs

AXIS remains a composition host: sources load history, streams advance the present, engines evaluate scripts, storage holds libraries. Capability work should preserve that separation.

Related docs

- Architecture overview — composition model
- Strategy and results (end user) — how strategy output is surfaced today
- Debugging (UI) — current debug surfaces
- Feature atlas — path index for implementation work

Research notes

- Synthesis date context: community and documentation signals through early–mid 2026.
- “Still open” means encoded as present limits or still treated as incomplete in high-engagement demand—not items already reversed by later product notes.
- Engagement is qualitative across forums and social posts; it is not a weighted global survey.
- Plan-gated historical depth and deep-backtest order ceilings partially address older “more bars / more trades” asks; those are de-emphasized relative to portfolio, execution, order-flow, and tooling gaps.

61 · REFERENCE

LEGACY SHELL

Pre-Solid static shell vs current AXIS product path — what to ignore and what still runs.

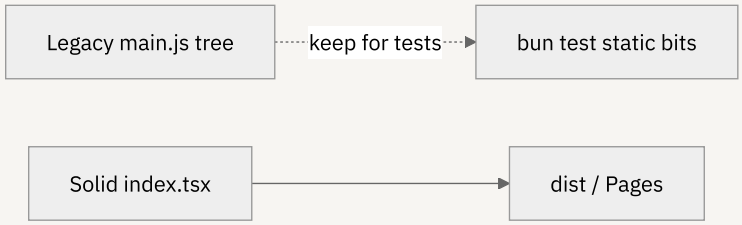
LEGACY SHELL

Abstract

AXIS's shipping product path is **Solid + Vite**. An older **static shell** remains in the repo for historical scripts and some Bun tests. Do not document or extend the legacy shell as the primary UI.

Source of truth: `frontend/LEGACY.md`.

Conceptual model



What is legacy

Path	Role
`frontend/src/main.js`	Old bootstrap (chart.js, legacy-topbar.js, …)
`frontend/style.css`	TV-blue-era tokens
`frontend/pyne-editor.js`	Pre-Solid CM6 wiring (renamed from `pine-editor.js`)
`frontend/server.ts`	Bun static server for old tree
Root / frontend `sw.js` (static shell SW)	Service worker for old shell

Makefile `make run-frontend` still prints AXIS and runs `bun run frontend/server.ts` on `:8081` — useful for legacy debugging, **not** the Solid product path.

What is current

Path	Role
`frontend/src/index.tsx`, `app.tsx`	Solid app
`frontend/src/chart/ChartHost.tsx`	LWC panes
`frontend/src/ui/*`	Topbar, Settings, Results, Manager, …
`frontend/public/` + `bun run build`	Production PWA assets
`frontend/axis_pwa_server.py`	Serve **dist** for demos

Migration residue

Residue	Handling
`pynescript.axis.plugins.v1`	Plugin install list
`pynescript.axis.library.v1`	Library localStorage fallback
`pynescript.axis.editor.doc`	Editor document backup
CF Pages project `axis`	Pages id; canonical `https://axis.hoox.sh`
CF Worker `worker-axis`	Script name; URL `https://worker.axis.hoox.sh`
Some docs/Makefile strings	Current brand is AXIS

App state namespace: `pynescript.axis.v1`.

Invariants

1. **Do not port new features into `main.js`.**
2. **Prefer Solid store** (`src/store`) over `state.js`.
3. **Plugin registry** is the Solid path (`src/plugins/registry.ts`); dual registries in old JS should be treated as dead ends.

4. **Pages deploy** should use Vite `dist/`, not the repo root static tree.

Failure modes

Confusion	Clarification
“UI doesn’t match docs”	You may be serving the legacy shell
Plugin manager missing	Legacy topbar lacks Solid Manager
Tests pass, UI wrong	Unit tests may still import legacy helpers

See also

- Feature atlas
- Build and serve
- AXIS hub

TESTING REFERENCE

AXIS unit, worker, security, coverage gate, and Playwright e2e conventions with paths and commands.

TESTING REFERENCE

Abstract

AXIS tests are **Bun-first** under `frontend/`, with Playwright for browser smoke and a scoped coverage ratchet. Canonical human guide: `frontend/TESTING.md`. This page is the reference map for agents and contributors.

Conceptual model

Syntax error in text
mermaid version 11.16.0

Commands

```
cd frontend

bun run test                # unit + worker tests
bun run test:unit           # frontend/tests only
bun run test:coverage       # lcov + text → coverage/
bun run test:coverage:gate  # coverage + 95% scoped gate
bun run test:security       # tests/security/
bun run test:e2e:smoke      # Playwright @smoke
bun run test:e2e:critical   # @smoke|@critical
bun run test:e2e            # all
bun run test:all            # gate + security
```

Root:

```
bun run test:frontend
bun run test:frontend:coverage
make test-frontend
```

Layout

```
frontend/tests/
  setup.ts
  fixtures/
  helpers/
  *.test.ts
  integration/
  security/
frontend/scripts/check-coverage.mjs
frontend/e2e/
  smoke.spec.ts
frontend/playwright.config.ts
frontend/worker/tests/
  auth.test.ts
  keys-*.test.ts
  scripts*.test.ts
  …
```

Coverage policy

Item	Policy
Gate	**95%** lines on scoped core
Ratchet history	70 → 80 → 83 → 87 → 90 → **95**
Include	plugins, storage (minus idb), store, results, sources, streams, data, chart pure helpers + series-factory/manager-access, worker auth/keys/runtime/scripts
Exclude from gate	`drawing-layer.ts`, pane-manager (unit elsewhere), runner chart apply, pyodide boot, legacy JS, Solid UI `.tsx`
Full report	`bun test --coverage` unscoped

Conventions

1. `import './setup'` first when tests touch store/plugins/storage.
2. **No real network** — mock `fetch` / `WebSocket`.
3. Table-driven tests for catalog built-ins.
4. Plugin fixtures in `tests/fixtures/plugins/`.
5. Prefer calling handlers with fake params over full browser for unit scope.

Playwright

Config (`playwright.config.ts`):

- `testDir: ./e2e`
- Chromium project
- `webServer`: `bun run build && bun run preview -- --host 127.0.0.1 --port 4173`
- `baseURL`: `http://127.0.0.1:4173` (override `AXIS_E2E_BASE`)
- Timeout 60s; CI retries 1

Smoke mocks `/run` and Binance. Use `data-testid` attributes (`axis-btn-load`, `axis-manager`, ...).

Security suite

`bun run test:security` covers:

- Plugin URL scheme rejection
- Storage-via-URL rejection
- Poisoned localStorage
- Worker partition isolation
- If-Match 409
- Admin keys

Adding a unit test

1. Create `frontend/tests/<area>.test.ts`.
2. Import setup if needed.
3. Use fixtures under `tests/fixtures/`.
4. Run `bun run test:unit` and `bun run test:coverage:gate`.

CI linkage

See CI and testing for workflow job names and nightly schedule.

Failure modes

Failure	Mitigation
Coverage drop	Add tests in scoped files or deliberately exclude only with policy change
E2E flake	Mock network; single worker; stable testids
IDB in unit	Local storage falls back to memory maps

See also

- CI and testing
- Plugin contracts
- Worker auth

PLUGIN EXAMPLES

Worked AXIS example plugins: CoinGecko source, Tiny Pine engine, Cloudflare DO stream — load paths and contracts.

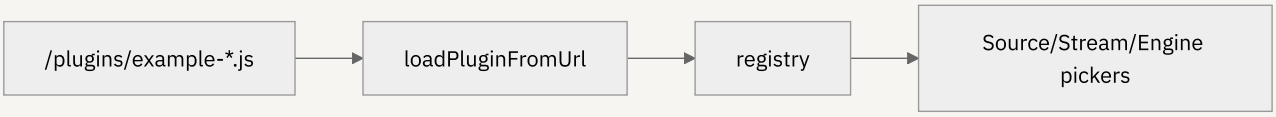
PLUGIN EXAMPLES

Abstract

AXIS ships three **loadable example plugins** as ES modules under `frontend/public/plugins/` (also mirrored for reference under `frontend/src/plugins/example-*.js`). They are not built-ins; install via Manager → Plugins → **Load from URL**.

Full narrative: `frontend/src/plugins/README.md`.

Conceptual model



Loading

1. Serve the PWA (Vite dev, preview, or `axis_pwa_server.py` on `dist`).
2. Open **Manager → Plugins**.
3. Paste a same-origin URL, for example:

```
http://127.0.0.1:8081/plugins/example-coingecko-source.js
http://127.0.0.1:3000/plugins/example-tiny-pyne-engine.js
http://127.0.0.1:8081/plugins/example-cf-do-stream.js
```

4. **Load** — registry updates; pickers show the new plugin.
5. Persistence: `localStorage` `pynescript.axis.plugins.v1` restores on next visit.

You may host modules on any origin that allows **module** CORS (same-origin is simplest).

Example catalog

File	id	kind	Purpose
`example-coingecko-source.js`	`coingecko`	source	Public CoinGecko market_chart → synthetic OHLC
`example-tiny-pyne-engine.js`	`tiny-pyne`	engine	In-browser JS DSL: sma/ema/rsi/plot/strategy
`example-cf-do-stream.js`	`cf-do`	stream	WS to Worker `/api/stream` SessionDO
External: `pyne-agent-worker` `/plugin/axis-pine-agent.js`	`pyne-agent`	component	NL → PYNE Agent chat (Cloudflare® Workers AI™); docs

Paths:

- Served: `frontend/public/plugins/`
- Source copies: `frontend/src/plugins/example-*.js`

CoinGecko source

Contract: `kind: 'source'`, `fetchHistorical`.

Config: `baseUrl` (default CoinGecko v3), `vsCurrency` (default `usd`).

Behavior:

- Maps common symbols (`BTC` → `bitcoin`, …).
- Requests `market_chart` for a day window derived from interval × bar budget.
- Synthesizes OHLC from consecutive price points (API is not full candles).

Limits: public rate limits (~10–30 calls/min); CORS depends on CoinGecko policy.

Use when: demo custom source without a key.

Tiny Pine engine

Contract: `kind: 'engine'`, `isReady`, `run`.

Behavior:

- Tokenizes a tiny expression language.
- Built-ins: `close`, `open`, `high`, `low`, `volume`, `sma`, `ema`, `rsi`, `plot`, `strategy.entry` / `strategy.close`.
- Returns `RunResult` with plots/events — **not** full PYNE fidelity.

Use when: offline demos without loading ~14MB Pyodide or hitting Flask.

Cloudflare DO stream

Contract: `kind: 'stream'`, `start` → stop.

Config: `endpoint` — Worker base URL (`https://…` or `http://127.0.0.1:8787`). Empty → error.

URL built:

```
{endpoint as wss}/api/stream?session={symbol}@{interval}&symbol=…&interval=…
```

Requires: Worker with `SESSIONS` Durable Object bound and deployed (durable objects).

Use when: many tabs share one upstream Binance kline subscription.

Minimal authoring templates

Source

```
  id: 'my-source',
  name: 'My Source',
  kind: 'source',
  description: 'Shown in Manager',
  configSchema: {
    baseUrl: { type: 'string', default: 'https://example.com', label: 'Base URL' },
  },
  async fetchHistorical({ symbol, interval, config }) {
    const res = await fetch(`${config.baseUrl}/bars?symbol=${symbol}&interval=${interval}`);
    return res.json(); // Bar[]
  },
};
```

Stream

```
  id: 'my-stream',
  name: 'My Stream',
  kind: 'stream',
  start({ symbol, interval, onBar, onError, onStatus }) {
    const ws = new WebSocket('wss://example.com/stream');
    ws.onopen = () => onStatus({ state: 'open' });
    ws.onmessage = (ev) => {
      const b = JSON.parse(ev.data);
      onBar({ time: b.t, open: b.o, high: b.h, low: b.l, close: b.c, volume: b.v });
    };
    ws.onerror = () => onError(new Error('ws error'));
    return () => ws.close();
  },
};
```

Engine

```
  id: 'my-engine',
  name: 'My Engine',
  kind: 'engine',
  async isReady() { return true; },
  async run({ script, bars }) {
    return { status: 'success', plots: bars.map((b) => b.close), events: [], series: {}, meta: {} };
  },
};
```

Storage plugins **cannot** be loaded from URL yet.

Invariants & edge cases

1. Treat plugin URLs as **executable code**.
2. `normalizePluginUrl` rewrites `/src/plugins/` → `/plugins/` for production.

3. Dangerous schemes rejected (`javascript:` , `vbscript:` , `data:text/html`).
4. After load, window may emit product-specific events (see README) for UI extensions.

Failure modes

Issue	Fix
Failed to restore URL	404 after deploy path change
CORS on CoinGecko	Offline mock or proxy
DO stream errors	Bind SessionDO; set endpoint
Tiny engine parse errors	Unsupported DSL constructs

See also

- Dynamic loader
- Contracts
- Worker durable objects

CHANGELOG

Release notes for AXIS.

All notable changes to **AXIS** (`hoox-sh/axis`) are documented in this file.

This changelog is **recursive**: it lists the full git history of the repository, grouped by month and conventional-commit type. Agents and humans **must keep it updated** on every release (see `AGENTS.md` § Changelog & releases).

Format roughly follows Keep a Changelog with commit SHAs for traceability.

Generated/updated: 2026-09-12 · 401 commits · describe-tag: `v2.6.5`

[Unreleased]

Added

- **Convert to Pine v6** in the editor overflow menu and command palette. Rewrites any older version (missing `//@version` counts as v1) toward v6: colors / `n` / timeframe names, `ta.*` / `math.*` / `request.*`, `study(` → `indicator(`, typed `input.*()`, `iff` / `offset`. Not a semantic v6 migrator.
- **Alerts panel**: create form covers percent-change, drawing-touch, and indicator plot conditions (not only price / on-chain). Delivery is browser notifications, a compact HTTPS webhook, and an optional L2 webhook (`channel: "l2"` payload). Live stream ticks evaluate the book.
- **Pine `alert()` / `alertcondition()`**: interpret runs export engine `alerts` into the book (`pine_alert`). Arm any/`alert()` / `alertcondition()` (optional title) on a strategy or indicator; last-bar watermark avoids backtest spam. Strategy `alert_message` fills are included. Pyodide interpret exports the same payload.
- **Compile-path Pine alerts**: in-browser compile records `alert()` / `alertcondition()` in the object-mode bar loop (PYNE's compiler used to emit empty statements for Numba). Remote Flask/pyne-worker compile still cannot collect Python alert events, so AXIS sends **interpret** when the script calls those builtins.

Fixed

- Fold marker labels were inverted (expanded blocks announced “Unfold”). They now read **Fold block** / **Unfold block**.

Changed

- **Workstation chrome polish**: two-bar command + module switch, 4/8px grid, 1px hairline, periwinkle `#8B9CFF` on `#07080C` / `#0C0E14`, Live as a 6px pulse, HUD capsules, 32px watchlist rows, 32×32 drawing rail, 26px status strip, tablet overlay drawers, mobile Chart/List/Editor/More. Same product — density and grouping only.
- **Cloudflare® rename**: Worker script `pynescript-axis` → `worker-axis` (`https://worker.axis.hoox.sh`). Pages project `pynescript-axis` → `axis` (`https://axis.hoox.sh`; previews `*.axis.pages.dev`). Health JSON `service` is `worker-axis`. Legacy `pynescript-axis.pages.dev` and `pynescript-axis.*.workers.dev` hosts still match in CORS / client heuristics.
- **Live stream on by default**: Settings → General → Enable live stream (`live.preferAfterLoad`, default on). Saving starts or stops the venue WebSocket; boot starts live when history is already loaded. Persist no longer writes a false default when the flag was missing.
- **Server/Pyodide execution default is Auto** (try compiler, fall back to interpreter). Strict Compiler is not the default: Flask/Numba is not prewarmed, first compile JITs, and some Pine stays object-mode. HUD and first-run now follow the engine schema instead of showing Interpreter when mode was never saved.
- **Editor fold gutter**: 14px Lucide chevrons replace the 10px `▾` / `▸` glyphs (down = expanded, right = folded), with a clickable `⋯` placeholder on folded blocks. Tab close/add and header Run/Library/Profiler use the same icon set and size.
- **Editor color tools**: one working color (picker overlay + any-form input + transparency) instead of a disconnected converter that stayed on `#939fff`. Chip captions use `#RRGGBB tN` so `color.new(...)` is not truncated; formats list copies the draft; Replace / Replace all / Jump; checkerboard swatches show Pine transparency.

[2.6.6] — 2026-09-12

Fixed

- **`axis secret put ADMIN_TOKEN` vs Cloudflare® 10053** (`@hoox-sh/axis-cli` 0.3.2): empty `[vars] ADMIN_TOKEN = ""` is still a binding, so Wrangler refuses the secret. `axis secret put` now comments the plaintext var, deploys once to drop it, retries on 10053, then sets the secret. `axis keys create` prompts for the token (CLI cannot read Worker secrets) and doctor warns on a leftover `[vars]` stub.

[2.6.5] — 2026-09-12

Added

- **AXIS CLI cloud-storage bootstrap** (`@hoox-sh/axis-cli` 0.3.1): `axis setup --prod` / `axis setup kv` create and bind `API_KEYS` KV; `axis keys create` / `validate` mint and check `pn_…` keys; `axis deploy` applies the remote D1 schema first; `axis health --scripts` and `axis doctor --remote` probe `/api/scripts` and `features.keys`. Typical prod path is now `setup --prod` → `secret put ADMIN_TOKEN` → `deploy all` → `keys create`.

Changed

- CLI `run()` now times out Wrangler (`wrangler whoami` in doctor) with SIGTERM then SIGKILL so `axis doctor --json` cannot hang.

[2.6.4] — 2026-09-12

Added

- **Script version history on every storage engine**: local (IndexedDB/`localStorage` snapshots, FIFO 50) and cloud Worker (`script_versions` + `GET /api/scripts/:id/versions`) now implement the same `listVersions` / `readAtRevision` contract as git. The library history panel is no longer git-only.
- **Worker URL + API key in Settings**: Script storage credentials for the cloud Worker (`endpoint` + `pn_…` key, generate demo key, test connection) live in Settings → General / Studio Settings as well as Script Library. Switching to Cloud no longer requires hunting the library panel first.
- **Workers inventory skeleton**: while backends are probing, the Workers page shows pulse placeholder cards (and an inspector skeleton) instead of an empty grid.

Fixed

- **Cloud script storage used the Pine engine URL**: `resolveCloudConfig` no longer falls back to `store.endpoint` (`:5002` / `/api/run`). Worker base defaults to wrangler `:8787` on localhost or the production workers.dev host.
- **Storage engine switch deleted the source library**: copy between engines never calls `remove` on the source. Dialog primary action is **Copy scripts**; **Switch without copying** leaves both libraries as-is.
- **Minimap height restored**: short files no longer stretch the overview canvas to the full editor column — 2px per line, compressed only when the document exceeds the host.
- **Update reload asks first**: Update now / Hard reload (and `hardReload` itself) open a confirm prompt with current → new version before the app reloads.
- **Studio always opens Settings**: the topbar / palette Studio button lands on Settings every time (no longer restores the last rail page). Settings is the first Workspace rail item. The Studio click handler no longer falls through `??` into Runtime then Workers because those callbacks return `void`.
- **Workers is a full-viewport studio page**: same overlay as Settings/Runtime/Wire/Plugins (not a half-width right sheet). Escape closes it.

Changed

- Storage-change dialog copy: “Copy scripts” / “Switch without copying” instead of migrate-and-delete / start fresh.

[2.6.3] — 2026-09-12

Fixed

- **Docker image build failed on `sync:versions`**: `bun run build` stamps Dockerfile/compose/bake VERSION fallbacks, but the image context did not copy those files → `ENOENT /app/Dockerfile` during GHCR bake. Missing stamp files are skipped, and the three files are copied into the build stage.

[2.6.2] — 2026-09-12

Added

- **Editor minimap (interactive)**: a scaled canvas overview docked to the editor right edge — one row per line, accent viewport box, cursor line. Click/drag scrolls the editor; auto-hides in narrow containers. Toggle in the editor overflow menu (View → Minimap, persisted `editorMinimapEnabled`, default on).
- **Foldable code sections**: indent-based folding for Pine (StreamLanguage has no syntax tree) with a void-styled fold gutter plus standard fold keybindings.
- **Indent grid**: faint `·` dots for leading whitespace plus one vertical guide per completed 4-space indent level, width-preserving and viewport-scoped.
- **Per-tab version/type badges + Save**: each editor tab shows `vN` (from `//@version=`, `–` fallback) and `STR` / `IND` / `LIB` (from the declaration call, `–` fallback) plus a Save button that persists that tab via the existing library/draft path.
- **Workers stays a right-side non-modal sheet**: the Workers overlay remains a right-aligned sheet with a left scrim that click-throughs to the chart/editor (modality unchanged: no focus trap, no Escape-steal) — not a full-width page like Runtime/Wire/Settings/Plugins.
- **Language Feature Bar above the editor status strip**: a chip row (Hover · Sig · Complete · Lint · Marks · Chips · Remote) where each chip is colored only while the feature is actually firing (open card / list / hint, running check, rendered marks) — dimmed while switched off — so the bar shows which features the current script uses and which could be disabled. Click toggles the master switch, long-press (~550ms) or right-click opens a popover with the group's full settings. The row itself is a persisted “Language Feature Bar” option (Settings → Editor intelligence → View, default on; `editorFeatureBarEnabled` in the store).
- **Standalone CLI binaries (bun compile)**: `bun run build:binaries` (`packages/cli/scripts/build-binaries.ts`) cross-compiles the CLI into single-file executables for 7 targets — linux x64/arm64 (glibc + musl), macOS x64/arm64, Windows x64 — with a native `--version` / `--help` smoke test. Own package metadata is now embedded at bundle/compile time (`src/own-package.ts` static JSON import) and preflight recognizes a new `binary` install context (`$bunfs` → "standalone binary (bun compile)"), so version display, engine checks, and drift detection work without a package.json on disk. Binaries ship as `axis-cli-<version>-bun-<target>` release assets and are covered by `SHA256SUMS` (bare asset filenames — a flat download verifies with one `sha256sum -c`).
- **`VERSION` single source of truth**: the repo-root `VERSION` file now owns the release number — `bun run sync:versions` (runs with `bun run build`) stamps `package.json`, `src-tauri/tauri.conf.json`, `src-tauri/Cargo.toml`, `src-tauri/Cargo.lock`, `public/version.json` (served at `/version.json`), and the generated `src/version.ts` (`APP_VERSION`, also shown in About) + `worker/src/version.ts` (`WORKER_VERSION`, surfaced on `/` + `/health`). `bun run check:versions` asserts the whole set; the Makefile reads `VERSION` directly.
- **Update manager + notification**: the app polls the deployed `/version.json` (5 min interval + focus/visibility/online, cache-busted) and a waiting service worker routes through the same state — a new deploy shows an `axis-update-banner` (Update now / Hard reload / Later), announces via screen-reader live region + status line, and fires a Browser Notification when permission was already granted (never prompts). Hard reload unregisters workers, deletes `axis-*` caches, and navigates cache-busted. Covered by `tests/update-manager.test.ts` + `tests/version.test.ts`.
- **Window-close confirmation**: `src/pwa/close-guard.ts` installs one `beforeunload` listener per page and prompts only while a registered predicate reports unsaved work — the tabbed editor registers dirty-tab state (both chart and popout shells). Master switch + custom message APIs included; covered by `tests/close-guard.test.ts`.

Fixed

- **Pyodide engine crashed on bars containing null (`eval_code` traceback + empty drawings)**: the browser bridge interpolated raw `JSON.stringify(bars)` into Python source, so JSON `null` / `true` / `false` became Python `NameError`s before Pine ever ran — surfacing only as a truncated `_pyodide/_base.py` traceback with `drawings: []`. `src/engines/catalog.ts`, `src/engines/index.js`, and `worker/src/pyodide_runtime.ts` now pass JSON strings via `globals` + `json.loads`, return `drawings: []` on bridge failures, and surface the full Python traceback (new `callPyodideRunScript` / `formatPyodideBridgeError`, covered by `tests/pyodide-bridge.test.ts`). `pyodide/pynescript_runtime.py` re-synced with the deployed `public/pyodide` copy (drawing-registry reset + compile-path GC).
- **Pyodide bridge errors kept their head, not the exception tail**: `formatPyodideBridgeError` cut long tracebacks with `full.slice(0, 6000)`, dropping the final `NameError: …` line that users need; it now trims the *middle* (keeps the `eval_code` head + the exception tail) in `src/engines/catalog.ts`, the legacy `src/engines/index.js`, and `worker/src/pyodide_runtime.ts` — which previously surfaced only `err.message` (Pyodide's `…`-truncated form). Covered by a new `tests/pyodide-bridge.test.ts` case.
- **Language Feature Bar nits**: clicking a chip whose settings menu is open now dismisses the popover instead of flipping the group's master switch (long-press opens the menu; the natural dismissal — a short press on the same chip — accidentally toggled the feature off). Number settings (Hover delay, Max options, …) now draft raw keystrokes and clamp/commit on blur or Enter, so values below a field's `min` can be typed instead of snapping mid-entry. Clicking a *different* chip while a popover is open switches the popover without toggling.
- **Test DOM stub Symbol-safety**: `tests/setup.ts` `dataset` proxy now passes through non-string keys (e.g. `Symbol.toPrimitive`) instead of throwing on `console.log` /inspect, and the traceback-trim budgets are named constants kept in sync across `src/engines/catalog.ts`, `src/engines/index.js`, and `worker/src/pyodide_runtime.ts`.
- **Docker builds were broken by the git-hooks `prepare` script**: root `package.json` gained `"prepare": "git config core.hooksPath .githooks"` (commit `d5f0cf89`), which runs on every `bun install` — including inside the image `deps` stage, where git is not installed → exit 127 on every bake since then (v2.6.1 GHCR images never published). The Dockerfile now installs with `--ignore-scripts` (no dep needs lifecycle scripts; native binaries ship as platform optionalDeps) — verified with a local `build` target bake (`dist/.version` = 2.6.1). Stale version fallbacks (`2.3.0` / `2.3.1`) refreshed to `2.6.1` across `Dockerfile`, `docker-bake.hcl`, and `docker-compose.yml`.
- **Service worker no longer cache-first `/version.json` probes**: cache-busted `/version.json?t=*` polls were classified as same-origin static and stuffed unique entries into the 96-entry runtime cache, evicting hashed chunks and `/pyodide/*`. Those probes now bypass the SW. Cache names bumped `v6` → `v7` so poisoned runtime caches are dropped on activate.
- **Close-guard disabled before update hard/soft reload**: the unsaved-work `beforeunload` prompt blocked Update now / Hard reload; the update path now disables the guard immediately before navigating.
- **Drawings e2e counts user-drawing SVG only**: the chart overlay SVG count no longer includes chrome/grid paths, so the trend-line persistence assertion is not a false pass.
- **`openStudio` no longer toggle-closes a slow overlay**: opening Studio while the sheet is still animating in no longer treats a second click as a close.
- **CLI binary install URL documents app tag vs CLI version**: GitHub Release tags use the app `VERSION` (`v2.6.2`); standalone binaries are named with the CLI package version (`axis-cli-0.3.0-bun-linux-x64`). The CLI README curl snippet now uses both current values so the documented URL is not a 404.
- **Docker build stamps `VITE_APP_VERSION` from the `VERSION` file**: compose/bake ARG fallbacks are no longer the Vite stamp, so `docker compose up --build` without `VERSION=` matches `/version.json`. `Dockerfile` `ARG VERSION=`, compose `${VERSION:-}`, and bake `variable "VERSION"` defaults join `sync:versions` / `check:versions`.
- **Feature bar long-press/contextmenu no longer race-dismiss; dialog focus**: the chip popover no longer closes on the same pointer that opened it, and the settings dialog receives focus.
- **Minimap pointercancel + full-height hit target**: `pointercancel` ends a drag the same way `pointerup` does, and the hit area spans the editor height so clicks near the gutter register.
- **Workers non-modal sheet restored**: Workers is a right-aligned sheet again (not full-width), with a left scrim that click-throughs to the chart/editor.
- **`IndentWidget.eq` so guides don't rebuild on every scroll**: indent-grid widgets compare equal across viewport updates, so CodeMirror does not reconstruct guides on each scroll.

Changed

- **Coverage push toward 80/85%**: ~260 new unit tests across gateway, signed-fetch, pine-logs, watchlist tickers/live, symbol catalog, bars-cache, streams catalog, workers probe, git-oauth, dataset sinks/expand, data-source-manager, credentials, and drawing coords (`createCoordContext` with stub scales — coords 36% → ~100%) — scoped core gate up 82.8% → ~90% (passes the 80% and 85% floors), app lines 80.1% → 82.6%. Fixed 5 wrong expectations in the generated data-coverage tests (walk-loop `expanded` semantics, same-bucket merge conflicts, sink-merge unions) and 2 cross-suite leaks (probe teardown deleting the shared `window` stub, a real-timer device-flow poll). Overall lcov sits at ~79.3% — the remainder is browser/chart/worker-runtime code (drawing layer, pane manager, pyne-lsp, durable session) that needs E2E rather than unit tests.
- **Drawings e2e + chart DOM harness**: new `e2e/charts.ts` harness (stubbed network, mock-bar boot, toolbar tool selection, chart clicks, SVG shape counting) and `e2e/drawings.spec.ts` — draws a trend line via the real toolbar, asserts the SVG overlay renders, reloads to prove storage persistence, and switches tools. Covers the browser-bound drawing layer that unit tests cannot reach (no layout engine/canvas in bun).
- **GitHub Release now carries CLI + worker artifacts**: a new `release-assets` job in `.github/workflows/release.yml` attaches, for every `v*` tag, the offline-install CLI tarball (`axis-cli-<cli-version>.tgz` via `npm pack` — verified with an offline `npm i -g` + `axis --version` smoke), a deployable worker source snapshot (`axis-worker-<tag-version>.tar.gz` — `LICENSE` + `worker/` minus `node_modules` / `.wrangler` / `.dev.vars`, sanity-checked for `wrangler.toml.example` + `src/index.ts`), and `SHA256SUMS-<tag-version>.txt`. The job waits for the desktop workflow to create the Release (creates it itself if desktop never does) and uploads with `--clobber`; existing tags can be backfilled via `gh workflow run release.yml --ref main -f tag=vX.Y.Z`.
- **CLI binary smoke test runs on host-native targets**: `packages/cli/scripts/build-binaries.ts` no longer assumes a linux-x64 host — it smoke-tests whichever wanted target matches the running platform/arch (e.g. `bun-darwin-arm64`) and skips cleanly when none does, so `bun run build:binaries` cross-compiling on macOS/ARM64 dev machines no longer fails on the smoke step. Classic CI (ubuntu-x64) smoke is unchanged.
- **Pyodide runtime surfaces drawing-GC failures instead of silently no-opping**: `pyodide/pynescript_runtime.py` (synced to `public/pyodide`) now warns once on stderr when `DrawingRegistry` export/limits or the compile-path `gc_exported_drawings` call fails (e.g. a pyne wheel API change/move) rather than swallowing the error while `meta` still advertises the caps. `max_*_count` declaration parsing now strips `//` and `/* */` comments so a cap mentioned in a comment isn't captured, and the redundant `"mode"` entry is dropped from `compile_meta`.

[2.6.1] — 2026-09-08

Fixed

- **Desktop bundles carried the wrong version**: the v2.6.0 GitHub Release shipped desktop assets named `2.4.1` because `src-tauri/tauri.conf.json`, `src-tauri/Cargo.toml`, and `src-tauri/Cargo.lock` were never bumped past v2.4.1 — `tauri-action` derives bundle versions from the Tauri config, not the git tag. All four version stamps are now `2.6.1`, and a new `bun run check:versions` guard (`scripts/check-versions.mjs`, enforced in CI + the desktop workflow) fails on any drift.

Changed

- **Script Logs moved into the editor**: the topbar `Script Logs` button and the shell-level `ScriptLogsPanel` (`src/ui/ScriptLogsPanel.tsx`, deleted) are gone — logs now live as an editor-local `ScriptLogsPane` (`src/editor/ScriptLogsPane.tsx`) above the editor statusbar, toggled by a `Logs` button beside `Symbols` (`axis-editor-logs-toggle` + `axis-editor-logs-badge`) and via the palette (`axis-editor-show-logs` event). Statusbar clusters now wrap as units (`flex-wrap`, actions `flex-shrink-0`) so buttons stay clickable in narrow docks instead of squeezing under each other. Guards: Playwright smoke `Script Logs lives in the editor bottom bar, not the topbar`.
- **Default timeframe is now 15m**: the chart-slot fallback (`src/chart/layout.ts`) and workspace-snapshot build/parse fallbacks (`src/storage/workspace-snapshot.ts`) changed from `1d` to `15m`, matching the store default — new chart slots and restored snapshots without an interval start on the 15-minute TF.
- **Studio opens on Settings**: `lastStudioPage` initial in `src/app.tsx` changed `runtime` → `settings` — the first Studio open (topbar button / palette) lands on Settings (subsequent opens still remember the last page).
- **Workers page is non-modal**: `AppPage` (`src/ui/studio/AppPage.tsx`) gains a `modal` prop; `StudioHost` renders Workers non-modally — transparent click-through scrim (new `.ax-page-backdrop--nonmodal` sheet in `studio.css`), no focus trap, no Escape-steal — so the chart and editor stay interactive while Workers is open. Modality follows the active rail page; other pages are unchanged.
- **Workers top status grid removed**: the aggregate Healthy / Health / Active backend summary deleted from `src/ui/workers/WorkersPage.tsx` — status now exists only per worker (inventory cards + detail pane + footer probe line). Guards: Playwright smoke `Workers page is non-modal` (scrim pointer-events, no top status, Escape does not close).

Added

- **Versioned git hooks** (`.githooks/`): **pre-commit** runs Biome (`check`) on staged files only (fast); **pre-push** runs `bun run typecheck` (tsc) + `bun run lint` (errors fail, warnings pass). Auto-configured via the root `prepare` script (`git config core.hooksPath .githooks`) on every `bun install`; bypass with `--no-verify`.
- **CLI installation self-check** (`@hoox-sh/axis-cli` 0.3.0): every `axis` command self-verifies on start (`packages/cli/src/utils/preflight.ts`) — engine compliance (Node ≥ 20 / Bun ≥ 1.2), install context (npm install / npx cache / repo checkout), and **version drift** when a globally installed CLI is older than the repo checkout it operates on (warns `npm i -g @hoox-sh/axis-cli`). Warnings-only on stderr: skipped for `--version` / `--help`, silenced by `--quiet` / `--json`, never blocks or fails a command. `axis doctor` gains the optional **CLI installation** row (`cli-install` in `--json`). Tests: `packages/cli/tests/preflight.test.ts` (context detection, engines, drift, doctor row).

Changed

- **Docs are CLI-first**: all `bun run axis:*` invocations across `README.md`, `AGENTS.md`, `docs/**` (cloudflare, local-dev, index, topologies, vps-demo), `worker/README.md`, and `packages/cli/README.md` replaced with the primary `axis <command>` form; `bun run axis:*` is now documented as a repo-only alias (alias tables flipped in `docs/devops/cli.mdx` and `docs/enduser/getting-started/installation.mdx`). New **Installation self-check** section in the CLI docs.

[2.6.0] — 2026-09-08

Added

- **Keypress feedback for consumed shortcuts**: `dispatchShortcut` now emits an `axis-shortcut-fired` CustomEvent (id / chord / description — `src/ui/shortcuts/runtime.ts`) and a transient bottom-center chip (`src/ui/shortcuts/Feedback.tsx`, `data-testid="axis-shortcut-feedback"`, `aria-live="polite"`, pointer-events none, ~1.2 s auto-hide, platform-correct chord glyphs via `formatChord`) confirms every consumed binding — main shortcut keys no longer feel dead even when the side effect is subtle.

Changed

- **Docs synced to v2.5.0**: `docs/index.mdx` + `docs/enduser/getting-started/installation.mdx` version stamps; `docs/ui/ui-shell.mdx` gains **Keyboard shortcuts** (dispatch hub, scopes, recorder, Shortcuts modal) and **Mobile shell (phones / tablets)** sections; `docs/ui/index.mdx` subsystem table mentions shortcuts + mobile shell; `docs/ui/results-and-strategy.mdx` and `docs/enduser/guides/strategy-and-results.mdx` document the v2.4.3 results overhaul (Events Stream/Open⇄Close position-cycle views, single-column strategy tab, saved-run stats snapshots, shared `walkStrategyEvents` walker).

Fixed

- **Keyboard shortcuts were dead: `ShortcutHub` was never mounted** (the component had zero importers, so the capture-phase keydown dispatcher in `src/ui/shortcuts/Hub.tsx` and the whole `src/ui/shortcuts/actions.ts` registration side effect never loaded) — `Ctrl/Cmd+K` / `Ctrl/Cmd+Shift+P` (command palette), `Shift+?` (shortcuts help), `Ctrl/Cmd+,` (settings), `Ctrl/Cmd+\` (toggle editor), `Esc`, and every `chart.*` binding (zoom / pan / tool letters / grid layouts) silently no-oped. The product shell (`src/app.tsx`) now mounts `(plus)`. Only the parallel `Mod-S` / `Mod-G` handler in `src/editor/tabbed-editor.tsx` survived, which is why save worked while the palette was unreachable by keyboard. Guard tests: `tests/shortcut-feedback.test.ts` and the Playwright smoke `Mod-K opens and closes the command palette (shortcut hub guard)` in `e2e/smoke.spec.ts`.

[2.5.0] — 2026-09-07

Added

- **Mobile-first responsive shell** (phones ≤767px, tablets 768–1023px; desktop ≥1024px unchanged):

- **Viewport signal** (`src/ui/responsive.ts`): reactive `phone | tablet | desktop` mode + `pointer: coarse` detection — all JS layout branching keys off this instead of CSS-only hacks, because panel geometry is persisted px state.
- **App-style phone shell**: compact header (brand · venue picker · symbol picker via SymbolModal · interval select) + bottom tab bar (Chart · Panels · Editor · Studio · More) replace the desktop Topbar/StatusBar; Panels list and More drawer (Run, Live, Studio pages, Settings, theme, About) open as nav overlays.
- **Panel sheets** (`FloatableShell` mobile mode): docked/floating panels render as full-viewport sheets under the header / above the tab bar — one active sheet at a time (`src/ui/panels/mobile-sheet.ts`); drag-to-undock, resize handles, and hover-slide are disabled on phones. Sheets dismiss via **swipe-down** on the title bar (grabber affordance + live translate feedback) or the close button. Desktop geometry is **bypassed, never overwritten**, so phone↔desktop switches preserve the user's layout. Phones boot chart-first (open panels don't steal the viewport).
- **Chart force-single on phones**: 2H/2V/4 layouts collapse to the active slot with a slot-switcher chip row; other slots stay preserved in `store.chartLayout`.
- **Tablet**: desktop shell kept, topbar becomes horizontally scrollable.
- **Dialogs → bottom sheets** on phones (`sc-dialog`), safe-area insets (`viewport-fit=cover`, `env(safe-area-inset-*)`), `100dvh` app root, `interactive-widget=resizes-content` for soft-keyboard handling, ≥44px touch targets in mobile chrome (`pointer: coarse`).
- **Studio tables** wrap cells on phones instead of forcing horizontal overflow.

- **PYNE runtime 0.5.0**: vendored `pynescript-0.5.0-py3-none-any.whl` (built from the sister pyne repo via `scripts/sync-pyne-wheel.sh`); stale 0.4.x wheels dropped; hard-coded wheel paths in `src/engines/catalog.ts` / `src/engines/index.js` updated (`hoox_pyne-0.4.2` / `pynescript-0.4.0` → `pynescript-0.5.0`). Pine language builtins unchanged.

Changed

- **Zero Biome lint errors** (368 → 0) across 641 checked files:
  - **Tooling**: Biome 2.5 (`biome.json`, lint-only — Prettier keeps formatting), `lint` / `lint:fix` scripts, Prettier 3.9 pinned (`.prettierrc` single-quote/width-100, `.prettierignore`), project `opencode.json` enabling LSP + formatter.
  - **Real fixes**: regex `exec`-loops → `matchAll()` (pine-colors, preevaluate, pyne-lsp, script-inputs, library-publish, …), `type="button"` on ~18 buttons, keyboard handlers (Enter/Escape) + `tabIndex` /ARIA value semantics on interactive widgets, `role="region"` divs → `<section>`, global-shadowing renames (`escape` → `escapeHtml`, `valueOf` → `readFieldValue`), duplicate object keys removed, `const enum` → `enum` (isolatedModules safety).
  - **Justified suppressions** (each with reason): intentional control-char sanitization regexes (injection safety), CSS `env(safe-area-inset)` fallbacks, backdrop click-away surfaces, custom-widget rows with nested buttons, sticky-regex tokenizer.
  - Verified regression-free: `tsc --noEmit` clean; test suite identical to pre-change baseline (pre-existing `server.test.ts` hook timeout + `window.matchMedia` cross-file pollution unaffected).

[2.4.3] — 2026-09-07

Added

- **Results panel — events views & richer reports** (spec: `.opencode/specs/2026-09-07-results-views-design.md`):
  - **Events tab view switcher**: `Stream` (classic chronological list, now enriched with entry fill count / running position qty+avg, `from_entry`, remaining qty after closes, and per-fill P&L on close/exit rows) and **Open ⇄ Close** — a two-column grid with one row per position: entries on the left (fill list, pyramiding averages into the position, total qty @ avg price), exits on the right (each close fill with P&L, partial-close remainder, Open/Closed badge). Re-opens after a full close form new cycles (`id #2`); `close_all` flattens every open cycle; rows click-jump to the fill on the chart.
  - **Strategy tab — single column**: fill options → stats cards (12: trades, win rate, W/L, PF, net profit, **Return %** (compounded per-trade), max DD, avg trade/win/loss, gross profit/loss) → equity chart → closed-trades table below it, with new **Bars** (bars held) and **Fills** (pyramided entry count) columns.
  - **Saved runs tab — rich stats**: `buildResultMeta` now persists a JSON-safe strategy stats snapshot (`StrategyStatsSnapshot`) computed from the run's events (`profitFactor: null` = ∞); saved rows render a stats grid (trades, W/L, win %, PF, net P&L, return %, max DD, avg trade); legacy saved rows fall back gracefully.
  - **Pyramiding parity**: entry fills into a live position now average into the trade P&L (`entry = Σ(p·q)/Σq`) matching the pyne broker `(px − avg) × close_qty`; per-position cycles and the enriched stream are computed by a single shared walker (`walkStrategyEvents` in `src/results/strategy.ts`, re-exported via `src/results/positions.ts`).

[2.4.2] — 2026-09-07

Added

- **Global keyboard shortcuts**: app-level chord dispatch hub (`src/ui/shortcuts/`) with capture-phase handling, dialog-skip guard, per-user persisted overrides on the app store, and a live shortcut registry — wired to the command palette, editor, and chart canvas.
  - **Recorder**: Settings → Keyboard → record chord, plus a **Shortcuts modal** listing default bindings with live rendering.
- **Editor line ops** (`src/editor/cm-line-ops.ts`): `Prec.high` keymap extensions (duplicate line, toggle comment, indent/unindent, delete line) for the Pine editor; palette extensions render recorded chords.
- **CLI npm release pipeline** (`.github/workflows/release.yml`): full publish flow for `@hoox-sh/axis-cli` via org secret `NPM_TOKEN_HOOXSH` —
  - tag/version guard: `cli-v*` tags must equal `cli-v<packages/cli version>` (hard fail on mismatch); root `v*` tags publish the CLI when its version is missing from the registry and skip cleanly otherwise; `workflow_dispatch` gains a `force` re-attempt input.
  - post-publish **registry verification** (`npm view` with propagation retries).
  - **Node-only smoke test**: fetches the published tarball, installs it into a clean directory with `--ignore-scripts`, and asserts `axis --version` / `axis --help` on Node (no Bun, no repo).
  - step summary now reports tag kind, registry verify, and smoke results.

- **`@hoox-sh/axis-cli` 0.2.2** — first CLI-only release via the new pipeline: `cli-v0.2.2` tag → `Release (npm)` workflow (provenance publish, registry verification, Node-only smoke test of the published tarball). Code identical to 0.2.1; release exercises the full publish path end-to-end.

Fixed

- **UI review findings (`/.opencode/review/`)** — 23 of the 25 consolidated findings (editor-audit + pass1/2/3) fixed across the editor, drawing chrome, panels, and themes:
  - **Escape friction cluster** (`AXIS-ED-ESC-OVERLAY`, `AXIS-P1-ESC-CANCEL-DRAFT`, `AXIS-UI-DRAW-MEASURE-ESC`, `AXIS-UI-KB-ESC-RECORD`): new CodeMirror Escape command dismisses suggest / hover / signature overlays in order; drawing drafts (measure, …) now always cancel on Escape (capture-phase handler immune to other preventDefault consumers); the Keyboard shortcut recorder records Escape as a chord with `stopImmediatePropagation` + a `data-axis-recording-chord` guard so the Studio page no longer closes mid-recording; conflict flags are scope-aware (`app.escape` vs `chart.cancel-draft` layer by design, no false conflict).
  - **Drawing flyout clipping** (`AXIS-P1-DRAW-MENU-CLIP`, `AXIS-UI-DRAW-MENU-CLIP`): Line / Fib / Shapes / Annot / Measure flyouts and the settings popover clamp to the nearest clipping ancestor — tall menus (13-item Shapes) scroll instead of spilling past the viewport, and flip upward when there is no room below.
  - **Callout / text prompts** (`AXIS-UI-DRAW-CALLOUT`, `AXIS-UI-DRAW-TEXT-FOCUS`): all blocking `window.prompt` calls (callout, note, anchored text, text tool, double-click re-edit) replaced with a non-blocking in-app dialog (`src/chart/drawings/tools/text-prompt.ts`) that autofocuses the input, commits on Enter, cancels on Escape — also fixes Tauri webviews where `window.prompt` is unsupported.
  - **Bar replay dismiss** (`AXIS-P1-REPLAY-DBL-CLOSE`, `AXIS-UI-REPLAY-DBLCLICK`): topbar toggle trusts both the subscribed UI state and the module session (a missed emission can no longer turn an exit click into a start), and Escape exits replay.
  - **Results staleness** (`AXIS-ED-RESULTS-STALE`): the store tracks the newest run (`newestRunId`); the script selector shows a "new run" marker when focus lags the footer.
  - **Status / System Logs clash** (`AXIS-UI-STATUS-LOGS-CLASH`): the System Logs expanded body is clamped to the viewport (reserving topbar + workspace + status strip), so an oversized persisted height can no longer push the Status strip off-screen.
  - **Diagnostic cascade** (`AXIS-ED-DIAG-CASCADE`): the pre-eval balance scanner no longer carries an unterminated string across lines (one bad quote suppressed every later structural diagnostic), and remote parse errors only shadow local errors on the same line — semantic errors after a parse failure now surface.
  - **A11y live region** (`AXIS-ED-A11Y-LIVE`): diagnostics changes announce politely via the shared SR live region; the Problems count is a `role="status"` live element.
  - **Long-line clip** (`AXIS-ED-LINE-CLIP`): `overflow-wrap: anywhere` on editor content so wrap mode breaks long tokens instead of clipping.
  - **Contextual autocomplete** (`AXIS-ED-AC-CONTEXT`): completions rank by prefix quality, cursor position (statement start boosts keywords, value position boosts series / user locals) and in-buffer local bindings.
  - **`input.int` hover** (`AXIS-ED-HOVER-INPUTINT`): exact catalog member beats the last-segment type fact, so hovers show signature + params instead of namespace docs.
  - **Contrast** (`AXIS-ED-LIGHT-CONTRAST`, `AXIS-UI-THEME-GRAPHITE-CONTRAST`): theme bridge enforces contrast floors for `--color-text-dim` / `--color-text-faint` (WCAG-ish 4:1 / 3.2:1) with correct mix direction per base; static light-theme faint token darkened (#8b8e9c → #666a78).
  - **Focus + labels** (`AXIS-ED-FOCUS`, `AXIS-ED-ICON-LABELS`): visible focus rings on editor tool / status / tabbar buttons and the CM surface; editor tab strip uses proper `tablist` / `tab` semantics with a real (non-nested) close button; aria-labels on icon-only git bar, topbar panel toggles, and Problems copy button.
  - **Eraser on locked drawings** (`AXIS-UI-DRAW-ERASER-LOCK`): locked erase / delete attempts now show a transient toast + assertive SR notice instead of a silent no-op.
- **Architecture apply guard**: `applyArchitecture` only preloads Pyodide when a real DOM append target exists. The unit-test document stub now exposes `querySelector`, which previously flipped the browser-detection gate and made `prefetchPyodideAssets` throw on `document.head.appendChild` under `applyArchitecture` tests (`commits Offline Lab`, `clears the stream slot`).

[2.4.1] — 2026-09-06

Fixed

- **Worker typecheck**: clean `tsc --noEmit` under the strict `worker/tsconfig.json` (`noUncheckedIndexedAccess`, `exactOptionalPropertyTypes`).
  - `auth.extractBearer`: regex capture non-null assertion — match guarantees the capture.
  - `scripts.rowToMeta` / `listD1`: build `ScriptMeta` with optional `description` / `path` only when set (no implicit `undefined`).
  - `scripts.handleScripts` draft put: include `name` only when defined; `parts[0]` non-null after the `_draft` / empty-path guards.

No runtime behavior change; only type compliance so `worker tsc` matches the pre-push gate.

[2.4.0] — 2026-09-06

Added

- **DSM-first data plane**: AXIS now loads datasets first — the chart paints instantly from the DatasetStore (repaired), then a background DSM job auto-completes the series (validate → classify gaps → sliced walk-back backfill) with progressive repaints as slices land. Venue REST remains the fallback/seed path; the Reload button forces a venue refetch (`preferDataset: false`).
- **DatasetStore** (`src/data/dataset-store.ts`): single source of truth for OHLCV datasets with pluggable persistence sinks — `session` (memory only), `local` (IndexedDB, default), `git` / `worker` (storage-plugin documents, debounced + retried, never blocking the chart). All consumers (chart loads, DSM jobs, live ticks, compare, CSV upload) read/write through it.
- **Dataset validation & repair** (`src/data/dataset-validate.ts`): `repairBars` drops corrupt rows, dedupes, sorts, and snaps individually off-phase bars to the series' dominant open-time phase (≥3 bars, ≥60% agreement — consistent venue offsets are preserved). `classifyGaps` splits gaps into **fillable** vs **legitimate** (weekend closures for session venues, declared maintenance windows) so the DSM stops chasing phantom gaps. `validateDataset` composes both into a report.

- **Dataset merge with conflict detection** (`src/data/merge-datasets.ts`): overlapping timestamps whose OHLCV disagree beyond a relative tolerance are conflicts, auto-resolved by policy (`newest-wins` default, `keep-current`, `prefer-incoming`); non-overlapping bars always union.
- **Settings → Data → Dataset persistence** main switch: `Session | Local | Git | Worker` (default Local), persisted as `store.datasetPersistence` and synced to the DatasetStore on boot/change.
- **Sliced-backfill progress chip** in the topbar Data group: live `loaded / expected` bar count while a DSM job completes the current dataset in the background.

Fixed

- **DatasetStore reads** are memory-first and never wipe the mirror on a sink miss; switching persistence copies in-memory series into the new sink.
- **Git/Worker sinks** return pending bars from `get` (no empty merge during DSM walk-back), serialize per-key flushes so a retry cannot clobber a newer put, and write-through to IndexedDB as a durable front. `onSinkError` lands in system logs.
- **Progressive paint** no longer goes through `loadBars` (that re-fitted the viewport every slice). `paintDataset` is generation-guarded so a slower prior load cannot paint over a newer symbol.
- **DSM jobs** repair + `validateDataset` / fillable gaps (not raw `validateBarCoverage`); `ensureDatasetComplete` uses the source calendar class. In-flight jobs for the same key are reused; the progress chip counts bars in the target window.

Changed

- **Topbar order**: the Venue picker moved to the front of the Market group (Venue → Symbol → Interval → Type → Compare) since it determines valid symbols/intervals; the Data group keeps plugin config, upload, datasets, Load, and Reload.
- **DSM job engine** reads and writes now route through the DatasetStore (session-mode safe, progressive repaints fire on every merged slice); `expand-cache` (cache→now + live ticks) and the Data Manager source do the same, with a repair pass on Data Manager resolves and CSV uploads.
- **Reload** shows the same loading spinner / `loadSeq` as Load.

Tests

- `tests/dataset-validate.test.ts` — repair (corrupt rows, phase-aware snap, dedupe, sort), gap classification (24/7 vs session venues, maintenance windows), dataset reports.
- `tests/merge-datasets.test.ts` — conflict detection tolerance, merge policies, union/sort behavior.
- `tests/dataset-store.test.ts` — sink routing (session vs local), conflict-resolved merges, replace/remove, change subscription, persistence-mode switch.
- `tests/dataset-sinks.test.ts` — remote `get` returns pending bars before flush.
- `tests/integration/load-symbol.test.ts` — DSM-first cache paint without hitting the venue.

[2.3.1] — 2026-09-05

Fixed

- **Doctor**: missing gitignored `worker/wrangler.toml` is a warning until `axis setup`, not a hard fail.
- **Live badge**: show Reconnecting… / Offline while the websocket is down; one reconnect warning per disconnect burst.
- **Static PWA server**: `GET /health` is `ok` (not SPA HTML); OPTIONS 204; favicon served.
- **Versions**: product identity is 2.3.1 everywhere (About, Tauri, docs).
- **Desk / PYNE path**: sibling repo is `../pyne` (fallback `../pynescript`).

Added

- Modest **Install app** chip on `beforeinstallprompt` (no nag).
- Quick Start defines SRC / STR / ENG / STO in plain language.

Changed

- **Default editor width** is 30% of the viewport (was 50%) on first load and layout reset.
- **Default chart interval** is `15m` (was `1d`).
- **Timeframes**: topbar / settings / DSM / watchlist interval list now includes `3m`, `30m`, `2h`, `6h`, `8h`, `12h`, `3d`, and `1M` alongside the previous set.
- **Topbar Inputs**: removed the Inputs button and its Studio/Settings visibility toggle. Script inputs still open from the Scripts list, command palette, and the existing inputs modal.

[2.3.0] — 2026-09-05

Fixed

- **CI unit tests**: plugin bootstrap now re-registers after `registry.clear()` (catalog flags reset together). Store/runner tests reset `resultsFocusId` so `setLastRun` is not skipped. Data-manager selection is cleared in stream catalog tests. CI pins Bun `1.3.14` (`latest` was racing global store/registry). `bun test --max-concurrency=1` for deterministic unit runs. Worker origin URL helpers live in `src/data/worker-origin.ts` so `store` → HTTP client → market-worker → on-chain proxy no longer hits a TDZ on import.

Changed

- **Repo description & tags**: GitHub / package / PWA / desktop copy no longer frames AXIS as “for Pine Script™”. Tagline covers CEX OHLCV, drawings, on-chain overlays, and Pine Script™ via PYNE; extra GitHub topics and package keywords.

Added

- **Per-tool drawing settings**: style bar grows context chips (extend L/R, show price, lock, custom color) plus a gear popover for fib levels, text/font size, risk/reward, arrow caps, and stats toggles. Last-used extras persist per kind in `drawingPrefs.byKind`. Paint honors extend flags, custom fib ratios (including reverse), highlighter width, long/short RR, and double-click text re-edit.

Tests

- **Tool settings catalog** — `tests/tool-settings.test.ts`: every drawing kind has controls; ray/highlighter/long defaults; fib + RR helpers; normalize keeps `meta.fibLevels`.
- **Store `byKind` merge** — `setDrawingPrefs` deep-merges per tool without wiping siblings.

[2.2.0] — 2026-09-04

Added

- **Results → Raw JSON tree**: collapsible VOID-colored viewer (keys / strings / numbers / bools) with key filter, Expand / Collapse, and Tree | Source. Large `plots` / `series` arrays page in (32 at a time) so a full run does not mount thousands of nodes. Click a primitive to copy it. Footer Copy / JSON still export the full payload.

Changed

- **Studio fullscreen canvas**: Runtime, Wire, Settings, Workers, and Plugins (plus nested tabs) now use the full overlay width instead of a reading-column cap. Wide layouts are two columns — Runtime (engine | endpoint), Settings General / Data / Topbar / Editor / Theme token grids, Workers inventory | inspector, Plugins catalog cards, Results strategy table. Legacy `sc-settings-*` chrome on Editor intelligence, exchange keys, and stacked plugin fields is replaced with studio `ax-*` primitives.
- **Theme color field**: Settings → Theme uses one `StudioColorInput` per token (swatch picker + text). Accepts hex, `rgb()` / `rgba()`, CSS names, and Pine `color.*` enums; the picker rewrites in the same form.
- **Editor color chips**: clicking an inline script color chip opens a native picker and replaces the source in the original form (`#hex`, `color.red`, `color.rgb`, `color.new`), keeping transparency.
- **Studio scroll**: nested panes, tables, code blocks, and library lists no longer scroll on their own. Only the modal canvas (`.ax-page-canvas`) scrolls.
- **Studio primitives**: status pills stay compact (no longer stretch full-width inside cards); Idle/Skipped/Degraded get the same pill chrome as Healthy/Down; cards, stats, inline buttons, and feature rows tightened. Same language applied across Runtime, Wire, Settings (all tabs), Plugins (catalog / install / library), Workers, and Results: capability badges are `ax-cap`, catalog rows are entity lists, chips-as-tags stay small, and compact `sc-*` chrome hosted in the overlay is scaled up to studio density.
- **Results overlay**: true fullscreen studio shell (`.ax-page-backdrop` + `.ax-page`, no dialog padding/blur). Header holds auto-open / script picker / saved-runs chip; Copy / JSON / CSV sit in the sticky footer. Canvas is the only scroller. Stacks above other studio pages (`z-index: 1050`) so a finished strategy is not trapped under Wire or Workers. Events, Strategy, Optimise, Plots, Metrics, Raw, and Saved use the same `ax-*` primitives as the rest of studio.

Fixed

- **Strategy Saved-runs loop**: live silent re-runs (every-tick / bar-close) were calling `setLastRun` without `persistence: 'skip'`, so local `saveResult` wrote a new IndexedDB row about once a second. Durable save is now user-initiated Run only; deferred / skipped / superseded payloads never persist. Fire-and-forget saves are serialized so FIFO trim cannot race.
- **E2E smoke vs Studio overlay**: topbar Architecture / Runtime / Settings / Plugins hooks are `hidden` (Playwright no longer treats `sr-only` 1×1 clips as clickable, which hung behind the fullscreen button). Smoke, critical, and library journeys open those pages through the visible Studio button + rail (`axis-studio-rail-*`).
- **Editor false typo on `syminfo.mintick`**: pre-eval "unknown built-in member" only knew `syminfo.prefix` / `syminfo.ticker` from `pyne-builtins.json` (the two callables). Host variables (`mintick`, `tickerid`, `timezone`, `currency`, …) are now allowlisted with the other Pine constants, matching pyne's implemented `syminfo.*` surface.
- **Market proxy (Worker)**: MEXC public REST is served through `/api/market/mexc/{klines,ticker/24hr,exchangeInfo}` because `api.mexc.com` does not send `Access-Control-Allow-Origin`. `fetchMexcJson` tries the Worker first and the direct host only as last-ditch fallback (offline lab, `skipWorkerProxy` tests, or Worker 5xx / network). Worker 4xx (allowlist / not found) is thrown immediately so the browser does not pay a CORS-blocked extra hop. Shared `resolveMarketWorkerBase` lives in `src/data/market-worker.ts` (re-exported from `binance-http.ts` / `mexc-http.ts`). Call sites: `watchlist-tickers.ts`, `sources/catalog.ts` (`mexcRest.fetchHistorical`), `symbol-catalog.ts` (exchangeInfo). Interval allowlist is AXIS TFs plus venue `1M` (`1h` → `60m`, `1w` → `1W`; no `1s` / `3m` / `2h` / `6h` / `8h` / `12h` / `3d`). `ticker/24hr` allowlists optional `symbol=` (single object, cached per symbol); there is no `symbols=` batch — omit the query for the full book. Watchlist uses per-symbol requests when the list is small (≤8), else the full book. No signed MEXC path yet.

Tests

- **Run-result persist** — `tests/run-results-persist.test.ts`: `persistence: 'skip'` and ephemeral meta do not write Saved runs; interactive Run writes once; three silent `liveTick` re-runs do not append.
- **Studio / Results e2e** — `e2e/studio.ts` opens rail pages without clicking hidden topbar hooks; critical Results close uses `axis-results-close` because the fullscreen overlay covers the topbar toggle.
- **Pre-eval `syminfo.*` host vars** — `tests/preevaluate.test.ts`: `syminfo.mintick` / `tickerid` / `timezone` / `currency` are known; `syminfo.minitck` still suggests `mintick`.
- **MEXC client fetch order** — `tests/mexc-http.test.ts`: Worker first, `skipWorkerProxy` goes direct, abort does not try the next host, Worker 400 does not call `api.mexc.com`, Worker 5xx still falls through. Watchlist small-list `symbol=` vs full-book path.
- **MEXC Worker ticker `symbol=` + `1M`** — `worker/tests/market-proxy.test.ts`: optional `symbol=` forwarded and cached separately from the full book; `symbols=` rejected; native `1M` klines accepted.
- **Color rewrite helpers** — `tests/pine-colors.test.ts`: `formatPickedChipColor` keeps kind/transparency; `rewriteColorKeepingFormat` preserves hex / rgb / rgba / named.

- **JSON tree helpers** — `tests/studio-json.test.ts`: kind / preview / stats, expand-all skips huge arrays, filter does not walk 200-point plot series.

[2.1.1] — 2026-08-29

Fixed

- **CORS product-host list**: retired the legacy `*.pynescript.ai` branch from `PRODUCT_ORIGIN_RE` (worker `pickOrigin`) — production URL is `pynescript.online`, not `.ai`. The fallback default in `env.ALLOWED_ORIGIN` (and the inline fallback when none is configured) now also points at `https://pynescript.online`. Legacy `.ai` requests fall through to the allowlist fallback instead of being echoed. `isOAuthProxyBase` (git-oauth client proxy allowlist) drops the `.ai` check for symmetry. Docs (README, `docs/devops/cors-and-origins.mdx`, `docs/worker/bindings.mdx`, `docs/architecture/evaluation.mdx`, `docs/llm.txt`) updated.

Tests

- **CORS regression**: `worker/tests/cors-origin.test.ts` adds a `does NOT echo the retired pynescript.ai product host` case that confirms `https://pynescript.ai` and `https://app.pynescript.ai` requests fall back to the configured `ALLOWED_ORIGIN` first entry (i.e. they're no longer auto-allowed as product hosts).

[2.1.0] — 2026-08-29

Added

- **Local storage: persisted run results**: `localStoragePlugin` now implements the optional `saveResult` / `loadResult` / `listResults` / `removeResult` methods so completed strategy/indicator runs can survive reloads. IndexedDB schema bumped from v1 → v2 — additive migration only, with a new `results` object store keyed by compound `[scriptId, runId]` and a `byScript` index on `meta.scriptId` for cheap per-script listing. FIFO trim keeps at most `MAX_RESULTS_PER_SCRIPT = 50` runs per script (oldest by `meta.startedAt` evicted; the just-saved run is always preserved). A `LOCAL_STORAGE_VERSION = 2` constant is exported for downstream schema detection, and the plugin now advertises `capabilities.results = true` so UI can render saved-runs affordances. localStorage JSON + in-memory `Map` fallbacks mirror the same semantics for SSR/tests (`pynescript.axis.results.v1` key + `_getMemResultsForTests` helper). Plugin contract extended in `src/plugins/types.ts` with a `results?: boolean` capability flag.
- **Storage change confirmation dialog**: new `StorageChangePrompt` (`src/ui/StorageChangePrompt.tsx`) is mounted once at the app root and hosts the existing `StorageChangeDialog` whenever the user switches storage engines. A new `promptStorageChange(oldId, newId)` helper in `src/storage/service.ts` is the single entry point — short-circuits silently when the engine is unchanged or unset (first-time set), and otherwise opens the dialog so the user can pick *Migrate scripts* or *Start fresh* before the active plugin flips. The migration itself still runs through the existing dialog (it calls `getStorage(fromId)` / `getStorage(toId)` directly so the active plugin stays pinned to the source until commit). All four storage-change call sites now route through it — `ScriptLibraryPanel` dropdown, `SettingsDialog` general tab Save, `PluginsPage` catalog activate, `PluginManager` modal activate — so there is exactly one shared dialog state, not four duplicated ones.
- **Per-panel icon in FloatableShell header**: `FloatableShell` now renders the panel-specific glyph next to the hamburger menu (testid `axis-panel-header-icon-{panelId}`), matching the Topbar panel toggle. Single source of truth is `PANEL_ICON: Record<PanelId, IconName>` in `src/ui/icon-map.ts`; TypeScript exhaustiveness guarantees every `PanelId` has an entry. Missing mappings fall back to the hamburger-only chrome without breaking the panel.

Fixed

- **Status bar chip 1px hover wobble**: `ConnectionHud` chips (`ChipShell`, `TickPulse`, `LiveBadge`) no longer shift pixels on hover/active. Locked border-width to `border-[1px]` (Tailwind v4 `border` no longer relies on theme indirection), added `transition-none` to prevent any color flicker, and pre-applied the `ring-1` outline on `LiveBadge` so only the ring color animates between `ring-transparent` and `ring-accent` (no box-shadow structure delta). `.sc-btn-ghost` gained an explicit `box-sizing: border-box` (defensive — Tailwind v4 preflight already sets it on `*`).
- **Duplicate "Open in new tab" in editor dock menu**: the editor panel's dock menu showed both the generic DOCK_MENU "New tab" entry and an editor-specific "Open in new tab" entry passed via `menuExtra`. The `menuExtra` is now `undefined` (per the `FloatableShell` JSX usage in `EditorPane`), so the dock menu shows the generic "New tab" entry exactly once. The editor's own "Open in new tab" affordance lives only in the right-side `EditorOverflowMenu` (testid `axis-editor-btn-new-tab-overflow`), where it pops the editor out into a full browser tab instead of a docked window.

Tests

- **Storage result round-trip + FIFO + fallback** — `tests/storage-local-results.test.ts` covers the new IndexedDB v2 `results` object store: add / load / list / remove, FIFO trim at `MAX_RESULTS_PER_SCRIPT = 50`, the `byScript` index, SSR fallback (`pynescript.axis.results.v1` + `_getMemResultsForTests`), and additive schema migration from v1.
- **Service façade + plugin opt-in** — `tests/storage-service-results.test.ts` exercises `saveRunResult` / `loadRunResult` / `listRunResults` / `removeRunResult` / `supportsRunResults` through the active plugin, including the local-only fallback when the active engine (git / cloud) doesn't implement the optional result methods.
- **Storage change prompt** — `tests/storage-change-prompt.test.ts` asserts the dialog is mounted exactly once at the app root, opens on engine change, and short-circuits silently when the engine is unchanged.
- **Editor menu dedupe** — `tests/editor-menu.test.ts` locks the regression: `EditorPane` no longer passes a `menuExtra` with a "New tab" entry, the dock menu still contains exactly one "New tab", and `EditorOverflowMenu` still owns the overflow "Open in new tab".
- **Panel header icon routing** — `tests/panel-icon.test.ts` verifies `PANEL_ICON` covers every `PanelId`, each value is a valid `IconName`, and `FloatableShell` reads `PANEL_ICON[props.id]` and emits the `axis-panel-header-icon-{panelId}` testid.

[2.0.31] — 2026-08-28

Changed

- **About modal**: removed ethos manifesto list and author section; added AXIS + PYNE version badges and brief no-walled-garden philosophy.
- **Studio consistency**: `ThemePanel` (Settings → Theme) now uses the studio `ax-*` primitives (`StudioSection`, `StudioField`, `StudioInput`, `StudioToggle`, `StudioChip`, `StudioButton`, `StudioHint`) instead of legacy `sc-settings-*` classes, matching Runtime / Wire / Workers / Plugins. Studio `--ax-*` tokens promoted to `:root` so shared panels render correctly outside the modal.
- **Theme color input**: each token uses a single color input (mono text field + non-interactive preview swatch) instead of a paired swatch picker and text box; number tokens use one number input.
- **Studio modal radius**: added a scoped `--radius-surface` token (8px inside `.ax-page`, defaulting to 3px elsewhere). Studio modal surfaces — `.ax-card`, `.ax-btn`, rail nav items, color preview, equity tooltip, HPO chart wrap, results auto-open toggle, and the Results error box — now use the higher radius. Chart/panel `<input>` / `<select>` and small chips/badges keep their existing 2–3px radius for hierarchy.

Added

- **Results fullscreen studio modal**: replaced the docked Results panel (`FloatableShell`) with a focused fullscreen studio overlay (`ResultsModal`) opened via `store.resultsPanel.open`. Six restyled subpages — Events, Strategy, Optimise, Plots, Metrics, Raw — share one studio canvas (`ax-page` / `ax-*`) with `StudioStat`, `StudioTabs`, and `ax-list` / `ax-card` primitives. `StrategyReport` and `HpoPanel` rebuilt on studio classes (stat cards, equity card, ax-field controls, ax-list search space). Results auto-open only for `strategy()` scripts (indicators no longer auto-open).
- **Results auto-open setting**: new persisted `resultsAutoOpen` flag (default `true`) gates the strategy-only auto-open in the runner. A visible toggle sits at the top of the Results modal header ("Auto-open on strategies"); the same control is exposed in Settings → General ("Auto-open results on strategies"). Persisted via localStorage and hydrated on boot.
- **Strategy equity graph**: replaced the flat SVG line with a rich `EquityChart` — gradient area fill, zero baseline, gridlines with money-axis labels, underwater drawdown shading, and a hover crosshair + tooltip showing time, equity, and drawdown.
- **Results subpage polish**: Events list now shows colored type badges and long/short direction pills; Plots shows a sparkline card per series (positive/negative color); Optimise shows an in-sample / out-of-sample trial score chart with legend; Metrics shows a status pill plus derived stats (profit factor, max DD, plot points, script type).
- **Results full-width + 2-column subpages**: the Results modal canvas now uses the full studio width (`ax-page-canvas--wide`, no reading-column cap). Optimise uses a two-column split — search configuration + search space on the left, live study chart + trials table on the right. Strategy uses a two-column split — stats + equity curve on the left, closed-trades table on the right (stacks to one column under 64rem).
- **Topbar settings**: new 'Topbar' tab in Settings with toggles for each topbar group (brand, market, data, compute, layout, panels, system) and individual panel buttons (watchlist, editor, library, scripts, inputs, layers, DSM, on-chain, alerts, values, results, script logs, system logs, status). Settings are persisted via localStorage.
- **HPO (Optimise) polish**: the search config (trials, sampler, objective, validation, holdout/walk-forward sizes, min trades, and per-param bounds) now persists across reloads; the trial chart pins its x-axis to the trial index (IS/OOS lines no longer desync when trials error), marks the best trial, and shows the score scale; the results header gained a status chip, best IS/OOS scores, and a Clear button; categorical params show their choices and the search-space list has an empty state.
- **HPO tier gating**: added a persisted `tier` (`free` / `pro` / `self-hosted`) — the free tier disables walk-forward validation (shown as "(Pro)") with an upgrade hint, while Pro / Self-hosted show a tier badge and unlock it. The engine-runs cap was raised from 400 → 1000 so walk-forward fits within the trial ceiling.

Fixed

- **Workers probe non-blocking**: initial backend probe deferred past first paint (double `requestAnimationFrame`) and a non-blocking "Probing backends..." hint shows while the first snapshot loads — the Workers panel is ready and interactive immediately, regardless of slow/failed probes.
- **Indicator engine correctness**: styled Pine plots (`histogram` / `area` / `columns` / `stepline` / `baseline` / `line` / `cross` / `circles`) were silently dropped because `splitSeriesByKind` only accepted `plot` / `hline` kinds — they now route to line plots (missing-kind and unknown-kind both default to line). Engine `status: 'success'` is no longer downgraded to error by a stray non-fatal `error` / `message` string. Series↔time-axis alignment is now derived from the exact bars the engine evaluated (passed through `getOhlcvTimesForApply`), fixing misaligned/blank newest bars on live ticks and HPO holdout slices. Explicit `overlay=true` is no longer auto-demoted to a sub-pane by the oscillator-scale heuristic. The `getOhlcvTimesForApply` cache is reused for the common `store.bars` case (the bypass path only applies to a genuinely different axis) and invalidates on a `store.bars` reference change.
- **Results panel dock cleanup**: removed the dead `results` panel from the dock registry (`PanelId`, `PANEL_META`, `PANEL_IDS`, `DOCK_STACK_ORDER`) so the bottom dock no longer reserves ~220px of empty space — Results is now a fullscreen modal driven solely by `resultsPanel.open`.
- **Input override no-op on chart**: changing a script input (e.g. RSI length 14→56) now repaints the whole series. `PaneManager.syncOverlayLines` / `syncOverlayOhlc` / `syncBgcolorBands` gained a `forceFull` owner option; `runAndApplyInner` passes `forceFull: opts.liveTick !== true` so interactive runs / input recomputes bypass the tip-only smart-apply (which only patched the last bar when length + last time were unchanged). Live ticks keep the fast path via `liveTick: true` in the multiplex re-run loop.
- **Wire page**: removed separate scrolling sections; now uses single-scroll ax-page-canvas pattern matching Runtime, Settings, Workers, and Plugins.
- **Settings polish**: consolidated duplicate `sc-settings-*` CSS into single definitions, added `sc-settings-section--plain` and `sc-settings-content--compact` variants to replace `!important` overrides, added `color-mix` fallback, `focus-visible` on color swatches, and `prefers-reduced-motion` guard; replaced `!mt-0 !border-t-0 !pt-0` with semantic variant and fixed ThemePanel compact gap override.
- **Workers probe**: fixed `requestAnimationFrame` leak — now cancels on unmount and removes dead `hasLoaded` signal; probe defers to next frame without blocking studio paint.
- **Vendor**: removed obsolete `pynescript-0.4.0/0.4.1` wheels (replaced by `hoox_pyne-0.4.2`).

[2.0.30] — 2026-08-26

Fixed

- **plot display= in compile mode**: `plot(..., display=display.data_window)` was not carried in `plot_meta` because the compiler didn't fold Pine `display.*` constants. Both axes now resolve `display.data_window` / `display.pane` /etc. to bitfield integers and pass them through to `plot_meta`.
- **Pyne wheel** — vendor `pynescript-0.4.2` (was 0.4.1). Compiler display folding + server compile-path backfill.

[2.0.29] — 2026-08-26

Changed

- **Engine default mode** — both Pro and Pyodide engines now default to `auto` (try compile, fall back to interpret) instead of `interpret`.

Fixed

- **hline linestyle in compile mode**: server-side compile path did not include `linestyle` / `color` / `price` in `plot_meta` because `plot_attrs` is empty for hline/fill. Backfill from `__drawings` now fills missing visual attrs.
- **Pyne wheel** — vendor `pynescript-0.4.1` (was 0.4.0). Server compile-path backfill fix.

[2.0.28] — 2026-08-26

Fixed

- **hline linestyle in interpret mode**: the interpret path (Pyodide default) did not synthesize `plot_meta` from `__drawings`, so LWC price lines for `hline()` always rendered solid regardless of `linestyle=`. Both interpret and compile paths now synthesize `plot_meta` with kind/linestyle/color/price for hline, fill, and plotshape drawing types.

[2.0.27] — 2026-08-26

Changed

- **Pyodide PYNE wheel** — vendor `pynescript-0.4.0` (was 0.3.7). Engine catalog, legacy JS engine, SW cache test, and editor builtins synced from pyne 0.4.0.

Fixed

- **hline linestyle**: `hline(linestyle=hline.style_dashed)` rendered solid — the drawing path only checked `r.style` (never `r.linestyle`), and the compile-mode overlay path lacked `plot_meta` entirely so the LWC price-line never received the style. Both paths now covered: `pyne-drawings.ts` falls back to `r.linestyle`, and both Pyodide runtimes synthesize `plot_meta` from `__drawings` (hline/fill/plotshape kinds with linestyle, color, price).
- **multi-chart**: same indicator on every grid cell showed identical values — the runner evaluates Pine against `store.bars` (active plane), but focusing another slot never swapped in that slot's cached history, so every Run/add computed from whichever chart was loaded last. `setActiveChartSlot` now restores the focused slot's own bars (+ gen bump), stops stale bar replay, restarts live for the slot's market, and silently re-runs applied scripts so each chart shows its own values.
- **multi-chart**: switching to a 2×2 grid could overwrite the active chart with a sibling's symbol — ChartHost "prefetched" inactive slots via `loadSymbolData`, which always writes the active plane (last-mounted sibling won). Removed; empty cells load on focus instead.
- **tables**: render Pine `table.cell` `text_valign` (top/middle/bottom) and `text_size` tokens/numeric points in the table HUD — parsed previously but rendered fixed 10px/middle; defaults unchanged when absent.
- **bgcolor**: carry band titles into `syncBgcolorBands` and apply them as series titles on create/update, matching the offline `buildPlotVisuals` path.

[2.0.26] — 2026-08-24

Added

- **Pages security headers** — `public/_headers` ships the nginx/server.ts hardening baseline to Cloudflare® Pages deploys (CSP with self scripts + `wasm-unsafe-eval` for Pyodide, https/wss connect, plus nosniff / referrer / frame-deny), so axis.hoox.sh gets enforced headers instead of only Page Shield's Report-Only placeholder.
- **Native MEXC CEX** — `mexc-rest` / `mexc-ws` as a built-in venue (public spot klines, no CCXT). Venue picker lists MEXC with Binance/OKX/Bybit; signed REST uses `X-MEXC-APIKEY`. MEXC was removed from the pinned CCXT shortlist.

Fixed

- **`hline()` levels vanished once `bgcolor()` rendered** — `priceLineHost` picked the first non-overlay series, which after the bgcolor fix could be a `bgcolor_*` histogram living on its own hidden 0–1 price scale; hlines attached there mapped far off-viewport. Host selection now skips bgcolor underlays (TradingView® paints backgrounds *behind* panes — they never host price lines).
- **`bgcolor()` / `barcolor()` drew nothing** — response normalization coerced every series sample through `Number()`, turning engine color arrays (`rgba(8, 153, 129, 0.298)` per bar) into all-`null`, so the histogram bands were filtered out before reaching the chart. CSS color samples now pass through normalization untouched (`coerceSeriesSample`); value plots still coerce as before.
- **Workers status chips stayed "Unknown"** — `StudioStatus` closed over the first `props.status` (`unknown` while probes ran). The CSS class updated (green/red dot) but the word did not. Label is now read from props on each render.

Changed

- **Topbar Studio** — one **Studio** button replaces Wire, Runtime, and Settings. Those pages stay in the studio rail (and ⌘K). The button reopens the last studio page.
- **Wire recipe grid** — the hatch is a behind-layer at lower opacity. The right-edge fade no longer masks preset names.

- **PYNE Pro API origin is `https://pynescript.online`** — Hetzner VPS (nginx → gunicorn `:5002`). `axis.hoox.sh` is the Cloudflare® Pages PWA, not same-origin with the API. Default Backend URL, Workers Manager probe, datafeed gateway, and Settings/Runtime presets follow the new origin; saved `axis.hoox.sh` / old VPS IP endpoints remap on load.

[2.0.25] — 2026-08-23

Fixed

- **Script Settings inputs no longer lose to editor defaults** — re-running an applied script from the editor keeps per-instance `inputValues` / strategy properties. Empty `{}` bags no longer wipe saved settings. Overrides persist only when they differ from the script default and are keyed by title, id, and LHS var name so the engine actually applies them.

Changed

- **npm publish uses org secret `NPM_TOKEN_HOOXSH`** — `release.yml` no longer looks for a repo `NPM_TOKEN`. Root Bun tests ignore `packages/**` so `axis-cli` suites run only after `packages/cli` deps are installed. Authenticated skip detection treats an already-published version as success.

[2.0.24] — 2026-08-23

Added

- **Production-grade `@hoox-sh/axis-cli` 0.2.0** — first npm publish of the AXIS CLI. Node-compatible `dist/` + `npm i -g @hoox-sh/axis-cli`, AGPL `LICENSE` in the tarball, provenance on tag publish, CI coverage for `packages/cli`.

Fixed

- **`plot(..., display=)` was ignored at apply time** — editor already knew `display.data_window` / `none` / `pane` / `price_scale` / `status_line` / `all`, but every series still painted on the pane and in the Data Window. AXIS now reads `plot_meta.display` (token or Pine bitfield): `none` hides everywhere, `data_window` keeps Data Window and skips the chart, `price_scale` / `status_line` gate last-value labels. PYNE packs `display=` on plot/hline/shape/fill/bgcolor.
- **E2E smoke vs studio pages** — Plugins is opened from the Runtime rail (topbar hook is sr-only). Wire Apply waits for an exact `Wire · Offline Lab` title so a drifted `Offline Lab +N` plan is not applied as Live Crypto.
- **`axis setup --remote-d1` skipped local D1** — `setup d1` treated `--remote` as exclusive of local even when `local: true` was passed, so bootstrap never applied the schema locally. `--remote` alone stays remote-only; `--local --remote` and `--remote-d1` apply both.
- **Worker `/health` probe accepted any HTTP 2xx** — `probeHealth` OR'd the JSON `status === "healthy"` check with `res.ok`, so a 200 HTML page counted as healthy. Requires a JSON `status` of `healthy` or `ok`.
- **Failed `axis deploy worker` ran wrangler deploy twice** — on inherit/quiet failure the CLI re-invoked deploy to capture stderr. Capture once, parse the workers.dev URL, never redeploy on error.
- **`--json` mixed human banners into stdout** — `--json` now implies `--quiet` so doctor/health/deploy emit parseable JSON only.
- **Docker GHCR version tags could skip on `v*`** — `docker.yml` path-filtered `push`, and GitHub applies that filter to tag pushes. Tag/main image builds no longer use a path filter (PRs still do).
- **Unknown `axis` commands dumped a generic wrapper error** — Commander help/version/unknown-command now exit with Commander's own code instead of `handleError`.

Changed

- **npm release workflow** — pin Bun 1.3.14, `id-token: write` + `npm publish --provenance`, pack dry-run (LICENSE/dist/bin required), org secret `NPM_TOKEN_HOOXSH`. CI unit job now typechecks and tests `packages/cli`. Dead `@clack/prompts` dependency removed.

[2.0.23] — 2026-08-23

Changed

- **Drawings / plot parity plan** — `docs/architecture/drawings-parity.mdx` lists P0–P3 work: user-tool move/handles, Pine pane routing, PYNE export gaps (tables, fill meta, plotshape Y, OHLC).

Fixed

- **Pre-eval treated `chart.fg_color` as a typo** — `pyne-builtins.json` only lists `chart.point.*` callables, so `chart.bg_color` / `chart.fg_color` (and the `color_background` / `color_foreground` aliases, plus viewport `left_visible_bar_time` / `right_visible_bar_time` / `is_standard`) were flagged "not a built-in member". Allowlisted with the other Pine host constants.
- **1-point drawings did not body-move** — `shiftDrawing` only special-cased `text` / `priceLabel`. Note, flag, crossline, anchored text, and arrow marks now shift `p1` (and `points[]` when present). Third-anchor `p3` handles hit-test and resize.
- **Linefill GC was a no-op** — `garbageCollectScriptDrawings` ignored `skip.linefill` on the early-return. Overflowing linefill-only batches now trim oldest. Polyline/linefill copy `force_overlay`; polyline `fill_color` maps to fill.
- **`overlay=false` bgcolor/fill always hit the price pane** — bgcolor bands take `paneId`; fills use the script-pane drawing layer when the indicator is not overlay. Empty `drawings: []` clears script SVG even on silent live re-runs.
- **plotshape on oscillator panes** — `setShapeMarkers` takes a pane id; overlay=false scripts attach LWC markers to the first overlay series on `ind_*` instead of price candles. Trade markers stay on price.

- **Drawing tool settings** — Fill slider covers channel, triangle, gann boxes, ranges, highlighter; highlighter stroke no longer multiplies width ×6; horizontal ray places on one click; XABCD / head-and-shoulders commit on the 5th point.
- **Hide drawings hid only user tools** — the eye toggle now also clears Pine script SVG and `fill()` bands (every pane layer). `label.style_text_outline` paints outlined text; `yloc.abovebar` / `belowbar` sit 8px off the wick. First Add Indicator re-owns plotshape / fills / script drawings from `__editor__` to the script id. Histogram overlays honor `plot_meta.histbase` when the engine sends it.

[2.0.22] — 2026-08-23

Added

- **Topbar Venue picker** — Data group is now Venue (native CEX / pinned CCXT / CCXT… / local) instead of a raw source-plugin list. Picking Bitget (CCXT) writes `ccxt-rest` + `ccxt-ws` and both `exchange` bags together; `store.provider.id` is `ccxt:bitget`. Exchange ID stays on the topbar only for **CCXT…**. Stream remains hidden while paired.
- **CCXT symbol browse uses that exchange's markets** — with Source = CCXT (Gateway), the symbol modal loads `GET /markets?exchange=` and keeps unified `BTC/USDT` tickers (slash is no longer stripped). Stream is hidden on the topbar while it still matches the source pairing (Wire / HUD Fix if it drifts).
- **CCXT session API keys** — Settings → Data can store a key + secret (+ passphrase / uid) per CCXT exchange id (`ccxt:bybit`, …) in the RAM vault (never `pluginsConfig` / localStorage). Save POSTs `POST /datafeed/session`; Load/Live then pass only `cred=` on OHLCV/watch. Public candles still work without a key. Topbar shows an **API key** chip next to the CCXT exchange picker. Gateway `auto|pyne|sidecar` moved to Settings (advanced). Sidecar accepts `/ohlcv` and `/datafeed/ohlcv`. PYNE `POST /datafeed/session` accepts `apiKey` / `secret` / `password` in the JSON body.
- **npm release pipeline for the AXIS CLI** — `@hoox-sh/axis-cli` (`packages/cli`) is now publishable: dropped `"private"`, and a new `.github/workflows/release.yml` typechecks, tests, builds and publishes the package to npm on `v*` tags using an `NPM_TOKEN` repository secret. A registry-version guard skips cleanly when the version is already published (safe tag re-runs); run summary reports local vs registry versions.
- **"What's in this repo" section in README** — table mapping every part of the repo to its role and paths: app/PWA, datafeed pipeline, calc engines + Workers Manager, vendored Pyodide runtime, on-chain plane, Cloudflare® Worker (API/WS/D1/KV/R2), allowlisted `/api/onchain` proxy, Tauri desktop shell, npm-published CLI, tests/ops.
- **Plugin config `advanced` fields + Settings section** — schema fields can now be marked `advanced` (API base URL, Bars, Synthesize on failure, …). They are hidden from the Topbar config row and rendered in Settings → "Source & stream plugins" with full descriptions, using the shared form classes.
- **Full unified ccxt exchange dropdown** — the Exchange ID select is fed by the gateway's complete ccxt exchange list (`/health` → new `ccxt_exchanges`, mirrors `ccxt.exchanges`; ~103 venues) instead of only the native adapters. PYNE side: `backend/api/datafeed.py` health payload extended accordingly.

Changed

- **Studio full-page overlay** — Runtime, Wire, and Settings are no longer right-edge drawers. They share one full-viewport page (`src/ui/studio/`, `ax-*` classes) with a left rail, 16px type root, and large fields. Compact `sc-dialog` stays for Symbol / Script Settings / About / ⌘K.
- **Runtime uncoupled from Workers and Plugins** — Topbar **Runtime** is the active engine / endpoint / exec mode / health. Workers (backend inventory) and Plugins (contract catalog) are sibling pages from the rail, ⌘K, or Runtime cards — not tabs inside Runtime. Engine/endpoint/mode moved out of Settings → General; storage stays a Wire slot.
- **Workers studio restyle** — Backend inventory is a full `ax-*` studio page (`WorkersPage`): large health cards + inspector (probe features, numbered install steps, Use as calculation backend). No inner Overview/Detail/Install/Configure tabs and no `sc-btn` chrome. Engine writes go through `saveEngineConfig`. Open Plugins is a sibling cross-link, not a nested catalog.
- **Plugins studio restyle** — Contract catalog is a full `ax-*` studio page (`PluginsPage`), sibling of Runtime. Catalog / Install / Script Library stay as `StudioTabs` (e2e tab name **Catalog**). Kind chips, large rows with Use + capability badges, URL install + example cards, Script Library embed, active src/eng/stm/stor footer + Done. Open Workers and Open Wire are sibling cross-links. Close stays on the AppPage shell (`axis-plugins-close`); `axis-manager` stays on the dialog only.
- **Data tab no longer duplicates bar count or Binance API host** — how many bars to load lives only under General → Historical bars. Native CEX host URLs stay as fetch defaults (Binance already has host + Worker fallback). Gecko proxy/network, CCXT gateway, and synthesize-on-failure remain.
- **Docker release images tagged by version, not sha** — `docker-bake.hcl` release targets now push `pwa-v<VERSION>` / `pwa-nginx-v<VERSION>` (e.g. `ghcr.io/hoox-sh/axis:pwa-v2.0.21`) instead of `pwa-sha-<sha>`. The git SHA remains on the image as the `org.opencontainers.image.revision` label and build arg; workflow docs updated to match.

Fixed

- **Plugin config row — first-save, apply, and exchange dropdown** — shared source/stream fields now (1) resolve a stored value from *any* target bag before falling back to a schema default (stream `exchange` is no longer hidden by a source default of `''`), (2) persist text/number keystrokes but only reload history on blur/Enter (selects/checkboxes still apply immediately), (3) treat the exchange control as a `<select>` only when the gateway list is actually populated, with a placeholder `Select exchange…` option and a free-text fallback when the fetch fails or returns empty, (4) cache the ccxt list per gateway mode, and (5) wire stacked-form `label for` to the control `id`. Gecko `baseUrl` / `network`, Coinbase granularity, and Mock Walk start price are Settings-only (`advanced`). Topbar Load no longer bails out while a previous fetch is in flight.
- **Pre-eval scanners close paired typographic quotes** — “…” / ‘…’ now close on the matching right quote (not only the same code point), so a pasted title string no longer marks the rest of the file as an unclosed string and block Run. Shared `isQuoteClose` in `pine-scan-util.ts`.
- **`array<string>` survived as `array` after Add types** — the declaration rewriter captured the collection token but dropped `<string>` when inserting a missing type1 qualifier. Generic args are preserved (`simple array<string> vals = …`).
- **Type methods and `export enum` missed as user bindings** — indented `method profit(this) =>` inside a `type` body and `export enum Name` now register as declared names (same as `export type`).

- **`const` (and line-leading keywords) colored inconsistently — green vs violet** — a float ending in a trailing dot (`2.`, `0.`) was split into number + lone `.`, leaving the tokenizer's `afterDot` state set at end-of-line. The state leaked across comments/blank lines, so the *next* code line's first word (`const`, `series`, …) rendered as a member property (green) instead of a keyword (violet accent). Number scanning now consumes trailing-dot floats and `afterDot` resets at every line start.
- **Pre-eval typo checker round 3 — validated against the real 2222-line grid strategy** — found via local reproduction on `grid.pine`: generic typed declarations (`var series array<string> TP1 = …`) were not parsed by the assignment regex, so TP1/SL1/TRL1-class vars were never registered as declared; library import coordinates (`import cryptolinx/String/1 as strx`) leaked into both the declaration collector and ident scanner (`String` / `Hoox` flags); and dotted method calls (`.init()`, `.show()`, `.abs()`) were treated as bare-call typos. All three fixed; checker now reports **0** typos on grid.pine, grid.2.pine.
- **Pre-eval typo checker round 2 — declaration coverage + copy-paste resilience** — remaining false positives addressed: `strategy(linktoseries=…)` whitelisted (pyne accepts arbitrary declaration kwargs); typographic/curly quotes (“”‘’) now treated as string delimiters in every scanner so pasted text no longer leaks as identifiers; `enum` members + enum/type names, `type` fields, and `for [i, v] in …` loop vars are collected as declared bindings; dotted-path roots (`Mode.SINGLE`) skipped by the ident scanner.
- **Pre-eval typo checker flooded false positives (~260 per script)** — pyne's LSP metadata only lists functions/namespaces, so the pre-eval linter treated core built-in series variables (`close`, `high`, `low`, `open`, `volume`, `ohlc4`, `hl2` …), type qualifiers (`simple` / `series` / `const`) and `text.align_*` / `text.wrap_*` constants as unknown identifiers and suggested nonsense ("Unknown `close` — did you mean `CROSS`?"). Added a curated `PINE_BUILTIN_VARS` supplement (OHLCV + derived, time/bar state, calendar parts) plus the missing `text.*` members to the known-path index; qualifiers also join the bare-call skip set. Regression tests cover the reported failure pattern (now zero typos).
- **Config row crash on first save** — `setField` wrote `pluginsConfig` through a deep Solid store path whose parent bag did not exist yet; solid-js/store does not auto-create intermediate nodes, so the first-ever config change threw `TypeError: can't access property "exchange"` and never persisted. Bag creation is now guarded (`writePluginField`), covered by regression tests and a browser e2e run.
- **Gateway select had no choices** — the ccxt `gateway` schema fields lacked `options`, so the dropdown rendered only its current value. `auto | pyne | sidecar` are now declared on both `ccxt-rest` and `ccxt-ws`.
- **Config row deduplication** — the Topbar renders one shared row for the union of active source/stream config fields (was: duplicated per picker); changes write through to all declaring plugins so ccxt-rest and ccxt-ws stay in sync.
- **Gateway loopback on remote pages (hardened VPS)** — `gatewayBase` resolved `pyne` / `auto` to `http://127.0.0.1:5002/datafeed` even when the page was served from a remote host, so browsers on `axis.hoox.sh` fired doomed cross-origin requests at the *visitor's* machine (CORS NetworkError). Remote pages now resolve to same-origin `/datafeed` on product hosts (nginx) or the product API origin cross-origin (Pages previews); loopback behavior is unchanged for local dev. Explicit endpoint overrides still win.
- **ccxt-ws `require()` actually removed** — v2.0.21 added the static `gatewayWs` import but left the shadowing CommonJS `require('../data/gateway')` call in place, so the browser bundle still threw on stream start. The `require` is now gone.

Changed

- **Plugin config moved to Settings → Data** — the "Source & stream plugins" section (with the advanced fields) now lives in the Data tab above Exchange API keys, with a note that CCXT Gateway venues serve public market data and need no keys. Native-source keys (Binance/OKX/Bybit/Coinbase/Kraken) remain session-only in the browser vault and sign requests client-side.
- **Config row styled like the Source picker** — inline plugin config fields now render through `TopbarField` (integrated uppercase label, shared `axis-tb-field` chrome, focus-within ring), matching Symbol/Source/Interval exactly; the Settings variant keeps the dialog's stacked form classes.
- **Stream config pass-through** — `multiplex.startLive` now passes `store.pluginsConfig[stream:<id>]` into `stream.start({ config })`, matching the source path, so ccxt-ws receives the configured exchange id.
- **ccxt plugins: clear error for unconfigured exchange** — `ccxt-rest` / `ccxt-ws` no longer fire requests with an empty `exchange=` param; they surface "exchange id not configured" instead.

[2.0.21] — 2026-08-21

Fixed

- **ccxt-ws stream: `require()` in browser ESM** — replaced CommonJS `require('../data/gateway')` with a static import. The `require` call threw `ReferenceError` in the Vite browser bundle when activating the `ccxt-ws` stream (Bun tests masked it since Bun honors `require` in ESM).
- **ccxt-rest backfill window inversion** — `endTime` was forwarded as CCXT `since`, which pages *forward* from that timestamp, breaking the Data Source Manager walk-back contract (bars must end at/before `endTime`). Now derives `since = endTime − limit·timeframe` so each page covers `[endTime − limit·tf, endTime]` and gap-fill converges.

[2.0.20] — 2026-08-21

Added

- **Datafeed gateway transport** (`src/data/gateway.ts`) — resolves gateway URL (auto/pyne/sidecar/direct), probes sidecar health with 30s TTL cache, and provides `gatewayFetch` / `gatewayWs` helpers for plugins that route through the PYNE datafeed gateway or local sidecar.
- **ccxt-rest source plugin** — long-tail exchange OHLCV via the PYNE datafeed gateway or sidecar. Routes `ccxt:<exchange>` venues through `GET /datafeed/ohlcv`. Credentials stay server-side; browser never holds API keys.
- **ccxt-ws stream plugin** — real-time klines via the datafeed gateway WebSocket. Routes through `WS /datafeed/watch` with exchange+symbol+timeframe params.
- **Gateway-aware active resolution** — when provider gateway is not `direct` and the venue is a `ccxt:<exchange>` long-tail id, active source resolves to `ccxt-rest` and stream to `ccxt-ws`. Native venues (binance, okx, etc.) are never swapped.

[2.0.19] — 2026-08-20

Fixed

- **Hex color lint false positive** — pre-eval `scanIdentRefs` no longer treats hex color tails (e.g. `d3ee` in `#22d3ee`) as unknown identifiers. Scans backward past hex digits to detect the `#` prefix and skips the entire literal.
- **Pre-existing TypeScript errors** — resolved 23 type errors across 7 files: null barIndex guard, CSS `TextAlign` narrowing, `PineTable` camelCase property reads via `Record<string, unknown>`, `linefill` missing from `DrawingLimits` / drawing caps/counts/skip, `cancelled` status narrowed type assertions, `ParamValue` import + explicit typing, legacy `panelWindows` / `dockLayout` access on `AppState`.

Changed

- **CI typecheck gate** — added `bunx tsc --noEmit` as a first step in the CI workflow (runs before unit + security tests). Added `typecheck` script to `package.json`.

[2.0.18] — 2026-08-20

Added

- **Provider session** — chart aggregators inherit a locked venue identity (`store.provider`: source + stream + market + auth mode). Source changes re-pair the live stream; HUD shows venue and a **Fix** chip when stream mismatches. Data Source Manager follows the chart source unless "different source" is checked.
- **`kraken-rest`** — public Kraken OHLC source, paired with `kraken-ws`.
- **Exchange API keys** — Settings → **Data** stores key/secret/passphrase in a **session vault** (RAM only). Saving a key for the active venue sets `provider.authMode = authenticated`. Secrets never go to localStorage, `pluginsConfig`, or error-share dumps.
- **Signed Binance klines** — with a vault key, history uses HMAC REST then the Worker `GET /api/market/binance/signed/klines` (request-scoped `X-Exchange-Key` / `X-Exchange-Secret`, not a Worker vault). Public allowlisted proxy is unchanged.
- **Venue HMAC signers** — thin Binance / OKX / Bybit / Coinbase / Kraken signers in `src/data/venues/` (no CCXT).
- **Test key** — Settings → Data **Test key** button verifies saved credentials with a signed 1-bar kline fetch. Shows success, 401/403 rejection, or CORS/network warnings.
- **ADR-016** — provider-locked market data architecture decision record.
- **SessionDO multi-venue** — Worker Durable Object relay now supports all 5 venues (Binance, OKX, Bybit, Coinbase, Kraken) via `venue` query param. Each venue's WS URL and subscribe message handled correctly. Browser → DO control messages accept `venue` for resubscribe.
- **Shared venue WS builders** — `src/streams/ws-venues.ts` provides `buildVenueWs()` for both browser-side StreamPlugins and the Worker DO. Single source of truth for URL patterns and subscribe messages.
- **`@hoox-sh/axis-datafeed`** — optional Bun sidecar (`packages/datafeed/`) using CCXT Pro for users who want local WS without running Flask. Mirrors the PYNE datafeed gateway contract: `GET /datafeed/ohlcv`, `GET /datafeed/markets`, `WS /datafeed/watch`, `POST /datafeed/session`. Run with `bun run dev:datafeed`.

Fixed

- **Mixed-provider quotes** — watchlist no longer falls back to Binance tickers for Kraken (or unknown venues).
- **Coinbase live bars** — stream uses Advanced Trade venue candles folded into the chart interval, not ticker buckets.
- **Binance synthetic fallback** — `fallback` defaults **off**, so a network error cannot look like real prices.

[2.0.17] — 2026-08-19

Visible pre-eval squiggles, Problems that drop on edit, and idle lint for named args and declared vars.

Added

- **Named-arg typos** — idle lint flags unknown function/method parameters (`plot(..., coltor=color.green)` → `color=`) from curated signatures, builtin docs, and user-function parameter lists.
- **User-var typos** — assignment / `var` / tuple-unpack / function-parameter names are added to the typo map (declaration is source of truth), so `length = 14` then `lenght` suggests `length`.

Fixed

- **Lint underlines** — pre-eval / error marks use a high-precedence squiggle (SVG + wavy underline + inset line), not CodeMirror `baseTheme`, so they stay visible against Tailwind. Empty decoration sets rebuild when underlines are on. The tab-switch effect no longer loops and cancel the idle lint timer, which left the buffer with Problems but no squiggles.
- **Stale Problems** — pre-eval rows drop as soon as the buffer diverges from the last linted source. Idle lint no longer stamps the live buffer onto old diagnostics, and rescheduling the same buffer does not wipe a completed lint (fixing a typo no longer leaves the old error in the list).

[2.0.16] — 2026-08-19

Restore editor hover, lint underlines, and chips after a dead Settings bag and clipped hover cards.

Fixed

- **Editor intel dead bag** — an all-off persisted Settings bag (checkbox mount / store-proxy spread) no longer disables hover, lint underlines, completions, and chips. Hydrate recovers defaults; user patches write `rev`.
- **Hover cards clipped** — Pine hover tooltips set `clip: false` so overflow-hidden editor chrome no longer hides cards on the first lines.
- **Hover hide-on-apply** — decoration / selection transactions no longer dismiss hover cards; only document edits do.

- **Editor hover catalog** — bare `close` / `new` no longer steal `strategy.close` / `array.new`. Hover facts win; completions still walk modules.
- **`display.pane`** — real Pine v5+ member is allowlisted (no more typo → `display.none`). Invented `math.isnan` / `math.isfinite` completions removed.
- **Editor intel timeouts** — Hover / Completion timeout settings now reach `/lsp/*` (were hardcoded).
- **Intel number fields** — clamp on blur / spinner, not every keystroke, so `1500` is typeable.
- **Diagnostic hover** — respects Show errors / warnings / typos / info flags.
- **Pre-eval flags apply live** — toggling lint generation immediately re-runs or clears marks.
- **Hide + tables** — `table.*` HUDs follow visibility; drawings / fills / `barcolor` are owner-scoped so hide with a sibling no longer leaves the hidden script's artifacts.
- **Hide exclusive pane** — empty oscillator strip is destroyed; show recreates it.
- **`*br` primitives** — `destroyPane` detaches line-break primitives; attach failure is not cached; segments are cached and culled to the visible range.
- **Last-bar drawings** — script paint no longer grows `rightOffset` up to 500 bars for wall-clock `timenow` (one-bar `bar_index+1` still maps).
- **PYNE Agent seed** — same-origin plugin install no longer writes a hardcoded Worker URL; chat stays off until the operator sets the endpoint.
- **Unstamped last-run marks** — engine underlines drop unless `meta.axisSource` (or `script`) still matches the buffer.

Improved

- **Hover facts** — `hl2` / `ohlc4` / `hlcc4` use midpoint / OHLC / HLCC wording (only `hlc3` is typical price).
- **`plotshape` / `plotchar`** — curated params include `format`, `precision`, `force_overlay`; `series` is `int/bool`.
- **Docs** — `areabr` is only on the broken-segment row; debugging mermaid includes pre-eval → Problems.

[2.0.15] — 2026-08-19

Editor intelligence settings, Supertrend `linebr` gaps, live script hide/run, and PYNE Agent chat markdown.

Added

- **Editor intelligence settings** — Settings → **Editor** (and command *Open Editor Settings*) exposes every lint / pre-eval / hover-card / signature-hint / autocomplete / underline / gutter / inline-chip option, plus idle / tab-switch / remote timeouts. Defaults match the previous hardcoded behavior; **Reset defaults** restores them.
- **Strategy chart trade size** — entry/exit markers print `Long N` / `Short N` (and exit id + qty) so the filled amount is visible on the chart, not only the order id.
- **Hide plot names on last-value labels** — Settings and chart **[T]** clear `RSI` / `Overbought` titles beside the last value; the numeric label stays. Independent of **[N]**.

Improved

- **Pre-eval timings** — idle lint stays **1s** after the last keystroke (configurable 200–3000 ms). Tab switch uses a separate **200 ms** beat (was a hardcoded 120 ms). Remote `/lsp/diagnostics` wait is **2s** (configurable) and only starts after idle; local marks still publish immediately. Stale "2s idle" editor comment removed.
- **PYNE Agent chat markdown** — replies render headings, lists, and **bold** instead of raw `###` / `1. **Foo**`. Plugin default API endpoint is empty (set in plugin config); no hardcoded Worker URL.
- **Editor hover + param checklist** — mouseover covers keywords, types/qualifiers, series (`close` / `bar_index`), namespaces (`ta.`), user inputs/functions, and hex colors. Call hints list type + default; current param is emphasized (no strikethrough on used). Hover cards use a void-theme header, parameter definition list, and inset example.
- **Pre-eval / Problems / debug chips** — idle lint is 1s; `study()` and bare `security()` warn; duplicate `indicator()` / `strategy()` warn; Problems label **pre-eval** vs **run**; debug chips have higher contrast and truncate long logs.
- **Live scripts auto-run** — starting a stream immediately re-runs every **visible** script on the chart (no extra Run). Hide pauses execution and clears that script's plots (series, markers, drawings, fills, `barcolor` when it is the last visible script); show re-runs it. `library()` sources stay out of the live / reapply loop.
- **Editor type highlighting** — `int` / `float` / `string` / … and declared UDTs / enums (`type Point`, `export enum Easing`, `m.Easing`) use the type token in **bold**. `series` / `simple` are bold only as type qualifiers; `m.SuperTrend(` stays a library member.
- **Strategy Properties wiring** — Apply persists only changed `strategy()` kwargs (so leverage is not overwritten by default margin %). Leverage ↔ margin UI no longer writes the auto-filled sibling into the bag. Editor Properties apply on isolate/HPO runs. Execution flags are documented as not implemented in PYNE.

Fixed

- **`plot.style_linebr` / `steplinebr` / `areabr` on `na`** — Lightweight Charts Line/Area series drop whitespace points and connect the remaining samples, so Supertrend-style `bull ? st : na` plots drew a diagonal through inactive stretches. `*br` styles now hide the LWC connector and stroke each finite run (isolated samples stay a tick). `plot.style_line` still spans `na`.
- **Hide mid-run** — an in-flight live apply no longer paints after Hide; empty-bar `startLive` no longer logs `No bars loaded`.
- **Last-value names** — compare / on-chain titles survive **[T]**; volume stays number-only unless a title was set. Default `exactOnCandle` puts `Long N` on the side arrow, not the in-bar circle.
- **Isolate / HPO strategy bag** — explicit `strategyProps: {}` means no rewrite; isolate no longer merges leftover editor broker settings onto a baked study.
- **Last-bar / `varip` drawings** — `line.new` / `box.new` / polylines / linefills that extend one bar past the series (`bar_index + 1`) are no longer snapped to the last candle (that flattened them into a vertical tick). Labels still clamp `timenow` onto the last bar. Historical `varip` **plots** already render as a normal 1-sample-per-bar series (AXIS does not send `realtime_last_bar`; enabling it would reset last-bar `varip` cells under pyne's current re-init).
- **Remote LSP cooldown** — aborting an in-flight `/lsp/*` request (typing, hover leave, pre-eval cancel) no longer starts the 30s failure cooldown, so idle parse diagnostics still reach pyne.
- **Editor Problems / underlines** — a clean pre-eval no longer wipes last-run runtime errors. Pre-eval and last-run marks are unioned while the buffer still matches the run source.
- **`plotshape` / `plotchar` / `plotarrow` `style=`** — completions offer `shape.*` instead of invalid `plot.style_*`.
- **Pre-eval block comments** — `"/*"` and `https://` inside strings are no longer treated as comments (which could block Run).

- **Inline debug line parse** — log values like `RSI: 55` no longer become a chip on line 55; structured `line` wins.
- **`color=` completions** — `color.new` / `color.rgb` / `color.from_gradient` stay in the list (including after `color=color.`).

See also

- Editor

[2.0.14] — 2026-08-17

Strategy hyperparameter search in Results, isolated trial runs, and `/optimize` proxy wiring.

Added

- **Hyperparameter Optimisation** — built-in component plugin (`hyperparameter-optimisation`) and Results → **Optimise** tab. Searches `strategy()` `input.*` values over N trials. Evaluation SoT is pyne `POST /optimize` (TPE / random / grid + holdout / walk-forward); Pyodide falls back to an isolated client loop. Apply winner **merges** into Script Inputs (unsearched fields stay). Strategies only.
- **`runScript({ isolate, bars, signal })`** — headless engine eval so HPO trials do not touch `lastRun`, the chart, or the engine HUD. Live multiplex defers while a study is running.
- Vite / Docker nginx **proxy** **`/optimize`** next to `/run` so same-origin VPS and `bun run dev` hit Flask.

See also

- Hyperparameter Optimisation

[2.0.13] — 2026-08-17

Call-parameter intelligence, richer Pine highlighting, script input enums, and Pine table lifecycle.

Added

- **Call parameter intelligence** — hover on `plot` / `input.int` / `ta.sma` (and other curated calls) shows parameter docs + an example. After `,` inside a call, completions offer remaining named args (`title=`, `minval=`) in a **Parameters** section and already-used ones under **Already used**. A signature hint below the cursor lists every parameter and marks used / current / unused.

Improved

- **Pine editor highlighting** — series builtins (`close`, `bar_index`, `time`, OHLC) use a cyan token distinct from user variables; `import … as` aliases and library export members (`m.Easing`) use a violet family; types get a warm token. New chrome vars `--color-cyan`, `--color-editor-builtin`, `--color-editor-lib`, `--color-editor-lib-member`, `--color-editor-type`.

Fixed

- **Script Inputs enum values** — `input.enum(m.Easing.linear)` no longer shows the import/Python path as a text field. Members are inferred from `enum` declarations (local + imported library source), values normalize to `Type.member`, and the modal seeds from that script's `runResults` (not another indicator's `lastRun`).
- **Script display name** — applied scripts / pane badges always use the Pine `indicator()` / `strategy()` / `library()` title (not engine `"plot"` or a stale file name).
- **Pine tables after delete** — table HUD only shows tables from still-applied scripts' `runResults` (orphans / `__editor__` leftovers no longer stick). First-run migrates run cache to the real script id; last-script delete clears script drawings.
- **Pine table layout** — grid size expands from cell extents when engine understates rows/columns; frame/border and cell align improved.

[2.0.12] — 2026-08-17

Editor color chips + type declarations, chart label/marker lifecycle, topbar icon map, debug UX.

Added

- **Editor inline color chips** — square swatches (line-height × line-height) before Pine color literals (`#hex`, `color.red`, `color.rgb(...)`, `color.new(...)`). Click selects the range.
- **Add type declarations** (editor overflow → Source) — after a successful first run, inserts missing Pine **type1** (`series` / `simple` / `const`) and **type2** (`int` / `float` / `bool` / `string` / `color` / …) on untyped assignments.
- **`ICON_MAP`** (`src/ui/icon-map.ts` + `Icons` in `icons.tsx`) — one Lucide glyph per product intent; topbar panel buttons each use a unique icon.

Improved

- **Panel hamburger menu** — consistent 12px label / 14px icons; **New window** → **New tab**.
- **System Logs** — clear control uses trash icon instead of ×.
- **Topbar panels** — reordered List → Editor → Library → Scripts → Inputs → Layers → DSM → On-Chain → Alerts → Values → Results → Script Logs → System Logs → Status.
- **Inline debug / Chart pins** — overflow Debug items include full how-to on hover; toggle shows status-bar tips.

Fixed

- **Duplicate last-value labels** on indicator panes (editor owner + real script id stacked series).
- **Strategy long/short labels** stayed after remove — trade markers are owner-scoped and cleared on detach.

- **Chart chrome** — removed top-right fullscreen / chart-only icon buttons (F11 / Shift+F / Esc still work).
- **Inline debug enable gate** — chips only when explicitly on; pin gutter class clears when empty.

[2.0.11] — 2026-08-17

Chart script lifecycle, per-script indicator panes, pyne-worker engine, panel drag UX.

Fixed

- **Panel title bar** — a plain click no longer undocks the panel to float; undock/move starts only after the pointer is dragged past a small threshold (hamburger still: click = menu, hold/drag = move).
- **Non-overlay indicators** — each `overlay=false` script gets its own sub-pane (`ind_<id>`) instead of stacking into a single shared `indicator` pane. Removing one script only clears that owner's series and no longer destroys siblings' full history (was left with 1–2 bars).
- **Chart scripts lifecycle** — add/remove/reopen stays correct: owner-scoped detach (no leftover series/empty panes), ChartHost no longer wipes re-applied scripts after history paint, centralized `reapplyChartScripts` restores visible scripts (with saved inputs/strategy props) after symbol load / cache paint; applied scripts are sanitized on hydrate and re-run on reopen.

Added

- **Engine** `pyne-worker` — built-in edge evaluator plugin (default `https://pyne-worker.cryptolinx.workers.dev`), API key field, Workers Manager / Settings presets. Distinct from AXIS data-plane Worker and Flask `server` .

[2.0.10] — 2026-08-16

Market data resilience, Script Settings input layout, and local-first editor LSP.

Improved

- **Script Settings → Inputs formatting** — Pine `group` , `inline` , `tooltip` (`\\n`), and `active` / `active=<ident>` layout in the modal (TV Settings-style sections, same-line clusters, `?` help, grayed inactive fields). `input.text_area` uses a multi-line control.
- **Editor LSP hover + autocomplete** — local builtins/doc annotations show **immediately** (no wait on a dead Backend URL); remote Pro API is short-timeout + 30s cooldown after failure. **⌘/Ctrl-Space** triggers completion; tooltip z-index raised so hover/suggest stay above chart chrome.

Fixed

- **Browser market data / plugins (CORS "status null")** — when the page cannot reach venue hosts or remote plugin modules (geo, firewall, port 9443, extensions), AXIS no longer hard-depends on a single cross-origin path:
  - **Binance REST** tries `api.binance.com` → `data-api.binance.vision` , then the Worker allowlisted proxy `GET /api/market/binance/{klines,ticker/24hr,exchangeInfo}` .
  - **Binance WS** rotates `stream.binance.com:9443` → `:443` → default 443 → `data-stream.binance.vision` .
  - **PYNE Agent plugin** ships same-origin at `/plugins/axis-pine-agent.js` (legacy remote URL migrates on restore); agent API endpoint still defaults to the pyne-agent Worker.
  - **Service worker** navigation never rejects `respondWith` (offline shell HTML); cache version **v5**.

Added

- **Worker market proxy** — `worker/src/market.ts` public allowlist for Binance klines / 24hr ticker / exchangeInfo (short isolate cache; not an open reverse proxy).

[2.0.9] — 2026-08-15

Editor symbol catalog, richer Pine highlighting, CORS for Pages previews, and CI greens.

Added

- **Editor symbol & emoji manager** — status-bar **Symbols** catalog of TV-editor-safe arrows, box drawing, blocks, marks, spaces, and chart emoji (IBM Plex Mono vs wide). Insert raw, quoted, or `plotchar(…)` .

Improved

- **Pine highlighting** — stateful multiline strings + `/* */` blocks, `\\n` / hex-color atoms, namespaces (`ta.` , `label.new`), types, control vs definition keywords, function names before `(` .
- **Format** — leaves continuation lines of an unclosed string or block comment untouched.

Fixed

- **CORS** — Cloudflare® Pages previews (`*.pynescript-axis.pages.dev`) can call `https://axis.hoox.sh` . Pyne always allowlists those product Origins and treats `GET /health` as a free CORS path (AXIS Settings probe). Worker `pickOrigin` already echoed the same hosts.
- **CI** — library publish stays on the local cache when leftover/invalid git credentials fail remote (was `GitHub: Bad credentials` in the full unit suite). Smoke waits on `axis-status-message` and opens Plugins via Runtimes. Vite build on macOS/Windows resolves Solid `Topbar.tsx` / `Watchlist.tsx` instead of same-stem legacy JS. Docker push uses the `release` bake group (no `:local` docker.io tags) and bake-action v6 env overrides.

Changed

- **Pyodide PYNE wheel** — vendor `pynescript-0.3.7` (`Runtime.run(libraries=)` for `import ns/Name/ver`, ring-series skip, runtime hot path). Engine catalog, legacy JS engine, SW cache test, and editor builtins synced from pyne 0.3.7.

[2.0.8] — 2026-08-15

Architecture modal, library publish emulator, drawings/plot parity, and the post-2.0.7 hardening stack.

Changed

- **Pyodide PYNE wheel** — vendor `pynescript-0.3.6` (UDT `array.binary_search*` `sort_field`, drawing export/delete/fold). Engine catalog, legacy JS engine, SW cache test, and editor builtin metadata synced from pyne 0.3.6.

Added

- **Library publish emulator** — `library()` scripts publish to git `{basePath}/published/{ns}/{Name}/{1,2,3…}/lib.pyne` (plus local cache). Library panel **Publish library**, or auto-publish on a successful Run. Consumers `import ns/Name/ver`; AXIS resolves the folder and sends sources to the engine (`register_library_source`).
- **Architecture modal** — topbar **Wire** (and command palette **Open Architecture**) opens a compose-recipe dialog: pick Offline Lab / Live Crypto / CSV Desk / On-Chain / Team Cloud, then swap or switch off any source · stream · engine · storage · dataset slot. Plan name records drift (`Live Crypto +1 −1`); Apply commits `setActivePlugin` + optional on-chain panel.
- **Pine `label` styles / yloc** — normalize passes through `style`, `yloc`, `size` / `text_size`, `textcolor`, and `color`; SVG paint supports `label.style_label_up|down|left|right|center` (and bare tokens), bubble tips, icon markers (`xcross`, triangles, …), and `yloc.abovebar` / `yloc.belowbar` when OHLC bars are available (default `yloc.price` + bubble for unknown styles).
- **About AXIS modal** — click the topbar HOOX/AXIS brand (or command palette **About AXIS** / Help → About) for product, author, and HOOX ethos from hoox.sh/manifesto, with links to AXIS / PYNE / docs.
- **Script settings → Properties** — when a `strategy()` is loaded, a **Properties** tab exposes broker parameters (initial capital, order size / pyramiding, commission, leverage / margin, process orders on close, calc flags). Overrides persist per script and are merged into `strategy()` on run without rewriting the editor buffer.
- **`linefill.new` paint** — normalize + SVG quad fill between two lines (pairs with pyne `export_for_api` linefill serialization).
- **`barcolor()`** — per-bar candle body/wick tint from `kind: barcolor` series (LWC color fields).
- **plotshape multi-script** — shape markers are owner-scoped by script id so one run does not wipe another script's shapes.
- **Non-overlay drawings** — `overlay=false` scripts paint geometry on the indicator pane Y-scale; `force_overlay` still routes to price.
- **Compile `line.set_*`** — pyne folds set events onto handles before export (final geometry for AXIS).

Improved

- **Editor default width** — factory / layout-reset docked editor width is **50vw** (was fixed 460px); existing persisted widths unchanged.
- **Editor gutters on demand** — line-number column stays content-width (digits only; overrides CM 20px min); diagnostic + inline-debug gutters hide until they have markers (profiler-style).
- **Modal focus trap** — shared `installFocusTrap` cycles Tab inside Settings / About and restores prior focus on close.
- **Chart a11y region** — ChartHost exposes `role="region"` with symbol/interval/bar-count label; load success/error uses polite/assertive SR announcer.
- **Session DO stream sanitize** — symbol `^[A-Z0-9]{1,20}$` + Binance interval allowlist before upstream WS URL is built.
- **Prod plugin remote allowlist** — production builds default-deny remote plugin hosts (same-origin `/plugins/*` + `data:` + allowlisted PYNE Agent host); override with `VITE_ALLOW_REMOTE_PLUGINS` / `VITE_PLUGIN_REMOTE_ALLOW`.
- **Plot fill apply** — skip SVG rebuild when fill fingerprint matches; pan/zoom paint uses visible-range cull + pixel-budget stride on dense bands.
- **Runner OHLCV times cache** — live re-apply reuses the times axis when `chartDataGen` + length + last time are unchanged (open-bar ticks).
- **Webhook URL policy** — https-only, no credentials, reject loopback/private IP literals; 8s AbortController timeout.
- **Presentation Escape** — detect AXIS modals via `[role=dialog][aria-modal=true]` (not only native `<dialog open>`).
- **CSP baseline** — nginx + Bun static server send Content-Security-Policy (self scripts, wasm-unsafe-eval, https/wss connect, frame-ancestors self).
- **First-paint bundle** — chart PWA critical path no longer statically imports CodeMirror/editor or `@tauri-apps/*`: `index.tsx` dynamic-imports App vs EditorApp; `EditorPane` is Solid `lazy` + `Show` when docked editor is open; Tauri shell install is dynamic-imported after a lightweight `isTauriShell` check.
- **Multi-chart slot bars reactive** — `chart-registry` stores per-slot `bars` / `chartDataGen` in a Solid store so inactive ChartHost slots re-paint when those paths change.
- **PWA SW update UX** — stop blind `SKIP_WAITING` without a reload path; post skip-waiting only when activating a waiting update, and soft-reload once on `controllerchange` (refreshing guard) so mixed old/new modules never stick. Still skips DEV and Tauri.
- **Heavy live re-runs** — at ≥10k bars, `every-tick` indicator re-runs are treated as `bar-close` so full OHLCV is not re-encoded for the engine on every open-bar tick (`effectiveLiveRerunMode`).
- **Offline history fallback** — when venue/source fetch fails, `loadSymbolData` paints warm `bars-cache` series and reports `Offline · N cached bars` (telemetry `degraded`).
- **SW runtime cache cap** — `axis-runtime-v4` soft-caps at 96 entries (FIFO trim after put); version bump clears prior unbounded `v3` runtime caches.
- **Watchlist quote batching** — multi-symbol WS ticks coalesce to one Solid price-map write per animation frame.
- **Live tick rAF coalesce** — multiplex keeps only the newest bar until the next animation frame so trade-ticker floods (Coinbase) do not thrash store + LWC every message.
- **Pane resize coalescing** — `PaneManager` ResizeObserver callbacks flush once per rAF (was synchronous `applyOptions` per pane entry).
- **Smart series apply fingerprint** — overlay tip no-op when length + last time **and** last value match (skips redundant LWC `update` on silent re-runs).

- **Overlay tip-only re-apply** — silent live `syncOverlayLines` updates the last point without a full `toLwcLineData` map when prior meta length + lastTime match (falls back to full setData on length/time change).
- **StatusBar / Results strategy reports** — no longer rebuild trade pairing on every live `store.bars` path update; depend on `lastRun` / `chartDataGen` / fill-mode prefs.
- **`loadBars` batching** — multi-field store writes run inside Solid `batch()` for a single reactive flush.
- **Vite critical path** — `modulePreload` drops CodeMirror / pyne-builtins from first paint; gzip enabled on Docker nginx; `X-Frame-Options: DENY`.
- **OHLCV bulk paint hardening** — `mapBarsToPriceData` / volume mapper drop non-finite rows so one NaN bar cannot blank LWC series.
- **Plot styles** — distinct series kinds for `plot.style_columns` (vs histogram), `plot.style_cross` (discrete markers, no connector), and `plot.style_stepline_diamond` (stepline + vertex markers). plotshape diamond/cross map to square marks with optional `+` / `×` glyphs. LWC still lacks column-width and native diamond/cross point markers — documented in charting docs.
- **`line.style_arrow_*`** — script lines paint SVG arrow heads for `arrow_left` / `arrow_right` / `arrow_both` (were solid stroke only).

Fixed

- **Editor Problems panel close** — auto-open only when diagnostic count *increases*; no longer re-forces open on every pre-eval/store tick while problems remain (statusbar toggle can stay closed).
- **Worker `/api/run` auth fail-closed** — when `EXTERNAL_BACKEND` is non-empty or `PYODIDE_IN_WORKER=enabled` and `ALLOW_OPEN_KEYS` is not `"1"`, `POST /api/run` requires `requireApiKey` (also still forced by `API_KEYS` / `REQUIRE_RUN_AUTH`); local demos keep open keys without burning unauthenticated compute on a real backend.
- **Script-only drawing layer leak** — `clearScriptPaneLayers()` on ChartHost unmount (RO + time-scale subs were retained after multi-chart dispose).
- **Companion panel postMessage** — origin-bound target + receiver check (no more `*`).
- **Legacy results event CSS class** — allowlist kind tokens before interpolating into `class` (attribute breakout).
- **ResizeHandle multi-touch** — ignore non-primary pointer ids mid-drag.
- **bars-cache IDB integrity** — `pagehide` / `beforeunload` flush pending debounced puts.

Tests

- **CI workflow** — `.github/workflows/ci.yml` runs `bun run test`, `test:security`, and Playwright `@smoke` (Chromium) on PR / main.
- **Optional firehose bench** — `tests/bench/` (`AXIS_BENCH=1` / `bun run test:bench`): 500k `appendBar` ticks on ~50k history with soft wall budget; skipped by default.
- **Drawings / plot-visuals** — unit coverage for `force_overlay` normalize, `linefill` edge cases, `barcolorSeriesToMap` / `coerceBarColor`, and GC with `linefill` present.
- **plot-visuals** — columns / cross / stepline_diamond kind mapping, histogram-family helpers, diamond/cross shape + glyph defaults.
- **chart-type / heavy-data** — non-finite OHLC filter on price + volume mappers.
- **store-append-scale** — always-on `appendBar` same-time + cap invariants.
- **multiplex-heavy-rerun / sw-strategy / load-symbol** — heavy re-run mode, runtime cache FIFO cap, offline bars-cache fallback.
- **register-sw** — pure update helpers (`shouldSoftReloadOnControllerChange`, skip-waiting) + controllerchange single-reload mocks.
- **security / runner / fills** — remote plugin prod allowlist, webhook URL policy, OHLCV times cache, plot fill signature.

[2.0.7] — 2026-08-12

Pine drawings diagnostics and plotarrow direction; pairs with pyne 0.3.5 last-bar line export.

Fixed

- **`line.new` / drawings apply** — prefer DrawingLayer fallback when applying Pine drawings; warn when the engine returns drawings that normalize to zero or when no layer is mounted (load bars first).
- **`plotarrow`** — negative series samples render as down arrows (were always up).

[2.0.6] — 2026-08-12

Test suite isolation and store hydrate defaults hardening.

Fixed

- **Tests** — isolate store/state defaults across suites; strategy-extra uses event prices (not bar fills); engine `isReady` mock returns JSON health; hydrate never restores plane telemetry from disk.
- **Store seed** — createStore no longer aliases nested `DEFAULTS` objects (telemetry/panels/live), so path updates cannot poison factory defaults.

[2.0.5] — 2026-08-12

Panel manager defaults / reset / chart overlay; classic Status + System Logs chrome; editor open-in-new-tab.

Added

- **Panel manager** — per-panel **default position** (dock + size + float coords in `PANEL_META`); panel menu **Reset to default**.
- **Chart overlay** — per-panel toggle and bulk **Chart overlay: all on/off** (`setAllPanelsChartOverlay`); edge-docked panels float over the chart without shrinking it.
- **Editor** — **Open in new tab** in overflow panel options (and left dock menu).

Changed

- **Status bar** — restored classic fixed footer look (no panel title chrome).
- **System Logs** — restored classic collapsible strip (expand/collapse body); topbar shows/hides the strip. Label **System Logs** (not bare “Logs”).

[2.0.4] — 2026-08-11

Script Logs / System Logs naming and Status / System Logs topbar toggles.

Changed

- **Script Logs / System Logs** — Script `log.*` pane titled **Script Logs** (editor tool + panel); app telemetry titled **System Logs**.
- **Status & System Logs** — topbar toggles for **Script Logs**, **System Logs**, **Status**.

[2.0.3] — 2026-08-11

Runtimes hub, editor save-before-run / idle lint, Workers Manager reliability, and hardened VPS probe routing.

Fixed

- **Workers Manager endless probing** — open-effect no longer tracks `store.endpoint` (backend changes re-fired probes); every probe gets a hard timeout merged with the modal abort; Pyodide probe uses HEAD instead of downloading full `pyodide.js`; `probeAllWorkers` uses `allSettled`.
- **Workers Manager / hardened VPS** — pyne Pro probes use same-origin `https://axis.hoox.sh/health` (nginx → loopback :5002) instead of client `127.0.0.1:5002`, which UFW correctly blocks on the public VPS.

Changed

- **Runtimes hub** — single dialog for **Status** (Workers Manager) and **Plugins** (catalog / install / library); topbar **Runtimes** entry; cross-links between sections; models remain separate.
- **Save before Run** — clicking Run (editor header, topbar split, Mod-Enter, command palette) auto-saves the active script to the library when the tab is dirty or not yet bound (`libraryId`). Save failure blocks the run.
- **Idle lint** — editor underlines (e.g. `plt()` → unknown) reappear after **2s** without typing, not only after Save/Run. Mid-keystroke marks stay cleared; bare call typos like `plt()` flagged locally.
- **Workers Manager UX** — distinct icon per catalog worker; **Usage / When to use** copy on Detail + Install; docs updated.
- **Editor line numbers** — gutter width follows digit count only (no fixed min-width / heavy padding).

[2.0.2] — 2026-08-11

Editor chrome polish and docked panel **hover slide**.

Added

- **Panel hover slide** — docked panels can collapse to a peek strip and slide open/closed on pointer enter/leave. Toggle **Slide on hover** in the panel hamburger menu (left/right/bottom only). Store APIs: `setPanelHoverSlide`, `togglePanelHoverSlide`, `isPanelHoverSlide`. Preference persists on `panelChrome[id].hoverSlide`.

Changed

- **Editor chrome polish** — header tools always show **icon + label** (no hover slide-in); **Format** removed from the editor header and tab strip (still available via overflow **Format document** and `Shift+Alt+F` / `Mod+Shift+F`).
- **Library from editor** — primary **Library** tool opens the script library panel; overflow menu also has **Open library**.

Fixed

- **Tab close / switch** — closing a tab left of the active one no longer jumps focus; CM buffer is snapshotted before add/close/switch; `setDoc` no-ops on identical content.

[2.0.1] — 2026-08-11

Security and performance release from the multi-agent **harden-perf** audit (4 specialist scanners + synthesis). Report: `docs/devops/harden-perf-audit-2026-08-11.md`.

Security

- Fail-closed Worker auth when D1 is bound without `API_KEYS` KV (`API_KEYS_REQUIRED`).
- CORS: product hosts + project-scoped `*.pynescript-axis.pages.dev` only (no open `*.pages.dev`).
- `/api/run`: auth when `API_KEYS` or `REQUIRE_RUN_AUTH`; per-IP/key rate limit (30/min); script/bars size caps; upstream proxy timeout.
- Git OAuth: env `GITHUB_OAUTH_CLIENT_ID` / `GITLAB_OAUTH_CLIENT_ID` wins over body `clientId`; tighter start/poll rate limits.
- Watchlist: build rows with `textContent` (no unescaped symbol `innerHTML`).

Performance & reliability

- bars-cache: memory-first `getCachedBars`, IDB hydrate, `getCachedBarCount` / range without full clone when warm.
- DSM gap-fill progress uses `getCachedBarCount` instead of full series loads.
- engine-ws: 45s dead cool-down — keep dead client so live re-runs do not thrash reconnects.

Commits

- `0a668e00` — fix(security,perf): gate /api/run, OAuth clientId, WS cool-down
- `156221b9` — fix(security,perf): fail-closed Worker auth, CORS, bars-cache

Full history (recursive)

2026-09 (82 commits)

Features

- `1dc6c089` (2026-09-12) — feat(cli): cloud-storage setup, keys, and D1 schema deploy
- `c111371a` (2026-09-12) — feat(storage): copy-on-switch, Worker settings, version history
- `a9f7bb6e` (2026-09-12) — feat(release): VERSION single source of truth with update manager and close guard
- `319403fb` (2026-09-10) — feat(editor): minimap, folding, indent grid, tab badges, workers full width
- `8e9e8d6b` (2026-09-10) — feat(editor): Language Feature Bar with live activity chips
- `4d565427` (2026-09-08) — feat(cli): standalone binaries via bun compile; fix Docker prepare-hook break
- `1fef94c3` (2026-09-08) — feat(ui): default TF 15m, studio opens on Settings, non-modal Workers, no top status
- `77d8600e` (2026-09-08) — feat(cli): installation self-check on start + doctor cli-install row
- `716a26eb` (2026-09-07) — feat(ui): mobile-first responsive shell (phone sheets, tab bar, force-single chart)
- `74418b05` (2026-09-07) — feat(results): events views (Open⇄Close), single-column strategy tab, rich saved stats
- `e00b6c97` (2026-09-06) — feat(shortcuts): shortcuts modal + Settings → Keyboard recorder

- `6efa590a` (2026-09-06) — feat(shortcuts): palette extensions with live shortcut rendering
- `47706d85` (2026-09-06) — feat(shortcuts): wire chart keymap through the dispatch hub
- `acc5ce93` (2026-09-06) — feat(shortcuts): editor keymap extensions with Prec.high line ops
- `e07ecb8c` (2026-09-06) — feat(shortcuts): wire app-level bindings through the dispatch hub
- `2795ac72` (2026-09-06) — feat(shortcuts): capture-phase dispatch hub with dialog-skip guard
- `50467b3d` (2026-09-06) — feat(shortcuts): persisted overrides slice on the app store
- `902741d1` (2026-09-06) — feat(shortcuts): chord parser, registry, and default bindings
- `da5e96d7` (2026-09-06) — feat(data): route all dataset paths through the DatasetStore
- `f597806b` (2026-09-06) — feat(ui): dataset persistence switch + sliced-backfill progress chip
- `257342d1` (2026-09-06) — feat(data): DSM-first load pipeline with progressive paint
- `a5caf0ab` (2026-09-06) — feat(data): DatasetStore with pluggable persistence sinks
- `d105e161` (2026-09-06) — feat(data): dataset merge with conflict detection & policies
- `19c92dd1` (2026-09-06) — feat(data): dataset validation & repair with gap classification
- `a63c3b69` (2026-09-06) — feat(ui): move Venue picker to front of Market group
- `be830abe` (2026-09-05) — feat(chart): per-tool drawing settings (v2.3.0)

Fixes

- `99eb5f20` (2026-09-12) — fix(ui): Studio Settings home, Workers fullscreen, minimap height
- `9e963eab` (2026-09-12) — fix(docker): skip missing VERSION stamp files in image build
- `ff4674a1` (2026-09-12) — fix: harden PWA, editor, e2e, and Docker; release 2.6.2
- `3a5de985` (2026-09-12) — fix(pwa): retry transient fetch failures in SW cache-first
- `cb089b18` (2026-09-12) — fix(e2e): retry twice on CI runners
- `b3915c90` (2026-09-12) — fix(e2e): wait for boot in openStudio, upload failure artifacts
- `8f8c91cd` (2026-09-12) — fix(docker): copy version-stamped files into build context
- `60aa2c4e` (2026-09-12) — fix(ci, docker): copy VERSION into build stage, harden studio helper
- `bd8d1c0c` (2026-09-11) — fix(pyodide, editor, cli): bridge traceback tail, feature-bar popover, host-native smoke
- `364776c5` (2026-09-09) — fix(pyodide): pass bars via globals + json.loads, surface full traceback
- `e4888c72` (2026-09-08) — fix(ci): SHA256SUMS used CI paths for binaries — sha256sum -c unusable
- `7d16a7bb` (2026-09-08) — fix(ci): CLI tgz mv target — packages/cli/.. is packages/, not the root
- `144ef3d3` (2026-09-08) — fix(ci): GITHUB_ENV var casing — release-assets referenced empty vars
- `cdb188b9` (2026-09-08) — fix(ci): worker bundle sanity check — wrangler.toml is gitignored
- `cb6f00f5` (2026-09-08) — fix(ci): check-versions fails on Windows runners — CRLF line endings
- `d4082f4c` (2026-09-08) — fix(shortcuts): mount ShortcutHub + keypress feedback
- `c06a14cf` (2026-09-07) — fix(test): make CI test suite green (server hook timeout, matchMedia stubs)
- `81f24c76` (2026-09-07) — fix(lint): resolve all Biome lint errors (368→0); add Biome + Prettier tooling
- `7d9f2dec` (2026-09-07) — fix(ui): resolve 23 UI-review findings across editor, drawings, panels, themes
- `464a9497` (2026-09-06) — fix(architecture): only preload Pyodide when the DOM can host asset links
- `8d5daf5c` (2026-09-06) — fix(worker): clean up tsc errors under noUncheckedIndexedAccess & exactOptionalPropertyTypes
- `7ee49cc4` (2026-09-06) — fix(data): DatasetStore memory-first, DSM paint/gaps, v2.4.0
- `520869dc` (2026-09-05) — fix: setup review — doctor, live HUD, static health, 2.3.1
- `16499ac7` (2026-09-04) — fix(results): skip live-tick persist and unstick studio e2e

Refactors

- `ac0d8a90` (2026-09-08) — refactor(editor): move Script Logs into the editor bottom bar

Documentation

- `477084bf` (2026-09-08) — docs(changelog): note bare-filename SHA256SUMS for binaries
- `055d73d5` (2026-09-08) — docs: CLI-first command surface (axis command primary, bun run axis:* as repo alias)
- `9b75c459` (2026-09-08) — docs(readme): real badge row, stack cross-links, unified stack footer
- `efa7e87c` (2026-09-08) — docs(readme): add official Codecov coverage badge
- `e38a72f9` (2026-09-07) — docs: sync docs to v2.5.0 — shortcuts, mobile shell, results views
- `eecca17d` (2026-09-06) — docs(changelog): regenerate full history for v2.4.0
- `e9df22fe` (2026-09-06) — docs(changelog): DSM-first dataset store, validation, persistence switch
- `d10a9ea4` (2026-09-05) — docs(screenshots): wire stills into README and matching guides
- `adcb846d` (2026-09-05) — docs(screenshots): close editor except on editor-focused stills
- `3211bfe0` (2026-09-05) — docs(screenshots): recapture at 1920x1080 and 2560x1440
- `9b820a77` (2026-09-05) — docs(screenshots): capture PWA, Studio, and CLI stills plus GIFs

CI

- `60a0c8d6` (2026-09-08) — ci(release): attach CLI tarball + worker source bundle to GitHub Releases
- `d5f0cf89` (2026-09-08) — ci(hooks): versioned .githooks — pre-commit biome (staged), pre-push tsc + lint

- `cdb68ff9` (2026-09-08) — ci: split Codecov uploads per flag, bump action to v7, target 75%
- `9fe171a6` (2026-09-08) — ci: upload unit + CLI coverage to Codecov
- `637cfa87` (2026-09-06) — ci(cli): full npm release pipeline with tag guard, registry verify, Node smoke

Tests

- `079f1569` (2026-09-12) — test(e2e): drawings overlay spec with chart DOM harness
- `3e6778ea` (2026-09-12) — test(coverage): lift scoped core gate 82.8% to 89.9%
- `6527e3de` (2026-09-10) — test(editor): cover folding, indent grid, minimap class
- `2c9643b3` (2026-09-05) — test(pwa): stub beforeinstallprompt without DOM Event
- `c0d88523` (2026-09-05) — test(load-symbol): isolate live stream and leftover scripts

Chores

- `b327a28a` (2026-09-10) — chore: stop tracking .grok/ and AGENTS.md (local-only)
- `28420c09` (2026-09-08) — chore(release): v2.6.1 — sync desktop version stamps + check:versions guard
- `67723020` (2026-09-08) — chore(release): v2.6.0 — shortcut hub fix, keypress feedback
- `a78a7fca` (2026-09-07) — chore(release): v2.5.0 — mobile shell, pyne 0.5.0, zero lint errors
- `8d6a7c70` (2026-09-07) — chore(engines): vendor pynescript wheel 0.5.0 from pyne
- `9ae3e3da` (2026-09-07) — chore(release): v2.4.3 — results events views, single-column strategy, saved stats
- `aaf4d9ac` (2026-09-07) — chore(release): v2.4.2 — UI review fixes, keyboard shortcuts, line ops
- `fa49a6c1` (2026-09-06) — chore(cli): release @hoox-sh/axis-cli 0.2.2
- `36ba501c` (2026-09-06) — chore(release): v2.4.1 — worker tsc compliance
- `e84b8325` (2026-09-04) — chore(release): prepare v2.2.0

2026-08 (237 commits)

Security

- `d2b50220` (2026-08-13) — a11y,security: focus trap, chart region, SessionDO stream sanitize
- `4321c4b9` (2026-08-13) — security,perf: plugin allowlist, webhook policy, CSP, fill/runner hot paths
- `72869743` (2026-08-13) — perf,security,ci: multi-agent optim stack (registry, overlay tip, auth, SW, bundle, CI)
- `ebfc9955` (2026-08-07) — harden chart, data, and run paths for performance and crash resilience

Features

- `783a1003` (2026-08-30) — feat(ui): fullscreen studio overlay, Results JSON tree, and market proxy hardening
- `9081a559` (2026-08-29) — feat(worker): MEXC public REST through /api/market/mexc/* proxy
- `fb193c25` (2026-08-29) — feat(panel): per-panel icon in FloatableShell header
- `9196e4cb` (2026-08-29) — feat(ui): storage-change migration dialog
- `008fb22a` (2026-08-29) — feat(storage): persist strategy/indicator run results via StoragePlugin
- `eb86c103` (2026-08-28) — feat(results): fullscreen results studio, HPO polish, and tier gating
- `28a1f202` (2026-08-27) — feat(settings): add topbar tab to SettingsPage (studio view)
- `339f6a05` (2026-08-27) — feat(settings): add topbar visibility settings tab
- `67b1558d` (2026-08-24) — feat(security): enforce CSP baseline on Cloudflare® Pages deploys
- `3fb40122` (2026-08-24) — feat(data): native MEXC spot venue + PYNE Pro origin pynescript.online
- `09e047cd` (2026-08-24) — feat(ui): single Studio button replaces Wire/Runtime/Settings
- `cd0148ca` (2026-08-23) — feat(cli): ship production-grade @hoox-sh/axis-cli 0.2.0
- `0d45a6f2` (2026-08-23) — feat(ui): full-page studio overlay for Runtime, Wire, and Settings
- `6900c093` (2026-08-22) — feat(ui): slide-over drawers for Settings, Runtimes, and Wire
- `c685a2d4` (2026-08-22) — feat(ui): Venue picker pairs native CEX and CCXT in one control
- `fe0ed9c7` (2026-08-22) — feat(ui): CCXT symbol catalog and hide paired stream
- `95144094` (2026-08-22) — feat(data): CCXT session keys and plugin-config/editor fixes
- `339175f0` (2026-08-21) — feat(ui): advanced config fields in Settings + full ccxt exchange list
- `203b4492` (2026-08-21) — feat(ui): topbar plugin config row for source/stream configSchema
- `6b4d5543` (2026-08-21) — feat(plugins): gateway-aware active source/stream resolution
- `1631975f` (2026-08-21) — feat(stream): ccxt-ws gateway stream plugin
- `c1887afd` (2026-08-21) — feat(source): ccxt-rest gateway source plugin
- `a47baf04` (2026-08-21) — feat(data): gateway transport layer for CCXT datafeed
- `3c72d089` (2026-08-20) — feat(release): AXIS v2.0.18 multi-venue DO, credential vault, CCXT sidecar
- `c42992e8` (2026-08-19) — feat(release): AXIS v2.0.17 editor lint underlines and typos
- `720237a1` (2026-08-19) — feat(release): AXIS v2.0.16 editor intel recovery
- `fd23d4a8` (2026-08-19) — feat(release): AXIS v2.0.15 editor intelligence and linebr
- `cbe6369f` (2026-08-19) — feat(editor): richer hover, param checklist, and pre-eval
- `a2178fc9` (2026-08-18) — feat: live script execution, chart labels, and strategy properties
- `35557be3` (2026-08-17) — feat(release): AXIS v2.0.14 strategy hyperparameter optimisation

- `9ec62e96` (2026-08-17) — feat(release): AXIS v2.0.13 params, highlighting, input enums, tables
- `db3b42c1` (2026-08-17) — feat(release): AXIS v2.0.12 editor UX, chart lifecycle, icon map
- `f96319e3` (2026-08-17) — feat(release): AXIS v2.0.11 chart scripts lifecycle and pyne-worker engine
- `7f85d664` (2026-08-16) — feat(release): AXIS v2.0.10 market proxy, inputs layout, local-first LSP
- `2cae2b72` (2026-08-15) — feat(editor): symbol catalog and richer Pine highlighting
- `9411a16b` (2026-08-15) — feat(pyodide): vendor pynescript 0.3.7 wheel
- `fe6dfb18` (2026-08-15) — feat(library): git versioned publish emulator for import ns/Name/ver
- `4d1e8647` (2026-08-13) — feat(ui): architecture modal to wire plugin slots from recipes
- `814cb21f` (2026-08-13) — feat(editor): 50vw default width; content-sized gutters
- `80f25012` (2026-08-13) — feat(pyodide): vendor pynescript 0.3.6 wheel
- `904ec110` (2026-08-12) — feat(drawings): paint line.style_arrow_left/right/both heads
- `669ee31d` (2026-08-12) — feat(drawings,plot): label styles/yloc, plot columns/cross/diamond
- `8b80166b` (2026-08-12) — feat(drawings): non-overlay pane Y + force_overlay routing
- `46229069` (2026-08-12) — feat(drawings,plot): linefill paint, barcolor, multi-script shapes
- `4b379391` (2026-08-12) — feat(ui): strategy Properties tab in Script settings
- `4bf973e4` (2026-08-12) — feat(ui): About AXIS modal on logo click
- `e96114b0` (2026-08-12) — feat(editor): handle PYNE Agent insert/open script events
- `e29aa644` (2026-08-12) — feat(ui,panels): panel defaults, chart overlay, classic logs chrome
- `a00ad891` (2026-08-11) — feat(ui): dockable Status and System Logs; Script Logs naming
- `7f44667e` (2026-08-11) — feat(runtimes,editor): hub, idle lint, save-before-run, tight line gutters
- `0c4a812d` (2026-08-11) — feat(editor,panels): hover-slide docks and editor chrome polish
- `7abea423` (2026-08-10) — feat(chart): price-scale decimals with auto from symbol/bars
- `235079b7` (2026-08-10) — feat(editor): autoformat, color tools, and chrome redesign
- `9ce5e0f3` (2026-08-10) — feat: plot style chart parity, preeval constants, and storage versions
- `dee91540` (2026-08-10) — feat(ui): fullscreen modes, price-pane scale controls, brand density
- `8aa9dddc` (2026-08-10) — feat(ui): symbol modal and richer Scripts panel cards
- `60502d08` (2026-08-10) — feat(cli): add AXIS CLI for install, setup, deploy, and doctor
- `da9dd6fd` (2026-08-10) — feat: Workers Manager modal and Tauri desktop shell
- `cf938db1` (2026-08-09) — feat(examples): add Pine v6 script library starter pack
- `a21dc811` (2026-08-08) — feat(plugins): install PYNE Agent component and enable component URL load
- `fae48240` (2026-08-08) — feat(onchain): refresh jobs, CSV export, and popular protocol presets
- `61c9993c` (2026-08-08) — feat(onchain): alerts UI, data view, commands, and ADR-015
- `8ad97cb8` (2026-08-07) — feat(onchain): alerts, layers, HUD health, and DSM walk-back polish
- `f91a10f7` (2026-08-07) — feat(onchain): data plane with DefiLlama TVL, DEX OHLCV, and events
- `f62611a0` (2026-08-07) — feat(editor): typo marks, wrap toggle, resizable Problems
- `259204e5` (2026-08-07) — feat(editor): pre-eval marks wrong code and disables Run
- `6a945396` (2026-08-07) — feat(drawings): merge parallel agent tool packs with full tests
- `9427a565` (2026-08-07) — feat(drawings): Gann, fib extras, patterns, and forecast tools
- `963f5f87` (2026-08-07) — feat(drawings): seed types and catalog for next parity packs
- `aaa2cca5` (2026-08-07) — feat(drawings): pitchfork, brush, callout, and more parity tools
- `1abe0795` (2026-08-07) — feat(drawings): registry-based tools toward charting platform parity
- `807262a4` (2026-08-07) — feat(data): dataset manager load window, venue live, expand to now
- `0e4ae885` (2026-08-06) — feat(data): raise historical bars cap to 100k
- `58ff7be5` (2026-08-06) — feat(data): cached datasets browser and Data Manager chart source
- `c1d45d00` (2026-08-06) — feat(ui): high-end theme presets, Scripts panel, Run accent only while busy
- `4a6c5e49` (2026-08-06) — feat(data): background Data Source Manager for multi-page OHLCV backfill
- `b7887c92` (2026-08-06) — feat(editor): slide-in tool labels on hover
- `4d533975` (2026-08-06) — feat(strategy): close fills, slippage next-open, marker prefs; fix script settings focus
- `4705cb96` (2026-08-06) — feat: symbol-scoped drawings, chart refresh, labels, editor docs
- `43b0de57` (2026-08-04) — feat(docker): full Buildx bake + Compose stack for AXIS PWA
- `b51879b0` (2026-08-04) — feat(ui): move chart Theme Manager to its own Settings tab
- `5f6863f0` (2026-08-03) — feat: chart Theme Manager (Pine chart.bg_color/fg_color) and UI hardening
- `26a8358d` (2026-08-03) — feat: GitHub/GitLab OAuth connect, multi-script results focus, hoox-sh org
- `902f3d4f` (2026-08-03) — feat: multi-agent PWA hardening for rock-solid reliability
- `b844a76f` (2026-08-02) — feat(ui): library panel, editor run/tab, live-after-load default
- `ed812a67` (2026-08-02) — feat(chart): render Pine fill(plot1, plot2, color=…) as SVG bands
- `7b481574` (2026-08-02) — feat(library): default git scripts to .pyne under pyne-library
- `52e2aa8c` (2026-08-02) — feat(library): drag-drop .pine files into script library
- `0b0df8e7` (2026-08-01) — feat(editor,topbar): parallel polish — ruler, diagnostics, git, pins, problems
- `10dd5074` (2026-08-01) — feat(editor): show cursor line:col in stats strip

- `c44f4cd9` (2026-08-01) — feat(chart): script action icons on pane name badges
- `ee31517a` (2026-08-01) — feat: TV-class power features from parallel agents
- `a6cd8e1b` (2026-08-01) — feat(editor): inline debug, wrap, stats; layers left slide-in
- `9309f9a2` (2026-08-01) — feat(layers): list and select user drawings in Layers panel
- `f45bf320` (2026-08-01) — feat(drawings): vline, extend, arrow, ellipse + toolbar polish
- `40430f6b` (2026-08-01) — feat(chart): multi-chart layouts, indicator align, editor full height
- `bacedac8` (2026-08-01) — feat(ui): chart reflow with docks, reload/reset, float editor fixes
- `0f9c4433` (2026-08-01) — feat(ui): stack multiple docked panels one below the other
- `f982e0d8` (2026-08-01) — feat(chart): multi-pane sync, scale controls, drawing GC, history depth

Fixes

- `defaccea` (2026-08-29) — fix(data): try MEXC Worker proxy first, direct host as fallback
- `167921f2` (2026-08-29) — fix(worker): update CORS product hosts to pynescript.online (retire .ai)
- `8426f3d9` (2026-08-29) — fix(editor): remove duplicate 'Open in new tab' entry
- `6178bced` (2026-08-29) — fix(ui): status bar chips wobble on hover
- `e0b8d4a8` (2026-08-27) — fix(chart): repaint full series on input override recompute
- `5fe39320` (2026-08-27) — fix(ui): fine-tune settings polish and worker probe
- `ad473102` (2026-08-27) — fix(ui): polish settings pages and fix worker health probe blocking
- `bb702efb` (2026-08-27) — fix(wire): remove separate scrolling sections from wire page
- `f38a0239` (2026-08-26) — fix(compile): display= folding into plot_attrs for data_window-only plots
- `cb9c1773` (2026-08-26) — fix(compile): hline linestyle via plot_meta backfill + default auto mode
- `5bc18f5d` (2026-08-26) — fix(hline): synthesize plot_meta in interpret path for hline linestyle
- `d417025f` (2026-08-26) — fix(multi-chart): restore slot history on focus so each cell computes independently
- `5025302f` (2026-08-26) — fix(hline): carry linestyle through both drawing and overlay paths
- `b07befb9` (2026-08-25) — fix(chart): render table cell valign/text-size; carry bgcolor band titles
- `86b86bd7` (2026-08-24) — fix(chart): never host hlines on bgcolor underlays
- `53cef490` (2026-08-24) — fix(chart): finish PlotSample widening in runner types
- `2db34902` (2026-08-24) — fix(chart): preserve engine color series so bgcolor/barcolor render
- `68b13118` (2026-08-23) — fix(settings): keep user inputs on applied scripts
- `ea13ba50` (2026-08-23) — fix(ui): seed Wire plan on mount and expose source for e2e
- `70d0b378` (2026-08-23) — fix(chart): honor Pine plot display.* flags
- `bed28c62` (2026-08-23) — fix(chart): hide Pine drawings, label pad, histbase, re-own
- `a05e42d8` (2026-08-23) — fix(chart): plotshape pane routing and drawing tool settings
- `098bfee4` (2026-08-23) — fix(chart): drawing move, linefill GC, overlay pane routing
- `55e26ddd` (2026-08-22) — fix(editor): stop afterDot state leak from trailing-dot floats
- `d98048a8` (2026-08-22) — fix(editor): pre-eval typo round 3 — generics, import paths, dotted calls
- `47beaa62` (2026-08-22) — fix(editor): pre-eval typo round 2 — decl forms, curly quotes, linktoseries
- `5f0c3f36` (2026-08-22) — fix(editor): stop pre-eval typo flood on built-in vars, qualifiers, text.*
- `2f8ff1d2` (2026-08-22) — fix(ui): plugin config first-save crash + gateway select options
- `a6323558` (2026-08-21) — fix(ui): single shared plugin config row + exchange dropdown
- `7e74a266` (2026-08-21) — fix(gateway): remote-page loopback resolution + drop shadowed require
- `67361dc6` (2026-08-21) — fix(datafeed): ccxt-ws browser ESM require + ccxt-rest walk-back window
- `aefa3baf` (2026-08-20) — fix(editor): skip hex color tails in pre-eval identifier scanner
- `d408f110` (2026-08-20) — fix: resolve all 23 pre-existing TypeScript errors + CI typecheck gate
- `804c5662` (2026-08-20) — fix(test): reset store provider state in credentials beforeEach
- `b160498e` (2026-08-19) — fix: review follow-up for hover, hide, linebr, and agent seed
- `35efaed7` (2026-08-19) — fix(plugins): render PYNE Agent markdown and drop default endpoint
- `7ccd745d` (2026-08-19) — fix(chart): break plot.style_linebr on na
- `dbcf1f0a` (2026-08-18) — fix: review follow-up for live hide, labels, and strategy props
- `fa57f639` (2026-08-17) — fix(editor): LSP pre-eval marks and last-bar drawings
- `f0235ac1` (2026-08-15) — fix(ci): desktop Watchlist case clash and GHCR-only docker push
- `42cb454f` (2026-08-15) — fix(ci): green unit, smoke, desktop build, and docker bake
- `4d34d82a` (2026-08-13) — fix(editor): allow Problems panel to stay closed
- `eb61d218` (2026-08-12) — fix(drawings,plot): line.new last-bar export path + plotarrow direction
- `299ec83c` (2026-08-12) — fix(test,store): green suite — isolate defaults, health mock, bar fills
- `41099e71` (2026-08-11) — fix(plugins): clearer errors when cross-origin plugin import lacks CORS
- `3bc43323` (2026-08-11) — fix(panels): harden hover-slide reflow for non-DOM test windows
- `0a668e00` (2026-08-11) — fix(security,perf): gate /api/run, OAuth clientId, WS cool-down
- `156221b9` (2026-08-11) — fix(security,perf): fail-closed Worker auth, CORS, bars-cache
- `9bfc801b` (2026-08-10) — fix(editor,run): defer syntax lint; enum completions; protect interactive Run

- `24f9cf68` (2026-08-08) — fix(run): Re-run replace, add-instance menu, clear stuck Running
- `77d2727e` (2026-08-08) — fix(onchain): health probe always targets Worker base
- `b24fbd19` (2026-08-08) — fix(onchain): honor empty health endpoint override
- `f8e3ae47` (2026-08-08) — fix(onchain): always use AXIS Worker proxy; allow product CORS origins
- `794ecadb` (2026-08-08) — fix(onchain): do not proxy DefiLlama via axis.hoox.sh SPA host
- `6d672e34` (2026-08-07) — fix(editor): recognize strategy qty constants; copy problems
- `3557ac99` (2026-08-07) — fix(drawings): keep trendline draft when tool re-syncs mid-place
- `b81991da` (2026-08-06) — fix(deploy): same-origin Pro API on https://axis.hoox.sh
- `3e1f3a76` (2026-08-06) — fix(git): harden storage integrity and OAuth device flow
- `6f870e52` (2026-08-06) — fix(data): backfill from now with gap validate; theme chrome; editor tools
- `ebddd006` (2026-08-06) — fix(data): multi-page walk-back without startTime+endTime trap
- `fdb4c6b1` (2026-08-06) — fix(ui): repair DrawingToolbar JSX after badge layout comment
- `1fd8b64f` (2026-08-06) — fix(ui): keep script badges above drawing toolbar chrome
- `9140e31d` (2026-08-04) — fix(strategy): correct PnL for Long/Short reverse entries
- `260a59c4` (2026-08-04) — fix(strategy): stop zeroing net PnL from placeholder prices/profit
- `0867b1ce` (2026-08-04) — fix(ui): make log copy buttons work on HTTP (clipboard fallback)
- `10313f67` (2026-08-04) — fix(strategy): qty-aware PnL, equity curve, and Strategy results tab
- `4b32864d` (2026-08-04) — fix(pyodide): align vendored wheel dist-info with pynescript filename
- `7f516350` (2026-08-04) — fix(ui): stop overlapping PRICE, indicator, and symbol badges
- `73fd9d3d` (2026-08-03) — fix: strategy events, plot colors, and price-scale labels
- `16a381d1` (2026-08-03) — fix(ui): icon-only editor tools; new tab in panel hamburger
- `a4cc7b27` (2026-08-02) — fix(chart): collapse stacked same-text Pine status labels
- `146bb090` (2026-08-02) — fix(results): drop no-fill closes; recover Source inputs; clamp future labels
- `134414dc` (2026-08-02) — fix(library): strip TradingView® Expand (N lines) chrome on import
- `3d89f4ac` (2026-08-02) — fix(library): open every dropped .pine as its own editor tab
- `abcf5eb2` (2026-08-02) — fix(library): block browser file-open on pine drag-drop
- `85e037be` (2026-08-01) — fix(dock): side-by-side panels on left/right (indicators left of editor)
- `33305aec` (2026-08-01) — fix(chart): plot indicators on stable sub-pane id
- `4fd29aea` (2026-08-01) — fix(chart): stop oscillator scripts vanishing on the price pane
- `8a11fdc5` (2026-08-01) — fix(editor): restore full height so lines are editable
- `3a77976d` (2026-08-01) — fix(replay): start with full history, not a single candle
- `f89b9cc6` (2026-08-01) — fix(editor): sync full Pine builtins corpus from pyne

Performance

- `de00cbf3` (2026-08-13) — perf(pwa): heavy live re-run throttle, offline bars fallback, SW v4 cap
- `a9375440` (2026-08-13) — perf: coalesce live ticks, resize, and strategy UI; harden paint/security
- `cb901eb4` (2026-08-10) — perf: full-app hardening — live ticks, overlays, load abort, boot split
- `a9f57958` (2026-08-10) — perf(chart): heavy history paint, conflation, and O(1) live ticks

Refactors

- `afb85816` (2026-08-26) — refactor(about): remove ethos/author, add versions and philosophy
- `6041c2d6` (2026-08-22) — refactor(ui): move plugin config section into Settings → Data tab
- `f19fe9a2` (2026-08-08) — refactor: rename remaining pine-* modules to pyne-*
- `18eedc78` (2026-08-08) — refactor: rename pine-language/lsp/builtins editor stack to pyne-*
- `096a799f` (2026-08-08) — refactor: rename pine-editor CSS, DOM, and PyneEditor component
- `b52526af` (2026-08-08) — refactor: rename pine-editor.js to pyne-editor.js
- `f0a1446b` (2026-08-02) — refactor: rename chart palette TV→VOID; document Pine naming parity

Documentation

- `0a705de9` (2026-08-27) — docs(changelog): add topbar settings and wire page fix entries
- `35dacd21` (2026-08-24) — docs: studio shell, MEXC venue, and PYNE Pro origin across guides
- `dce471b0` (2026-08-21) — docs(devops): list /datafeed/ among nginx-proxied Pro API paths
- `27b5ca13` (2026-08-21) — docs(changelog): datafeed gateway phase 4 additions
- `36900319` (2026-08-15) — docs(cors): document Pages preview origins for axis.hoox.sh
- `d3172dbb` (2026-08-11) — docs: AXIS 2.0.1 docs, CLI-first ops, Pine Script™ v6 examples
- `78c22e85` (2026-08-11) — docs(release): recursive changelog, AGENTS release workflow, v2.0.1
- `f7c30e04` (2026-08-09) — docs: prefix internal links with /axis/docs
- `3d6fe1ab` (2026-08-08) — docs(ui): clarify on-chain uses public APIs unless Worker endpoint
- `f21dcc47` (2026-08-08) — docs: update llm.txt for PyneEditor rename
- `95e181c3` (2026-08-08) — docs: document on-chain data plane in README, hubs, and worker
- `df7b77ab` (2026-08-07) — docs: add PYNE Agent (pyne-agent-worker) plugin guides

- `5d1dc0f0` (2026-08-07) — docs: add PYNE Agent (pyne-agent-worker) plugin guides
- `dc62332f` (2026-08-03) — docs: regenerate llm.txt and llms.txt agent corpora
- `5dc92693` (2026-08-01) — docs: align UI and end-user guides with last 15 commits

CI

- `0c963fc6` (2026-08-23) — ci(cli): do not fail release when access public is forbidden
- `88ca0133` (2026-08-23) — ci(cli): treat already-published 0.2.0 as success and set public
- `c35de1fe` (2026-08-23) — ci(cli): auth registry checks and always set package public
- `e07c59df` (2026-08-23) — ci(cli): force @hoox-sh/axis-cli public after npm publish
- `dd3ebf85` (2026-08-23) — ci(cli): publish with NPM_TOKEN_HOOXSH and isolate CLI tests
- `75c4543c` (2026-08-22) — ci: versioned docker tags and npm publish for axis-cli
- `3ddc4193` (2026-08-10) — ci(docker): fix image build, enhance make targets, add GHCR workflow
- `5b4bbef4` (2026-08-10) — ci: build Tauri desktop app on every main push

Tests

- `5d5844ae` (2026-08-23) — test(e2e): open Plugins via studio rail; exact Offline Lab apply

Chores

- `862631e6` (2026-08-29) — chore(release): prepare v2.1.1
- `40668dbc` (2026-08-29) — chore(release): prepare v2.1.0
- `f34c76cf` (2026-08-26) — chore(release): bump to v2.0.27 — PYNE wheel 0.4.0, hline linestyle, multi-chart fix
- `d323c8fd` (2026-08-24) — chore(release): v2.0.26
- `a01be631` (2026-08-23) — chore(release): AXIS v2.0.23
- `22966e62` (2026-08-23) — chore(release): AXIS v2.0.22
- `3e7236ef` (2026-08-21) — chore(release): v2.0.20
- `b4b3227b` (2026-08-15) — chore(release): AXIS v2.0.8
- `236d042f` (2026-08-12) — chore(release): AXIS v2.0.7
- `5f6d7a9a` (2026-08-12) — chore(release): AXIS v2.0.6
- `12cffe27` (2026-08-11) — chore: ignore local multi-agent sync notice files
- `33e11930` (2026-08-06) — chore(worker): gitignore wrangler.toml and ship example
- `9cb4dad0` (2026-08-06) — chore: vendor latest pyne 0.3.0 wheel (parity Aug 6)
- `9fba8aaf` (2026-08-03) — chore: rename SuperChart branding to AXIS including CF project id
- `0cbda10b` (2026-08-01) — chore(pyodide): vendor pynescript 0.3.0 wheel with drawing GC

Style

- `21676494` (2026-08-22) — style(ui): render inline plugin config fields via TopbarField

Merges

- `938835cb` (2026-08-08) — Merge pull request #1 from hoox-sh/feat/onchain-data-plane

Other

- `e6491985` (2026-08-08) — merge: resolve main conflicts for on-chain docs

2026-07 (79 commits)

Security

- `8c4a635c` (2026-07-24) — Harden + bugs + polish (16 changes)

Features

- `593bb8aa` (2026-07-31) — feat(inputs): OHLC source enums and cross-indicator plot sources
- `501761db` (2026-07-30) — feat(ui): make all panels border-resizable down to 1px
- `852d2fa2` (2026-07-30) — feat(ui): Scriptlogs rename; editor FloatableShell + tools
- `745e3426` (2026-07-30) — feat(ui): Pine Logs panel and Editor Profiler mode
- `a2ed35e6` (2026-07-30) — feat(chart): multi-type price series and fix settings save race
- `d0e52bb2` (2026-07-29) — feat: live watchlist WS quotes and TV-style drawing chrome
- `9487673c` (2026-07-29) — feat(ui): softer inputs and collapsible panel dock menu
- `4e9a6100` (2026-07-29) — feat(ui): density scale slider and floatable dockable panels
- `4e3f029d` (2026-07-29) — feat(ui): data window, layers, script inputs; plot visuals + strategy
- `60c7e2f9` (2026-07-29) — feat(settings): presets for VPS UI + local pyne compile
- `52cd2668` (2026-07-29) — feat(editor): prefer pyne Pro API LSP for completion and hover
- `08fdbf24` (2026-07-29) — feat(editor): Pine completion and hover from LSP builtin metadata
- `ebb4db8d` (2026-07-29) — feat(hud): ENG/RUN/MODE/PATH chips with sticky hover info
- `ba480687` (2026-07-29) — feat(settings): expose PYNE execution mode (interpret/compile/auto)
- `7dc63117` (2026-07-27) — feat(axis): connection HUD, WSS engine path, chart stability

- `7cf3d357` (2026-07-27) — feat(axis): quality stack at 95% coverage, e2e, and ops polish
- `55a5a0c2` (2026-07-26) — feat(axis): self-host Pyodide and preload on idle
- `e264c91e` (2026-07-26) — feat(axis): source-aware watchlist with interval and refresh settings
- `5de17948` (2026-07-26) — feat(axis): plugin capability badges and storage status chrome
- `b56e4f61` (2026-07-26) — feat(axis): git script storage and Pine table HUD
- `3e00e1e1` (2026-07-26) — feat(axis): script library, storage plugins, and drawing polish
- `6322cd7b` (2026-07-26) — feat(axis): render Pine drawings and drag user drawings
- `c9d2a0e8` (2026-07-26) — feat(axis): interactive chart drawing tools
- `31b7eed8` (2026-07-26) — feat(axis): jump-to-trade from Results and status-bar P&L
- `9c0e3b06` (2026-07-26) — feat(axis): strategy trade markers, equity pane, and event normalization
- `bd0a106b` (2026-07-26) — feat(axis): ship AXIS charting PWA, evaluator corpus fixes, and /run/batch
- `c2f27a02` (2026-07-25) — feat: SuperChart rewrite + Pine evaluator runtime fidelity
- `acee8d1c` (2026-07-24) — feat: slide-in editor pane with push-layout and drag resize
- `2a6a2163` (2026-07-23) — feat: Pyodide engine rewrite, watchlist sidebar, URL hash sync, demo scripts
- `40db7f47` (2026-07-23) — feat(pwa): wire multi-tab editor (CodeMirror 6 with tab persistence)
- `68da3cce` (2026-07-23) — feat(pwa): plugin manager, script library, theme switcher, symbol autocomplete, plugin examples
- `fe90be7b` (2026-07-23) — feat(pwa): settings dialog, multi-pane, volume, time presets, strategy tester
- `9be7d819` (2026-07-23) — feat(frontend): modular PWA + Cloudflare® Worker backend
- `28b57862` (2026-07-20) — feat(v6): multiline strings, export const runtime, strategy state, library import

Fixes

- `d5e487f2` (2026-07-31) — fix(profiler): compute gutter % from Σ line cost, not wall ms
- `4d5ad0a0` (2026-07-31) — fix(profiler): send profiler over WS and auto-run on enable
- `34cb26a3` (2026-07-31) — fix(editor): render LSP hover markdown and round tooltip corners
- `0abecff8` (2026-07-31) — fix(profiler): pass profiler flag to API and map profile/logs
- `33308a88` (2026-07-29) — fix(ui): stop pine drawing flicker; indigo loader accent
- `3c6dbbae` (2026-07-29) — fix(engine): REST fallback after WS timeout must not reuse dead AbortSignal
- `7b4b951e` (2026-07-29) — fix(settings): clearer endpoint probe errors for NetworkError
- `156e8c7f` (2026-07-29) — fix(hud): keep ENG plane in sync with selected engine
- `8ffa6c5c` (2026-07-29) — fix(ui): pyodide load hint, clarify MODE vs ENG chips, clip tick pulse
- `9dfeab37` (2026-07-29) — fix(pyodide): install local wheels with deps=false
- `f4dd5d0b` (2026-07-29) — fix(settings): always show execution mode for server/pyodide
- `57814bd1` (2026-07-29) — fix(settings): show execution mode for pyodide engine too
- `7d0021d2` (2026-07-29) — fix(engine): tolerate bare NaN in REST /run responses
- `b97649fd` (2026-07-29) — fix(engine): send execution mode in REST /run body
- `22178b3a` (2026-07-27) — fix(api): accept null optional symbol on WS live re-runs
- `431b3da2` (2026-07-26) — fix(axis): serve pyodide wheels from public/ and stop SPA HTML fallback
- `2d10a39a` (2026-07-26) — fix(axis): open Indicators panel from topbar toggle
- `f75867d8` (2026-07-26) — fix(axis): return multi-plot series from /run and harden trade pairing
- `49ddff4e` (2026-07-26) — fix(axis): multi-plot series with colors and bar-mode crossover
- `cf36ed70` (2026-07-26) — fix(axis): serve example plugins from /plugins in production
- `754b433c` (2026-07-26) — fix(axis): split plugin-badge utils for bun tests; drop wrangler d.ts
- `1ef06799` (2026-07-26) — fix(axis): resolve marker time from bar_time and kind from parity events
- `4252dc9e` (2026-07-26) — fix(evaluator): Console UDT multi-dispatch, bare na, and bar-loop lock
- `95479073` (2026-07-24) — fix(chart): fix chart and volume pane CSS layout
- `6642164d` (2026-07-24) — Fix chart overlap: absolute-position the loading skeleton
- `086277c3` (2026-07-24) — Fix data window positioning: use fixed coords + getBoundingClientRect
- `aaff9631` (2026-07-24) — Fix 4 layout/UX issues: loading skeleton, auto-run demos, watchlist, data window
- `4bc3132f` (2026-07-24) — fix: stdev always returns None due to AdvancedIndicators dispatch + PineSeries initial None
- `e0854e83` (2026-07-24) — fix: dict/NoneType errors in Pyodide engine (input.int, color.new, arith)
- `40568a06` (2026-07-24) — fix: builtin dispatch only passes kwargs if handler accepts them
- `7f2d7928` (2026-07-24) — fix: SMA constant values and RSI None crash in Pyodide engine
- `c35b12d6` (2026-07-23) — fix: merge kwargs into positional args for builtins in Pyodide runtime

Documentation

- `6bd9c848` (2026-07-29) — docs: inline TSDoc and module headers across the codebase
- `a3ee4a19` (2026-07-29) — docs(cors): allow any localhost/127 port on Worker; skip 0.0.0.0
- `2b09f248` (2026-07-28) — docs(license): AGPL-3.0-only (not or-later)
- `8fe2058f` (2026-07-28) — docs: cross-link HOOX / pyne / axis with hoox.sh websites

Tests

- `1e314369` (2026-07-29) — test(engine): assert REST /run body includes execution mode

Chores

- `42bf3ba9` (2026-07-29) — chore(pyodide): vendor latest pyne wheel and sync bridge
- `e6040569` (2026-07-28) — chore: bootstrap standalone axis repo from pyne frontend/
- `8dff3a71` (2026-07-28) — chore(license): relicense to AGPL-3.0-or-later under jango_blockchained
- `202a0499` (2026-07-23) — chore: switch frontend + worker to Bun + TypeScript 6

Style

- `1ee7f8ea` (2026-07-26) — style(axis): larger drawing toolbar icons
- `432dcf51` (2026-07-26) — style(axis): swap Hell Flieder for void indigo from landing pack

Other

- `514b90d1` (2026-07-24) — Add copy buttons with SVG icons to status bar + results panels

2026-01 (1 commits)

Features

- `bb224279` (2026-01-13) — feat: Google Server Setup and Closed Source Transition

2025-10 (2 commits)

Features

- `38a00b82` (2025-10-24) — feat(script): Add overlay series for line chart and improve data handling

Other

- `5613175b` (2025-10-20) — Add technical analysis built-ins and corresponding tests

---

How to update

1. Edit **[Unreleased]** (or the new version section) by hand for the story.
2. Regenerate the recursive history block:

```
python3 scripts/generate-changelog.py
```

3. Commit changelog with the release; then tag, push, publish, sync (see `AGENTS.md` § Changelog & releases).