

Original Authored Invariants And Primitives

A high-confidence inventory of the repo-native contracts, proof carriers, settlement units, authorship rails, governance artifacts, and world-runtime manifests that are clearly authored inside this stack.

10 Foundational invariants	19 Primitive clusters	37 Typed identifiers in appendix
-------------------------------	--------------------------	-------------------------------------

Method note: no local repo can prove that nobody else on Earth has ever named something similarly. This atlas therefore uses a stricter and testable bar instead: an item only counts if it is repo-native in naming, defined in code or schemas, repeated across surfaces, and enforced or narrated as a contract.

- Included: only constructs with a name, a contract, and implementation evidence.
- Excluded: generic framework patterns, vendor names, plain crypto nouns, and purely presentational components.
- Scope: this is a full high-confidence inventory, plus an appendix of the typed identifiers discovered across the repo.

Workspace. /Users/bjklock/Kai-Turah/receiz

Output. /Users/bjklock/Kai-Turah/receiz/output/pdf/receiz-authored-invariants-and-primitives-atlas.pdf

SECTION I

Foundational Invariants

These are the repo-wide rules that determine what counts as truth, what counts as delivery, and what is never allowed to silently redefine the system's authority model.

INVARIANT

Original Bytes Are Authority

Does. Receiz treats the exact original file bytes as the source of verification truth. If the bytes match, the artifact verifies; if the bytes change, verification fails.

Why it counts. This is not a slogan in the repo. It is repeated in the README, formal standard text, the seal pipeline, and the verifier/runtime paths.

Evidence.

- README.md
- docs/receiz-standard-v1.md
- app/lib/documentSeal/receizeDocumentBytesCore.ts
- app/lib/receizArtifactVerifier.ts

INVARIANT

Published Manifest Is Historical Truth

Does. Accepted historical state is supposed to resolve from a published deterministic manifest, not from mutable client fetches or feed reconstruction.

Why it counts. The repo defines a hard historical model: canonical manifest first, then live append-only delta, then augmentation, then local experience state.

Evidence.

- README.md
- docs/verified-history-first-principles.md

INVARIANT

Settlement Authority Is Append-Only Replay

Does. Settlement authority comes from replayable integer custody fields plus deterministic rate math, not from whatever the UI currently displays.

Why it counts. The economy pages, shared standards copy, and conversion helpers all lock the same model: posted tuple first, replay second, presentation last.

Evidence.

- README.md
- app/lib/copyStandards.ts
- app/economy/page.tsx
- app/lib/wallet/amounts.ts

INVARIANT

Trust Claims Must Map To Code And Evidence

Does. Public trust statements are expected to be backed by source paths, requirement IDs, and evidence artifacts rather than loose marketing language.

Why it counts. This repo operationalizes the rule through formal standards pages plus conformance routes that literally enumerate source needles and runtime checks.

Evidence.

- docs/receiz-standard-v1.md
- app/api/economy/conformance/route.ts
- app/api/issuance/conformance/route.ts
- app/api/world/conformance/route.ts

INVARIANT

Truthful First Paint Only

Does. Deterministic trust surfaces may first-paint real server truth, a real navigation snapshot, or nothing. They may not first-paint fake loaders, skeletons, or correction frames.

Why it counts. The repo treats this as a contract, not a stylistic preference. The standard names the banned patterns and the tests enforce them.

Evidence.

- README.md
- docs/deterministic-surfaces.md

INVARIANT

Truth Is Split Into Four Non-Interchangeable Layers

Does. Canonical truth, live delta, augmentation, and experience state are separate buckets with different authority. They are not allowed to blur together.

Why it counts. This separation appears in both the verified-history contract and the world/public-twin model, which is a strong signal that it is foundational to the stack.

Evidence.

- docs/verified-history-first-principles.md
- docs/virtual-identity-world.md
- app/lib/world/blueprint.ts

INVARIANT

Weaker Payloads Cannot Downgrade Stronger Truth

Does. A weaker or partial payload is not allowed to replace a stronger known canonical block. New transport convenience must not reduce established certainty.

Why it counts. This is one of the clearest repo-specific architecture rules because it appears as a named hard invariant and as a required question for any new feature.

Evidence.

- docs/verified-history-first-principles.md

INVARIANT

Assets Are Realizations, Not Rediscovery

Does. Once a deterministic block already knows the asset identity, surfaces should realize and reuse those bytes or variants instead of rediscovering them through weaker fetch paths.

Why it counts. The repo ties this directly to both media handling and route navigation. It is part of the same authoritative-core worldview as manifests and ledger replay.

Evidence.

- README.md
- docs/verified-history-first-principles.md

INVARIANT

PresenceBound Runs On One Canonical Realm

Does. PresenceBound WebAuthn verification is pinned to one production origin and one RP ID: ``https://receiz.com`` and ``receiz.com``.

Why it counts. The repo names this a hard requirement, implements constants for it, and handles realm mismatch as a first-class failure mode in authorship verification.

Evidence.

- docs/presencebound-canonical-realm.md
- app/lib/pbi/realm.ts
- app/api/pbi/authorship/verify/route.ts

INVARIANT

AI Interpretation Is Never Proof

Does. AI can rank, summarize, suggest, and augment, but it must never become evidence of ownership, payment, authorship, or consent.

Why it counts. The world stack repeats this as a guardrail because the public-twin layer is explicitly allowed to feel alive without crossing the boundary into fabricated authority.

Evidence.

- docs/virtual-identity-world.md
- app/lib/world/blueprint.ts

SECTION II

Repo-Native Primitive Clusters

These are the named building blocks that actually carry the stack: proof bundles, custody chains, settlement units, action canons, world manifests, runtime packs, and governance artifacts.

PRIMITIVE

Kai Pulse / Kai Klock

Does. A repo-native deterministic time lattice rooted in a genesis anchor. Its pulse index and rendered Kai Klock line are embedded into proof bundles, settlement flows, custody events, and UI timing.

Why it counts. This vocabulary is unmistakably stack-native: `kaiPulseEternal`, `kaiKlok`, pulse math, lattice constants, and repo-wide usage from sealing to world playback.

Evidence.

- app/utils/kai_pulse.ts
- public/standards/receiz.schema.v1.json
- app/lib/documentSeal/receizeDocumentBytesCore.ts

PRIMITIVE

Receiz Proof Bundle

Does. The canonical proof payload carried across sealed artifacts. It holds timestamps, verify coordinates, pulse markers, claim identifiers, basis hash, proof data, signatures, and optional sidecars.

Why it counts. This is the clearest authored core primitive in the repo: one named payload, one schema, one encoder/decoder, many carriers.

Evidence.

- public/standards/receiz.schema.v1.json
- app/utils/receizProofBundle.ts
- docs/receiz-standard-v1.md

PRIMITIVE

Trusted Verification Profile

Does. A hard-fail overlay on the proof bundle that requires anchor ID, Poseidon hash, real Groth16 envelope, artifact basis hash, and signatureV4 device-cert evidence.

Why it counts. It is not just a boolean trusted flag. The repo defines it as a stricter profile with its own schema and explicit device certificate type.

Evidence.

- public/standards/receiz.trusted-profile.v1.json
- public/standards/receiz.schema.v1.json

PRIMITIVE

Receiz Bundle Envelope

Does. A JSON envelope format that packages the original bytes and the proof bundle together as `application/vnd.receiz.bundle+json`.

Why it counts. This is a distinct authored carrier primitive rather than a generic zip or attachment wrapper. The envelope itself is versioned, typed, and validated.

Evidence.

- public/standards/receiz.bundle.schema.v1.json
- app/lib/documentSeal/bundleContainer.ts

PRIMITIVE

Real Groth16 Envelope

Does. A normalized `g16:` payload form that stores a real Groth16 proof plus public signals under `receiz.g16.real.v1`.

Why it counts. The repo does not stop at "there is a zk proof." It authors a transport shape, a schema, and a generation path for the proof envelope itself.

Evidence.

- public/standards/receiz.groth16.real.schema.v1.json
- app/lib/documentSeal/realGroth16Proof.ts

PRIMITIVE

Document Seal Multi-Carrier Pipeline

Does. A single sealing core that derives code, pulse, verify path, anchor, proof bundle, and carrier-specific embedding across PNG, PDF, trailer, SVG, JSON, and bundle outputs.

Why it counts. This is a strong authored primitive because the repo models “seal once, carry anywhere” as one contract instead of many disconnected exporters.

Evidence.

- `app/lib/documentSeal/receizeDocumentBytesCore.ts`
- `public/standards/receiz.document-seal.schema.v1.json`

PRIMITIVE

WireProof Attestation

Does. A payment-proof sidecar embedded inside the proof bundle with an anchor ID, claim hash, and verifier path under `wireproof.v1``.

Why it counts. The repo extends the core proof primitive into a payments-specific authored attestation instead of inventing a separate unrelated artifact family.

Evidence.

- `app/utis/receizProofBundle.ts`
- `public/standards/receiz.schema.v1.json`
- `app/api/issuance/conformance/route.ts`

PRIMITIVE

AssetSeal Claim

Does. A canonical origin-and-custody claim format that fixes genesis provenance and preserves or extends custody history instead of rewriting it.

Why it counts. The code explicitly canonicalizes it, guarantees a genesis custody event, and renders it as a first-class variant of the Receiz artifact surface.

Evidence.

- `app/lib/assetseal/claim.ts`
- `app/api/assetseal/mint/route.ts`
- `app/api/receiz/route.tsx`

PRIMITIVE

AssetSeal Custody Event Chain

Does. An append-only hash-chained custody event payload tied to exact parent coordinates, PBI receipt hash, Kai pulse, and previous event hash.

Why it counts. The repo treats custody continuity as its own authored integrity primitive, not just a list of notes in a table.

Evidence.

- `app/api/assetseal/custody/append/route.ts`
- `app/api/issuance/conformance/route.ts`

PRIMITIVE

Settlement Ledger Unit (SLU)

Does. The canonical non-USD custody unit used to keep ledger posting, replay, and offline note settlement deterministic while preserving ``phi*`` protocol field compatibility.

Why it counts. This repo explicitly names the unit, explains the compatibility strategy, and threads it through economy pages, account state, routes, and note flows.

Evidence.

- `app/lib/copyStandards.ts`
- `app/economy/page.tsx`
- `app/economy/ledger/page.tsx`

PRIMITIVE

Portable Account State Family

Does. A portable snapshot root ``receiz.account.state.v2`` that carries profile data plus embedded ``receiz.action.ledger.v1`` and ``receiz.calendar.v1`` subdocuments.

Why it counts. This is a repo-authored portability primitive for moving state between environments without reducing the determinism of ledger and calendar payloads.

Evidence.

- `app/lib/ledger/portableState.ts`

PRIMITIVE

Action Ledger Hash Chain

Does. A JCS-canonicalized append-only ledger whose entry hashes are built from versioned material including prev-hash, route, deterministic key, and canonical payload text.

Why it counts. It is a formal chain primitive, not just an events table. The hashing material and zero-hash rules are spelled out in code.

Evidence.

- app/lib/ledger/actionLedger.ts
- app/lib/ledger/portableState.ts

PRIMITIVE

PresenceBound Authorship Action Canonons

Does. Line-based canonical action texts hashed before passkey challenge and verification, binding exact artifact coordinates or identity-key actions to a PresenceBound commit.

Why it counts. This is unusually stack-specific: authorship attach, invite accept, identity-key enrollment, and login each have distinct canonical strings and hashes.

Evidence.

- app/lib/pbi/action.ts
- app/api/pbi/authorship/challenge/route.ts
- app/api/pbi/authorship/verify/route.ts

PRIMITIVE

Kairos Note Packet

Does. A sealed offline note packet embedded into exported note artifacts so value can be transferred out of band and later claimed back into the deterministic ledger.

Why it counts. The repo gives offline settlement its own named packet format and embeds it both as metadata and as PNG text chunks.

Evidence.

- app/lib/wallet/kairosNoteArtifact.ts
- app/api/wallet/notes/claim/route.ts
- app/api/connect/README.md

PRIMITIVE

Public Twin Mind Bundle

Does. A signed export artifact for owner identity, agent profile, avatar state, world presence, showcase corpus, memory, and voice sample under `receiz.world.public-twin.mind`.

Why it counts. This is a bespoke world-state artifact with PNG embedding, payload hashing, and signature material, not a generic profile export.

Evidence.

- `app/lib/world/twinMindTransport.ts`
- `app/lib/world/twinMind.ts`

PRIMITIVE

Avatar Identity Pack

Does. A versioned manifest for avatar source media, identity assets, and geometry capture metadata under `receiz.world.avatar.identity-pack`.

Why it counts. The repo treats avatar embodiment as a continuity-bearing manifest rather than a single mutable image field.

Evidence.

- `app/lib/world/avatarStudio.ts`

PRIMITIVE

Voice Clone Enrollment Manifest

Does. A versioned pack for voice sample, calibration assets, geometry calibration, transcript, and profile ID under `receiz.world.voice-clone.enrollment`.

Why it counts. It shows the stack's pattern clearly: even voice enrollment becomes a durable, typed manifest instead of an ad hoc blob.

Evidence.

- `app/lib/world/performanceStudio.ts`

PRIMITIVE

Twin Promotion And Promotion History

Does. Current and historical manifests for approved, rejected, or quarantined photoreal public-twin promotion, carrying evidence assets, quality gates, mastering state, and reference verification.

Why it counts. The repo does not model public twin realism as a loose flag. It models it as a manifest lineage with history and downgrade paths.

Evidence.

- app/lib/world/twinApproval.ts
- app/api/world/conformance/route.ts

PRIMITIVE

Owned World Runtime Manifest Family

Does. A local manifest protocol spanning `receiz.world.owned-model-pack`, `receiz.world.owned-engine-bundle`, `receiz.world.bundle-v1`, `receiz.world.resident-worker`, and `receiz.world.resident-cache`.

Why it counts. This family is distinctly authored: model packs, engine bundles, resident workers, GPU pinning, and render bundles all speak one stack-specific runtime language.

Evidence.

- models/world-owned/render/selfhosted-live-twin/manifest.json
- scripts/world_owned_twin_runtime.mjs
- scripts/world_runpod_render_backend.mjs
- app/lib/world/ownedTwinLatents.ts
- scripts/world_resident_twin_pack.mjs

PRIMITIVE

Governance Artifact Manifest

Does. A signed public manifest that hashes required governance artifacts, anchors them to registers, and verifies them through Ed25519 keyring material.

Why it counts. This is not generic docs housekeeping. The repo turns governance publication itself into a cryptographically bound artifact family.

Evidence.

- public/governance/artifact-manifest.v1.json
- docs/governance/governance-artifact-signature-standard.v1.md

PRIMITIVE

Conformance Artifact Family

Does. Repo-native report, attestation, and history-entry families for settlement, issuance, verification, identity, interoperability, world, and release integrity.

Why it counts. This stack repeatedly turns claims into machine-checkable conformance artifacts instead of leaving them as prose-only policy.

- Evidence.
- app/api/economy/conformance/route.ts
 - app/api/issuance/conformance/route.ts
 - app/api/world/conformance/route.ts
 - public/governance/release-attestation.schema.v1.json

Primitive pattern summary: this repo keeps authoring the same way across domains. It names a carrier, gives it a schema or canonical form, binds it to deterministic fields, and then makes higher-order routes answer to that primitive instead of replacing it.

SECTION III

Full Identifier Appendix

A grouped appendix of the stack-native schema and kind identifiers surfaced during repo inspection. This is the exhaustive typed inventory behind the narrative clusters above.

Artifact, Authorship, And Settlement Carriers

Typed identifiers that carry or bind proof, custody, authorship, and offline value.

receiz.proof_bundle	Canonical proof payload embedded across Receiz carriers.
receiz.bundle.v1	Bundle envelope carrying original bytes plus proof bundle.
receiz.g16.real.v1	Decoded Groth16 proof envelope stored inside the proof bundle.
receiz.device.v1	Device certificate type used by the trusted verification profile.
wireproof.v1	Payment-proof attestation sidecar inside proof bundles.
assetseal.v1	Origin and custody claim flavor rendered through the Receiz artifact route.
assetseal.custody.v1	Hash-chained custody event payload with PBI receipt binding.
kairos_note_packet_v2	Offline note packet embedded into exported note artifacts.

Account, Portability, And Connect Shapes

Identifiers used to move durable account state or bootstrap connected login flows.

receiz.account.state.v2	Portable account snapshot root.
receiz.action.ledger.v1	Embedded append-only action ledger document.
receiz.calendar.v1	Embedded calendar document inside the portable snapshot.
receiz.login.bootstrap.v1	Username-scoped connect/OIDC bootstrap document.

Governance And Conformance Families

Identifiers that turn trust claims, releases, and domain-specific controls into typed attestable artifacts.

receiz-governance-artifact-manifest-v1	Signed governance artifact manifest schema marker.
receiz-release-attestation-v1	Release integrity attestation artifact.
receiz-conformance-report-v1	Shared conformance report shape.
receiz-conformance-attestation-v1	Base or economy-oriented conformance attestation.
receiz-conformance-history-entry-v1	Base or economy-oriented conformance history entry.
receiz-verification-conformance-attestation-v1	Verification-domain attestation artifact.
receiz-verification-conformance-history-entry-v1	Verification-domain history entry.
receiz-issuance-conformance-attestation-v1	Issuance-domain attestation artifact.
receiz-issuance-conformance-history-entry-v1	Issuance-domain history entry.
receiz-identity-conformance-attestation-v1	Identity-domain attestation artifact.
receiz-identity-conformance-history-entry-v1	Identity-domain history entry.
receiz-interopability-conformance-attestation-v1	Interoperability-domain attestation artifact.
receiz-interopability-conformance-history-entry-v1	Interoperability-domain history entry.
receiz-world-conformance-attestation-v1	World-stack attestation artifact.
receiz-world-conformance-history-entry-v1	World-stack history entry.

World Runtime Manifest Family

Typed manifests used to express avatar identity, voice, public twin state, promotion state, and owned-runtime execution state.

receiz.world.avatar.identity-pack	Versioned avatar source and geometry manifest.
receiz.world.voice-clone.enrollment	Versioned voice sample and calibration manifest.
receiz.world.public-twin.mind	Signed export bundle for public-twin identity and memory.
receiz.world.twin.promotion	Current photoreal twin promotion manifest.
receiz.world.twin.promotion.history	Historical promotion, rejection, or quarantine manifest.
receiz.world.bundle-v1	Render request bundle manifest used by the world render backend.
receiz.world.owned-model-pack	Local model-pack manifest for owned twin engines.
receiz.world.owned-engine-bundle	Per-request engine bundle manifest for owned runtime execution.
receiz.world.resident-worker	Resident worker telemetry and warm-state manifest.
receiz.world.resident-cache	Resident cache manifest for warmed identity, voice, and render state.

Bottom line: the stack’s authored distinctiveness does not come from one feature. It comes from one repeated worldview: bytes are authority, manifests carry historical truth, ledger replay carries settlement truth, and every higher-order surface is expected to stay accountable to those lower-level contracts.