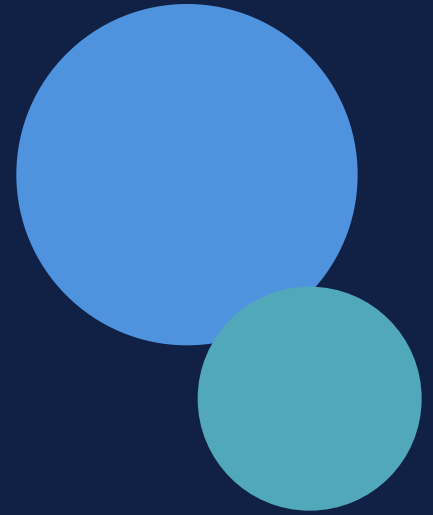




RECEIZ
PUBLIC REPORT / v98.0.0



Proof-Native Artifact System

The object carries truth. Everything else is projection.

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

1,158,623

TRACKED FILES

5,591

TEST LINES

95,287

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

v98 makes Receiz programmable without moving truth.

SDK, MCP, webhooks, public proof, market, ledger, sports, and agents now build around the proof object instead of replacing it.

Receiz v98.0.0 locks the post-v97 developer runtime, proof graph, ledger settlement, sports performance, profile continuity, webhook delivery, MCP agent rail, and Pitch Command archive into one proof-native release.

For users

Profiles, wallets, cards, transactions, market views, and public proof routes keep known verified truth first. Background work may append; it must not replace settled proof.

For developers

@receiz/sdk and @receiz/mcp-server are aligned at v98.0.0. SDK, CLI, MCP, OI DC, webhooks, app-state, and public-store rails remain beneath proof objects.

For institutions

Public proof graph, transaction proof inspection, ledger projections, conformance suites, governance, and visual-evidence contracts keep proof surfaces inspectable.

For markets and sports

Market/profile projection freshness is a write-side obligation after first paint is protected. Sports performance work stays beneath ownership, settlement, and closeout truth.

The v98 lock

Baseline: v97.0.0. Reviewed range: v97.0.0..6cd0af58a. Scope: 65 commits / 535 paths before v98 edits.

Release-freeze passed. Signed attestation still requires non-empty release signing environment values.

What Receiz Replaces

Traditional software makes the server, database, session, dashboard, or platform account the root of truth. Receiz changes the authority order.

The authority shift

The proof object carries identity, ownership, settlement, media, provenance, witness material, and append history. The app projects that truth. The server syncs it. The database indexes it.

SDK, MCP, webhooks, and agents build around it. Receiz is not another app category. It is a different record model.

Old model

The platform remembers for you.

Authority concentrates in server, database, session, dashboard, or account state.

Receiz model

The object carries its own memory.

Lower rails project, sync, publish, automate, and index beneath proof truth.

Why this bridge matters

Normal readers need to know what world Receiz replaces before they understand why v98 matters. The replacement is authority order, not a decorative interface category.

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 Still 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.

Chronos-only display change must not force proof rewrite.

Kai is proof state

Kai/Klok state lets Receiz ask what verified additions exist after the known head.

That preserves causality and avoids rediscovering known truth.

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

Why v98 performs better

v98 adds more developer and public rails, but the speed law stays the same: known verified truth paints immediately; weaker rails append, index, publish, or hydrate beneath it.

This is why SDK, MCP, webhooks, public proof graph, ledger, sports, and market/profile work can grow without becoming a new truth source.

The release lock

v98.0.0 keeps Kai Klok as the proof object state machine across SDK, MCP, public proof, ledger, transaction, market, profile, sports, Pitch Command, and offline verification surfaces.

Anatomy Of A Receiz Proof Object

A Receiz proof object is the durable object that carries the truth Receiz projects. This page names the object parts without adding a new authority layer.

Identity

The object has deterministic identity and can be inspected as itself.

Payload / Media

The object can carry or reference the work, card, record, transaction, certificate, file, or event material.

Provenance

The object carries source, authorship, custody, witness, and origin context.

Kai/Klok State

The object has a proof-time coordinate for ordering, appending, and verifying state without making wall-clock time the authority.

Verified Appends

New truth attaches after the known head. Old truth does not need to be rewritten.

Verification Surface

Users, developers, public routes, and institutions can inspect the object and its proof material.

Bottom line

A Receiz proof object is not a file with metadata. It is a durable state-carrying artifact.

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 v98 Surfaces Carry

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

Identity

Receiz ID, PBI recovery, Identity Record, Identity Seal, Receiz Key, OIDC/Connect delegation, and local restore 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, funded purchases, market trades, payouts, and governed transfers are settlement primitives.

Public Proof Graph

Proof graph, market-cap projections, transaction proof inspection, public ledgers, and conformance badges are public proof surfaces.

Sports Arena

Live cards, player images, vault projections, card memory, event proof, closeout, and append history stay proof-native.

Developer Runtime

SDK, CLI, MCP, OIDC agent tokens, webhooks, OpenAPI, public-store, and app-state rails expose Receiz without creating new authority.

One truth, many projections

The same verified proof additions can appear in calendar, account, card history, wallet, transaction, proof graph, market, public proof, and MCP surfaces. The correct system does not rediscover them 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.

Institutional Pilot Paths

Receiz can be piloted without replacing an institution's entire system. The first pilots should target records where continuity, custody, verification, and offline resilience matter more than dashboard convenience.

Pilot 1 - Certificates and credentials

Issue certificates as proof-native artifacts with identity, issuer, recipient, verification, and append history.

Pilot 2 - Permits and public records

Turn permits, claims, notices, approvals, and revocations into inspectable proof objects instead of mutable rows hidden behind a portal.

Pilot 3 - Custody and evidence packages

Carry custody, transfer, inspection, and witness state with the object itself.

Pilot 4 - Grants and procurement

Represent awards, milestones, payments, receipts, and reconciliations as settlement-linked proof trails.

Pilot 5 - Archives and cultural records

Preserve sports, creator, civic, and historical records as proof-native public artifacts.

Best first pilot

The right first pilot is not the biggest database. It is the record people cannot afford to lose, fake, rewrite, or misplace.

The v98 Developer Surface

@receiz/sdk and @receiz/mcp-server are version-locked to the app at v98.0.0. They expose proof-native development rails while preserving the authority boundary: proof objects first, developer convenience second.

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

Install the typed SDK package for proof objects, public proof, wallet, Sports, Twin, World, Connect, webhooks, and public-store work.

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

Run packaged fixture conformance with zero production network calls and zero database calls.

```
npx @receiz/sdk inspect ./receiz-asset-manifest.json
```

Inspect one proof payload, project display rows, admit it into memory, and print the known Kai/proof head.

```
npx -y @receiz/mcp-server
```

Run the official MCP server for agent-callable Receiz SDK/API tools beneath proof truth.

What developers receive

Typed clients, schemas, validators, projections, durable proof memory, local storage, identity artifact helpers, wallet reads, Sports proof integration, public proof lookup, Connect/OIDC delegation, webhook signature verification, MCP tools, docs, fixtures, and copy-paste integrations.

What developers do not receive

They do not receive service-role authority, database authority, private production secrets, or a way to fabricate proof truth. SDK and MCP help developers use Receiz primitives correctly.

SDK And MCP Runtime Lock

v98 promotes the developer surface from SDK-only release evidence into a broader app runtime: SDK package, MCP package, OIDC delegated agent tokens, webhook management, public-store publishing, and public app-state rails.

@receiz/sdk@98.0.0

Typed client, CLI, proof memory, schemas, projections, public proof, wallet, Sports, Twin, World, Connect, webhooks, media, domains, release pinning, sandbox helpers, and React exports.

The SDK is convenience beneath proof truth.

@receiz/mcp-server@98.0.0

Agent-callable tools for diagnostics, scopes, app-state, public-store, verification, public rails, transfer previews, pack previews, webhooks, sandbox, and resource templates.

MCP authority is false.

Delegation without replacement

Receiz-issued Connect/OIDC agent tokens authorize scoped mechanics. They do not replace Receiz ID, PBI, Identity Seal, Receiz Key, proof objects, ownership appends, settlement rows, or verified local truth.

Merchant public-store authority

Merchant-owned public-store state can publish with Identity Seal / Receiz Key proof authority without delegated bearer-token authority. Delegated MCP/server writes remain a separate mechanics rail.

MCP Agents Beneath Proof Truth

The MCP server makes Receiz buildable by agents without changing what Receiz is. Every MCP tool is a lower-authority interface over SDK/API/public proof rails.

Agent setup

Run `receiz_doctor` and `receiz_capabilities`, request required scopes, use `receiz_mcp_login` or `receiz_authorize_url` for Connect/OIDC, then resolve known public-store or app-state truth for cold start.

Render known verified projection truth immediately. Publish verified additions only through the relevant delegated rail.

Tool families

Diagnostics, identity sessions, durable public app state, commerce/store projections, proof verification, public ledgers, world snapshots, pitch witnesses, card history, transfer previews, pack previews, marketplace templates, and webhook workflows.

Authority boundary

MCP may inspect, resolve, publish through delegated SDK calls, and report. MCP must never replace, rename, or outrank proof objects, identity primitives, settlement primitives, ownership surfaces, or public proof surfaces.

Public Proof Graph

v98 makes the public proof graph a richer public proof surface without turning it into decorative analytics. It is rooted in real proof, economy, flow, asset, agent, ledger, market-cap, and Kai pulse data.

What the graph carries

Live Kai pulse state, proof/economy/flow/asset/agent views, detailed panels, mobile graph navigation, public ledgers, market-cap projections, settlement rollups, transaction volume, and asset pulse charts.

What keeps it honest

Graph data remains a public proof surface. Downsampling, viewport fit, panel layout, and chart controls are performance mechanics beneath proof truth, not a license to simulate proof.

Public transaction inspection

`/transactions/[slug]/[code]/[pulse]` projects transaction proof inspection with wallet ledger links, proof-bundle projection, copy affordance, and route contracts. Transaction truth remains proof-bundle and settlement truth, not generic page state.

Ledger, Market, And Settlement

v98 keeps value language exact. Wallet rows, market positions, certificate trades, notes, reserves, transaction pages, and ledger routes must preserve settlement, transfer, proof-bundle, and Kai/proof coordinates.

Ledger proof

Public ledger pagination, cursor repair, transfer value snapshots, proof-bundle extraction, transaction prioritization, kai_upulse / Kai Klok display, and stronger sorting/rollup handling.

Ledger is proof projection, not generic activity feed.

Market proof

Market-cap and LiveChart work added downsampling, OHLC candles, sampling budgets, chart controls, asset pulse projection, and only-open asset chart construction.

Rich views stay light beneath visible proof surfaces.

Profile and market first paint

Visible public profile originals belong in market. Profile/market write paths own public projection freshness; read paths must not regain speed by rediscovering all profile truth before deterministic first paint.

Sports Performance And Pitch Archive

v98 hardens Receiz Sports and Pitch Command as proof-native live surfaces. Performance work is allowed only when it reduces weaker work beneath proof, ownership, settlement, media, and closeout truth.



Sports vault and arena

owner_current projection mode, sport-aware portrait/card image behavior, base-card BASE 1:1 language, deferred surfaces, bounded rollup workers, active-job dedupe, transfer routes, value snapshots, closeout contracts, and performance contracts.



What must never move

Known proof, media, ownership, closeout, settlement state, Main Event scores, and card memory must not move behind idle/background rediscovery. They are already admitted proof truth when the app has them.



Pitch Command archive

Day-proof bundles through July 7, 2026, manifest/projection updates, strict archive parsing, workflow catch-up behavior, and lint-safe proof-register typing are release-carried proof archive material, not disposable static content.

Security And Authority Boundary

Receiz does not become safer by inventing new authorities above implemented proof primitives. v98 adds many developer rails, so the release has to name their lower authority explicitly.

SDK and MCP do not own truth

The SDK exposes schemas, validators, projections, helper clients, CLI inspection, local conformance, and integration examples. MCP exposes agent-callable tools. Neither exposes service-role authority, database authority, private production secrets, or a way to mint valid proof objects without the required proof material.

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.

Webhook and token boundary

Webhook endpoint state, delivery logs, agent tokens, app-state rows, and public-store rows are lower-authority rails. They may publish, deliver, index, and verify mechanics, but they cannot replace proof objects or verified appends.

What Receiz Is Not

Receiz is not a blockchain wrapper.

Receiz is not an NFT marketplace.

Receiz is not a dashboard with verification badges.

Receiz is not a cache strategy.

Receiz is not an AI-agent app.

Receiz is not a generic SaaS platform.

What Receiz is

Receiz is a proof-native artifact system. Blockchains, servers, databases, APIs, agents, SDKs, dashboards, and public routes may witness, project, publish, automate, or synchronize Receiz truth.

They do not become the source of that truth.

Category boundary

If a weaker rail can erase the record, it was never the authority. If the object still verifies when the rail disappears, the primitive is real.

Why v98 Performs

Receiz performance comes from honoring proof law. v98 does not make the system faster by hiding truth; it makes the system faster by reducing lower-authority work beneath settled proof surfaces.

Known truth first

Profile, market, sports, ledger, wallet, transaction, and public proof surfaces paint known proof first.

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

Bounded projection

Chart downsampling, rollup workers, active-job dedupe, deferred panels, and route extraction reduce load without reducing visible proof truth.

Performance stays beneath the primitive.

The profile contract lesson

The production profile route intentionally preserves a deterministic first-paint fallback when public profile projection throws. The v98 contract now locks that fallback instead of requiring an outdated direct-call shape.

User experience impact

Lower database and chart pressure is not only infrastructure. It is product law: less lag, fewer blank surfaces, fewer fake loaders, better offline continuity, and faster access to proof truth.

v98 Release Evidence

The v98 release locks the public developer/runtime surface with executable package, app, conformance, governance, and release-freeze evidence. This PDF is a public explanation layer; the source of truth remains repo law, release docs, tests, packages, and generated artifacts.

APP VERSION

v98.0.0

SDK VERSION

98.0.0

MCP VERSION

98.0.0

COMMITTS

65

Release-blocking gates

Full pnpm test:release-freeze passed for v98.0.0. It includes release lock, lint, build, deterministic surfaces, page-first-paint, visual evidence, primitive binding, repo-only reasoning, profile/world/sports/lease/kiosk/contact contracts, SDK/MCP/package gates, conformance suites, value-loop, API/security/chaos/chat/perf, and governance controls.

Attestation boundary

pnpm release:attest is the signed promotion artifact after gates pass. It requires active release signing environment values. This PDF is not the signed release attestation.

Public report evidence

This report is generated by scripts/generate_receiz_v98_public_report_pdf.py and written to output/pdf/receiz-v98-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 v98

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



Users

Use Receiz to keep identity, media, wallet, card, market, calendar, transaction, 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.



Agents

Run @receiz/mcp-server, use delegated Connect/OIDC scopes only when needed, and keep every MCP action beneath proof objects, verified appends, ownership, settlement, and identity primitives.



Institutions

Pilot proof-native records where continuity, offline verification, custody, settlement, public proof, conformance, 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.

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/OIDC

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

SDK

Developer convenience beneath proof truth, not an authority layer.

MCP

Agent-callable tools beneath SDK/API/proof rails. MCP authority is false.

Webhook

Signed event delivery mechanics. Delivery does not outrank the proof object, append, ownership, or settlement event it reports.

PUBLIC RELEASE STATEMENT

Receiz v98 makes proof programmable.

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



For the public

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



For builders

Receiz gives developers and agents canonical SDK and MCP rails 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.

Build on Receiz. Pilot Receiz. Verify with Receiz.

Users: open, own, inspect, and carry proof-native artifacts.

Developers: install @receiz/sdk and run local conformance.

Institutions: start with one record class where custody and verification matter.