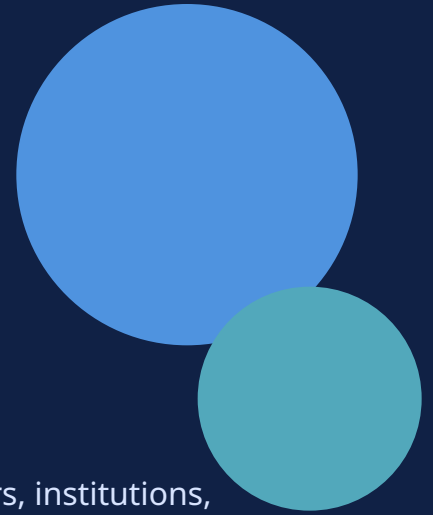




RECEIZ
PUBLIC REPORT / v96.0.0



Proof-Native Artifact System

A public architecture report for users, inventors, developers, builders, institutions, governments, and partners who need to understand what Receiz is and why it matters.

PROOF OBJECTS

KAI KLOK STATE

OFFLINE VERIFICATION

RECORD/SEAL, VERIFY, APPEND FOREVER

SOURCE LINES

1,126,260

TRACKED FILES

5,242

TEST LINES

90,810

CONFORMANCE

local

Receiz is not a generic app with proof features. Receiz is a proof-native artifact system where identity, ownership, settlement, media, public witness, and developer integration are projections of implemented proof primitives.

01 / EXECUTIVE SUMMARY

Receiz lets truth travel with the object.

Receiz makes digital identity, ownership, media, settlement, public proof, and developer integration operate from proof objects first. Servers and databases sync and publish additions; they do not decide whether already sealed proof truth is true.

For regular users

Your known truth is carried by proof objects and durable local truth, not by a fragile loading screen.

Your identity, wallet history, cards, media, and public proofs can paint from verified proof truth first.

Network sync may add verified additions, but it must not erase or downgrade what is already known.

For builders and developers

@receiz/sdk exposes validators, projections, durable proof memory, local conformance, CLI inspection, webhook helpers, Connect, and copy-paste integration rails.

The SDK is convenience beneath proof truth, not a new authority.

For inventors and creators

Receiz turns work into public proof objects with provenance, ownership/custody, inspection, verification, and durable witness surfaces.

A creator can prove what exists and what changed without asking a platform to be truth.

For institutions

Receiz provides inspectable, append-only records, offline verification, public witness surfaces, deterministic identity, and settlement primitives.

Auditability is built into the object model, not bolted on after the fact.

The operating law

Use known verified truth immediately. Ask only what verified additions exist after the known proof/Kai head. Sealed proof truth remains true; new verified truth appends.

The Truth Ladder

Receiz preserves a strict authority order. A weaker layer may decorate, index, hydrate, or publish stronger truth. A weaker layer must never replace stronger truth.

01

Receiz law and product invariants

The release-blocking rules that define what the system is.

02

Sealed artifact truth and embedded proof

The proof object carries the strongest portable truth boundary.

03

Deterministic proof object state

Identity, payload, provenance, inspection, and Kai/Klok state.

04

Verified local truth and durable proof memory

The stored verified prefix. It is not cache and it is not stale.

05

Verified appends and register blocks

New additions after the known proof/Kai head.

06

Server, DB, session, API, SDK, and CLI

Sync, projection, indexing, convenience, and developer ergonomics.

What this means in practice

If the user already has a valid proof object, the product must project it immediately. A database timeout may delay global sync, but it must not make known proof disappear.

This is not fallback behavior. Offline verification and durable proof memory are Receiz primitives.

Kai Klok Is The State Machine

Kai Klok is not an aesthetic label, metadata, or optional timestamp. Complete Receiz proof objects carry Kai. The pulse is the living coordinate that lets proof objects order, append, and verify without submitting truth to a wall-clock authority.

Chronos is display

Chronos remains useful for human calendars, receipts, and external compatibility.

It is not the primary backend authority for proof object order.

When Chronos-only metadata changes, it must not force proof rewrites.

Kai is state

Kai/Klok state lets Receiz ask: what verified additions exist after my known head?

That question preserves causality and avoids rediscovering known truth.

It is the basis for record/seal first, verify, then append forever.

Why this improves performance

Legacy systems repeatedly ask databases to prove that old rows still exist. Receiz records or seals proof objects, verifies them, stores the verified prefix, and uses sync for verified additions.

That removes needless reads, removes broad rewrites, preserves user continuity, and makes the experience feel instant because the product starts from already-held truth.

The release lock

v96.0.0 locks the same language into the SDK: Kai Klok is the proof object state machine, kaiPulseEternal / kai_upulse is the pulse unit, and complete Receiz proof objects carry Kai.

How Receiz Operates

Every main surface is a projection of proof primitives. The same truth can appear as a profile, wallet ledger, calendar point, sports card event, public proof route, developer SDK object, or offline verifier object without becoming different truth.



The database role

The database is an index, sync rail, projection store, queue, and global coherence mechanism. It is not higher authority than sealed artifact truth, deterministic proof object state, or verified local truth.

This is why database health matters for sync and global append distribution, but it must not become a gate that hides already known proof objects from the user.

The user experience result

Known truth first paints immediately. Background work appends and refines. The user should experience continuity, not repeated discovery.

What Receiz Surfaces Carry

Receiz screens are not independent islands. They are different projections of proof objects, ownership state, settlement state, identity continuity, and public witness.

Identity

Receiz ID, PBI recovery, Identity Record, Identity Seal, Receiz Key, local restore, and Connect delegation preserve one account continuity primitive.

Ownership

Vault objects, sports cards, media artifacts, transfers, custody, provenance, and public proof routes expose ownership surfaces.

Settlement

Wallet ledger, reserve value, certificates, notes, purchase routes, payouts, and governed transfers are settlement primitives.

Public Proof

Public routes show proof objects, witness state, provenance, ownership/custody, media, and verification affordances.

Sports Arena

Live cards, pitch proof, play proof, event proof, append history, and card memory demonstrate proof-native real-time experience.

Developer Surface

SDK, CLI, OpenAPI docs, quickstarts, fixtures, conformance, webhooks, and Connect expose Receiz without creating new authority.

One truth, many projections

The same verified proof additions can appear in calendar, account, card history, wallet, and public proof surfaces. The correct system does not rediscover them independently as unrelated data.

Receiz In The Real World

The simplest way to understand Receiz is to start from ordinary moments. A person records something, owns something, or settles something. Receiz makes that moment carry proof truth, then lets later verified additions append without asking old truth to become true again.

A creator seals an original work

The visible action is simple: the creator records and seals the work. Underneath, the artifact carries deterministic identity, sealed payload or media, provenance, inspection behavior, and Kai/Klok state.

Later sales, transfers, public witness, rights evidence, or disputes append to the same proof object. The original work does not become a loose upload that must be rediscovered by a server.

A fan owns a sports card that appends live events

The fan sees a card, but the object carries athlete identity, ownership/custody state, event proofs, score deltas, pitch and play witnesses, and durable card memory.

When a live event happens, verified additions append to the card. Reopening the card projects admitted history first; sync only asks what verified additions exist after the known head.

A business accepts payment into a proof-native settlement trail

The payment does not disappear into a generic balance. It becomes a settlement trail with value state, reserve, certificate or note context, payer/payee evidence, and proof-native completion.

Receipts, refunds, transfers, audits, and reconciliation append beneath the same settlement primitive instead of replacing the settled record.

Same law in all three

The object carries truth. The app projects it. Servers and databases publish, index, and sync additions beneath it. Chronos helps humans read the moment; Kai orders the proof state.

What Users Benefit From

Receiz gives ordinary users a direct experience of proof-native software without making them understand infrastructure. The product should feel faster, more durable, and more trustworthy because the system starts from known proof truth.

Continuity

The app uses verified local proof truth first, so known history, identity, ownership, and media do not need to be rediscovered on every open.

Airplane-mode access to known proof is a product law, not a fallback aspiration.

Trust without theater

A proof object can be inspected and verified. The user is not asked to trust a spinner, a session, or a database row as the root of truth.

Background sync adds truth; it does not make old truth disappear.

Everyday examples

A profile loads known public proof. A wallet shows known settlement history. A sports card shows verified player events and appends. A calendar shows verified proof points. A public route shows witness material. All are projections of proof truth.

Why speed improves

The fastest request is the one Receiz does not need to make. Once truth is verified and stored, the product does not refetch or re-prove it before using it. It only syncs verified additions.

Why Inventors And Creators Need It

Receiz gives inventors and creators an artifact system where originality, publication, authorship, media, transfer, witness, and verification are first-class proof surfaces.

From file to public proof object

A creator can seal an artifact, attach media and payload material, expose inspection and verification behavior, and make the object public without reducing it to generic metadata.

The public proof surface remains tied to the proof object. It is not merely a share page.

Provenance that survives

Receiz preserves source, ownership/custody, append history, public witness, and verification affordances. Later additions enrich the object without rewriting what was already sealed and verified.

A better market primitive

Market surfaces are deterministic proof bundles, not generic listings. Visible public profile originals belong in market first paint, and hidden originals remain absent until visible again.

Why Governments And Institutions Should Care

Institutions need records that can be inspected, verified, reconciled, and carried without making one server the permanent source of truth. Receiz gives them proof objects and append-only evidence rather than mutable dashboard state.

Audit and custody

Receiz objects can carry deterministic identity, provenance, ownership/custody state, witness state, and inspection behavior.

That makes audit a property of the object model.

Offline verification

A sealed object can remain verifiable when a network, vendor, or database is unavailable.

That matters for resilience, continuity, records, emergency use, and cross-organization handoff.

Institutional use cases

Certificates, permits, custody records, public claims, grants, procurement records, creator rights, sports and cultural archives, identity proofs, settlement records, and evidence packages can be represented as proof-native artifacts.

Governance boundary

Receiz does not ask an institution to abandon servers. It demotes servers into projection, sync, queue, and publication rails beneath proof truth, which is the correct authority order.

The Developer Surface

@receiz/sdk is the canonical developer convenience layer. It is version-locked to the app, exposes proof validators and projections, and now ships an executable CLI for local inspection, conformance, and starter generation.

```
npm install @receiz/sdk
```

Install the public SDK package.

```
npx @receiz/sdk conformance
```

Validate packaged fixtures locally with zero network calls and zero DB calls.

```
npx @receiz/sdk inspect ./proof.json
```

Inspect one proof payload and print the known proof/Kai head.

```
npx @receiz/sdk init ./receiz-integration
```

Generate a local-first durable proof memory starter.

What developers receive

Typed clients, schemas, validators, projections, durable proof memory, local storage and in-memory stores, identity artifact helpers, wallet reads, Sports card/event proof integration, Connect transfers, webhook signature verification, CLI tooling, docs, fixtures, and copy-paste integrations.

What developers do not receive

The SDK does not expose service-role authority, database authority, private production secrets, or a way to fabricate proof truth. It helps developers use Receiz primitives correctly.

v96 SDK A+ Lock

v96 turns the developer surface from strong documentation into an executable, package-verified integration surface. The SDK proves itself from install to conformance to starter.



Local conformance

Packaged asset, sports card, and webhook fixtures verify, project, and store in durable proof memory.

The contract reports 3 fixtures, 3 proof memory entries, zero network calls, and zero DB calls.



Package Evidence

SDK tests, typecheck, build, CLI conformance, npm pack dry-run, package contents, and no-authority-leak checks are part of release evidence.

The app and SDK versions remain locked.



Passive observations

ReceizSdkObservation reports proof memory record, append, persist, and clear events without leaking payloads and without changing proof verification or storage. Evidence remains evidence; it cannot become proof authority.



Developer default

The SDK teaches developers to verify first, project immediately, persist durable proof memory, ask for additions after the known head, and use Connect only when delegated Receiz API access is required.

Security And Authority Boundary

Receiz does not become safer by inventing new authorities above implemented proof primitives. It becomes safer by preserving the exact authority order and making every convenience layer subordinate to proof truth.

Why the SDK does not let someone steal Receiz

The SDK exposes schemas, validators, projections, helper clients, CLI inspection, local conformance, and integration examples. It does not expose private signing authority, service-role access, production database authority, or a way to mint valid Receiz proof objects without the required proof material.

A developer can build experiences for Receiz assets because Receiz gives them the public integration rail. They cannot turn the SDK into Receiz authority.

Correct verification pattern

Verify the stronger Receiz artifact, identity proof, ownership proof, settlement proof, or sealed file first. Then parse, project, and render embedded subpayloads.

A subpayload hash validates deterministic projection only. It does not outrank the enclosing stronger primitive.

External systems boundary

Browsers, frameworks, databases, APIs, and registries inform mechanics. They do not define Receiz product truth.

Why It Performs

Receiz performance comes from honoring proof law. The system becomes lighter when it stops rediscovering known truth, stops rewriting Chronos-only metadata as proof change, and treats databases as sync/index rails beneath durable truth.

No rediscovery

Known verified truth first paints from durable proof memory.

Background sync asks only for verified additions after the known head.

The user gets truth sooner because the app starts from truth it already has.

No false authority

DB/server/session state can publish, index, repair, and sync.

They cannot erase sealed artifact truth or durable local truth.

This removes unnecessary pressure and preserves continuity.

The sports breakthrough

Sports card append history proved the law in a high-pressure live surface: once player events are verified and stored, they stay on the card and do not need to hydrate again. New appends extend the proof history.

That lesson applies across Receiz: profile, wallet, calendar, account, sports, market, identity, and developer surfaces all benefit from record/seal, verify, then append forever.

User experience impact

Lower database pressure is not only an infrastructure win. It is a product win: less lag, fewer blank surfaces, less repeated loading, better offline continuity, and faster access to proof truth.

Release Evidence

The v96 release locks the public developer surface with concrete package evidence. The PDF is a public explanation layer; the source of truth remains the repo law, release docs, tests, SDK package, and generated artifacts.

APP VERSION

v96.0.0

SDK VERSION

@96.0.0

SOURCE LINES

1,126,260

TEST LINES

90,810

Release-blocking gates

test: sdk-release-contracts verifies app/SDK version lock, SDK runtime constant, CLI docs, copy-paste docs, no Supabase authority leaks, passive observations, SDK tests, SDK typecheck, SDK build, CLI conformance, and npm pack dry-run package contents.

test: release-freeze includes the SDK release contract so developer evidence cannot drift away from product truth.

Public report evidence

This report is generated by `scripts/generate_receiz_v96_public_report_pdf.py` and written to `output/pdf/receiz-v96-public-architecture-report.pdf`.

It is designed to be shared with users, developers, inventors, partners, institutions, and governments without requiring private repository access.

How To Benefit From Receiz

Receiz can be adopted at different depths without weakening the primitive model. Every adoption path starts from proof truth and uses sync, APIs, and SDKs beneath it.



Users

Use Receiz to keep identity, media, wallet, card, market, calendar, and public proof truth as durable objects that can be inspected, shared, and verified.



Developers

Install @receiz/sdk, run local conformance, inspect a proof object, create durable proof memory, render known truth immediately, and ask Receiz only for verified additions after the known head.



Inventors and builders

Use Receiz to turn artifacts, products, assets, media, proofs, and state transitions into inspectable proof-native objects that preserve provenance and support public witness.



Institutions

Pilot proof-native records where continuity, offline verification, custody, settlement, public proof, and auditability matter more than ordinary database visibility.

Plain-Language Glossary

Proof object

A Receiz object with deterministic identity, verifiable payload, provenance, inspection behavior, durable state, and Kai/Klok state.

Record / seal

The proof object creation boundary. A record or sealed artifact must be verified before it is stored as known proof truth.

Durable proof memory

The stored verified prefix. It is not cache and it is not stale.

Kai Klok

The proof object state machine. `kaiPulseEternal` / `kai_upulse` is the pulse unit.

Verified append

New verified truth appended after the known proof/Kai head.

Public proof surface

A public route that presents a proof object, witness, provenance, ownership/custody, and verification affordance.

Settlement primitive

Proof-native value, reserve value, certificate value, note issue value, funded purchase state, or governed transfer state.

Connect

Delegated Receiz API access after proof identity, not the identity proof root.

SDK admission

A developer/storage mechanic for placing already verified proof truth into durable memory. It is not sealing and it is not authority.

SDK

Convenience for developers beneath proof truth, not an authority layer.

PUBLIC RELEASE STATEMENT

Receiz makes proof usable.

The practical breakthrough is not only that Receiz can prove objects. It is that Receiz makes proof objects livable: immediate first paint, durable history, offline verification, public witness, ownership, settlement, identity continuity, and developer integration all operate from the same primitive law.



For the public

Receiz gives people stronger digital continuity: identity, artifacts, public claims, wallet settlement, and verified history that do not vanish because a session, server, or database is slow.



For builders

Receiz gives developers a canonical SDK and CLI so they can build proof-native experiences without re-inventing authority, verification, durable memory, or append-sync law.



For institutions

Receiz gives governments, organizations, and serious operators a model for records and assets where verification and continuity are embedded in the object, not dependent on a fragile dashboard.

Record. Seal. Verify. Append forever.