

Integrated UI Low-Level Architecture

Surfaces (app)

Identity, access and feature control

Explorer domain components

Trading and policy / on-chain components

Admin (Asset Routing) subcomponents

Shared libraries and cross-cutting modules

Data/stream integration helpers

High-Level References

```
---
config:
  layout: elk
  theme: neutral
---
flowchart TB
  subgraph Browser["Browser"]
    WEB["Next.js App (Landing, Explorer, Trading, Admin)"]
    WALLET["Wallet (Wagmi/RainbowKit)"]
  end

  subgraph APIs["Phase-4 backends"]
    EXPL["Explorer API (public/protected)"]
    ROUTE["Routing API (Asset Routing)"]
    POLICY["Token Policy API (resolve, issue claim)"]
    CAD["Compliance Adapter (ACE cached/fresh)"]
  end
```

end

subgraph Chains["Audit & On-chain"]

 L2["ReFIN L2 (roots, DePIN)"]

 L1["Ethereum L1 Anchor"]

end

WALLET \leftrightarrow WEB

WEB \rightarrow EXPL

WEB \rightarrow ROUTE

WEB \rightarrow POLICY

POLICY \leftrightarrow CAD

EXPL \leftrightarrow L2

EXPL \leftrightarrow L1

config:

 layout: elk

 theme: neutral

flowchart LR

Root[app]

Root \rightarrow Landing[app/landing]

Root \rightarrow Auth[app/auth]

Root \rightarrow Explorer[app/explorer]

Explorer \rightarrow Resolve["app/explorer/[hash]"]

Explorer \rightarrow Search[app/explorer/search]

Explorer \rightarrow Timeline[app/explorer/timeline]

Root \rightarrow Admin[app/admin]

Admin \rightarrow Routes["app/admin/routes/[symbol]"]

Admin (Asset Routing) subcomponents

Purpose

Provide a secure, RBAC-gated admin surface to **list, search, preview, create, edit, and bulk import** routing decisions for assets. Ensures **optimistic concurrency** via `config_hash`, **cache invalidation** on writes, **audit events** on change, and clear **broker fallback** semantics for unknown symbols.

Ownership & boundaries

- **Owned in UI:** Routes table & filters, “Check route” decision preview drawer, Create/Edit form, 409 **conflict diff** modal, Bulk CSV import, Ops counters (p95, cache-hit, invalidations, recent updates).
- **Backends:** Asset Routing Read/Admin APIs, Redis read-through cache, Mongo `AssetRouting` store, audit emission on updates.

External interfaces

- `GET /v1/route/{symbol}` → **RouteDecision** `{ route, driver_hint, params, policy_flags.need_fresh_ace, config_hash }`. `404` means **default broker**; UI shows **derived=true** banner. Short-TTL cache.
- `PUT /v1/route/{symbol}` (admin RBAC) → upsert with **optimistic concurrency** (`if-match: config_hash`) and **cache invalidation**; emits `audit.evt asset_route_update`.
- `POST /v1/route/bulk` (admin RBAC) → CSV import, per-row validation, failures CSV.

RBAC and auth: Admin endpoints require admin role claim; UI attaches SIWE session cookies and `x-correlation-id` header to all requests.

Data model (reference)

`AssetRouting` record fields: `symbol (pk)`, optional `venue`, `asset_class`, `chainId`, `route` (`"broker"` | `"onchain"`), `driver_hint` (`"snaptrade"` | `"onchain"`), `enabled`, `slippage_bps?`, `mev_protect?`,

`max_gas_gwei?` , `token_policy_required?` , `token_policy_ref?` , `updated_by` , `updated_at` , `config_hash` .

Validate symbol format, chainId/venue consistency, numeric bounds.

Component map

config:

layout: elk

theme: neutral

flowchart TB

admin["Admin UI"] → list["Routes Table and Filters"]

admin → preview["Decision Preview Drawer"]

admin → form["Create and Edit Form"]

admin → bulk["Bulk CSV Import"]

admin → diff["409 Conflict Diff Modal"]

admin → counters["Ops Counters p95 and Cache Hit Ratio"]

- **Routes Table and Filters:** `{symbol, route, driver_hint, enabled, updated_at, config_hash}` with filters `route, asset_class, venue` .
- **Decision Preview Drawer:** `GET /v1/route/{symbol}` ; shows `{route, driver_hint, params, policy_flags, config_hash}` and **derived=true** banner for fallback.
- **Create and Edit Form:** all route fields; derived defaults and validation hints; sends `config_hash` .
- **Bulk CSV Import:** drag-drop, per-row statuses, export failures.
- **409 Conflict Diff Modal:** server vs local with `config_hash` .

UI flows

A) Optimistic concurrency on save

sequenceDiagram

participant UI as "Admin UI"

```

participant API as "Routing Admin API"
UI->>API: "GET /v1/route/{symbol}"
API->>UI: "config with config_hash=abc"
UI->>API: "PUT /v1/route/{symbol} with config_hash=abc"
alt "Write conflict"
  API->>UI: "409 with latest config_hash=def and diff"
  UI->>UI: "Show conflict diff modal then retry with config_hash=def"
else "Success"
  API->>UI: "200 OK and cache invalidated and audit.evt emitted"
end

```

Covers 409 diff, cache invalidation and audit emission.

B) Bulk CSV import

```

sequenceDiagram
  participant UI as "Admin UI"
  participant API as "Routing Admin API"
  UI->>API: "POST /v1/route/bulk with text/csv and x-correlation-id"
  API->>UI: "Per-row validation statuses and failures list"
  UI->>UI: "Render success rows and provide failures CSV download"

```

Implements per-row validation and export of failures.

C) Decision preview and fallback semantics

```

sequenceDiagram
  participant UI as "Admin UI"
  participant API as "Routing Read API"
  UI->>API: "GET /v1/route/{symbol}"
  alt "Configured symbol"
    API->>UI: "RouteDecision with config_hash and params and policy_flags"
  else "Unknown symbol"
    API->>UI: "404 defaults broker upstream"
  end

```

```
UI→>UI: "Show derived true banner and Defaulting to broker route chip"  
end
```

404 is not an error for users; it surfaces broker default and derived flag.

States, copy, and error taxonomy

- **Loading**: skeleton table and form. **Empty**: "No routes yet. Create your first route." **Partial**: section warning banners. **401/403**: lock icon and sign-in CTA. **404**: "Symbol not configured. Falling back to broker routing." **409**: "This route changed while you were editing. Review updates and retry." **429**: "Too many requests. Please wait and retry." **5xx**: "Service unavailable. We're retrying."

Security & privacy

- **Auth**: SIWE session; Admin endpoints require **RBAC**; attach `x-correlation-id` headers in middleware.
- **Headers**: HSTS, X-Frame-Options=DENY, CSP with nonce, Referrer-Policy, Permissions-Policy set by app config.

Observability

- UI shows small counters for **p95 lookup latency**, **cache hit ratio**, **invalidations**, and an **updated recently** row badge. Backend emits `audit.evt` `asset_route_update` on writes.

Performance & resilience

- Read path uses **Redis** read-through cache with **short TTL**; if cache misses, fallback to Mongo; unknown symbol returns **default broker** with `derived=true`. Writes are single upserts guarded by `config_hash`.

Feature flags

- Admin surface is flaggable per environment; with flag off, navigation hides Admin routes. Policy flags hint integration remains visible on preview but write

actions remain RBAC-gated.

Accessibility & i18n

- Keyboard-first, visible focus states; ARIA labels on table headers, tooltips, and diff modal controls; live-region toasts on save/409 events; locale-aware numbers and ISO-8601 timestamps. (Global a11y and copy taxonomy apply.)

Implementation notes

- **List & Filters:** Query local state and debounce server queries; include filter chips for `route`, `asset_class`, `venue`. Decision preview drawer shows `{route, driver_hint, params, policy_flags, config_hash}`.
- **Create/Edit:** Front-load validation hints (symbol, bounds); send `config_hash` from last read; on `409`, render **line-by-line diff** with server values and allow **accept/retry**.
- **Bulk Import:** Accept `.csv` with header row; post as `text/csv` with `x-correlation-id`; render per-row result table and expose "Download failures" action.
- **Fallback semantics:** Unknown symbol is **not an error**; show "Defaulting to broker route" chip and mark preview as **derived**.

Testing

Unit

- Table row rendering and filters; preview drawer mapping; validation hints.
- Save flow with `config_hash`: success and **409 diff** branches.
- Bulk import parser and failures CSV exporter.

Integration

- `GET /v1/route/{symbol}`: configured vs 404 fallback paths.
- `PUT /v1/route/{symbol}`: cache invalidation, audit emission observed in test hooks.
- `POST /v1/route/bulk`: mixed success/failure dataset renders correct statuses.

E2E

- RBAC enforced on write routes.
- 409 conflict modal resolves and resubmits successfully.
- Counters display and update after import and edits; unknown symbol shows broker fallback chip.

Exit criteria

- Admin can **list, search, preview, create/edit, and bulk import** routes under RBAC;
- Writes use **optimistic concurrency** with 409 conflict resolution;
- **Cache invalidation** and **audit.evt emission** on change;
- Unknown symbols surface **broker fallback** with **derived** banner;
- Ops counters visible and meaningful.

Data/stream integration helpers

Purpose

Provide a **unified, typed client layer** for consuming **live and near-real-time data** in the UI. The helpers normalize transport concerns (WebSocket, SSE, long-poll), apply **backoff and backpressure**, expose **frame reducers** for high-rate topics, propagate `x-correlation-id`, and present **idempotent, cache-friendly** selectors to UI components. They back the Trading views (market ticks and lifecycle events) and Explorer/Timeline reads (public GETs).

Ownership and boundaries

- **Owned in UI:** `packages/api-clients` stream adapters, `apps/web/src/lib/streams/*` hooks (`useWebSocketBase` , `useEventCoalescer` , `useMarketStream` , `useOrdersLifecycle`), selectors, and schema guards.
- **Relies on backends:** Market provider WS (Polygon demo); HTTP Explorer endpoints (`resolve` , `search` , `timeline` , `preimage`); lifecycle events surfaced by the platform from `orders.evt` (UI consumes via backend surfaces).
- **Out of scope:** Proof construction, anchoring jobs, and event production; those are backend responsibilities.

External interfaces

Market data (demo lineage)

- **WebSocket** `wss://socket.polygon.io/stocks` with auth; subscribe to `T.*` and `Q.*`; expose `marketData` , `connectionStatus` , `error` , `connect()` , `disconnect()` . Exponential backoff and terminal error after cap.

Explorer reads (public and protected)

- `GET /v1/resolve/{hash}` , `GET /v1/batch/{id}` , `GET /v1/root/{id}` , `POST /v1/verify/inclusion` , `GET /v1/search` , `GET /v1/anchor/{bundle_id}` ; **protected:** `GET /v1/preimage/{hash}` , `GET /v1/timeline?correlationId=...` . All echo `x-correlation-id` .

Trading lifecycle context

- Platform emits `orders.evt { clientId, status submitted|mined|reverted|failed_retry|rejected, ts, correlationId, ... }`; brokered vs on-chain lifecycles share schema; backend reconciles and surfaces to UI.

Error and retry policy (client)

- Retry idempotent GETs only** (429 with jitter). **Do not retry writes** in browser. Standard copy for 4xx/5xx.

Component map

```

---
config:
  layout: elk
  theme: neutral
---
flowchart TB
  subgraph "Stream Helpers"
    WSB["useWebSocketBase"]
    COL["useEventCoalescer"]
    MKT["useMarketStream"]
    ORD["useOrdersLifecycle"]
    EXP["useExplorerClient"]
  end

  Provider["Polygon Provider WS"] --> WSB
  WSB --> COL --> MKT
  Backend["Backend Explorer and Lifecycle Surfaces"] --> EXP
  Backend --> ORD

  UI["UI Consumers"] --> MKT
  UI --> ORD
  UI --> EXP

```

- `useWebSocketBase` abstracts connect, auth, subscribe, reconnect with backoff.

- `useEventCoalescer` batches high-rate ticks into animation-frame or timed batches to reduce render thrash. (Productionization item 1).
- `useMarketStream` composes base WS and coalescer; falls back to a **deterministic simulator** when live feed is unavailable.
- `useOrdersLifecycle` reduces lifecycle updates (from backend-surfaced `orders.evt`) into stable UI chips and feeds.
- `useExplorerClient` thin GET client honoring `x-correlation-id`, error taxonomy, and retry rules.

Data and state model

Market frames

- Symbol-indexed buffers for Trades `T.*` and Quotes `Q.*`; selector derives `MarketData[]`, `positions`, `portfolioMetrics`, connection badges.

Lifecycle reducer

- Input: normalized lifecycle items from backend derived from `orders.evt`.
- Output: `{ byId: Map<clientId, { status, ts, reason?, txHash? }>, feed: TradeFeedItem[] }`. Mirrors statuses `submitted|mined|reverted|failed_retry|rejected`.

Explorer reads

- Idempotent GETs with SWR-style cache; deep links carry and echo `x-correlation-id`.

Core flows

A) WebSocket lifecycle and fallback

```
sequenceDiagram
    participant UI as "Consumer"
    participant WS as "useWebSocketBase"
    participant PV as "Provider WS"
    UI->>WS: "connect and subscribe"
    WS->>PV: "auth then subscribe T.* and Q.*"
```

```

PV→>WS: "tick stream"
WS→>UI: "batched frames via useEventCoalescer"
alt "auth or connection failure"
  WS→>UI: "connectionStatus error"
  UI→>UI: "switch to simulation mode"
end

```

Auth, subscribe, message routing, fallback and timers per demo lineage.

B) Explorer resolve and timeline (idempotent GETs)

```

sequenceDiagram
  participant UI as "Explorer UI"
  participant EC as "useExplorerClient"
  participant API as "Explorer API"
  UI→>EC: "resolve or search"
  EC→>API: "GET with x-correlation-id"
  API→>EC: "ExplorerRecord or 404 or Pending"
  EC→>UI: "normalized record and states"
  UI→>EC: "timeline or preimage under SIWE"
  EC→>API: "GET /v1/timeline or /v1/preimage"
  API→>EC: "events or redacted artifact"

```

Contracts and protected routes per Phase-3 Explorer surface.

C) Orders lifecycle projection

```

sequenceDiagram
  participant UI as "Trading UI"
  participant OL as "useOrdersLifecycle"
  participant BE as "Backend Surface"
  BE→>OL: "lifecycle items from orders.evt"

```

OL → >UI: "status chips and feed rows"

Lifecycle semantics and statuses mirror `orders.evt` payloads.

Backpressure, batching, and cleanup

- **Coalescing strategy:** accumulate ticks and flush on `requestAnimationFrame` or a 50–100ms timer under load; trade feed reduces to append-only items. (Gap noted for prod; implemented in helpers.)
- **Symbol lifecycle:** explicit subscribe/unsubscribe with debounce when watchlist changes.
- **Intervals registry:** central registry cancels intervals/timeouts on unmount to avoid leaks.

Error, retry, and state taxonomy

- **Market WS:** invalid/demo keys → terminal `error` status; reconnection backoff with cap. UI flips to simulator.
- **Explorer GETs:** 404, Pending Anchor, 429 (retry affordance), 5xx copy per taxonomy; GETs may retry with jitter.
- **Writes:** never retried by browser; handled elsewhere (admin upserts, claims).

Security and privacy

- **No client secrets** persisted; WS auth messages only; SIWE cookies used for protected Explorer reads.
- **Preimages** are served **redacted from GCS**; UI streams artifacts; Mongo stores pointers only.

Observability

- **Client telemetry:** connection status badges, latency metrics exposed by hooks; RUM dashboards include FE error rate and p95 resolve/search.
- **Correlation:** propagate `x-correlation-id` on all requests and deep links.

Performance guardrails

- Avoid heavy client proofs; rely on Explorer endpoints and server memoization (ExplorerCache). Use SWR for GET APIs; route-split screens.

Testing

Unit

- WS base: auth, subscribe, backoff, cleanup; coalescer flush logic; schema guards for ticks and lifecycle items.
- Explorer client: error mapping for 404, Pending, 429, 5xx; correlation header propagation.

Integration

- Market fallback: invalid key triggers simulator; UI reflects status and renders simulated data.
- Timeline and preimage under SIWE: protected GETs succeed when authenticated.

E2E

- Explorer resolve/search/verify flows; timeline deep links carry correlation ID; trading feed reflects lifecycle updates surfaced from `orders.evt`.

Exit criteria

- `useMarketStream` exposes stable frames with **coalesced ticks** and **simulator fallback**;
- `useOrdersLifecycle` renders correct statuses from backend-surfaced `orders.evt`;
- Explorer client honors **correlation IDs**, **retry policy**, and **state taxonomy**;
- Hooks clean up intervals and subscriptions; observability and a11y patterns are consistent across consumers.

Appendix: Reference contracts

- `orders.evt` schema and identical lifecycle for brokered and on-chain; Explorer public/protected endpoints; Preimage and Timeline privacy posture.

Explorer domain components

Purpose

Provide a live, verifiable view of the audit trail. Users can resolve any identifier, inspect Merkle inclusion proofs, see L2 and L1 receipts, and—under SIWE—access redacted preimages and correlation timelines.

Ownership and boundaries

- **Owned in UI:** Explorer surfaces and domain components (Search, Resolve renderer, Inclusion Verify, Preimage Viewer, Timeline, Deep-linking).
- **Contracts:** Public endpoints `resolve`, `batch`, `root`, `search`, `verify/inclusion`, `anchor`; protected `preimage`, `timeline`. SIWE + Redis rate limits.
- **Out of scope (UI):** Proof construction logic, chain client receipt fetching, anchoring jobs—these are backend responsibilities. UI consumes their results.

External interfaces

Public REST

- `GET /v1/resolve/{hash}` • `GET /v1/batch/{batch_id}` • `GET /v1/root/{root}` • `POST /v1/verify/inclusion` • `GET /v1/search?...` • `GET /v1/anchor/{bundle_id}`.

Protected REST (SIWE)

- `GET /v1/preimage/{hash}` (redacted GCS artifact) • `GET /v1/timeline?correlationId=...`.

Inputs accepted by Explorer UI

- `hash`, `batch_id`, `root`, `correlationId`, `clientOrderId`, `intent_id`.

Response baseline

- Normalized `ExplorerRecord` joining Leaf → Interval → ReFIN L2 → L1 Anchor, carrying `correlationId`.

Component map

```

---
config:
  layout: elk
  theme: neutral
---
flowchart TB
  ui["Explorer UI"] → searchbar["Global Search Input"]
  ui → record["ExplorerRecord Renderer"]
  ui → verify["Inclusion Verify Action"]
  ui → preimage["Preimage Viewer (Protected)"]
  ui → timeline["Timeline View (Protected)"]
  record → chips["Status Chips L2 and L1"]
  searchbar → api["Explorer API Client Hooks"]

```

- **Global Search Input:** Parses any supported identifier and dispatches the appropriate call.
- **ExplorerRecord Renderer:** Renders Leaf, Interval, L2 Receipt, L1 Anchor, and a preimage ref. Copy buttons on domain keys.
- **Inclusion Verify Action:** Posts proof payload and shows pass or fail.
- **Preimage Viewer:** Redacted JSON preview with gated download; never store PII in Mongo.
- **Timeline View:** Correlated events (risk → execution → audit) by `correlationId`.

Data contracts

ExplorerRecord (normalized)

- `leaf` : { `leaf_hash`, `emitter`, `emitted_at`, `seq_no`, `preimage_ref?` }
- `interval` : { `batch_id`, `leaf_index`, `root`, `leaf_count?` }
- `l2_receipt` : { `l2_tx_hash`, `block_no`, `confirmed_at?` }
- `l1_anchor` : { `l1_tx_hash?`, `messageId?`, `status`: "anchored"|"pending" }
- `correlationId` : string

- `links` : { `I2_explorer_url?`, `I1_explorer_url?` }

Rendered into key/value cards; pending L1 shows a "Pending" badge.

InclusionVerify request/response

- **Req:** { `batch_id`, `leaf`, `proof?`, `index?` }
- **Res:** { `ok`: boolean, `root`, `index`, `references` }

(Proof building done server-side; client posts leaf and receives boolean with references.)

UI flows

A) Resolve or Search then Verify

```
sequenceDiagram
    participant U as "User"
    participant X as "Explorer UI"
    participant E as "Explorer API"
    U->>X: "Enter hash or correlationId"
    alt "Hash path"
        X->>E: "GET /v1/resolve/{hash} with x-correlation-id"
        E->>X: "ExplorerRecord with leaf, interval, L2, L1, preimage_ref"
    else "Search path"
        X->>E: "GET /v1/search?correlationId|clientOrderId"
        E->>X: "Record list and selected record"
    end
    X->>E: "POST /v1/verify/inclusion"
    E->>X: "boolean result with references"
```

Accepts `hash` | `batch_id` | `root` | `correlationId` | `clientOrderId` . Verify shows a modal or toast. 404 and 429 have standard copy.

B) Preimage and Timeline (protected)

```
sequenceDiagram
    participant U as "Authenticated User"
    participant X as "Explorer UI"
    participant E as "Explorer API"
    U->>X: "Open Preimage or Timeline"
    X->>E: "GET /v1/preimage/{hash} or GET /v1/timeline?correlationId=..."
    E->>X: "Redacted JSON stream or correlated events"
    X->>U: "Preview and download under auth or timeline list with deep links"
```

Preimage requires SIWE; artifacts are stored in GCS and streamed; Mongo holds pointers only. Timeline groups risk → execution → audit.

States, copy and chips

- **Loading, Empty, Partial, 4xx, 5xx, 429** have standardized copy. "Anchor pending" appears when L2 receipt exists but L1 is not yet anchored.
- **Status chips:** `root_submitted` and `anchored` per receipts; tooltips describe source.

Deep linking and correlation

- Every request carries and echoes `x-correlation-id`. Explorer exposes deep links for resolve and timeline, and prints `correlationId` near results.

Security and privacy

- **Auth:** Protected routes require SIWE JWT; rate limits enforced in Redis.
- **Preimage posture:** Serve **redacted** artifacts from GCS only; Mongo stores pointers; no PII at rest in Mongo.

Performance and caching

- Use Explorer endpoints (do not rebuild heavy proofs client-side). Memoized receipts and proofs on server; UI uses SWR for idempotent GETs and progressive rendering.

Accessibility and i18n

- Keyboard-first flows, focus rings, ARIA labels on chips and copy buttons, live regions for toasts. Messages localized; timestamps ISO-8601 with UTC labels.

Observability

- RUM and error rate by endpoint, p95 latency for resolve and search, anchor fetch error counters. Network and chain badges in UI.

Error and retry model

- Respect service errors; **retry idempotent reads with jitter; never retry writes**; show 429 retry affordance. Clean copy for 404 and pending-anchor partials.

Testing

Unit and integration

- Resolve, batch, root, inclusion verify, search by correlationId; contract mocks for timeouts and 4xx/5xx.

E2E

- Public resolve renders ExplorerRecord and verify succeeds; protected preimage/timeline enforce SIWE and render for authenticated users.

Exit criteria

- Search and Resolve render a complete ExplorerRecord with deep links and standard states; Inclusion Verify works; protected Preimage and Timeline function behind SIWE and rate limits.

Appendix: Backend reference

- Deterministic joins and proof building are performed server-side; receipts are attached and cached; anchors may be “pending” until L1 confirmation. UI surfaces those results exactly.

Identity, access and feature control

Purpose

Provide a unified layer for **authentication (SIWE)**, **authorization (RBAC & route guards)**, and **feature gating (flags)** across all UI surfaces (Explorer, Trading, Admin). Ensures cookie-based sessions with CSRF protection, consistent error/UX copy, deep-link safety, and fail-safe flags.

Ownership & boundaries

- **Owned in UI:** Auth provider/context, route guards, protected link wrappers, feature-flag service, error/UX mapper.
- **Relies on backends:** SIWE endpoints (`/siwe/nonce` , `/siwe/verify` , `/auth/refresh` , `/auth/logout`), protected Explorer endpoints (preimage/timeline), Admin APIs (RBAC), Token Policy (policy/claim).

External interfaces

AuthN/AuthZ (SIWE)

- **Public:**
`GET /siwe/nonce` (domain/origin/URI/chainId bound), `POST /siwe/verify` (sets **HTTP-only** cookies), `POST /auth/refresh` (rotation), `POST /auth/logout` (revoke).
- **Operating mode:** Web uses **secure, HTTP-only cookies**; CSRF token on state-changing requests.
- **RBAC claim:** Admin routes require **admin role** in JWT/session.

Protected Explorer routes

- `GET /v1/preimage/{hash}` (redacted stream from GCS), `GET /v1/timeline?correlationId=...` (rate-limited; SIWE required).

Feature flags

- Flags gate **Admin, Timeline, Preimage, Policy/Claim** UI; default **off** per environment, with graceful placeholders.

Token Policy hooks (for on-chain features)

- Read policy (`GET /policy/{chainId}/{token}`) and (flagged) `POST /claim/issue` (idempotent, `423 POLICY_BLOCK` on deny/review). UI only **surfaces** requirement & result.

Data & state model (UI)

- AuthContext state:** `{status, account_id, wallet_id, roles[], kyc_status}` ; derives from cookies/verify/refresh. Session badge & guarded navigation update on changes.
- Cookies:** Access/refresh (rotating) in **HTTP-only** cookies; names/domains aligned with backend; **SameSite** settings; CSRF header.
- Correlation:** Every request includes `x-correlation-id` ; propagated into deep links.

Core flows (sequence)

A) SIWE login → refresh → logout

```
sequenceDiagram
    participant U as User
    participant W as Web App
    participant A as Auth API
    U->>W: Connect wallet
    W->>A: GET /siwe/nonce (bound to domain/origin/uri/chainId)
    A->>W: nonce
    W->>U: Sign EIP-4361
    U->>W: signature
    W->>A: POST /siwe/verify
    A->>W: Set HTTP-only cookies
    W->>U: Session badge
    W->>A: POST /auth/refresh (rotation)
```

A→>W: New cookies
U→>W: Logout
W→>A: POST /auth/logout (revoke)
A→>W: Clear cookies

Flow and cookie/CSRF requirements per Phase-1/2 specs.

B) Route guard & RBAC

- On navigation to **/admin** or protected Explorer routes, guard checks `AuthContext.status === "auth"` and `roles.includes("admin")` (for Admin). Failing guards land on auth prompt with standardized copy.

C) Feature-flag gating

flowchart TD

User → Web

Web → Flags{Flag on?}

Flags -- yes → Feature[Render feature]

Flags -- no → Placeholder[Graceful placeholder]

Flags default **false**, opt-in per env; Policy/Claim remains behind a flag until Phase-4 backends are green.

Error model & UX copy

Map backend codes to concise user copy (toast/page):

- `NONCE_INVALID`, `SIGNATURE_INVALID`, `POLICY_VIOLATION`, `CHAIN_DENIED`, `ACCOUNT_BLOCKED`, `REFRESH_REVOKED`. Include `correlationId` in logs.
- Protected Explorer without session → **401/403**; clean **404/Pending anchor** messages for Explorer results.
- 429**: retry affordance for idempotent GETs; **never** retry writes from browser.

Security controls (UI edge)

- **CSP/HSTS/SRI**, dependency pinning, SBOM; OAuth redirect allow-lists align with SIWE validators; **no client secrets** in bundles.
- **Fetch Metadata** (Sec-Fetch-*) policy; **Trusted Types**; CSP **report-only** in preview cycles.
- **Cookie hardening**: exact domain binding from nonce→verify; **no-store** on protected GETs; `Vary: Authorization, Cookie`.
- **Revocation & multi-tab sync**: BroadcastChannel to fan-out logout/revoke to all tabs.

Feature policy & token policy interplay

- When routing indicates **on-chain** and policy requires **fresh ACE**, the (flagged) claim UI calls `POST /claim/issue`; UI blocks on `423 POLICY_BLOCK` and shows reasons.
- Claim issuance/verifications are **backend-owned**; UI surfaces states and TTL only.

Observability

- Thread **x-correlation-id** through requests and deep links; dashboards for auth 401 churn and 429 spikes.

Performance notes

- SIWE hot paths are **Redis-backed** on server; UI aims for minimal round-trips and caches idempotent reads with SWR where safe.

Accessibility & i18n

- WCAG 2.1 AA: keyboard-first, visible focus, ARIA for auth/guard/flag affordances; ICU messages; locale/ISO-8601 timestamps.

Testing

Unit

- Auth: nonce lifecycle, error mapping, refresh rotation, logout/revoke.

- Flags: render on/off, placeholder fallback, deny-by-default.

Integration/E2E

- Guarded routes (Admin, Preimage/Timeline) enforce SIWE; Explorer 404/"Pending" copy; 429 affordance.
- Admin RBAC and 409 conflict flows covered elsewhere; included here for access checks.

Exit criteria

- SIWE login/refresh/logout/revoke work with **HTTP-only cookies** and CSRF; **RBAC** enforced on Admin; protected Explorer routes require session.
- Feature-flag gates for **Policy/Claim, Timeline, Preimage, Admin** are default-off with graceful placeholders.
- Error taxonomy and retry/backoff behavior presented consistently; deep links carry **correlationId**.

Shared libraries and cross-cutting modules

Purpose

Provide the **reusable substrate** for all UI surfaces: a typed **API client layer**, a **design system** with accessible primitives, a centralized **state and error taxonomy**, **feature flags**, **observability utilities** (web-vitals, JS error/breadcrumbs, correlation IDs), **security headers/middleware**, and **i18n/a11y helpers**. These modules standardize behavior, copy, and performance across Explorer, Trading, and Admin.

Ownership & boundaries

- **packages/ui** — Design system primitives, tokens, composition helpers; a11y defaults.
- **packages/api-clients** — Generated OpenAPI clients (Explorer, Routing, Auth, Token Policy), thin fetch wrapper, error mapping, retry policy. Specs are **source of truth**.
- **packages/config** — Shared config (headers, security, flags), `x-correlation-id` middleware contract, copy taxonomy.
- **apps/web/src/lib/** — App-specific adapters and hooks that consume the above.

External interfaces

- **Contracts:** OpenAPI v3 HTTP (`/v1/...`) and named AsyncAPI topics. RouteDecision (routing), TokenPolicy (policy/claims) are authoritative schemas.
- **Routing:** `GET /route/{symbol}` → `RouteDecision` with `config_hash`, `policy_flags.need_fresh_ace`; 404 defaults to **broker**. Short-TTL cache noted.

- **Token Policy:** `GET /policy/{chainId}/{token}` and `POST /claim/issue` (EIP-712 claim). UI may gate by **feature flag**.
- **Security & headers:** CSP nonce, HSTS, SRI; middleware assigns and forwards `x-correlation-id`.

Module catalog

1) Design system (`packages/ui`)

Scope: Accessible, themeable primitives (Button, Badge, Card, Tabs, Switch, Progress, Tooltip), `cn()` helper, `PersistentPopoutTab`, icon shim.

Responsibilities: keyboard-first focus, ARIA via Radix, dark-mode baseline, pure presentational components.

Testing: snapshot per variant, interaction tests for Tabs/Switch/Tooltip/Progress.

```

---
config:
  layout: elk
  theme: neutral
---
graph LR
  subgraph "Design System"
    B["Button"]:::p
    G["Badge"]:::p
    C["Card"]:::p
    T["Tabs"]:::p
    S["Switch"]:::p
    P["Progress"]:::p
    TT["Tooltip"]:::p
    PP["PersistentPopoutTab"]:::c
  end
  U["cn utility"] → B
  U → G
  U → C
  U → T

```

$U \rightarrow S$

$U \rightarrow P$

$U \rightarrow TT$

```
classDef p fill:#eee,stroke:#999,color:#111;
```

```
classDef c fill:#f7f7f7,stroke:#aaa,color:#111;
```

2) API client layer (`packages/api-clients`)

Generation: Clients generated from **OpenAPI**; freeze contracts, tag SDKs, and block breaking changes behind previews.

Fetch policy:

- **Idempotent GETs** (e.g., resolve, search, route, policy) may retry with jitter on 429;
- **Writes** (e.g., admin PUT, claim issue) are **never retried** by the browser; claim uses **Idempotency-Key**.

Error mapping: map `NONCE_INVALID` , `SIGNATURE_INVALID` , policy blocks, 404 default broker, 409 conflict to standardized copy.

Examples present (demo lineage): typed ACE/VAR/Anchor clients exhibiting **fail-closed** behavior.

3) State and copy taxonomy

Single table of **Loading, Empty, Partial, 400, 401/403, 404, 409, 429, 5xx** copy and skeleton patterns; chips for `root_submitted` and `anchored` , and `orders.evt` statuses.

4) Observability utilities

Client telemetry: Web-Vitals, JS errors, action breadcrumbs (PII stripped).

Dashboards track FE error rate, API error rate by endpoint, p95 resolve/search, anchor fetch errors.

Correlation: generate a session/page-scoped **correlationId**, forward `x-correlation-id` on every request, propagate in deep links.

```

---
config:
  layout: elk
  theme: neutral
---
sequenceDiagram
  participant U as "User"
  participant W as "Web App"
  participant M as "Metrics"
  participant S as "Error Tracker"
  participant API as "Backend"
  U->>W: "Action"
  W->>M: "Web Vitals with correlationId"
  W->>S: "Breadcrumbs or error with correlationId"
  W->>API: "Request with x-correlation-id"
  API->>W: "Response with x-correlation-id"

```

5) Security headers and middleware

Middleware sets `x-correlation-id`; responses attach **HSTS, CSP (nonce), X-Frame-Options, Referrer-Policy, Permissions-Policy**; SRI enforced; **no client secrets**.

SIWE: cookie-only auth between Web and Auth; CSRF header on writes; refresh rotation; revoke path.

6) Feature flags

Flags gate **Admin, Timeline, Preimage, Policy/Claim** with deny-by-default rollout and graceful placeholders.

7) i18n and accessibility helpers

A11y: keyboard-first flows, focus rings, ARIA labels for chips, copy, and conflict modals, live regions for toasts. **i18n**: message keys, locale numbers/dates, **ISO-8601** timestamps with UTC labels, pseudo-locale and RTL safe layouts.

Data contracts and types

- **RouteDecision** and **TokenPolicy** shapes come from OpenAPI; UI never invents fields—consumes generated types.
- Type modules for preview results and trading models exist in the demo lineage; production stance is to **generate** from OpenAPI and optionally validate at runtime (Zod).

Directory and routing references

Target repo tree shows `packages/ui`, `packages/api-clients`, `packages/config`, and app routes; CODEOWNERS per path.

Error and retry model

- **429**: backoff & retry for **idempotent GETs** only;
- **Writes**: admin PUT and claim issue are **not retried** in browser; present copy and let user act.
- Canonical copy for 400/401-403/404/409/5xx comes from taxonomy.

Performance guardrails

LCP \leq 2.5s, CLS \leq 0.1, TBT \leq 200ms; route-split heavy screens; avoid heavy client proofs—use Explorer endpoints; SWR for GET APIs; asset formats AVIF/WebP with immutable caching and SWR for API reads.

Testing utilities and coverage

- **Contract tests**: SDK generation jobs and smoke contract tests ensure clients compile against frozen OpenAPI.
- **UX tests**: axe scans, keyboard flows, Percy snapshots.
- **Experience E2E**: Auth (nonce/verify/refresh/logout), Explorer (resolve/search/verify/timeline), Admin (edit with 409).

Dependency overview

```

---
config:
  layout: elk
  theme: neutral
---
flowchart TB
  subgraph "Shared Libraries"
    Ulkit["packages/ui Design System"]
    SDKs["packages/api-clients OpenAPI SDKs"]
    CFG["packages/config Security and Flags"]
    OBS["Telemetry Correlation and Web Vitals"]
  end
  Explorer["Explorer App"] → Ulkit
  Explorer → SDKs
  Explorer → CFG
  Explorer → OBS
  Trading["Trading App"] → Ulkit
  Trading → SDKs
  Trading → CFG
  Trading → OBS
  Admin["Admin App"] → Ulkit
  Admin → SDKs
  Admin → CFG
  Admin → OBS

```

Implementation notes

- **Freeze contracts** before client changes; tag SDKs; preview behind flags.
- **Headers:** set `x-correlation-id` in middleware; pass through to APIs and vendors.
- **Security:** strict CSP with per-response nonce; SRI; dependency pinning; SBOM in CI; zero critical CVEs as gate.

Exit criteria

- **Design system** adopted by all apps; a11y and variants tested.
- **SDKs** generated from current OpenAPI; retry policy and error mapping aligned to taxonomy.
- **Flags** control Admin, Timeline, Preimage, Policy flows; deny-by-default.
- **Observability** emits web-vitals, JS errors, and breadcrumbs with correlation IDs; dashboards populated.
- **Security** headers and CSP enforced; SIWE cookie posture and CSRF on writes; no secrets in bundles.

Surfaces (app)

Purpose

Implements the product's user-facing applications—**Landing/Auth**, **Explorer (public & protected)**, **Trading App**, and **Admin (Asset Routing)**—as a single Next.js App Router surface with shared design system, API clients, error/state taxonomy, and feature flags. These apps consume the Explorer API, Asset Routing, Token Policy, and (indirectly) Compliance Adapter via the backend, enforcing SIWE auth and RBAC where required.

Ownership & Boundaries

- **Apps:** `apps/web/app/{landing,auth,explorer,admin}` map 1:1 to routes. Protected areas: `/admin`, `/explorer/timeline`, `preimage` actions.
- **Shared UI:** `packages/ui` (primitives, tables, diff modal, badges, code blocks).
- **API layer:** generated clients + hooks for **Explorer**, **Asset Routing**, **Token Policy**.

Routing Map

```
---
config:
  layout: elk
  theme: neutral
---
flowchart LR
  Root["app"] --> Landing["app/landing"]
  Root --> Auth["app/auth"]
  Root --> Explorer["app/explorer"]
  Explorer --> Resolve["app/explorer/[hash]"]
  Explorer --> Search["app/explorer/search"]
  Explorer --> Timeline["app/explorer/timeline (protected)"]
```

```
Root → Admin["app/admin (RBAC)"]  
Admin → Routes["app/admin/routes/[symbol]"]
```

Routes match the file tree; Admin and Timeline are gated; deep links carry `correlationId`.

External Interfaces (by app)

Explorer (public & protected)

- **Public:**

`GET /v1/resolve/{hash}`, `GET /v1/batch/{batch_id}`, `GET /v1/root/{root}`, `POST /v1/verify/inclusion`, `GET /v1/search`, `GET /v1/anchor/{bundle_id}`. Renders **ExplorerRecord** with Leaf → Interval → ReFIN L2 → L1 Anchor; "Pending" badge when L1 isn't available.

- **Protected** (SIWE):

`GET /v1/preimage/{hash}`, `GET /v1/timeline?correlationId=...` (redacted preimage stream from GCS; correlated event timeline). Rate-limited via Redis; no PII in Mongo.

Trading App

- **Compliance-gated preview/submit** (backend consults Compliance Adapter; UI is fail-closed with reasons). Route badge indicates **broker|onchain** from Asset Routing decision. On-chain shows `txHash` with block-explorer link and chips for `submitted|mined|reverted|failed_retry|rejected(POLICY_BLOCK)`.

Admin (Asset Routing)

- `GET /v1/route/{symbol}`, `PUT /v1/route/{symbol}`, `POST /v1/route/bulk` with RBAC; decisions include `config_hash`. Unknown symbols surface **broker fallback** (`derived=true`). Optimistic concurrency with 409 diff modal; bulk CSV with per-row validation and failures export. Small ops counters (p95, cache-hit, invalidations, recent updates).

Auth (SIWE)

- Nonce → EIP-4361 sign → verify → cookie session; refresh rotation; logout/revoke. Cookies are HTTP-only; CSRF header on writes.

Key UI Flows (sequence)

A) Explorer Resolve / Search / Verify (public, with protected branches)

```
sequenceDiagram
    participant U as User
    participant W as Web App
    participant E as Explorer API

    U->>W: Paste hash / search

    alt Resolve by hash
        W->>E: GET /v1/resolve/{hash} + x-correlation-id
        E->>W: Leaf, Interval, ReFIN L2, L1 Anchor
    else Batch/root paths
        W->>E: GET /v1/batch/{batch_id} / GET /v1/root/{root}
        E->>W: Results
    end

    W->>E: POST /v1/verify/inclusion
    E->>W: boolean

    opt Protected
        W->>E: GET /v1/preimage / GET /v1/timeline
        E->>W: redacted JSON / correlated events
    end

    Note over W,E: 429 → retry affordance (idempotent reads) ; never retry writes
```

Inputs accepted: `hash, batch_id, root, correlationId, clientId` . Preimage/timeline require session. Standard "Pending anchor" and clean 404 copy.

B) Admin Edit with Optimistic Concurrency

```
sequenceDiagram
    participant UI as Admin UI
    participant API as Routing API
    UI->>API: GET /v1/route/{symbol}
    API->>UI: config + config_hash=abc
    UI->>API: PUT config + config_hash=abc
    API->>UI: 409 + latest config_hash=def + diff
    UI->>UI: Show diff modal → user accepts
    UI->>API: PUT config + config_hash=def
    API->>UI: 200 OK + invalidate cache
```

Shows server vs local; 409 path fully tested; RBAC required for writes.

State & Error Taxonomy

Unified across apps:

- **Loading / Empty / Partial ("Pending") / 400 / 401-403 / 404 / 409 / 429 / 5xx** with standardized copy + skeletons + retry affordances. Explorer chips show `root_submitted` / `anchored` ; trades use `submitted|mined|reverted|failed_retry|rejected(POLICY_BLOCK)` with reason tooltips.

Feature Flags

Flags gate **Admin, Timeline, Preimage, Policy/Claim** UI. With flags off, routes degrade to placeholders and no dead links remain; badges show network/flag state.

Security & Privacy

- **SIWE cookies only** between Web and Auth; names/domains match; CSRF header on writes. No client secrets in bundles.
- **Headers/middleware:** set `x-correlation-id` , HSTS, CSP (nonce), X-Frame-Options, Referrer-Policy, Permissions-Policy.

- **Preimage privacy:** redacted artifacts only; access requires SIWE; no PII stored in Mongo.

Observability

- Pass/echo **x-correlation-id**; deep links carry `correlationId`. Web-vitals, JS errors, retry/429 telemetry; ops counters visible in Admin.

Performance & Delivery

- Budgets: **LCP ≤ 2.5s, CLS ≤ 0.1, TBT ≤ 200ms**; route-split heavy views; use Explorer endpoints vs client proofs; asset caching + SWR for GET APIs.

Component Contracts (by app)

Explorer

- **Inputs:** `hash|batch_id|root|correlationId|clientId`.
- **Outputs:** normalized **ExplorerRecord** + inclusion boolean; protected preimage stream; timeline events.
- **States:** Loading/Empty/Partial/404/429/5xx; Anchor "Pending".

Trading

- **Preview:** displays compliance verdict & reasons; **fail-closed** behavior.
- **Execution hints:** **Route badge** (broker/onchain); on-chain tx lifecycle chips; `txHash` link.

Admin

- **Records:** `{symbol, route, driver_hint, enabled, updated_at, config_hash}` with filters; decision preview panel shows `{route, driver_hint, params, policy_flags, config_hash}`; 409 diff modal. Bulk CSV import/export failures.

Accessibility & i18n

WCAG 2.1 AA; keyboard-first; visible focus; ARIA on chips/copy/diff modal; live-region toasts. ICU message keys; locale-aware numbers/dates; ISO-8601 with

UTC labels; RTL-safe layouts; units with tooltips (bps, gwei, seconds).

Testing

- **Auth:** nonce/verify/refresh/logout; 401/403 copy.
- **Explorer:** resolve/search/verify success; 404/429 copy; protected preimage/timeline enforce session.
- **Admin:** list/edit/bulk; 409 diff flow end-to-end; RBAC on writes.

Non-functional Guardrails

- Respect 429 (retry idempotent reads with jitter); never retry writes from browser; show standardized copy for errors.
- Small runtime counters in Admin; Explorer/Trading use badges for network/chain and status.

Open Integrations

- **On-chain route visualization** depends on OrderIdMap and driver events (`orders.evt`). UI links to block explorers using `txHash` .

Surfaces

- **Explorer:** public + protected endpoints fully integrated; inclusion verify works; anchor "Pending" surfaced; redacted preimage & timeline under auth.
- **Trading:** live compliance gating; route badges; on-chain lifecycle chips + explorer links.
- **Admin:** CRUD + bulk with RBAC; optimistic concurrency (409 diff) and ops counters.

Trading and policy / on-chain components

Purpose

Deliver a production trading experience with **live compliance-gated previews**, **deterministic routing badges** (broker vs on-chain), **policy claim enforcement** when tokens require **fresh ACE**, and **on-chain lifecycle visualization** with `txHash` and block-explorer links. The UI consumes Asset Routing, Token Policy, and Compliance Adapter outcomes via the platform backends and renders a fail-closed posture for user actions.

Ownership & boundaries

- **Owned in UI:** Trading screens, **Compliance-Gated Preview** widgets, **Routing Badge**, **Policy Claim flow UIs**, **On-Chain Lifecycle chips** and explorer links, error/state taxonomy for trading actions.
- **Provided by backends:** Asset Routing `RouteDecision`, Token Policy `PolicyDecision` and `ClaimResponse`, Compliance Adapter verdicts, on-chain driver `orders.evt` and `audit.evt`.

External interfaces (UI client layer)

- **Asset Routing:** `GET /v1/route/{symbol}` → `RouteDecision { route: broker|onchain, driver_hint, policy_flags.need_fresh_ace, params, config_hash }`. Default route **broker** on miss; 404 means default upstream broker.
- **Token Policy:** `GET /policy/{chainId}/{token}` → `PolicyDecision { require_fresh_ace, policy_version, claim_ttl, required_claims[], attester_pubkey }`; `POST /claim/issue` (Idempotency-Key) → `ClaimResponse` or **423 POLICY_BLOCK**.
- **Compliance Adapter (indirect via backends):** cache-first **ACE** verdicts; never returns ALLOW without a valid verdict (fail-closed).
- **On-Chain Driver events:** `orders.evt { clientOrderId, txHash, status submitted|mined|reverted|failed_retry|rejected POLICY_BLOCK, reason?, ts, correlationId }` and `audit.evt`.

`OrderIdMap { clientId → txHash }` for idempotent reconcile.

Data & state model (UI)

- **RoutingBadgeState**: derived from `RouteDecision.route` and `route=onchain` badge with tooltip; **derived** or fallback indication if decision defaults to broker.
- **PreviewContext**: `{ account_id, symbol, qty, ... }` → preview call; UI renders **verdict ALLOW|REVIEW|DENY** with reasons from backend; **submit disabled** unless ALLOW. (Fail-closed.)
- **PolicyHints**: from `PolicyDecision.require_fresh_ace`; when true and route is on-chain, UI requires claim issuance before enabling submit.
- **OnChainTxnState**: chips drawn from `orders.evt.status`; links use `txHash` resolved via `OrderIdMap`.

Component map

```
---
config:
  layout: elk
  theme: neutral
---
flowchart TB
  trading["Trading Screen"] --> badge["Routing Badge"]
  trading --> preview["Compliance-Gated Preview"]
  trading --> claim["Token Policy Claim UI"]
  trading --> lifecycle["On-Chain Lifecycle Chips"]
  preview --> verdict["Verdict and Reasons Panel"]
  lifecycle --> link["Block Explorer Link from txHash"]
```

- **Routing Badge**: decodes `RouteDecision` and shows **broker** or **on-chain** with hint; shows “derived” when falling back to broker.
- **Compliance-Gated Preview**: shows backend verdict and reasons; **submit** gated by ALLOW.

- **Token Policy Claim UI:** issues short-lived EIP-712 claim when `require_fresh_ace=true` ; blocks on **423 POLICY_BLOCK**.
- **Lifecycle Chips:** `submitted|mined|reverted|failed_retry|rejected(POLICY_BLOCK)` ; explorer link via `txHash` .

Core flows (sequence)

A) Route and preview with fail-closed gating

```
sequenceDiagram
    participant UI as "Trading UI"
    participant R as "Asset Routing API"
    participant B as "Backend Preview"
    UI->>R: "GET route for symbol"
    R->>UI: "RouteDecision with need_fresh_ace and params and config_hash"
    UI->>B: "POST preview with order context"
    B->>UI: "Compliance verdict and reasons"
    UI->>UI: "Render Routing Badge and enable Submit only on ALLOW"
```

Default route is **broker** on cache/miss; preview always **fail-closed** on non-ALLOW or backend error.

B) On-chain with token policy claim

```
sequenceDiagram
    participant UI as "Trading UI"
    participant R as "Asset Routing API"
    participant P as "Token Policy API"
    participant D as "On-Chain Driver Backend"
    UI->>R: "GET route for symbol"
    R->>UI: "RouteDecision route=onchain and need_fresh_ace=true"
    UI->>P: "GET policy for chainId and token"
    P->>UI: "PolicyDecision require_fresh_ace true and policy_version and claim_ttl"
```

```

UI→>P: "POST claim issue with Idempotency-Key"
alt "ALLOW after fresh ACE"
  P→>UI: "ClaimResponse with claim and expiry"
  UI→>D: "POST submit with claim attached"
  D→>UI: "orders.evt submitted then mined or reverted"
else "DENY or REVIEW"
  P→>UI: "423 POLICY_BLOCK with reasons"
  UI→>UI: "Disable submit and show reasons"
end

```

Freshness and claim issuance are **required** only when policy demands; otherwise proceed normally.

Rendering details

- **Routing Badge:** pill with text `Broker` or `On-chain` ; tooltip shows `driver_hint` and `params` (e.g., `mev_protect` , `slippage_bps`). **Derived** chip for broker fallback.
- **Verdict Panel:** maps backend verdict to color and message; reasons list; **Submit** button disabled unless ALLOW; error surface for adapter unavailable handled as rejection.
- **Claim UI:** shows `policy_version` and `claim_ttl` ; retries on idempotent **409** to fetch existing claim; blocks on **423** with reason codes.
- **Lifecycle Chips:** transitions drawn from `orders.evt` ; when status is `submitted|mined|reverted|failed_retry|rejected` , show timestamp and optional reason tooltip; **txHash** anchors to the correct L2 explorer.

Error and retry model

- **Preview path:** UI treats adapter/backends returning non-ALLOW or transport errors as **blocked**; show standardized copy; do **not** retry writes from the browser.
- **Routing read:** idempotent GET; safe to retry with jitter; 404 renders broker fallback badge.

- **Claim issuance:** send **Idempotency-Key**; on **409** render success using returned claim; on **423** display **POLICY_BLOCK** reasons; no auto-retry.
- **On-chain events:** if `failed_retry` or `reverted`, chips show reason; reconciliation continues idempotently due to `OrderIdMap`.

Security & privacy

- Claims are **EIP-712 signed** by the module; UI only transports them and never stores secrets; SIWE session required for actions; policy and ACE flows audited via `audit.evt`.
- No PII displayed; reasons are non-PII codes from Compliance Adapter or Token Policy.

Observability

- Thread `x-correlation-id` through route, preview, claim, and submit calls; surface per-action toasts with the same ID.
- Admin/ops dashboards (outside UI) track **route lookup p95, policy read p95, claim rate and 423s, orders.evt** lag; UI shows basic telemetry chips where appropriate.

Performance

- **Routing read** from Redis-backed cache; short-TTL default to **broker** on miss.
- **Policy reads** and claim issuance are service-side; the UI minimizes round-trips by only invoking claim when both `route=onchain` and `require_fresh_ace=true`.

Feature flags

- Gate claim UI and on-chain visualization during rollout; with flags off, show broker-only badges and hide claim prompts. (Backends still enforce policy.)

Accessibility & i18n

- Keyboard-first operation; ARIA roles on chips, badges, and tooltips; live region announcements for verdict changes and claim issuance; messages localized;

timestamps ISO-8601 with UTC labels.

Testing

Unit

- RouteDecision mapping to badges and tooltips.
- Preview verdict mapping and **submit gating**.
- Claim issuance: Idempotency-Key handling (200 vs 409), 423 block reasons.

Integration

- End-to-end on-chain path: route→policy read→claim issue→submit→ `orders.evt` renders chips and explorer link.
- Reconcile retry: duplicate `orders.cmd` does not duplicate UI state thanks to `OrderIdMap`.

E2E

- Asset routed **broker** shows broker badge; **on-chain** shows claim UI and blocks until claim present; denial reasons shown on 423; lifecycle chips progress to **mined** or **reverted** with reason.

Exit criteria

- UI consistently shows **Routing Badge**;
- **Preview** is live and **fail-closed**;
- **Policy Claim** enforced only when required and correctly blocks or enables submit;
- **On-Chain Lifecycle** chips and **txHash** explorer links render from `orders.evt` ;
- Error/state taxonomy and retries follow platform rules; telemetry and correlation IDs are threaded through all steps.

Phase 1 UI Checkpoints

Scope alignment

Core **Phase 1** delivers ingest→signal, risk, and **brokered** execution; **SIWE/identity is Phase 2**. UI Phase 1 ships without SIWE and does not depend on Phase-3 Explorer or Phase-4 Admin/Asset Routing backends.

Phase Checkpoints:

- a. UI Phase 1 Pre-Start Monorepo and Setup
- b. Component Migration and Static Integration
- c. Dashboard Scaffolding and Stubbed Data Feeds
- d. API Client and Broker Integration (stubbed)

Phase 1 Exit (UI) — Alignment & Guardrails

- **No SIWE dependency**; wallet UI allowed, but **auth flows land in Phase 2**.
- **No Explorer or Admin write-backs** (they depend on Phase-3 audit and Phase-4 Asset Routing).
- **Observability & perf baselines** in place (RUM, JS errors, correlation id; LCP/CLS/TBT budgets).

Result: After Core Phase 1 completes, the UI is **feature-complete visually and interactively** against **simulated/mock**ed contracts, with repo/CI standards and route structure that match the migration plan—ready to light up live identity (Phase 2), Explorer (Phase 3), and Admin/Asset Routing (Phase 4) without refactors.

Phase 2 UI Checkpoints

Scope Alignment

Core **Phase 2** provides **SIWE**, **CCID/KYC**, and the **Compliance Adapter**. UI Phase 2 replaces mocked identity/compliance with live flows, while the passive market/positions feed remains simulated; user-initiated actions (preview/submit) hit live endpoints and must fail-closed on non-ALLOW.

Phase Checkpoints:

- a. Foundational Identity (SIWE Integration)
- b. User Onboarding and Attestation (CCID/KYC Flow)
- c. Live Compliance Gating in UI

Phase 2 Exit (UI) — Alignment & Guardrails

- **Authentication is live** with cookies, CSRF, rotate, revoke; protected routes enforce SIWE/RBAC.
- **Onboarding/attestation is live** and drives adapter refresh; UI stores no PII.
- **Compliance gating is live** for previews/submissions; **fail-closed** posture preserved; passive data feed remains simulated.
- **Operational checks:** alerts on 401 churn/429 spikes/CSP violations; budgets and accessibility checks applied to new routes.

Result: After Core Phase 2, the UI runs with **live identity and compliance**, gating all trading actions while keeping passive feeds simulated—ready to adopt Phase-3 audit/Explorer integrations and Phase-4 admin/policy without structural refactors.

Phase 3 UI Checkpoints

Scope Alignment

Core **Phase 3** exposes the **Explorer API** (resolve, batch, root, search, inclusion verify, anchor) plus **protected** preimage and timeline under SIWE, with Redis rate limits and correlation-aware responses. The UI replaces mocks with these live endpoints and surfaces standard states (Loading, Empty, Partial “Anchor pending”, 404/429, 5xx).

Phase Checkpoints

- a. Explorer Foundation and Public Endpoints
- b. Protected Data (Preimage) and Correlated Timeline

Phase 3 Exit (UI) — Alignment & Guardrails

- **Live Audit Trail:** All public Explorer routes live; proofs verified; anchors displayed with “Pending” when appropriate.
- **Auth Integration:** SIWE unlocks **preimage** and **timeline**; no PII in Mongo; Redis rate limits enforced.
- **API Parity & DX:** UI matches Explorer API surface and UX specs; deep links carry **correlationId**.

Result: After Core Phase 3, the UI provides a **live, user-verifiable** audit trail with public and protected views, ready to incorporate **Admin/Policy** and on-chain features in Phase 4 without structural refactors.

Phase 4 UI Checkpoints

Scope Alignment

Core **Phase 4** adds **Asset Routing** (admin CRUD + bulk), **Token Policy Module** (claim issuance for fresh ACE), **On-Chain Driver** (OrderIdMap/txHash, submitted→mined/reverted), **zk-VaR Attestations**, and **DePIN (QoS Oracle + Leaderboard/Staking)**. UI Phase 4 lights these up behind RBAC and feature flags where noted.

Phase Checkpoints:

- a. Admin UI for Asset Routing
- b. On-Chain Policy and Execution UI
- c. zk-VaR & DePIN Integration

Phase 4 Exit (UI) — Alignment & Guardrails

- **Admin UI live** with RBAC: list/edit/bulk, optimistic concurrency (409 diff), ops counters.
- **On-chain execution surfaced**: route badges, **PolicyClaim** issuance (flagged), and **orders.evt** lifecycle incl. **txHash** & block-explorer links.
- **zk-VaR + DePIN visible**: attestations resolved/verified in Explorer; QoS **leaderboard** with settlement links; privacy posture intact.
- **UX/System posture**: feature-flag controls; standard state taxonomy; budgets (LCP/CLS/TBT); security headers/CSP; deep links & correlation IDs.

Result: UI is **feature-complete** and aligned with all four core phases—production-grade, secure, observable, and fully integrated across **Admin, Explorer, On-Chain execution, zk-VaR, and DePIN**.

Phase Checkpoints for UI Integration

Scope Alignment

As per the **Bridge Plan** documentation, each phase of UI integration will be completed only **after** it's corresponding core development phase has been completed.

Phased Integration:

Phase 1 UI Checkpoints

Phase 2 UI Checkpoints

Phase 3 UI Checkpoints

Phase 4 UI Checkpoints

a. UI Phase 1 Pre-Start Monorepo and Setup

0) Conventions & Standards

- ☐ **Monorepo layout:** `apps/web` (Next.js App Router), `packages/ui`, `packages/api-clients`, `packages/config`.
- ☐ **Routes (initial stubs):** `/landing`, `/explorer`, `/admin`, `/auth` (placeholder only), `/api` (Next handlers as needed).
- ☐ **TypeScript strict + ESLint/Prettier + Husky/Commitlint; Conventional Commits** required on PR.
- ☐ **Owners:** CODEOWNERS for `apps/web/**`, `packages/ui/**`, `packages/api-clients/**`.

1) Repository Bootstrap

- ☐ Initialize Next.js (App Router) under `apps/web` and create shared packages (`ui`, `api-clients`, `config`).
- ☐ **Tailwind** config scans both app and `packages/ui`.
- ☐ **TS path aliases:** `@ui/*`, `@api/*`, `@lib/*`.

2) CI/CD and Branch Protection

- ☐ CI on PRs: **typecheck, lint, unit, E2E smoke, SBOM/SCA, bundle-size budgets**, preview deploy.
- ☐ **Artifact provenance:** inject `BUILD_SHA`, `BUILD_TIME` at build.
- ☐ Protect `develop` / `main` with required checks (as above).

3) Smoke Tests

- ☐ CI builds `apps/web` successfully; lint passes; pre-commit hooks block violations.
- ☐ Basic E2E smoke opens `/landing` and `/explorer` stubs.

b. Component Migration and Static Integration

1) Component & Asset Migration

- ☐ **Dashboard modules** (`src/components/dashboard`) → `apps/web/app/explorer/_components` .
- ☐ **Trading modules** (`src/components/trading`) → `apps/web/app/admin/_components` (static only in Phase 1).
- ☐ **Wallet provider files** → `apps/web/app/_providers/wallet` (not gated by SIWE in Phase 1).
- ☐ **Public site** pages/components → `/landing` ; unify styles and tokens in `apps/web / packages/ui` .

2) Routing and Layout

- ☐ Remove Vite/SPAs; replace with **App Router** layouts/pages; Sidebar uses `<Link>` .
- ☐ Create `apps/web/app/(explorer)/layout.tsx` with shell (Header/Sidebar).
- ☐ Add static routes: `/explorer/overview` , `/explorer/positions` , `/admin` (read-only view).
- ☐ Onboarding flow scaffolded as static route group (no session persistence in Phase 1).

3) Testing & Validation

- ☐ Components compile (TS strict) in `packages/ui` and `apps/web` .
- ☐ **Storybook** for `packages/ui` (optional) for visual regression.
- ☐ **Styling parity** with demo on static pages.
- ☐ State/status chips conform to **orders.evt** mapping when rendered statically.

4) Exit Criteria

- Demo UI assets migrated and organized; **App Router shell renders** with static navigation; **no Vite/SPA router** remains.

c. Dashboard Scaffolding and Stubbed Data Feeds

1) Assemble Primary Views

☐ **Overview** page composes `MetricsOverview` , `PerformanceChart` , `LivePositions` , `LiveTradesFeed` .

☐ **Trading Interface** screen composes same primitives for focused order/position view.

☐ **Risk/Gamification** cards render with **mock** data only (no backend dependencies in Phase 1).

2) Stubbed Data Service

☐ Implement a **client-side simulation hook** (e.g., `useSimulation`) that emits synthetic **orders.evt** lifecycle and derives `trades` , `positions` , `portfolioMetrics` .

☐ Ensure shapes align with the **demo contracts** (e.g., `OrderPreviewResult` , `Position` , `Order` types).

3) UI Wiring

☐ All dashboard components consume the simulation hook and **update in real time**.

☐ A visible **"Simulated Data"** badge appears; **no WebSocket/broker** connectivity in Phase 1.

☐ Wallet UI may render, but **no SIWE** or gated flows yet (Phase 2).

4) Exit Criteria

- Interactive dashboards powered by **stubbed data**, clearly labeled as simulated.
- Data objects comply with **contracted types** to avoid drift when backends arrive.

d. API Client and Broker Integration (stubbed)

1) API Client Scaffolding

- ☐ Generate initial **TypeScript SDKs** from **OpenAPI** into `packages/api-clients`.
- ☐ Wrap SDKs with React hooks (`useOrders` , `useRiskPreview`), handling `loading/error/success`.
- ☐ Add **MSW** (or equivalent) to intercept calls and return **schema-compliant** mocks for `/orders/preview` , `/orders` , `/snaptrade/*`.
- ☐ Keep TS ↔ OpenAPI in **lock-step** (types mirror `components.schemas.OrderPreviewResult`).

2) Broker Connectivity UI

- ☐ Integrate the **broker connect modal** (e.g., SnapTrade portal) on a settings page; wire actions to API hooks; responses are **mocked**.
- ☐ Maintain **local** connection state (`connected|disconnected`) from mocked results.

3) Trading Actions (mock)

- ☐ "Preview Trade" → `useOrders.preview` (MSW returns `OrderPreviewResult`).
- ☐ "Submit Trade" → `useOrders.execute` (mock accept/409 paths).
- ☐ Verify UI error states map to contract semantics (e.g., 409/unavailable).

4) Exit Criteria

- **API client layer** exists; all network calls **intercepted** with spec-accurate mocks; **trading UI flows** render success/error states without live backends.

a. Foundational Identity (SIWE Integration)

0) Conventions & Standards

- ☐ **Session model:** Web uses **HTTP-only Secure cookies** with CSRF protection; refresh **rotation** required.
- ☐ **Error model (UI copy & handling):** `NONCE_INVALID, SIGNATURE_INVALID, POLICY_VIOLATION, CHAIN_DENIED, ACCOUNT_BLOCKED, REFRESH_REVOKED`.
- ☐ **Security headers & RBAC readiness** applied to auth routes; no service secrets in bundles.

1) SIWE Authentication Flow

- ☐ **Nonce:** `GET /siwe/nonce` wired from "Connect Wallet".
- ☐ **Verify:** EIP-4361 sign → `POST /siwe/verify`; enforce **domain/origin/URI/chainId** allow-lists.
- ☐ **Session creation:** Store access/refresh (cookies), update UI auth state with `account_id` and `wallet_id`.
- ☐ **Mapping:** Bind wallet to account (create if missing; status pending until KYC).

2) Session Lifecycle

- ☐ **Auth context** provider exposes `status, account, wallet` to routes.
- ☐ **Refresh:** `POST /auth/refresh` with rotation; **Logout:** `POST /auth/logout` clears cookies; support device-wide revoke.
- ☐ **Route guards:** Gate authenticated sections (settings, trading) behind SIWE.

3) Observability & Tests

- ☐ UI surfaces **session badges**; alerts on 401 churn; correlationId on requests.
- ☐ Unit/E2E: nonce success/invalid, verify success/ `SIGNATURE_INVALID`, refresh rotation, logout clears cookies.

4) Exit Criteria

- End-to-end **SIWE login/refresh/logout** works with cookies+CSRF, standardized errors, and guarded routes; RBAC ready for later phases.

b. User Onboarding and Attestation (CCID/KYC Flow)

1) Flow Integration

- ☐ **Gate by status:** Show onboarding only when `kyc_status ∈ {pending,incomplete}` from session/profile.
- ☐ **Start:** `POST /ccid/start` → receive provider URL/token; handoff to provider.
- ☐ **Return & poll:** Poll `GET /ccid/status` until `approved|denied|review`.
- ☐ **Webhook awareness** (test env): simulate `/ccid/webhook/provider` and ensure idempotent update.

2) Completion & Triggers

- ☐ On **approved**, update UI session/profile, transition to main dashboard; schedule refresh before expiry.
- ☐ Fire Adapter **invalidate/refresh** on issuance/revocation/expiry so compliance cache updates.

3) UX & Security

- ☐ Continuous status feedback (`pending, review, approved, denied`) with accessible messages.
- ☐ Do **not store PII**; only provider refs and hashed evidence appear in UI or logs.

4) Tests

- ☐ Start→webhook→issue_or_refresh→status happy path; revoked/expired paths schedule re-issue; adapter refresh observed.

5) Exit Criteria

- Live onboarding completes; **attestation exists**, session reflects new status; adapter notified; UI routes unlocked accordingly.

c. Live Compliance Gating in UI

1) Remove Mocks on Compliance/Preview

- ☐ Disable MSW for **/orders/preview** (and related compliance checks); keep simulation only for passive feed.

2) Integrate Compliance Adapter

- ☐ `useOrders.preview` calls live backend, which consults **Compliance Adapter**; UI displays **status=ALLOW|REVIEW|DENY**, `reasons[]`, and `source=cache|fresh`.
- ☐ Enforce **fail-closed**: non-ALLOW disables **Submit Trade** with explanatory tooltip; adapter errors treated as rejection (`COMPLIANCE_UNAVAILABLE`).

3) SLOs & Telemetry

- ☐ Telemetry shows **cache-hit ratio** and p95 preview latency; cache-hit path \approx negligible, fresh path within ACE budget.

4) Tests

- ☐ Happy paths: **miss**→**ACE**→**upsert**→**ALLOW** then **cache-hit**; deny and adapter-down paths block submit and surface reasons.

5) Exit Criteria

- Trade **preview** and gating run **live** against the adapter; UI correctly enforces verdicts and error handling (fail-closed).

a. Explorer Foundation and Public Endpoints

1) Explorer UI Scaffolding

☐ Global search accepts **hash, batch_id, root, correlationId, clientOrderId**; render an **ExplorerRecord** (leaf, interval, L2, L1, preimage_ref).

☐ Remove MSW for Explorer; wire live calls:

`GET /v1/resolve/{hash}` , `GET /v1/batch/{batch_id}` , `GET /v1/root/{root}` , `GET /v1/search` , `GET /v1/anchor/{bundle_id}` , `POST /v1/verify/inclusion` .

☐ Record layout includes copy-buttons and block-explorer links for L2/L1 receipts; show **Pending** when L1 anchor not yet available.

2) Inclusion Proof Verification

☐ "Verify inclusion" posts to `/v1/verify/inclusion` and displays a clear success/failure result.

3) States, Deep Links, and Errors

☐ Standard states and copy: Loading, Empty, **Partial** ("Anchor pending"), 404, **429 retry affordance**, 5xx.

☐ Expose **deep links** (resolve & search results) with visible **correlationId** in the UI and share URLs.

☐ Respect read-retry rules (jitter for idempotent GETs; never retry writes from browser).

4) Observability & Budgets

☐ Pass/echo **x-correlation-id** on requests; dashboards track **p95 resolve/search latency** and anchor fetch errors; show network/chain badge.

☐ Keep Web-Vitals within budgets (LCP, CLS, TBT) and cache GETs with **stale-while-revalidate**.

5) Tests & Exit Criteria

☐ E2E: search by known **hash** renders ExplorerRecord; inclusion-verify returns success; 404/429 copies appear as specified.

- **Exit:** Public Explorer fully live: resolve/search/verify/anchor display from **leaf** → **interval** → **L2 receipt** → **L1 anchor**, with deep links and standard states.

b. Protected Data (Preimage) and Correlated Timeline

1) Authenticated API Integration

- ☐ Gate protected UI affordances behind **SIWE session**; include bearer JWT on calls. Endpoints:

`GET /v1/preimage/{hash}` , `GET /v1/timeline?correlationId=...` .

2) Preimage Viewer

- ☐ Show **redacted JSON preview** inline; enable **Download Preimage** only when authenticated; never store PII in Mongo.
- ☐ Stream artifact from GCS with checksum; enforce SIWE scope; keep privacy posture (artifacts/pointers only).

3) Correlated Timeline View

- ☐ When searching by **correlationId**, call `/v1/timeline` and render a chronological view spanning **risk** → **execution** → **audit**; each event links to its own resolve view.
- ☐ Ensure rate-limits are respected (Redis) and protected access is enforced.

4) States, Deep Links, and Errors

- ☐ Unauthenticated access yields **401/403** with standard copy; authenticated users see preimage/timeline data and shareable **deep links** with correlationId.

5) Tests & Exit Criteria

- ☐ E2E: unauthenticated preimage/timeline → 401/403; authenticated user successfully fetches **preimage** and **timeline**; inclusion verify still passes.
- **Exit:** Protected Explorer features live and **gated by SIWE**; UI streams **redacted** artifacts from GCS and renders end-to-end **timelines**.

a. Admin UI for Asset Routing

1) Admin Scaffolding & RBAC

- ☐ Protect `app/admin/**` with **SIWE session + admin role**; enforce RBAC on writes.
- ☐ Remove mocks; wire `GET /v1/route/{symbol}`, `PUT /v1/route/{symbol}`, `POST /v1/route/bulk` with bearer JWT.

2) Route Management UI

- ☐ **List & filter** table: `{symbol, route, driver_hint, enabled, updated_at, config_hash}`; filters: `route, asset_class, venue`. Decision preview shows `{route, driver_hint, params, policy_flags, config_hash}`.
- ☐ **Create/Edit** form fields: `route, driver_hint, enabled, slippage_bps, mev_protect, max_gas_gwei, token_policy_required, token_policy_ref, chainId?, venue` with **derived defaults** + validation hints.
- ☐ **Optimistic concurrency**: send `config_hash`; on **409** show diff modal and allow retry.

3) Bulk Ops & Observability

- ☐ **CSV import** (drag-drop) → `POST /v1/route/bulk`; per-row validation; **export failures CSV**.
- ☐ **Decision preview** drawer: `GET /v1/route/{symbol}`; show **derived=true** banner on broker fallback.
- ☐ Inline ops telemetry: **p95 lookups, cache-hit ratio, invalidations, recent updates** counters.

4) States & Errors

- ☐ Standard copy for **404** ("Defaulting to broker route"), **409** diff modal, **429/5xx** retry affordance.

5) Tests & Exit Criteria

- ☐ Unit: list/read/edit/bulk; simulate **409** and verify modal flow. E2E: edit with 409 succeeds on retry. **Exit**: Admin CRUD + bulk live with RBAC and ops counters.

b. On-Chain Policy and Execution UI

1) Feature Flags & Route Display

- ☐ Wrap new on-chain UI behind flags (e.g., `NEXT_PUBLIC_FLAG_ONCHAIN_EXEC`); badges show flag/network.
- ☐ In trading views, surface **route badge** ("On-chain") from Asset Routing **RouteDecision**; show broker fallback chip on default.

2) Token Policy Claim Flow

- ☐ Read token **policy** (`GET /policy/{chainId}/{token}`) and **require_fresh_ace** hinting in UI.
- ☐ Before submit (for routes needing fresh ACE), call `POST /claim/issue` with **Idempotency-Key**; block on **DENY/REVIEW (423)**; show reason codes.
- ☐ Keep **claim issuance** behind a feature flag until fully ready.

3) On-Chain Trade Lifecycle

- ☐ Update **LiveTradesFeed** to map on-chain **orders.evt** chips: `submitted | mined | reverted | failed_retry | rejected(POLICY_BLOCK)` with `reason` tooltips.
- ☐ Display **txHash** (via **OrderIdMap**), add **block-explorer** links for the correct L2.
- ☐ Ensure gateway parity: preview still **fail-closed** if policy blocks; reconcile like broker path once events arrive.

4) Tests & Exit Criteria

- ☐ End-to-end on-chain: **onchain route selected** → **claim issued (if required)** → **tx mined** → **events rendered** → **links work**.
 - **Exit:** Route badges accurate; claim flow enforced; feed reflects **submitted/mined/reverted** with explorer links; features under flags.

c. zk-VaR & DePIN Integration

1) zk-VaR Attestation Display

- ☐ Light up **DualProofGate**: show **proof_hash / verification status** from **ZkVarAttestations**; UI verify button hits `POST /v1/verify`.
- ☐ In Explorer, **resolve attestations**: `GET /v1/attestation/{id}` → public inputs + **GCS proof_ref**; render + allow verify.
- ☐ Maintain **privacy-by-design** (commitments only; no holdings/PII).

2) DePIN UI (Leaderboard & Staking)

- ☐ New **Leaderboard** view (public): `GET /v1/leaderboard?epoch=&kind=`; show ranks and link out to settlement receipts.
- ☐ Per-node cards: uptime/success/rewards from **NodeMirror/QoS Scores/RewardLedger** mirrors; expose **region/kind** filters.
- ☐ **Staking/Rewards** panel (read-only): show staked amount and claimable totals; **link to ReFIN L2** contracts/tx for actions.

3) States, Security & Ops

- ☐ Standard **429/5xx** retry affordance; deep links and network/chain badges.
- ☐ Metrics: leaderboard latency, verify success rate; alerts for VK mismatches/backlog.

4) Tests & Exit Criteria

- ☐ E2E: attestation resolve+verify; leaderboard renders with paging/filters; links to on-chain receipts valid.
- **Exit**: Trading UI and Explorer surface **live zk-VaR** info; **DePIN leaderboard** visible with rewards transparency.

Bridge Plan, Landing plus MVP to Production

Purpose

Define full client flows for Auth, Explorer, Timeline, Preimage, Admin, Token Policy. Specify UI states, copy, inputs, API linkage, errors, accessibility, i18n, metrics, tests. Align to phasing and release gates.

REFERENCES

MVP GUI zip:

<https://drive.google.com/file/d/1o9wTCQLYKGDYyUWM90zIEhLWVMKJJp5/view>

Current Public site zip:

<https://drive.google.com/file/d/1CRpYiPK8o0di7cf2ZkoNZFBpfxY1P5ys/view>

Migration Overview

Section A - Client Experiences, end to end

Section B - Bridge playbook from zip projects to the phased build

Section C - Diagrams and reviewer checklists

Section D - Testing strategy and specs

Section E - Performance Plan and Budget

Section F - Security and Compliance

Section G - Observability and ops

Section H - Success rate, risks, scorecard

Section I - Repo structure, ownership, and storage

Section J - Security gaps and risk hardening

Section K - Final notes for the receiving engineer

Migration Overview

TARGET PACKAGE

Single monorepo with Next.js App Router and TypeScript strict

apps/web for the app

packages/ui for tokens and headless components

packages/api-clients for generated SDKs

packages/config for shared config

ROUTES IN THE NEW APP

/landing for the public site

/explorer with [hash], search, timeline, preimage

/admin with routes/[symbol]

/auth for SIWE screens and actions

/api for SIWE route handlers when needed

MOVE MAP FROM THE ZIPS

From project-landing-page.zip

src/pages/* → **apps/web/app/landing**

src/components/* → packages/ui/components or

apps/web/app/landing/_components

styles and Tailwind entries → merge into apps/web

Supabase code stays inside /landing only

From project-mvp-page.zip

src/components/dashboard → apps/web/app/explorer/_components

src/components/trading → apps/web/app/admin/_components

src/components/wallet and src/config/wagmi.ts →

apps/web/app/_providers/wallet

src/lib/* → apps/web/src/lib after replacing direct fetch with SDK calls

openapi/openapi.yaml → packages/api-clients if authoritative

server/* remains out of the frontend package

WHAT WE REMOVE OR REPLACE

Remove Vite and SPA routers from both projects

Replace BrowserRouter and Routes with App Router pages and links

Move global CSS to app/globals.css

Replace direct fetches with generated clients in packages/api-clients

Keep any Supabase usage isolated to /landing

No service secrets shipped to the browser

SIWE AND RBAC

Nonce, sign, verify, refresh rotation, logout

HTTP-only Secure cookies with SameSite strict and CSRF

Device-wide revoke

Admin routes gated by role claims

OBSERVABILITY AND CORRELATION

Middleware sets x-correlation-id

Pass the id on every request

Web-Vitals, JS errors, action breadcrumbs

Dashboards for p95 resolve and search and error rates

SECURITY

Strict CSP with nonces

HSTS, X-Content-Type-Options, X-Frame-Options DENY, Referrer-Policy, minimal Permissions-Policy

SRI for third-party scripts

SBOM via CycloneDX, fail on critical CVEs

No secrets in client bundles

Domain allow-lists for auth redirects

PERFORMANCE

Route-level code splitting

next/image with AVIF and WebP

Immutable caching for hashed assets

Stale-while-revalidate on GET APIs

Budgets on mid devices: LCP 2.5 s, CLS 0.1, TBT 200 ms

CI AND RELEASE

Typecheck, unit, E2E smoke, SBOM, SCA, bundlesize budgets on every PR

Preview deploy per PR

DAST on authless endpoints

Release notes from Conventional Commits

Embed BUILD_SHA and BUILD_TIME

PHASES AND EXIT CRITERIA

0. Repo bootstrap

Monorepo created, TypeScript strict, linting and commit checks on main

Move maps executed and builds pass

1. Auth foundation

SIWE flows work end to end

Cookies, CSRF, revoke, RBAC in place

Auth unit and E2E pass

2. Explorer

Resolve by hash, batch, root

Verify inclusion

Search by correlationId and clientOrderId

Preimage and timeline protected by SIWE

429 retry affordance on reads

Explorer E2E green

3. Admin

List and filter routes

Edit with optimistic concurrency via config_hash

Bulk CSV import with per-row validation and failure export

Derived defaults and broker fallback banner

Counters for lookups, cache hits, invalidations, recent updates

Admin E2E green including 409 flow

4. Token Policy

Read policy, issue short-lived EIP-712 claim, optional verify

Feature-flagged until backend is ready

Policy tests green behind the flag

ACCEPTANCE CHECKS BEFORE GO LIVE

All protected routes enforce SIWE and RBAC

Explorer and Admin endpoints mapped and reachable

Preimage and timeline only after SIWE

Accessibility checks pass across routes

Budgets met on a mid mobile device

E2E suite green on staging

SBOM shows zero critical CVEs

CSP and SRI validated in a live preview

No secrets present in client bundles

RISKS AND GUARDS

Unfrozen API contracts → freeze OpenAPI, tag SDKs, block breaking changes behind previews

SIWE edge cases → negative tests for NONCE_INVALID and SIGNATURE_INVALID, refresh rotation, revoke-all

Rate limits → surface 429 with backoff on reads, never retry writes

CSP breakage → report-only burn-in, then enforce

RBAC drift → guards and E2E for 403 paths

Preimage privacy → server redaction, gating, no PII in logs

IMPLEMENTATION NOTES THE TEAM WILL USE

Tailwind content globs include apps/web and packages/ui

Path aliases for @ui, @api, @lib

Feature flags for Admin, Policy, Timeline, Preimage

Network and session badges in the header

Correlation id visible in logs and shareable deep links

REFERENCES FOR THE RECEIVING ENGINEER

MVP GUI zip

<https://drive.google.com/file/d/1o9wTCQLYKGDYyUWM90zIEhLWVMKJJp5/view>

Public site zip

<https://drive.google.com/file/d/1CRpYiPK8o0di7cf2ZkoNZFBpfxY1P5ys/view>

Section A - Client Experiences, end to end

OBJECTIVE

Define full client flows for Auth, Explorer, Timeline, Preimage, Admin, Token Policy. Specify UI states, copy, inputs, API linkage, errors, accessibility, i18n, metrics, tests. Align to phasing and release gates.

GLOBAL EXPERIENCE RULES

Session and auth. SIWE with nonce, sign, verify, refresh rotation, logout. HTTP-only Secure cookies with CSRF. Device-wide revoke. Admin routes require role claims. No service secrets in the browser. Copy and codes include NONCE_INVALID, SIGNATURE_INVALID, POLICY_VIOLATION, CHAIN_DENIED, ACCOUNT_BLOCKED, REFRESH_REVOKED.

Correlation and tracing. Generate a correlationId per page or session. Pass x-correlation-id on every request. Explorer supports correlation search and timeline.

Error and retry model. Respect service errors. Retry idempotent reads with jitter. Never retry writes from the browser. Show 429 retry affordance. Provide clear 4xx and 5xx copy.

Accessibility and i18n. WCAG 2.1 AA. Keyboard-first flows. Visible focus rings. ARIA labels for chips, copy buttons, modals. Live regions for toasts. ICU message keys. Locale number and date formatting with ISO-8601 timestamps and UTC labels. RTL-safe layouts. Units show bps, gwei, seconds with tooltips.

Design system. Tokens for color, spacing, type in packages/ui. Headless accessible components for Auth, Explorer, Admin, Policy.

EXPERIENCE 1, ONBOARDING AND SESSION

User goals. Link wallet, sign in, view session status, manage sessions.

Flow

1. Fetch nonce
2. User signs EIP-4361

3. Verify
4. Session cookies set
5. Rotation on refresh
6. Logout revokes and clears cookies
7. Device-wide revoke available

UI states and copy

Unauthorized. Lock icon and Sign in to continue

Error codes. Short messages mapped to the error model above

Wallet link and unlink supported

Phase and exit. Phase 1. Sign in, refresh rotation, logout, revoke, RBAC enforced. Auth unit and E2E pass

EXPERIENCE 2, EXPLORER RESOLVE AND SEARCH

User goals. Resolve a leaf or hash. Search by correlationId or clientOrderId. Inspect proofs and receipts. Fetch preimage under auth.

Inputs accepted. hash, batch_id, root, correlationId, clientOrderId

Routes and calls. GET /v1/resolve/{hash}. GET /v1/batch/{batch_id}. GET /v1/root/{root}. POST /v1/verify/inclusion. GET /v1/search. GET /v1/anchor/{bundle_id}. Protected, GET /v1/preimage/{hash}. Protected, GET /v1/timeline

Record layout

Leaf panel shows leaf_hash, emitter, emitted_at, seq_no

Interval panel shows batch_id, leaf_index, root

ReFIN L2 shows tx hash, block, confirmation time with link out

L1 Anchor shows tx hash or Pending

Preimage shows a redacted JSON preview with download under auth

Actions include Verify inclusion, Copy proof, Open in block explorer

Error copy

Hash not found. No leaf found for this hash

Anchor pending. Anchor pending, check back later

Limits and privacy. SIWE session required for preimages. Redis rate limits. No PII in Mongo

Phase and exit. Phase 3. Resolve, search, verify are public; /v1/preimage and /v1/timeline present but gated by SIWE auth and feature flags. 429 retry affordance present. E2E green for public endpoints.

EXPERIENCE 3, TIMELINE

User goals. View end to end events for a trade or order by correlationId

Source stitching. Audit leaves, Merkle intervals, ReFIN roots, L1 anchors. Timeline groups risk, execution, audit

Access. Timeline is protected. Requires SIWE session

Phase and exit. Phase 3. Timeline renders from correlation search with deep link per event. E2E includes timeline.

EXPERIENCE 4, PREIMAGE VIEWER

User goals. Inspect redacted preimage tied to a leaf

UI behavior. Redacted JSON preview. Expand and download only under auth.

Data safety. No PII in Mongo

Phase and exit. Phase 3. Protected route enforcement verified on staging.

EXPERIENCE 5, ADMIN, ASSET ROUTING

User goals. Search routes. Create or edit. Import bulk. Preview decisions. View cache and audit state

List and search. Table of symbol, route, driver_hint, enabled, updated_at, config_hash with filters for route, asset_class, venue

Decision preview. route, driver_hint, params, policy_flags, config_hash

Create and edit. Fields include route, driver_hint, enabled, slippage_bps, mev_protect, max_gas_gwei, token_policy_required, token_policy_ref, optional chainId, venue. Derived defaults shown with hints. On save, send config_hash. If 409, show conflict resolver

Bulk import. CSV drag and drop. POST to route bulk. Per-row validation. Export failures as CSV

Defaults and broker fallback. On read default, show derived true with broker fallback banner. Unknown symbol shows Defaulting to broker route chip

Observability. Small counters for p95 lookups, cache hit ratio, invalidations, recent updates. Row badge for recent updates

API linkage. GET route. PUT route. POST route bulk. Emits audit events on change. RBAC required

Conflict copy. This route changed while you were editing. Review updates and retry

Phase and exit. Phase 4. Admin read, edit, bulk import, conflicts, counters live. E2E includes 409 flow

EXPERIENCE 6, TOKEN POLICY, PHASE 4

User goals. Read policy for a token. Issue a short-lived claim. Verify when needed

Endpoints. GET policy. POST claim issue for EIP-712 PolicyClaim. Optional POST claim verify

UI surfaces require_fresh_ace and block reasons. Gated by feature flag until back end is ready

Phase and exit. Phase 4. Claim issuance behind flag with tests and copy

EXPERIENCE 7, RATE LIMITS AND SERVER FAILURES

429. Too many requests. Please wait and retry. Browser shows retry affordance with backoff for idempotent reads

5xx. Service unavailable. We are retrying. Link to status if present

EXPERIENCE 8, STATE TAXONOMY AND STANDARD COPY

Loading. Skeletons for tables and cards. Progress bar for long resolves

Empty. Admin, No routes yet. Create your first route. Explorer, No results. Try a different search or paste a leaf hash

Partial. Section warnings and Pending badge

Error 400. Inline hints. Invalid request. Check inputs and try again

Unauthorized 401 or 403. Lock icon. Sign in to continue

Not found 404. Symbol not configured. Falling back to broker routing

Conflict 409. Modal shows server config_hash and local. This route changed while you were editing. Review updates and retry

EXPERIENCE 9, PERFORMANCE AND NETWORK

Budgets. LCP 2.5 s. CLS 0.1. TBT 200 ms on mid devices

Behavior. Route split. Avoid heavy client proofs. Use Explorer endpoints

Assets and API caching. next/image with AVIF and WebP. Immutable cache for hashed assets. Stale-while-revalidate for GET APIs. Static via CDN with brotli

EXPERIENCE 10, METRICS AND ALERTS

RUM metrics, JS errors, action breadcrumbs with PII stripped

Dashboards track front end error rate, API error rate by endpoint, p95 resolve latency, p95 search latency, anchor fetch errors

Network badge and chain indicator in the UI

DEEP LINKS AND SHARING

Expose deep links for resolve and timeline with correlationId. Show correlationId near results and in share links

SECURITY CONTROLS IN EXPERIENCES

Strict CSP and headers. SRI. Dependency pinning. SBOM in CI. Builds fail on critical CVEs. OAuth redirect allow-lists align with SIWE validators. No secrets in client

TESTS BY EXPERIENCE

Auth. Unit for nonce lifecycle, error mapping, refresh rotation. E2E for onboarding and logout

Explorer. Unit and integration for resolve, batch, root, inclusion verify, search by correlationId. Contract mocks for timeouts, 4xx, 5xx, policy blocks. E2E for resolve and timeline

Admin. Unit for list, read, edit, bulk. Simulate 409 and show diff modal. E2E for edit with 409

Accessibility and visual. Axe scans. Keyboard and reader flows. Percy snapshots for key routes

PHASING AND OWNERSHIP

Phase 0. Repo, CI, design tokens, branch checks, artifact provenance. Exit when checks required on main. Owners per path defined

Phase 1. Auth experience complete with error model, cookies, CSRF, revoke. Exit when RBAC enforced. Tests pass

Phase 2. Explorer experience complete with resolve, search, verify, timeline, preimage under auth. Exit when E2E green. Protected routes validated on staging

Phase 3. Admin experience complete with list, edit, bulk, conflicts, counters. Exit when E2E green. Audit events visible

Phase 4. Token Policy gated. Exit when claim issuance and optional verify pass behind flag

ACCEPTANCE CRITERIA BY EXPERIENCE

Auth. Sign in, refresh, logout, revoke work. Error codes map to UI copy. RBAC enforced on Admin

Explorer. Inputs supported. Record layout complete. 429 and 5xx copy shown. Preimage gated. No PII stored

Timeline. Correlation view with risk, execution, audit. Deep links present

Admin. Defaults and broker fallback messaging. Bulk CSV with per-row validation and exported failures. Conflict modal on 409. Counters shown

Policy. `require_fresh_ace` and reasons surfaced. Claim issuance works behind flag

Performance. LCP, CLS, TBT within budget on mid devices. Route split. Images optimized. GETs use stale-while-revalidate. Static via CDN with brotli

Observability. Web-Vitals, JS errors, breadcrumbs captured. Dashboards for error and latency. CorrelationId in logs and deep links

Section B - Bridge playbook from zip projects to the phased build

SCOPE

Move both zip codebases into a single Next.js App Router monorepo. Replace SPA routing. Keep SIWE sessions. Wire Explorer and Admin APIs. Preserve useful UI. Remove dead code. Meet security, performance, testing, go-live gates.

PREFLIGHT

1. Create a clean workspace and git repo
2. Download both zips. Keep originals read-only

1. Node 20.x. pnpm 9.x. Git LFS for large images

SHELL BOOTSTRAP

```
mkdir refi-frontend && cd refi-frontend
git init
pnpm dlx create-next-app@latest apps/web --ts --eslint --src-dir --app --tail
wind --use-pnpm
mkdir -p packages/ui packages/api-clients packages/config tmp/zips
# Place zips into tmp/zips
unzip -q tmp/zips/project-mvp-page.zip -d tmp/mvp
unzip -q tmp/zips/project-landing-page.zip -d tmp/landing
```

PATH MAP, SOURCE TO TARGET

MVP GUI

- src/components/dashboard → apps/web/app/explorer/_components

- `src/components/trading` → `apps/web/app/admin/_components` or `shared`
- `src/components/wallet` and `src/config/wagmi.ts` → `apps/web/app/_providers/wallet`
- `src/lib/*` → `apps/web/src/lib` after audit and swap to SDK usage
- `server/*` → out of scope for web
- `openapi/openapi.yaml` → `packages/api-clients` if authoritative

Public site

- `src/pages/**/*` → `apps/web/app/landing`
- `src/components/**/*` → `packages/ui/components` or `apps/web/app/landing/_components`
- styles and Tailwind entries → merge into `apps/web`
- Supabase code stays in landing only

PHASE 0 DELIVERABLES

1. Monorepo layout with working Next.js app and shared packages
2. TypeScript strict. ESLint, Prettier, Husky, Commitlint
3. CODEOWNERS for `apps/web` and `packages` folders

COMMANDS AND EDITS

```
# Move marketing pages
rsync -av --exclude 'node_modules' tmp/landing/src/pages/ apps/web/app/la
nding/
rsync -av --exclude 'node_modules' tmp/landing/src/components/ packages/
ui/components/

# Move MVP widgets
mkdir -p apps/web/app/explorer/_components apps/web/app/admin/_compon
ents
rsync -av tmp/mvp/src/components/dashboard/ apps/web/app/explorer/_com
ponents/
rsync -av tmp/mvp/src/components/trading/ apps/web/app/admin/_compone
```

```
nts/  
rsync -av tmp/mvp/src/components/wallet/ apps/web/app/_providers/wallet/  
  
# Libraries  
rsync -av tmp/mvp/src/lib/ apps/web/src/lib/
```

TAILWIND UNIFY

1. Merge content globs to include app and ui package
2. Move shared tokens into packages/ui/tokens.ts

TAILWIND CONFIG

```
// apps/web/tailwind.config.ts  
import type { Config } from 'tailwindcss'  
  
const config: Config = {  
  content: [  
    './app/**/*.ts,tsx',  
    './../packages/ui/**/*.ts,tsx'  
  ],  
  theme: { extend: {} },  
  plugins: []  
}  
  
export default config
```

TS PATH ALIASES

```
// apps/web/tsconfig.json  
{  
  "compilerOptions": {  
    "strict": true,  
    "baseUrl": ".",
```

```

"paths": {
  "@ui/*": ["../../packages/ui/*"],
  "@api/*": ["../../packages/api-clients/*"],
  "@lib/*": ["./src/lib/*"]
}
}
}

```

PHASE 1 DELIVERABLES

1. Auth foundation with SIWE. Nonce, verify, refresh, logout. HTTP-only Secure cookies, CSRF
2. App shell with Header, NetworkBadge, SessionBadge
3. Wallet provider using wagmi. Replace RainbowKit screens if needed
4. Remove SPA routers. Use App Router

AUTH WIRING

```

// apps/web/app/layout.tsx
import WalletProvider from '../providers/wallet/WalletProvider'

export default function RootLayout({ children }: { children: React.ReactNode })
{
  return (
    <html lang="en">
      <body>
        <WalletProvider>{children}</WalletProvider>
      </body>
    </html>
  )
}

```

SIWE NONCE ROUTE

```
// apps/web/app/api/siwe/nonce/route.ts
import { NextResponse } from 'next/server'

export async function GET() {
  const nonce = crypto.randomUUID()
  return NextResponse.json({ nonce }, { headers: { 'Cache-Control': 'no-store' } })
}
```

PHASE 2 DELIVERABLES

1. Explorer pages. /explorer, /explorer/[hash], /explorer/search, /explorer/timeline
2. SDK client for Explorer. Correlation id forwarded
3. Preimage viewer behind session. Redacted preview, gated download
4. 429 messages and jittered retry for idempotent reads

EXPLORER SDK

```
// packages/api-clients/explorer/client.ts
export class ExplorerClient {
  constructor(private base = process.env.NEXT_PUBLIC_EXPLORER_BASE!) {}

  private async req(path: string, init?: RequestInit) {
    const r = await fetch(`${this.base}${path}`, {
      ...init,
      credentials: 'include',
      headers: {
        'x-correlation-id': getCid(),
        ...(init?.headers || {})
      }
    })
    if (r.status === 429) throw new Error('RATE_LIMIT')
    if (!r.ok) throw new Error(`HTTP_${r.status}`)
    return r.json()
  }
}
```

```

}

resolve(hash: string) { return this.req(`/v1/resolve/${hash}`) }
batch(id: string) { return this.req(`/v1/batch/${id}`) }
root(id: string) { return this.req(`/v1/root/${id}`) }
search(q: URLSearchParams) { return this.req(`/v1/search?${q}`) }
preimage(hash: string) { return this.req(`/v1/preimage/${hash}`) }
timeline(cid: string) { return this.req(`/v1/timeline?correlationId=${encodeURIComponent(cid)}`) }
}

```

EXPLORER PAGE STUB

```

// apps/web/app/explorer/[hash]/page.tsx
import { ExplorerClient } from '@api/explorer/client'

export default async function Page({ params }: { params: { hash: string } }) {
  const api = new ExplorerClient()
  const rec = await api.resolve(params.hash)
  return <ExplorerRecord record={rec} />
}

```

PHASE 3 DELIVERABLES

1. Admin pages. /admin, /admin/routes/[symbol]
2. Admin SDK with config_hash optimistic concurrency. Bulk CSV import
3. Conflict resolver modal

ADMIN SDK

```

// packages/api-clients/admin/client.ts
export async function upsertRoute(symbol: string, body: { config_hash: string
} & Record<string, unknown>) {
  const r = await fetch(`${process.env.NEXT_PUBLIC_API_BASE}/v1/route/${sy

```

```

mbol}`, {
  method: 'PUT',
  credentials: 'include',
  headers: {
    'content-type': 'application/json',
    'x-correlation-id': getCid()
  },
  body: JSON.stringify(body)
})
if (r.status === 409) return { conflict: await r.json() }
if (!r.ok) throw new Error('UPSERT_FAILED')
return r.json()
}

```

CSV BULK IMPORT HANDLER

```

// apps/web/app/admin/_components/BulkImport.tsx
export function BulkImport() {
  async function onDrop(file: File) {
    const text = await file.text()
    const res = await fetch(`${process.env.NEXT_PUBLIC_API_BASE}/v1/route/
bulk`, {
      method: 'POST',
      credentials: 'include',
      headers: {
        'content-type': 'text/csv',
        'x-correlation-id': getCid()
      },
      body: text
    })
    // Show per-row success and export of failures
  }
  return <DropZone onFile={onDrop} />

```

```
}
```

PHASE 4 DELIVERABLES

1. Policy module behind feature flag. Read policy. Issue short-lived EIP-712 claim. Optional verify
2. UI surfaces require_fresh_ace and reasons

FLAGS

```
// apps/web/src/lib/flags.ts
export const FLAGS = {
  admin: process.env.NEXT_PUBLIC_FLAG_ADMIN === 'true',
  policy: process.env.NEXT_PUBLIC_FLAG_POLICY === 'true',
  timeline: process.env.NEXT_PUBLIC_FLAG_TIMELINE === 'true',
  preimage: process.env.NEXT_PUBLIC_FLAG_PREIMAGE === 'true'
}
```

MIDDLEWARE, CORRELATION

```
// apps/web/middleware.ts
import { NextResponse } from 'next/server'
import { randomUUID } from 'crypto'

export function middleware(req: Request) {
  const res = NextResponse.next()
  const cid = req.headers.get('x-correlation-id') || randomUUID()
  res.headers.set('x-correlation-id', cid)
  return res
}
```

SECURITY HEADERS

```
// apps/web/next.config.mjs
export default {
  async headers() {
    return [{
      source: '/(.*)',
      headers: [
        { key: 'Strict-Transport-Security', value: 'max-age=31536000; includeSubDomains' },
        { key: 'X-Content-Type-Options', value: 'nosniff' },
        { key: 'X-Frame-Options', value: 'DENY' },
        { key: 'Referrer-Policy', value: 'strict-origin-when-cross-origin' },
        { key: 'Permissions-Policy', value: 'geolocation=(), microphone=(), camera=()' },
        { key: 'Content-Security-Policy',
          value: "default-src 'self'; script-src 'self' 'wasm-unsafe-eval' 'nonce-<nonce>'; style-src 'self' 'unsafe-inline'; img-src 'self' data: https:; connect-src 'self' https://api.example.com https://explorer.example.com; frame-ancestors 'none'; base-uri 'self';" }
      ]
    }]
  }
}
```

CI AND GATES

```
# .github/workflows/web-ci.yml
name: web-ci
on: [push, pull_request]
jobs:
  build-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: pnpm/action-setup@v4
```

with:

version: 9

- run: pnpm install --frozen-lockfile
- run: pnpm -C apps/web typecheck
- run: pnpm -C apps/web test
- run: pnpm -C apps/web build
- name: sbom
run: npx @cyclonedx/cyclonedx-npm --output-file sbom.xml --json
- name: audit
run: pnpm audit --prod --audit-level=moderate
- name: bundle-size
run: npx bundlesize

ENV EXAMPLE

```
NEXT_PUBLIC_API_BASE=https://api.example.com
NEXT_PUBLIC_EXPLORER_BASE=https://explorer.example.com
NEXT_PUBLIC_CHAIN_ID=11155111
NEXT_PUBLIC_FLAG_ADMIN=true
NEXT_PUBLIC_FLAG_POLICY=false
NEXT_PUBLIC_FLAG_TIMELINE=true
NEXT_PUBLIC_FLAG_PREIMAGE=true
```

Section C - Diagrams and reviewer checklists

GLOBAL REFERENCES

MVP GUI zip

<https://drive.google.com/file/d/1o9wTCQLYKGDYyUWM90zIEhLWVMKJJp5/view>

Public site zip

<https://drive.google.com/file/d/1CRpYiPK8o0di7cf2ZkoNZFBpfxY1P5ys/view>

1. System context

flowchart LR

Client[User Browser]

CDN[CDN and Edge]

Web[Next.js App Router]

Auth[SIWE Auth API]

Explorer[Explorer API]

Admin[Admin API]

Policy[Token Policy API]

Sentry[Sentry]

Metrics[Web Vitals]

DB[(Mongo)]

Redis[(Redis)]

Client → CDN → Web

Web → |x-correlation-id| Explorer

Web → |x-correlation-id| Admin

Web → |SIWE cookies| Auth

Web → Policy

Web → Sentry

Web → Metrics

Admin → DB

Auth → Redis

Checklist

Correlation id on API arrows. SIWE cookies only Web to Auth. No client secrets.
Names and domains match

1. SIWE sequence

```
sequenceDiagram
    participant U as User
    participant W as Web App
    participant A as Auth API
    U->>W: Connect wallet
    W->>A: GET /siwe/nonce
    A->>W: nonce
    W->>U: Sign EIP-4361
    U->>W: signature
    W->>A: POST /siwe/verify
    A->>W: Set session cookies
    W->>U: Session badge shows
    W->>A: POST /auth/refresh
    A->>W: Rotate cookies
    U->>W: Logout
    W->>A: POST /auth/logout
    A->>W: Clear cookies
```

Checklist

Error copy for NONCE_INVALID and SIGNATURE_INVALID. HTTP-only cookies.
CSRF header on writes. Revoke covered

1. Explorer resolve, batch, root, preimage, timeline

```
sequenceDiagram
    participant U as User
```

participant W as Web App
participant E as Explorer API

U→>W: Paste hash or search keys
alt Resolve by hash
 W→>E: GET /v1/resolve/{hash} + x-correlation-id
 E→>W: Leaf, Interval, ReFIN L2, L1 Anchor
else Fetch by batch or root
 W→>E: GET /v1/batch/{batch_id}
 E→>W: Batch with leaves and root
 W→>E: GET /v1/root/{root}
 E→>W: Root summary with anchors
end

W→>E: POST /v1/verify/inclusion
E→>W: Inclusion true or false

opt Preimage protected
 W→>E: GET /v1/preimage/{hash}
 E→>W: 200 redacted JSON or 401 when no session
end

opt Timeline protected
 W→>E: GET /v1/timeline?correlationId=...
 E→>W: Events for risk, execution, audit
end

Note over W,E: On 429 retry with jitter for idempotent reads. Never retry writes

Checklist

Inputs include hash, batch_id, root, correlationId, clientOrderId. Preimage and timeline show as protected. Correlation id on each call. Retry note for 429. Clean 404 and pending anchor copy covered in UI

1. Admin optimistic concurrency

```
sequenceDiagram
    participant UI as Admin UI
    participant API as Admin API
    UI->>API: GET /v1/route/XYZ
    API->>UI: config + config_hash=abc
    UI->>API: PUT with config_hash=abc
    API->>UI: 409 with latest config_hash=def and diff
    UI->>UI: Show diff modal
    UI->>API: PUT with config_hash=def
    API->>UI: 200 OK
```

Checklist

Diff modal shows server versus local. 409 flow tested end to end. Bulk CSV covered elsewhere. RBAC on writes

1. Feature flag gating

```
flowchart TD
    User --> Web
    Web --> Flags["Flags{Policy flag on}"]
    Flags -- yes --> PolicyUI["PolicyUI[Policy screens]"]
    Flags -- no --> Placeholder["Placeholder[Coming soon]"]
```

Checklist

Flags for Policy, Timeline, Preimage, Admin. Flags loaded from env or edge config. Badges show network and flag state. No dead routes with flags off

1. App routing map

```
flowchart LR
    Root[app]
    Root --> Landing["Landing[app/landing]"]
```

```
Root → Auth[app/auth]
Root → Explorer[app/explorer]
Explorer → Resolve["app/explorer/[hash]"]
Explorer → Search[app/explorer/search]
Explorer → Timeline[app/explorer/timeline]
Root → Admin[app/admin]
Admin → Routes["app/admin/routes/[symbol]"]
```

Routes match file tree. Admin and Timeline protected. Landing isolated. 404 and 403 states defined

1. Security headers and middleware

flowchart LR

```
Req[Request] → MW[Middleware sets x-correlation-id]
MW → Route[Route handlers]
Route → Res[Response]
Res → H[HSTS, CSP, X-Frame-Options, Referrer-Policy, Permissions-Policy]
H → Browser
```

Checklist

CSP with nonce. SRI required. frame-ancestors none. No client secrets

1. Route and policy data model

```
classDiagram
class Route {
    symbol string
    route string
    driver_hint string
    enabled boolean
    config_hash string
    params JSON
    updated_at datetime
    asset_class string
}
```

```

venue string
slippage_bps int
mev_protect boolean
max_gas_gwei int
token_policy_required boolean
token_policy_ref string
}
class TokenPolicy {
  token string
  require_fresh_ace boolean
  reasons JSON
  claim_ttl_s int
}
Route → TokenPolicy : token_policy_ref

```

Checklist

All Admin fields present. Derived defaults marked. Validation hints correct.
 config_hash in every write

1. CI and release flow

flowchart TD

PR[Pull request] → Checks[Typecheck, Unit, E2E smoke, SBOM, SCA, Bundl
 esize]

Checks → Preview[Preview deploy]

Preview → DAST[DAST on authless endpoints]

Checks → Gate{All green}

Gate -- yes → Merge[Merge to main]

Gate -- no → Fix[Push fixes]

Merge → Release[Tag and notes]

Release → Prod[Deploy]

Checklist

SBOM produced. Critical CVEs fail builds. Bundlesize thresholds per route. Axe on preview. Notes from Conventional Commits

1. Auth state machine

```
stateDiagram-v2
    [*] → SignedOut
    SignedOut → Signing : get nonce
    Signing → SignedIn : verify ok
    Signing → SignedOut : signature invalid
    SignedIn → Refreshing : refresh
    Refreshing → SignedIn : rotated
    SignedIn → SignedOut : logout or revoke
```

Checklist

Refresh rotation tested. Revoke clears sessions. Error copy mapped. Telemetry on transitions

1. Observability wiring

```
sequenceDiagram
    participant U as User
    participant W as Web App
    participant S as Sentry
    participant M as Metrics
    participant API as Backend
    U->>W: Action
    W->>M: Web Vitals with cid
    W->>S: Breadcrumbs and error with cid
    W->>API: Request with cid
    API->>W: Response with cid
    W->>U: UI update
```

Checklist

cid in middleware. cid forwarded to APIs and vendors. Dashboards for p95 resolve and search. Alerts on 401 churn, 429 spikes, CSP violations. Logs include correlationId, route, and config_hash on all Explorer/Admin requests.

1. i18n pipeline

flowchart LR

Source[Copy strings] → Keys[ICU message keys]

Keys → Extractor[i18n extractor]

Extractor → Locales[en, ar, fr]

Locales → Pseudo[Pseudo locale]

Pseudo → UI[UI run]

Locales → Build[Build artifacts]

Checklist

ICU keys for all user copy. Pseudo run passes. RTL checks pass. Units localized

1. Prelaunch checklist

flowchart TD

Start[Prelaunch] → PenTest[Pen test complete]

PenTest → Cookies[Cookie consent verified]

Cookies → CSP[CSP and SRI verified]

CSP → Limits[Protected routes rate limited]

Limits → Legal[Legal copy present]

Legal → Owners[CODEOWNERS and on call updated]

Owners → Done[Go live]

Checklist

Preview and staging pen tested. Cookie consent stored. CSP report-only cleared then enforce. Domain allow-lists active

Section D - Testing strategy and specs

UNIT AND INTEGRATION

Auth

- nonce success and invalid nonce
- verify success and SIGNATURE_INVALID
- refresh rotation
- logout clears cookies

Explorer

- resolve 200 and 404
- inclusion verify true and false
- preimage 401 without session
- search by correlationId and clientOrderId
- batch and root reads

Admin

- list with filters
- read default shows derived true banner
- upsert success
- upsert conflict 409 returns diff modal payload
- bulk CSV validation and failure export

Policy

- GET policy shape
- issue claim payload and TTL honored
- verify optional path behind flag

E2E WITH PLAYWRIGHT

- SIWE onboarding and session badge
- Explorer resolve known hash
- [Phase 3] Explorer search renders timeline
- [Phase 3] Preimage gated by session
- [Phase 4] Admin edit with 409 conflict flow
- [Phase 4] Policy claim issuance and verify flows

ACCESSIBILITY AND VISUAL

- Axe scans on all routes
- Keyboard-only traversal, Enter opens table rows
- Reader landmarks present
- Percy snapshots for Landing, ExplorerRecord, Admin list, Admin editor

Section E - Performance Plan and Budget

BUDGETS ON MID DEVICE

- LCP 2.5 s or better
- CLS 0.1 or lower
- TBT 200 ms or lower

TUNING CHECKLIST

- Route-level code split in App Router
- next/image with AVIF and WebP
- Immutable cache for hashed assets
- Stale-while-revalidate for GET APIs
- Avoid heavy client proofs. Use server Explorer endpoints

Section F - Security and Compliance

HEADERS AND POLICIES

- CSP strict with nonces
- HSTS
- X-Content-Type-Options nosniff
- X-Frame-Options DENY
- Referrer-Policy strict-origin-when-cross-origin
- Minimal Permissions-Policy
- SRI on third-party scripts

DEPENDENCY HYGIENE

- SBOM via CycloneDX in CI
- Builds fail on critical CVEs
- Lockfile pinning

AUTH INTEGRITY

- SIWE with CSRF for writes
- Refresh rotation
- Device-wide revoke

PRIVACY

- No PII in client logs
- Preimage gated. Redacted preview. Rate limited

Section G - Observability and ops

CLIENT TELEMETRY

- Web-Vitals and custom timings
- JS errors and stack traces
- Action breadcrumbs with redaction

CORRELATION

- Generate correlationId per session or page
- Forward x-correlation-id on every request
- Dashboards for p95 resolve and search, error rate by endpoint, anchor fetch errors

ALERTS

- Auth failure spikes
- 429 spikes
- CSP violation spikes
- Budget regressions

RUNBOOK

- On incident, flip flags to isolate Admin or Policy
- Revert by BUILD_SHA
- Force logout on auth incidents
- Clear caches after Admin config fixes

Section H - Success rate, risks, scorecard

BASELINE SUCCESS RATE

- 75 to 85 percent for first push with all gates green
- 90 percent plus with levers below

TOP RISKS AND MITIGATIONS

- Unfrozen contracts. Freeze OpenAPI. Tag SDKs. Block breaking changes behind previews
- SIWE edge cases. CSRF, refresh rotation, revoke all. Negative tests for NONCE_INVALID and SIGNATURE_INVALID
- Rate limits. Surface 429 with backoff on reads. No write retries. Negotiate Redis burst values on staging
- CSP breakage. Nonces, SRI, report-only burn-in, then enforce
- Performance drift. Route budgets, bundlesize gates, Web-Vitals dashboards
- RBAC drift. Central claims and route guards. E2E for 403 paths
- Preimage privacy. Server redaction. Gated download. No PII in logs

WEEKLY SCORECARD

- Contracts frozen and SDKs generated
- All protected routes enforce SIWE and RBAC
- E2E green for Auth, Explorer, Admin, Policy paths
- Web-Vitals within budget on mid device
- SBOM shows zero critical CVEs
- CSP and SRI verified in live preview
- No secrets in client bundles

Section I - Repo structure, ownership, and storage

REPO TREE TARGET

```
apps/  
  web/  
    app/  
      landing/  
      explorer/  
        search/  
        timeline/  
        [hash]/  
      admin/  
        routes/  
          [symbol]/  
      auth/  
      api/  
        siwe/  
          nonce/  
          verify/  
        auth/  
          refresh/  
          logout/  
      public/  
      src/  
        lib/  
        styles/  
      middleware.ts  
      next.config.mjs  
      package.json  
      tsconfig.json  
      packages/
```

```
ui/  
  tokens.ts  
  components/  
api-clients/  
  explorer/  
  admin/  
  auth/  
config/
```

OWNERSHIP

- CODEOWNERS per path
- apps/web owned by FE lead
- packages/ui owned by design systems
- packages/api-clients owned by platform

DOCS AND DIAGRAMS

- docs/diagrams with .mmd files
- exports folder with PNGs for Notion where needed

Section J - Security gaps and risk hardening

1. CSP nonce and dynamic headers

- Risk. The sample CSP shows a placeholder nonce. A static nonce defeats CSP protections.
- Fix. Generate a per-response nonce. Inject into script tags. Use a middleware or an instrumentation hook to attach the nonce header per request. Add object-src 'none'. frame-ancestors 'none'. base-uri self. form-action self. upgrade-insecure-requests. trusted-types default if feasible.

1. CORS and credentialed fetch

- Risk. Cookies will fail or be overexposed if CORS is loose.
- Fix. API must send Access-Control-Allow-Origin with the exact app origin. Never wildcard with credentials. Send Access-Control-Allow-Credentials true. Allow only needed headers like x-correlation-id and x-csrf. Preflights must be explicit.

1. SIWE replay and fixation

- Risk. Nonce reuse or long nonce TTL allows replay. Session fixation at verify time.
- Fix. Single use nonces. Short nonce TTL, under 5 minutes. Bind nonce to address, domain, and chainId. Regenerate server session id on verify. Expire any prior session for that device at login. Enforce EIP-4361 fields, domain allow-list, uri match, and max clock skew. Support EIP-1271 for contract wallets.

1. Refresh rotation and reuse detection

- Risk. Stolen refresh cookie stays valid during rotation.
- Fix. Rotate on every refresh. Store family ids. Detect reuse. Revoke the family and force logout on all devices.

1. CSRF coverage

- Risk. Verify or logout endpoints without CSRF allow cross-site triggers.
- Fix. Double submit token for any state-changing call. SameSite Strict for session cookies when possible. If different sites are needed, set SameSite None and Secure.

1. Admin optimistic concurrency details

- Risk. config_hash churn from non-canonical JSON. Conflicts spam users.
- Fix. Canonicalize payloads before hashing. Stable key order. Trim whitespace. Exclude updated_at from hash. Consider ETag and If-Match as a server fallback.

1. Bulk CSV import safety

- Risk. CSV formula injection. Oversized files. Content sniffing.
- Fix. Cap file size and row count. Reject non text/csv. Neutralize cells starting with = + - @. Validate per row. Return a failure CSV with escaped content.

1. Preimage handling

- Risk. Sensitive fields cached or logged.
- Fix. Redact on the server. Send Cache-Control no-store. Never store preimage in client state beyond render. Block copy of full payload unless user confirms. Log only codes and sizes, not content.

1. Headers hardening set

- Risk. Missing modern cross-origin protections.
- Fix. Add Cross-Origin-Opener-Policy same-origin. Cross-Origin-Resource-Policy same-site. Cross-Origin-Embedder-Policy credentialless if compatible. X-DNS-Prefetch-Control off. X-Permitted-Cross-Domain-Policies none.

1. Dependency and supply chain

- Risk. npm audit alone misses advisories. Actions not pinned.
- Fix. Add OSV-Scanner in CI. Pin GitHub Actions by commit SHA. Add provenance and SLSA attestations. Renovate for weekly updates. Use

overrides to pin risky transitive packages. Fail build on critical CVEs. Review wallet libraries and WalletConnect versions.

1. Secrets and env handling

- Risk. Public env leaks. Unscoped preview secrets.
- Fix. Audit NEXT_PUBLIC values. No tokens in public env. Scope preview env to least privilege. Block secrets on PRs from forks. Use OIDC to cloud. No long-lived deploy keys.

1. Rate limits and client retries

- Risk. Thundering herds on 429. Visible timing side channels.
- Fix. Exponential backoff with jitter on idempotent reads. Cap retries. Add client-side request de-dup with AbortController. Do not retry writes. Randomize UI retry timers.

1. Error messages and information leakage

- Risk. Stack traces or server enums leak.
- Fix. Map server errors to short user copy. Include correlationId. Hide raw messages in the UI. Keep details in Sentry only.

1. Cookie model across origins

- Risk. SameSite breaks if API and app are on different sites. Or cookies become third-party.
- Fix. Prefer a shared site and subdomain model. If not possible, set SameSite None and Secure. Test Safari ITP. Mirror staging to production cookies and domains.

1. Wallet and chain prompts

- Risk. Wrong chain signatures. Phishing via unknown RPC.
- Fix. Hard gate on chainId. Show clear network badge. Block unknown RPC endpoints. Disable auto-connect by default. Require user intent to connect.

1. Finality and reorg UX

- Risk. Users assume final states too early.

- Fix. Show confirmation counts for L2 and L1. Label pending anchors. Document reorg edge cases in copy and tooltips.

1. Logging and privacy

- Risk. Addresses and hashes treated as PII in some regions.
- Fix. Redact or hash user addresses in telemetry if required by policy. Document retention and purge windows. Turn off session replay by default.

1. Build artifacts and source maps

- Risk. Public source maps leak internals.
- Fix. Upload source maps to Sentry. Do not serve them publicly. Strip console output in prod bundles.

1. Testing depth

- Risk. Gaps on negative paths.
- Fix. Add tests for nonce reuse, refresh reuse detection, CSV poison rows, 429 retry stops after cap, preimage no-store caching, redirect allow-list, and ETag collisions.

1. Caching rules

- Risk. CDN caches protected responses or preimages.
- Fix. Cache-Control no-store on protected GETs. Set Vary on Authorization and Cookie. Use immutable max-age only for hashed static assets.

Nice to have, high return

- Fetch Metadata protection. Enforce Sec-Fetch-* checks on APIs. Drop cross-site requests that do not match your policy.
- Trusted Types. Add a default policy to block DOM XSS from dynamic HTML.
- Content Security Policy reports. Run report-only for a full preview cycle. Store reports. Fix all violations before enforce.
- Client feature flags fail-safe. Default flags to false. Make Admin and Policy opt-in per env.

- Multi-tab session sync. Use BroadcastChannel to broadcast logout and revoke to all tabs.
- CSV sandbox. Process large CSVs in a Web Worker to keep the UI responsive.

Concrete changes by phase

Phase 0

- Add dynamic CSP nonce model and complete header set.
- Pin Actions by SHA. Add OSV-Scanner and secret scan. Add Renovate. Add OIDC for deploy.
- Enforce CODEOWNERS. Block unreviewed merges.

Phase 1

- Bind nonce to address and domain. Single-use storage. Short TTL. Session id rotation. CSRF on all writes. Redirect allow-list.
- Add refresh reuse detection. Device revoke endpoint. Add E2E negative tests.

Phase 2

- No-store on preimage and timeline. Redaction verified with a unit test using a fixture. Retry with jitter on reads. Request de-dup.
- Finality UI. Pending anchor copy. Confirmation counts.

Phase 3

- Canonical JSON hash for config_hash. ETag and If-Match server fallback. CSV sanitizer and limits. Formula neutralization. Return failure CSV.

Phase 4

- Sign and verify PolicyClaim with strict TTL. Show require_fresh_ace reason codes. Add deny-by-default flag state.

Checks you can run now

- Threat model review for SIWE, Admin writes, Preimage download.
- CSP report-only in preview. Triage reports daily for one cycle.
- Safari ITP session test on staging.

- CSV injection test with =cmd|'/C calc' style formulas and long cells.
- Refresh reuse simulation. Confirm family revoke works.

Remaining deltas to watch

- Freeze OpenAPI and tag generated SDKs before Phase 2.
- Confirm staging matches production domains, cookies, headers.
- Decide subdomain vs cross-site cookie model and test Safari ITP.

Section K - Final notes for the receiving engineer

- Zips match the earlier audit
- MVP widgets fill Explorer and Admin features with light refactor to Next.js patterns
- Landing moves to /landing under a shared UI shell
- Replace direct fetch calls with generated clients
- Keep Preimage and Timeline behind SIWE
- Keep Policy behind flag until back end reaches Phase 4
- Follow scorecard weekly
- Treat diagrams as living source of truth in repo and Notion