Proof-Native Artifact System

OFFICIAL SYSTEM BRIEF

Receiz v95.0.0 Architecture Report

A repo-grounded executive and technical report for understanding what Receiz is, how it works, why it performs, and what must never be confused with generic app behavior.

APP v95.0.0

SDK @95.0.0

OPENAPI 2026-06-25.v2

5,153	1,121,482	90,076	10
TRACKED FILES	SOURCE LINES	TEST LINES	CONFORMANCE GATES

Receiz is not a generic app with proof features. It is a proof-native artifact system where sealed artifacts, identity records, ownership states, settlement states, public witness surfaces, developer SDKs, and verification routes are projections of implemented primitives.

The critical operating law is simple: use known verified truth immediately, then ask only for verified additions after the known head.

GENERATED 2026-06-27 / COMMIT D5F24B75D9E1 / DIRTY WORKING TREE: TRUE

What This Codebase Is

Receiz should be read from primitives first. Routes, APIs, database rows, SDK helpers, UI controls, and media players are mechanics beneath proof, identity, ownership, settlement, provenance, public witness, durable local truth, verified append, and Kai/Klok state.

Receiz is	Receiz is not
A proof-native artifact system.	A generic app with proof-flavored features.
A system where verified truth can be recomputed.	A dashboard whose truth depends on a live backend being available.
A durable local truth and append model.	A stale-cache web app that must rediscover old truth.
A Kai-coordinated proof object system with Chronos display.	A wall-clock feed that later decorates rows with Kai labels.
An SDK-backed developer platform beneath proof primitives.	An API-first system where SDK responses become authority.

PRIMITIVE ACCEPTANCE

- Proof objects carry deterministic identity, verifiable payload/proof material, provenance, inspection behavior, and durable state.
- Vault objects carry proof identity, media, ownership or custody state, provenance, transfer/inspect/share/action affordances, and public or private legibility.
- Settlement primitives carry proof-native value, reserve, certificate, note, funded event, purchase, or transfer finality.
- Identity primitives carry Receiz ID, PBI, deterministic identity, sealed identity file, account continuity, recovery, and proof-bearing login.

The burden of proof belongs on weaker layers. DB, session, API refresh, route cache, and SDK transport must prove they only project, index, sync, or append beneath stronger truth.

Truth Hierarchy

01	Receiz law and product invariants	Product law defines which layer is allowed to govern.
02	Sealed artifact truth and embedded proof	Portable artifacts must remain verifiable without believing the app.
03	Deterministic proof object state	Stable object identity, payload, provenance, and inspection behavior.
04	Verified local truth and durable proof memory	The admitted prefix is known truth, not cache.
05	Verified register block and append	New truth extends the known head without rewriting old truth.
06	Authenticated snapshot	Useful continuity projection when it cannot outrank proof.
07	Server append and global sync	Publication and discovery, not permission for known truth.
08	DB, session, UI state, external references	Mechanics only. Never proof authority.

Forbidden downgrade: a weaker layer may accelerate or decorate stronger truth, but it must not replace, erase, delay, rename, re-authorize, or rehydrate it before projection.

First Admission, Then Append Forever

The largest v92-v95 learning was that Receiz gets faster when it stops treating verified truth as mutable cache. The correct operation is not 'fetch less'. The correct operation is 'never rediscover known truth before using it'.

1 Admit

A node missing full proof memory may hydrate the missing verified prefix once.

2 Persist

The admitted prefix becomes durable proof memory and survives route/session/PWA framing.

3 Project

Open from known truth immediately on the next render.

4 Ask

Network and DB ask only for verified additions after the known head.

5 Merge

Additions merge by proof identity, Kai coordinate, and domain-specific invariants.

6 Repeat

The object becomes more complete without becoming heavier to open.

PRODUCTION PROOF

- Sports card histories that admitted full event memory opened instantly and kept that history after PWA restart.
- Wallet ledger ordering moved to Kai heads so older truth is not re-proven by wall-clock windows.
- Calendar points and account activity nodes are projections of the same proof objects, not independent competing truths.
- SDK proof registers package this law for external developers so they do not rebuild a mutable feed architecture around Receiz assets.

Implemented Surface Map

Surface	Primitive carried	Source of truth	Allowed lower layers
Profile and public proof	Public proof surface	Deterministic proof object / published snapshot	Session controls, owner actions, analytics after paint
Market	Public proof and settlement bridge	Bundled market proof object	Wallet ownership enrichment, trade panels, live additions
Wallet	Settlement primitive	Kai-ordered settlement ledger rows	Balance projection, public ledger APIs, Connect
Sports cards	Vault object and event proof memory	Card manifest, appends, event proofs	Bounded RPCs, indexes, economy projection
Calendar and account	Proof object projections	Admitted event/action proof objects	Route-specific visualization only
Identity	Identity primitive	Receiz ID, PBI, Receiz Key, Identity Record, Identity Seal	Session cookies, Connect delegation
SDK and OpenAPI	Developer convenience	Existing proof primitives	Typed clients, schemas, fixtures, docs

How to read every Receiz screen: identify the primitive, identify the strongest proof or admitted state, then verify the visible UI is only projecting or appending beneath it. If a surface makes known truth load again, the implementation has promoted a weaker layer.

Kai Klok Is The State Machine

Kai Klok is the proof object state machine. kaiPulseEternal / kai_upulse is the pulse unit carried by that state machine. A complete Receiz proof object carries Kai; without that living coordinate it is not complete offline-verifiable Receiz proof truth. Chronos is display and compatibility time projected beneath Kai.

Layer	Correct role	Forbidden role
Kai Klok	State machine for complete Receiz proof objects.	Vibe, branding, metadata, or frontend-only label.
kaiPulseEternal / kai_upulse	Pulse unit carried by the Kai Klok state machine; orders ledgers, appends, and known heads.	Optional tie-break label after Chronos already decided state.
Chronos	Human display, external API bridge, migration fallback.	Primary backend ordering for Kai-bearing proof objects.
DB created_at / provider IDs	Physical storage, lookup aid, import context, and debugging context.	Proof causality, settlement authority, or identity authority.

WHY THE KAI CHILD-OFFSET FIX MATTERED

- Every proof object is unique, but a DB trigger can still damage state if it overwrites app-supplied Kai child coordinates.
- The fix preserved app-supplied Kai upulses and enforced uniqueness at the ledger boundary instead of flattening children into the same coordinate.
- This removed unnecessary write pressure and stopped valid proof children from colliding inside a weaker projection layer.

Backend rule: Kai Klok is the state machine; kaiPulseEternal / kai_upulse is the unit; Chronos renders for humans. Never reverse that boundary.

Performance Model

The slow-query work did not move authority into the database. It removed broad rediscovery beneath already-settled proof surfaces. Indexes and RPCs are correct when they find verified additions faster without changing what the object is.

Pressure found	Fix pattern	No-regression reason
Owner-wide append scans	Bound by owner, sport, collectible, game, source date, or Kai head.	Same proof rows; smaller search path.
Event proof sort pressure	Created/source/attached collectible paging indexes and RPCs.	JSON proof payload remains source material.
Vault showcase upserts	Skip unchanged redundant writes and keep idempotent admission.	No unadmitted showcase is dropped.
Tournament closeout misses	Worker path and explicit settlement/refund RPCs.	GET routes stay light; settlement truth appends once.
Wallet old-history windows	Kai-aware cursor/head and durable ledger projection.	Old settlement truth remains present.

SCALE LAW

- Preserve the visible primitive and remove weak work beneath it.
- Known stronger truth paints first; background sync appends verified additions.
- Rich surfaces are allowed when driven by deterministic proof state instead of repeated rediscovery.
- A growing history should increase truth, not visible loading.

Sports Proof Economy

Sports is the clearest production proof of the model. Cards, appends, event proofs, tournaments, vault showcases, ownership appends, economy views, and live fields are all projections over proof truth.

Finding	Resolution
Some cards had instant full history while others hydrated slowly.	Admit missing card event memory once, persist it, then append new events forever.
MLB provider IDs changed for the same event.	Deduplicate credit through semantic event circumstances and state, not provider ID trust.
Sports Economy caused proof read pressure.	Bound raw proof reads through RPC/index paths while preserving proof payloads.
Main Event stayed locked after slate finality.	Worker route must reach closeout; underfilled tournament void/refund appends settlement truth.
Visible card append RPC pressure remained after transfer filtering.	Use known durable card memory first; skip broad visible append rediscovery when appends are already admitted.

User-visible result

Cards that know their play/event proof history open with that history already present. New live events append into the same memory and remain there.

System result

Broad DB proof rediscovery moves out of the visible path. The database finds additions; proof memory carries known truth.

Wallet, Calendar, Account

Wallet history, calendar points, and account activity nodes are not separate truth systems. They are projections of proof objects and ledger appends. When they follow Receiz law, the user sees old truth immediately and only new additions arrive later.

Domain	v95 public operating model	Must not regress
Wallet ledger	Kai-ordered settlement rows with known ledger heads.	No Chronos-first paging where Kai exists; no partial window replacing full history.
Calendar	Calendar points project admitted account/action/event proof objects.	No repeated full hydration after the node has admitted events.
Account brain nodes	Activity nodes project the same proof events as calendar/history.	No competing route-local copy of the same proof object.
Identity export	Receiz account state v3 carries portable proof domains.	Identity files are not convenience backups; they are file-carried proof memory.

The correct question for these surfaces is: what verified additions exist after my known ledger or proof head? The wrong question is: is my old ledger still true?

Developer Platform

The official SDK is a convenience layer beneath implemented Receiz primitives. It is designed so external developers can build Receiz-native apps without inventing a weaker authority model or mutable-feed history model.

@receiz/sdk	95.0.0	9	3
NPM PACKAGE	SDK VERSION	SDK DOCS	FIXTURES

Capability	Developer receives
Verification	Artifact verify, document seal/verify, manifest projection, schema validation.
Proof memory	Local proof register: admit once, persist, append verified additions after known head.
Identity	Receiz ID, Identity Record, Identity Seal, Receiz Key login/recovery helpers.
Wallet and Connect	Public ledger reads, Connect transfers, payment routes, delegated token flows.
Sports	Sports card manifests, event proof URLs, card/event projection helpers.
Webhooks	Signature verification and typed event schema validation.

SDK boundary: a helper can make the correct path easy, but it cannot outrank sealed artifacts, proof bundles, verified appends, ownership appends, or settlement ledger rows.

Governance And Release Gates

The repo encodes product law as contracts. Release readiness is not only typecheck/build success; it includes first-paint, primitive binding, visual evidence, conformance, security, chaos, settlement, identity, sports, world, and governance controls.

Gate family	Examples from package scripts
Release lock	release:lock, check_release_lock.mjs, release docs, service-worker version surfaces.
Primitive law	test:primitive-binding-contract, test:repo-only-reasoning-boundary, test:deterministic-surfaces.
First paint	test:page-first-paint, visual evidence contracts, deterministic route tests.
Conformance	settlement, issuance, verification, market, identity, interoperability, economy, sports, world, signal circuit.
Runtime safety	api-security, chaos-failures, performance-routes, Supabase advisor guard.
Attestation	release:attest and release:verify-attestation.

EVIDENCE BOUNDARY

- Repo contracts can prove local implementation law and prevent known regression classes.
- Production DB state, Vercel rollout, npm publication, Supabase schema cache, installed PWA uptake, sports feeds, and CDN media remain runtime evidence after deploy.
- Visual evidence outranks a claimed pass when it contradicts the primitive on screen.

Repository Evidence Inventory

These metrics are generated from tracked repository files at report build time. They are included to show the scale of the implemented system and the breadth of contracts protecting it.

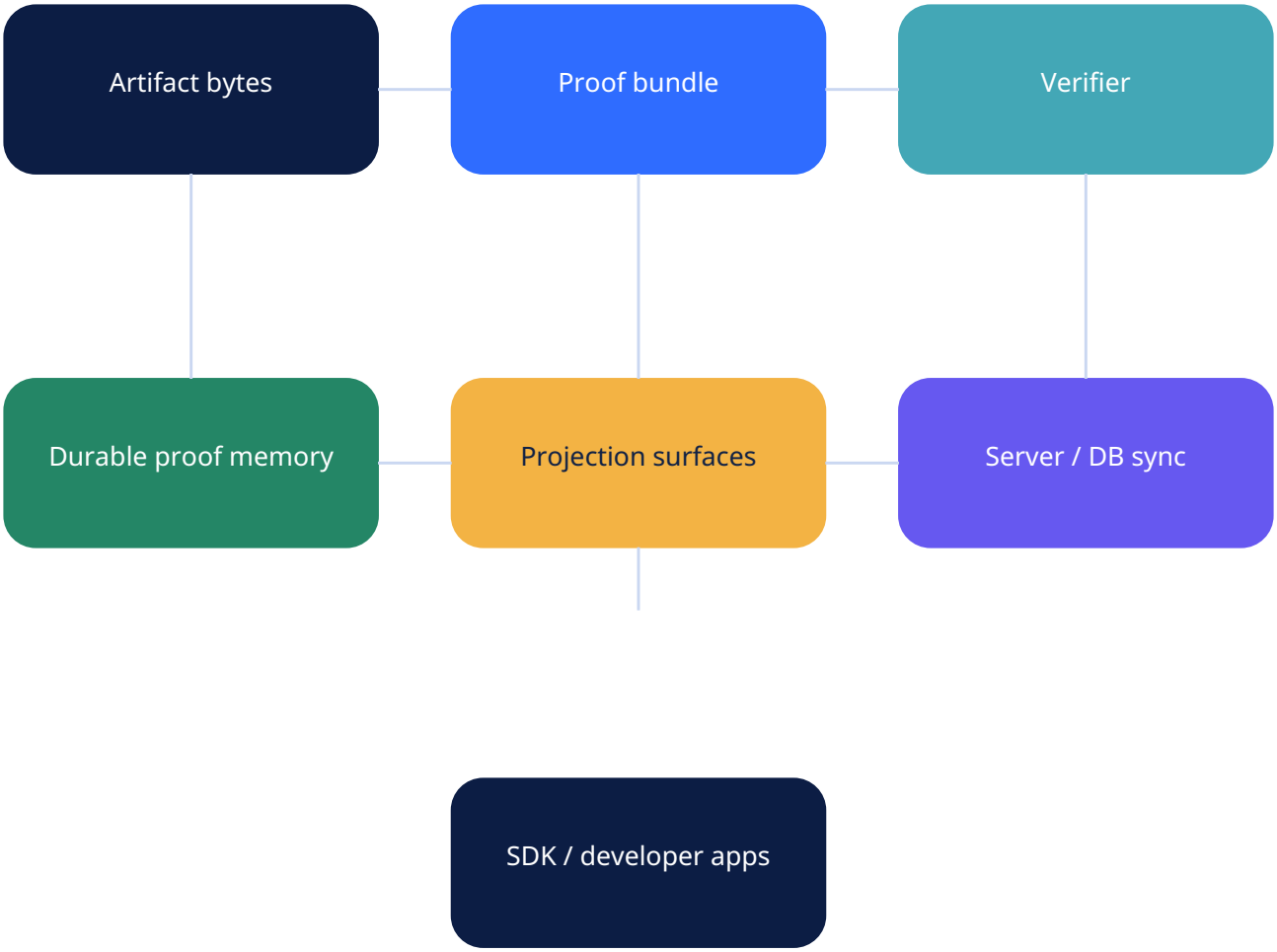
2,958	327	369	186	430
SOURCE FILES	DOCS FILES	TEST FILES	PAGE ROUTES	API ROUTES

Language	Files	Lines	Domain	Files	Lines
TypeScript	1,711	513,264	app/lib	756	270,433
React TypeScript	812	360,457	app/components	436	175,530
CSS	133	167,226	app/game	191	157,698
SQL	249	56,980	app/api	454	112,288
Node ESM	27	13,396	supabase	245	56,560
Python	10	6,408	app/u	28	53,928
JavaScript	6	2,601	scripts	99	30,102

Release corpus: 240 release documents, 11 public standard files, 25 SDK package files, and 10 conformance scripts.

System Flow

Receiz architecture becomes clear when weaker layers are placed beneath the proof object instead of above it. The app, database, APIs, SDK, and UI exist to project, inspect, sync, and append to proof truth.



The server can publish, mirror, index, discover, and return additions. It does not make already-held proof truth true again.

v92 Through v95 Learning Arc

Release arc	Learning locked
v92	Performance work exposed the difference between slow projection and missing proof truth.
v93.0	Sports Proof Economy made appends, event proofs, and economy reads explicit proof surfaces.
v93.1	Durable proof memory was locked: first admission only, append forever.
v93.2	SDK grew from low-level clients into proof-memory, projection, fixture, and schema infrastructure.
v94.0	Official SDK launch and durable proof operating model release.
v95.0	Official public table release: SDK, docs, and report state the canonical Kai/Klok proof object boundary exactly.

MAJOR FINDINGS

- Receiz gets faster by honoring proof ontology, not by hiding data.
- Known verified truth must paint before sync, route hydration, DB scans, or SDK responses.
- Kai is not aesthetic; it is the required state coordinate carried by complete Receiz proof objects.
- Databases and RPCs make additions faster. They do not become proof authority.
- Developers need the law packaged in the SDK so integrations do not reintroduce mutable feed behavior.

Release Contract

Invariant	Release-blocking regression
Known verified truth paints first.	A surface waits on server/session/DB before showing truth it already has.
Verified proof memory is not cache.	Reopen, restart, or route remount drops admitted history and hydrates from zero.
Complete proof objects carry Kai and order state by Kai.	Backend ordering reverts to Chronos-first for Kai-bearing proof objects.
SDK is convenience beneath proof.	SDK response, token, or API projection outranks sealed artifact or verified append.
DB is projection and sync.	A missing/slow DB row is treated as proof that local truth disappeared.
Visual evidence is product truth.	A screenshot showing changed primitive identity is overruled by a passing test.

The correct engineering question for every future feature is: what verified truth do we already have, and what verified additions exist after its known head?

If a feature asks whether old verified truth is still true, it is using the wrong model.

Why This Matters

Receiz has leverage because it is stricter than normal software at the core and simpler at the edge. The codebase now carries a coherent operating model: proof first, durable memory, Kai Klok as the proof object state machine, kaiPulseEternal as the pulse unit, append-only additions, projection beneath truth, SDK convenience beneath primitives.

A user should not watch Receiz discover what Receiz already knows.

A developer should not have to reverse-engineer Receiz law from production failure. The app, SDK, docs, conformance suite, and release gates must carry that law directly.

SYSTEM GRADE

- Architecture: A - proof-native primitives are explicit and increasingly obeyed in code.
- Experience continuity: A- - Sports, wallet, market, public proof, and console surfaces have strong first-paint and control continuity.
- Performance model: A- - slow pressure is being closed by bounded projection beneath known truth.
- Developer platform: B+/A- - official SDK is real; the next lift is deeper full-flow app templates and conformance fixtures.
- Codebase cleanliness: B+ - large legacy files remain, but contracts now protect behavior while modularization continues.