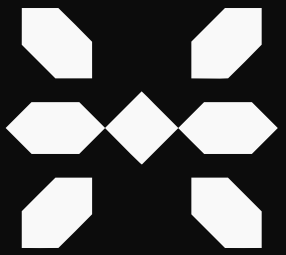
HOOX · AXIS · DOCUMENTATION



AUDITORS · QA · CONTRIBUTORS

REFERENCE MANUAL

Feature atlas, legacy shell, testing, plugin examples

Open charting PWA — own the axes, swap the engine

CONTENTS

FEATURE ATLAS

title: "Feature atlas" description: "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

```
mindmap
  root((AXIS))
    Plugins
      sources
      streams
      engines
      storage
      loader
    AXIS
      chart
      editor
      ui
      store
    Worker
      run
      keys
      scripts
      DO
    DevOps
      vite
      pages
      ci
```

Atlas tables

Plugin system

Feature	Primary paths
Contracts	`frontend/src/plugins/types.ts`
Registry	`frontend/src/plugins/registry.ts`
Bootstrap	`frontend/src/plugins/bootstrap.ts`
Active set	`frontend/src/plugins/active.ts`
Dynamic install	`frontend/src/plugins/loader.ts`
Barrel	`frontend/src/plugins/index.ts`
Manager UI	`frontend/src/ui/PluginManager.tsx` , `plugin-badges.tsx`
Docs (in-tree)	`frontend/src/plugins/README.md`

Data plugins

Feature	Primary paths
Sources	`frontend/src/sources/catalog.ts`, `upload-store.ts`
Streams	`frontend/src/streams/catalog.ts`, `multiplex.ts`
Engines	`frontend/src/engines/catalog.ts`
Load symbol	`frontend/src/data/load-symbol.ts`, `parse-bars.ts`
Indicator run	`frontend/src/indicators/runner.ts`

Storage / library

Feature	Primary paths
Catalog	`frontend/src/storage/catalog.ts`
Local IDB	`frontend/src/storage/local.ts`, `idb.ts`
Cloud	`frontend/src/storage/cloud.ts`
Git	`frontend/src/storage/git.ts`, `git-github.ts`, `git-gitlab.ts`
Service façade	`frontend/src/storage/service.ts`
Library panel	`frontend/src/ui/ScriptLibraryPanel.tsx`

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`
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/*`, `pine-drawings.ts`, `DrawingToolbar.tsx`
Editor	`src/editor/*` (CM6, diagnostics, ruler, git-sync, inline-debug)
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`
Store	`src/store/index.ts`, `types.ts`
Topbar / settings	`ui/Topbar.tsx`, `TopbarField.tsx`, `SettingsDialog.tsx`
Workspace snapshot	`src/storage/workspace-snapshot.ts`

Worker

Feature	Primary paths
Router	`frontend/worker/src/index.ts`
Run	`frontend/worker/src/runtime.ts`, `pyodide_runtime.ts`
Auth / keys	`frontend/worker/src/auth.ts`, `keys.ts`
Scripts	`frontend/worker/src/scripts.ts`, `schemas/scripts.sql`
Session DO	`frontend/worker/src/durable-objects/session.ts`
Config	`frontend/worker/wrangler.toml`, `RUNTIME.md`, `README.md`

Build / ops

Feature	Primary paths
Package scripts	`frontend/package.json` (`test:*` , `build` , `dev`)
Static server	`frontend/axis_pwa_server.py`
Public assets	`frontend/public/plugins` , `public/vendor` , `public/pyodide`
Makefile	`run-axis` (Vite), `pages-deploy` → `dist/` , `test-frontend`
CI	`.github/workflows/ci.yml` (coverage + e2e smoke), `axis-nightly.yml`
Coverage gate	`frontend/scripts/check-coverage.mjs` (scoped core ≥95%)

Examples & tests

Feature	Primary paths
Example plugins	`frontend/public/plugins/*.js` , `src/plugins/example-*.js`
Unit tests	`frontend/tests/**`
Worker tests	`frontend/worker/tests/**`
E2E	`frontend/e2e/**` , `playwright.config.ts`
Testing guide	`frontend/TESTING.md`
Legacy notes	`frontend/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
CF project	`pynescript-axis`
Health service	`pynescript-axis-worker`
Pyodide version	`0.26.2` (self-hosted path)

See also

- Plugins
- Worker
- Legacy shell

LEGACY SHELL

title: "Legacy shell" description: "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

```
flowchart LR
  LEG[Legacy main.js tree]
  CUR[Solid index.tsx]
  LEG -.->|keep for tests| BUN[bun test static bits]
  CUR --> SHIP[dist / Pages]
```

What is legacy

Path	Role
`frontend/src/main.js`	Old bootstrap (chart.js, topbar.js, …)
`frontend/style.css`	TV-blue-era tokens
`frontend/pine-editor.js`	Pre-Solid CM6 wiring
`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 project `pynescript-axis`	**Frozen** infrastructure id
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

title: "Testing reference" description: "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

```
flowchart TB
  U[Unit Bun] --> CORE[plugins storage store catalogs]
  W[Worker Bun] --> API[auth keys scripts runtime]
  S[Security] --> THREAT[URL schemes partitions If-Match]
  E[Playwright] --> SMOKE[@smoke PR]
  E --> FULL[nightly all]
  C[Coverage gate] --> SCOPED[95% scoped lines]
```

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

title: "Plugin examples" description: "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

```
flowchart LR
  URL["/plugins/example-*.js"]
  URL --> LOADER[loadPluginFromUrl]
  LOADER --> REG[registry]
  REG --> PICKER[Source/Stream/Engine pickers]
```

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-pine-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-pine-engine.js`	`tiny-pine`	engine	In-browser JS DSL: sma/ema/rsi/plot/strategy
`example-cf-do-stream.js`	`cf-do`	stream	WS to Worker `/api/stream` SessionDO

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