Kungfu: The Session Is Not the Work

Verified continuity for agent work that outlives the chat.

Project Cuts, `libkungfu`, and KFD: a local runtime and open responsibility protocol for long-running agent work.

User promise

Give your agent verified context. Keep the work when the chat ends.

The move for Agent builders

Embed `libkungfu` now. Deliver Project Cut continuity. Build your KFD-compatible Hub in parallel.

First user object	A Project Cut: the verified point from which a project can continue.
Reference runtime	`libkungfu`: embeddable Facts, Episodes, verification, recovery, and Project Cut support.
Open standard	KFD: portable responsibility and evidence semantics across independently owned systems.
Product boundary	Each vendor keeps its interface, models, accounts, policy, cloud, and customer relationship.
Current status	Alpha interfaces and first-party evidence; source-pinned integration is encouraged, certification is not claimed.

Keren Dong

Kungfu Origin Technology Limited keren.dong@kungfu.link

1 Executive Summary: The Work Must Outlive the Chat

The session is the right unit of interaction—and the wrong unit of continuity for long-running Agent work. A session can host valuable code, research, plans, operations, and decisions. It cannot, by itself, authoritatively settle what the project has become, what was accepted, which authority still applies, or where another participant may safely continue. When the session closes, a transcript, summary, or unverified memory is not enough. The model may be capable while the work remains fragile.

Kungfu closes this gap through three distinct layers. A **Project Cut** is the verifiable settlement and continuation coordinate presented to the next human or Agent. `libkungfu` is the local runtime that records, verifies, recovers, and resumes that continuity inside a participant's Hub. KFD is the open responsibility protocol that lets independently owned Hubs exchange continuity evidence without sharing one central cloud or surrendering their product boundary.

Together, these products provide the founding proof for an open Agent Supply Chain. KFD-3 makes product cooperation discoverable; Buildchain binds product claims to exact software artifacts; KFD-2 supports purpose-bound trust; libkungfu and `.kungfu` preserve durable work facts; and the public KFD Agent Hub profile lets independently owned Hubs carry bounded responsibility objects without sharing one vendor control plane.

This is a second strategic axis, not a claim that a multi-Hub market already exists. The current evidence covers public contracts, release mechanics, source-built runtime slices, and first-party qualification. It does not prove two independent production Hubs, external vendor adoption, industry-standard status, or blanket stable interoperability.

A Project Cut binds the accepted source state, the perspective used to understand it, the causal work that occurred, the decisions that admitted the result, unresolved risk, and the coordinate from which authorized work may continue. It does not replace those authorities. It gives the next participant one bounded answer to the question: *where can this project safely continue?*

> **The product promise**
>
> Give your agent verified context. Keep the work when the chat ends.

1.1 One Product Decision, Two Adoption Paths

Users should encounter Project Cut continuity before they need to understand the runtime beneath it. Agent builders, however, need an immediate technical path. Kungfu therefore separates two responsibilities:

1. embed `libkungfu` as the reference runtime for durable Facts, Episodes, verification, recovery, and Project Cut continuity; and
2. build a participant-owned Hub that retains the vendor's user experience, models, identity, accounts, policies, cloud, admission rules, and customer relationship while exchanging portable responsibility evidence through KFD.

These paths should proceed in parallel. A vendor does not need to rebuild a journal, content-addressed evidence store, recovery engine, and cross-language fact layer before delivering continuity. Nor should embedding a runtime force the vendor to surrender its product boundary to a central Kungfu service.

Recommended Agent Builder strategy

Build your Hub. Do not rebuild the runtime. Embed libkungfu now, prove one source-pinned Project Cut loop, and develop your KFD-compatible Hub boundary in parallel.

1.2 The Architecture in Five Responsibilities

Project Cut	The project settlement and continuation coordinate presented to users and Agents.
`libkungfu`	The reference implementation for runtime Facts, Episodes, journal authority, verification, export, and recovery.
KFD	The open engineering standard for responsibility, evidence, trust, and continuity across independently implemented systems.
Buildchain	Artifact, provenance, release, and evidence publication; not runtime or project authority.
Vendor Agent Hub	Product experience, models, identity, policy, accounts, admission, cloud, and customer relationship.

1.3 What This Paper Claims

Kungfu provides a concrete architecture, open source implementation, and first-party evidence for work that survives session and Agent replacement. It does not claim that a transcript is a Project Cut, that delivery is admission, that an Episode proves completion, or that authentication grants authority. It does not claim external vendor adoption, stable KFD Hub interoperability, certification, or broad production readiness. The public interfaces described here are alpha surfaces and should be adopted through exact source and evidence coordinates.

2 The Continuity Gap

Real work is not a prompt-response pair. It has accepted state, causal history, direction, authority, evidence, unresolved conflict, and consequences. It may cross context windows, models, devices, repositories, organizations, and years. A session is only one observer's temporary projection of that work.

This creates a product failure that better models alone cannot solve. A capable Agent can finish an action and still leave the project unable to continue. A new session may know what was said without knowing what was accepted; see a commit without knowing why it was chosen; inherit a task without inheriting the authority to perform it; or receive a confident summary that silently omits a conflict.

2.1 Context Is Not Project State

Common continuity mechanisms are useful but insufficient on their own:

Chat transcript	Preserves conversation, not admission, authority, or a deterministic project boundary.
Generated summary	Compresses context, but may hide loss, mix observation with decision, or drift when regenerated.
Vector memory	Helps recall, but similarity is not causality, acceptance, or permission.
Git history	Preserves source content and ancestry, but not the complete perspective, Episode, trust decision, or continuation Warrant.
Workflow database	Coordinates mutable process state, but does not necessarily publish an independently verifiable settlement.
Vendor session cloud	Can preserve provider context, but may strand continuity inside one product and one account boundary.

The missing object is not “more memory.” It is an explicit, bounded commitment about the project.

2.2 What a Successor Agent Needs

A fresh Agent should be able to determine, without trusting the previous session’s prose:

- the exact project settlement from which it is continuing;
- which source, perspective, causal, policy, and decision authorities the settlement binds;
- what work occurred and what result was actually admitted;
- what remains unknown, omitted, conflicted, rejected, or intentionally withheld;
- which continuing direction and bounded authority apply now; and
- how to verify the record and publish a successor without rewriting history.

This is the continuity gap Kungfu is designed to close. The product measure is not session length. It is whether one participant can hand real work to another through an exact, inspectable boundary with less human reconstruction and no silent expansion of trust.

3 The Strategic Choice for Agent Builders

An Agent vendor already owns the scarce product assets: the interface, model experience, accounts, permissions, policies, cloud, distribution, and customer relationship. It should preserve those assets. The strategic question is not whether to become a Kungfu-branded application. It is whether to spend years reconstructing a durable runtime before its users can keep work across chats.

The recommended answer is direct:[2]

> **Build your Hub. Do not rebuild the runtime.**
>
> Embed libkungfu now to deliver Project Cut continuity. In parallel, build your own KFD-compatible Hub so responsibility can move across products without moving product ownership.

3.1 The Two-Track Plan

Track A—embed the runtime. Pin an exact `libkungfu` source coordinate, integrate the smallest C, Node, or Python path, keep provider data and credentials outside the adapter, and prove one complete local loop: inspect a Project Cut, record bounded work, verify the evidence, settle a successor, and continue from it in a fresh Agent context.

Track B—build the Hub. Retain participant-owned identity, policy, admission, storage, product workflow, and customer controls. Add a KFD adapter that can negotiate exact capabilities, exchange rooted responsibility objects, write receiver-owned verdicts, preserve conflicts and disclosure boundaries, and export portable evidence.

The two tracks reinforce each other. Runtime integration creates immediate user value and real implementation evidence. Hub development prevents that value from becoming a local island. Waiting for a finished network before embedding the runtime delays the product; rebuilding the runtime before designing the Hub duplicates infrastructure without solving interoperability.

3.2 The Ownership Boundary

Keep in the vendor Hub	UX, models, prompts, tool policy, accounts, billing, identity mapping, customer data, cloud, admission decisions, and support.
Embed from `libkungfu`	Local runtime Facts, Episodes, journal and content roots, lifecycle, query, verification, export, recovery, and Project Cut substrate.
Exchange through KFD	Exact profile coordinates, capability roots, responsibility references, evidence, disclosure state, verdicts, conflicts, and residual risk.
Publish through Buildchain	Artifact digests, source coordinates, build and release provenance, release claims, and residual release risk.

3.3 What Embedding Does and Does Not Mean

Embedding `libkungfu` provides Kungfu's reference runtime implementation for KFD-governed Facts, Episodes, responsibility boundaries, verification, recovery, and Project Cut continuity. It is the fastest route to an executable implementation and the default recommendation for Agent builders.

It does not make KFD proprietary, force one Hub design, send data to a required Kungfu cloud, or automatically certify conformance. KFD remains open and independently implementable. Every compatibility claim must stay bound to the exact Profile, implementation, suite, platform, evidence, and residual risk that were actually assessed.

4 Project Cut: The First User Object

The first user layer in Kungfu is not a log viewer, model selector, or internal Fact table. It is the Project Cut: the highest ordinary interface through which a person or Agent inspects what a project has formally become and from where it may continue. This product decision keeps rigor below the surface while giving the user one actionable coordinate above it.[7]

4.1 What a Project Cut States

A Project Cut binds the following without replacing any of them:

Predecessor	The exact prior Project Cut or Cuts from which this settlement descends.
Source projection	The accepted Git or other source coordinate, under the source system's own authority.
Atlas	The declared perspective, sources, loss, and context used to interpret the project.
Episode increment	The admitted causal occurrence boundary: what happened, not merely what state now exists.
Decision and policy	The rules, Claims, Assessments, authorized Decisions, and Admission boundary used to accept the result.
Open boundary	Known omissions, conflicts, unknowns, rejected claims, withholding, and residual risk.
Continuation	The continuing Initiative or Pursuit, bounded Assignment and Warrant coordinates, verifiable root, and non-self-certifying receipt.

The result answers four user questions at once: What did the project officially become? What happened to produce that state? What was accepted rather than merely delivered or claimed? Where may authorized work continue?

4.2 A Settlement, Not a Fourth Fact Engine

The source repository remains authoritative for source. An Atlas or another perspective system remains authoritative for its declared context. The Kungfu runtime remains authoritative for admitted Facts and Episodes. Policy and decision authorities remain independently accountable for Admission. The Project Cut verifies and binds those roots; it does not absorb, reinterpret, or silently synchronize them.

This separation is essential. A large merged "project state" object would be easy to display but difficult to challenge, recover, or independently verify. A Project Cut instead publishes a deterministic macro commitment at an outer project coordinate while retaining the ability to inspect each authority and to reject a stale, missing, conflicted, unverifiable, or circular binding.

4.3 Why It Comes First

Users should not need to understand every Fact, Episode, Atlas source, Warrant, or receipt before resuming ordinary work. They should see the current verified Cut, active direction, bounded work, pending candidate, and actionable problems. Deeper evidence remains available when a decision, audit, failure, or high-consequence action requires it.

Project Cut is therefore a progressive-disclosure boundary. It compresses complexity without pretending the complexity disappeared. Product work should optimize time-to-first-cut and cut-to-cut continuity, not the number of kernel primitives visible in the interface.

4.4 Validity Is Not Completion

A structurally valid Project Cut proves that declared roots bind and verify under the stated protocol. It does not by itself prove that every objective is complete, the result is correct, the work is safe, or a release is fit for production. Those are explicit Claims and Assessments owned by their declared decision authorities. Project Cut preserves the conclusion and its boundary; it does not manufacture one.

5 How Work Continues Across Sessions

Continuity is a closed loop, not a saved transcript. Kungfu's target product loop is:[7]

1. inspect and verify the current Project Cut;
2. accept bounded work in a continuing Initiative or Pursuit, against the exact Atlas and Warrant roots;
3. execute while preserving causal occurrence in an Episode;
4. submit a completion or state-change Claim with evidence;
5. independently assess the Claim and record the authorized Decision;
6. admit only the accepted result and prepare a candidate successor Cut;
7. verify and explicitly settle the exact candidate; and
8. start a fresh session or Agent from that successor Project Cut.

In short:

session ends → Project Cut remains → fresh Agent verifies → work occurs → Episode records → Claim is decided → result is admitted → successor Cut is published

5.1 Agent Supply Chain

The public architecture separates five responsibilities that products often rebuild inside one private control plane:

1. KFD-3 exposes value, constraints, choices, commands, Exit, and records so products can discover how to cooperate.
2. Buildchain binds product-owned declarations to an exact source cut, build, artifacts, checks, and promotion evidence.
3. KFD-2 lets a receiver assess those claims for a declared purpose while retaining residual risk and decision ownership.
4. libkungfu and `.kungfu` preserve admitted work facts, Episodes, roots, exports, and recovery evidence; applications still own domain facts.
5. the KFD Agent Hub profile carries bounded responsibility objects across independently owned products, with receiver-owned admission.

The layers remain separable authorities. Buildchain does not invent product facts or make the trust decision. KFD does not become a central Hub, identity provider, database, or transport. Kungfu's JSON surfaces are edge projections, not a second authoritative data plane.

5.2 Conservative Context Projection

A new Agent does not need every token from every prior session. It needs a bounded projection that can be traced back to admitted Facts, causal Episodes, declared perspective, continuing direction, and current authority. A useful projection may summarize or omit material, but it must declare that loss and must not upgrade an unknown, conflict, or withheld field into a verified fact.

The full Project Cut remains the round-trip authority. A model-specific context pack is a replaceable view. If the view cannot preserve a load-bearing distinction, the system must expose the loss or refuse the projection rather than silently flattening it.

5.3 Session and Agent Replacement

The architecture treats sessions, models, and Agent processes as replaceable. The work should survive:

- context-window exhaustion or deliberate context reset;
- a new model or provider taking over the same bounded Assignment;
- process crash and recovery from the journal and content roots;
- movement between local and remote execution environments;
- human review between producing and continuing Agents; and
- exchange between independently owned Hubs that make their own admission decisions.

The successor does not trust the predecessor because both are called Agents. It verifies the cut, evidence, authority, and receipt under its local policy.

5.4 Failure Must Stay Visible

Continuity is invalid if the next session starts from a missing Episode, stale Atlas, mismatched receipt, incomplete parent acceptance, changed source after a Claim, widened Warrant, conflicting successor, or unverifiable root. Kungfu preserves such conditions as typed failure, conflict, unavailability, or intentional withholding. Last-write-wins and confident prose are not recovery strategies.

6 The Runtime Beneath the Cut

Project Cut is the first user object, but it can be trusted only because its inputs and lifecycle remain verifiable beneath the product surface. `libkungfu` is the reference implementation of that runtime substrate. It owns how local Facts and Episodes are recorded, ordered, stored, queried, verified, exported, and recovered while the embedding application owns its domain facts and product decisions.[5]

6.1 Fact and Episode

A **Fact** is admitted state under a declared authority. An **Episode** is a realized causal occurrence: work happened, observations were produced, or an attempted action failed. They are related but

not substitutable. State can change only through declared admission; occurrence can be preserved even when no completion Claim is accepted.

Action adds three independently addressable responsibilities:

- **Pursuit**: why the action continues and what success means;
- **Atlas**: what perspective, sources, and declared loss inform the action; and
- **Warrant**: who may do what, to which subject, under what limits, duration, consequence, and revocation boundary.

These coordinates prevent a context packet from impersonating intent or authority. A rich Atlas does not grant permission. A Warrant does not make a Claim true. An Episode does not prove the Pursuit complete.

6.2 Local Semantic Authority

The append-only journal is the highest local semantic authority for typed identity, admission, order, causality, lifecycle, Cuts, refs, and receipts. Content-addressed storage is authoritative for immutable body bytes only when they verify against a journal-committed root. SQLite, CLI, GUI, JSON, Python, Node, and other projections remain rebuildable access surfaces; they do not gain hidden authority by being convenient.

This design makes interruption recoverable. A process may crash after writing content but before admission, or after preparing a candidate but before settlement. Recovery derives the visible state from committed records instead of guessing from whichever projection was last updated.

6.3 Product and Distribution Layers

Project Cut	Binds accepted project authorities and exposes continuation.
Initiative and Assignment	Organize continuing direction and bounded work in the Agent Work Domain Profile.
Runtime substrate	Provides journal-backed Fact admission/query and Episode lifecycle/replay.
`libkungfu`	Embeds the same local truth-source loop in C, Python, Node, Rust hosts, desktop products, workers, or services.
Standalone CLI/TUI	Gives Agents and operators headless discovery, record, query, verification, maintenance, and export.
GUI	Presents the human control surface without owning storage semantics.
`.kungfu` **format**	Carries portable facts, declared schemas, verification, and preservation rules without requiring one GUI or language.

Each layer is an adoption boundary, not a bundle requirement. An Agent vendor may embed `libkungfu` without adopting Kungfu's GUI. A language SDK must not become mandatory authority for another host. A cloud service may synchronize or exchange runtime evidence without making cloud availability a prerequisite for local truth.

6.4 Buildchain Is Release Evidence, Not Runtime State

Buildchain binds source, toolchain, artifact, package, release, and publication evidence. It can prove which PDF, binary, package, or site bundle a release published and which residual risks were declared. It does not decide project Admission, record runtime Episodes, or replace a Project Cut. Keeping release provenance separate prevents a valid build from being presented as proof that the work itself was correct or complete.[1]

7 KFD and Independent Agent Hubs

KFD—Kung Fu Decisions—is an open engineering standard for a small, testable foundation for reliable action and continuity under uncertainty. It is a canonical public decision registry, not a Kungfu product API, a central service, a universal workflow language, or a belief test. Products, organizations, humans, Agents, and other bounded systems can adopt exact decisions within a declared scope and publish their own mappings, evidence, qualification, and residual risk.[3]

7.1 The Foundation

KFD begins with three commitments:

KFD-1	Facts must not drift.
KFD-2	Trust must start from facts.
KFD-3	Cooperation must start from trusted value.

The later decisions make the foundation operational. Views remain bound to declared perspectives. Primitive discovery separates genesis from qualification. Real-world action keeps state, occurrence, and action coordinates distinct. Numbered drafts then allocate emerging responsibility for Atlas, Pursuit, Warrant, Decision and Admission, software work, and Project settlement.

As of the alpha coordinate cited by this paper, the registry state is:

Decision	Status	Responsibility
`KFD-1`	active	Non-drifting facts and verifiable coordinates.
`KFD-2`	active	Fact-bound trust, assessment, and residual risk.
`KFD-3`	active	Cooperation through visible value, constraints, choices, and records.
`KFD-4`	active	Views bound to declared perspectives and loss.
`KFD-5`	active	Primitive genesis separated from qualification.
`KFD-6`	draft	Autonomous discovery grounded in causal experience.
`KFD-7`	active	Fact–Episode separation and action responsibility geometry.
`KFD-8`	draft	Atlas responsibility for admitted perspective.
`KFD-9`	draft	Pursuit responsibility for continuing direction.
`KFD-10`	draft	Explicit, bounded, revocable Warrant authority.
`KFD-11`	draft	Claim, Assessment, Decision, and Admission separation.
`KFD-12`	draft	Initiative and Assignment separation in software work.
`KFD-13`	draft	Project settlement that binds authorities without absorbing them.

Active decisions are normative within their declared adoption scope. A numbered draft has an allocated identity but an open activation gate. A pre-number KFD Candidate remains non-normative lineage. Product evidence for Project Cut does not silently activate KFD-13.

7.2 The Hub Boundary

An Agent Hub is a participant-owned control plane for execution, identity, policy, accounts, models, user experience, admission, and customer relationships. It may be a local process, device service, vendor cloud, organizational system, or federation member. KFD standardizes only the responsibility exchange between Hubs.

No KFD deployment requires one global identity provider, Hub registry, database, transport, clock, or KFD cloud. One Hub may use `libkungfu`; another may implement the semantics independently. Each keeps its internal workflow and local verdict authority.

7.3 The Agent Hub Alpha Profile

KFD publishes a transport-neutral Agent Hub alpha Profile. A Profile coordinate binds its identifier, version, manifest digest, and repository commit. Peers negotiate an exact capability intersection, then exchange rooted responsibility proposals, candidate Fact admissions, supersession, completion Assessments, or Warrant revocations. The same semantics can travel through a local call, IPC, file, HTTP, gRPC, message bus, or offline bundle.[4]

Every receiver retains the right and responsibility to verify authority, disclosure, evidence, conflict, and policy before writing its verdict. Retries are content-stable, conflicts stay visible, partial knowledge is typed, and unknown required semantics fail closed.

The Profile is alpha and is not itself a numbered KFD. It is not stable certification, proof of a conforming implementation, evidence of external vendor adoption, or proof of plural-Hub interoperability. Its purpose is to make the boundary implementable and falsifiable while independent builders gain real integration evidence.

8 Authority, Trust, and Evidence

Continuity is unsafe when a system collapses transport, identity, occurrence, judgment, and authority into a single success flag. Kungfu and KFD preserve the separations that keep responsibility attributable.

Three non-substitution rules

Delivery $\neq$ Admission. Occurrence $\neq$ Completion. Authentication $\neq$ Authority.

8.1 Delivery Is Not Admission

A transport receipt can prove that bytes reached a declared endpoint under a binding. It cannot decide whether the receiver accepts the source, profile, authority, disclosure, evidence, or result. The producer proposes; the receiver owns its local verdict. This remains true for an in-process call as well as a cross-company network exchange.

8.2 Occurrence Is Not Completion

An Episode proves that a declared occurrence was recorded and sealed under its runtime boundary. A process exit code, tool response, generated artifact, or merged change may support a Claim, but none independently settles the Pursuit. Completion follows a visible chain:

Claim $\rightarrow$ **Assessment** $\rightarrow$ **authorized Decision** $\rightarrow$ **Admission** $\rightarrow$ **Project Cut settlement**

The Claim states what is asserted. Assessment evaluates it against facts, criteria, evidence, and residual risk. Decision records what an authorized owner chooses. Admission changes accepted state or responsibility. Project Cut then binds the resulting authorities at the project level. A system may retain rejected and conflicted Claims without allowing them to change admitted state.

8.3 Authentication Is Not Authority

Authentication can prove control of an identifier under a particular binding. It does not prove permission, truth, completion, or fact admission. Warrant authority must be explicit, scoped, time-bounded where appropriate, revocable, and non-amplifying. A delegated Warrant must remain narrower than or equal to its parent across subject, action, consequence, time, disclosure, and further delegation.

This protects both users and Agent vendors. A Hub can map identities and enforce policy according to its own customer and regulatory obligations without pretending that another Hub's authentication system governs its local facts.

8.4 Trust Starts from Facts

KFD trust is not one undifferentiated score. A Claim should identify the facts and evidence that support it, the assessment method and decision owner, what remains unverified, and who owns the residual risk. Byte verification, structural conformance, execution evidence, independent review, and natural-language judgment are different evidence classes. None should borrow the confidence of another without an explicit bridge.

Buildchain applies this discipline to releases. A release Passport can bind an artifact to source, toolchain, checks, publication coordinates, declared impact, and residual risk. It does not certify the product's natural-language meaning or make an alpha Profile stable.[1]

For vendors, the immediate path is deliberately bounded: assign technical and product owners, run a 30-day assessment of the five Agent Supply Chain layers and their exact evidence, build one adapter or conformance spike, submit protocol gaps, and decide whether to adopt, co-shape, or monitor. The value of acting early is the chance to shape an open responsibility boundary before incompatible private state models harden. That is a migration-cost argument, not a claim of endorsement or an obligation to integrate.

Independent Hubs should compete on product experience, models, tools, policy, accounts, cloud, and customer relationships. The shared protocol removes the need to recreate discovery, provenance, bounded trust, durable work exchange, and Exit as proprietary lock-in. A second conforming production Hub remains an ecosystem milestone, not present evidence.

8.5 Disclosure and Exit

Portable evidence must not require portable secrets. A Hub may exchange roots, commitments, selected fields, or intentionally withheld states while keeping raw prompts, credentials, private files, and complete Episode bodies local. Redaction is not absence and must be declared. A receiving Hub decides whether the available disclosure is sufficient for its policy.

Exit remains a trust property. Export should preserve exact profile coordinates, roots, predecessor relations, verdicts, conflicts, disclosure commitments, and binding-independent payload digests. A

user's work should not become unverifiable merely because a vendor account, model, or transport is no longer available.

9 Current State, Adoption, and Roadmap

This paper is an alpha positioning and architecture document. It recommends immediate implementation work while keeping capability claims below the public evidence ceiling.

9.1 What Exists Today

As of 21 July 2026, the public surfaces support the following statements:

Surface	Supported statement	Open boundary
Project Cut	Protocol, settlement implementation, retained first-party Cuts, and a qualified three-actor continuation case exist.	The complete ordinary user product loop and v4 release qualification remain open.
`libkungfu`	Open source runtime code, exact-source quickstarts, runtime Facts, Episodes, verification, recovery, and product-layer contracts exist.	Builders should use source-pinned proofs of concept and cite exact package and platform facts; broad production readiness is not claimed.
KFD	Package `1.0.0-alpha.40` publishes active KFD-1, 2, 3, 4, 5, and 7 plus numbered draft KFD-6 and KFD-8 through KFD-13.	Pre-stable meanings, draft activation, adopter scope, and residual risk remain explicit.
Agent Hub Profile	Transport-neutral `0.1.0-alpha.1` schemas, manifest, state model, and implementer guidance exist.	No stable certification, external vendor adoption, or plural-Hub production proof.
Runtime Profile	A rooted 100-vector alpha suite, including 35 Core and 65 Experimental vectors, and two reference adapters make the contract executable.	Passing is non-qualifying and binds only the exact adapter, suite, platform, transcript, and report.
Buildchain	Artifact, package, source, publication, and release evidence is used across the public Kungfu system.	Release evidence is not project Admission, runtime authority, or proof of product meaning.

The fixed public dogfood snapshot at `2026-07-21T04:45:00Z` records 2,740 merged public pull requests across 10 repositories in the preceding 30-day window and 148 retained public Project Cuts with 148 valid receipts at the Kungfu source commit recorded in the snapshot. These are work and evidence counts, not feature counts, Agent identity counts, external adoption, or proof that every review was an approval.[6]

9.2 The Builder Path: Inspect, Embed, Prove, Connect

1. **Inspect.** Read the exact source-linked quickstart, KFD decision coordinates, Agent Hub Profile manifest, runtime facts, and known limits. Choose the smallest host-language path and define what data remains outside the adapter.

2. **Embed.** Link `libkungfu`, initialize an isolated local runtime only through explicit product action, and project a narrow set of lifecycle events. Deliver Project Cut inspect, verify, settle, recover, and resume behavior inside the vendor's existing UX.

3. **Prove.** Exercise clean initialization, deterministic roots, crash recovery, stale and conflicting candidates, revoked or widened Warrants, independent Claim assessment, and fresh-Agent continuation. Bind every public claim to exact source, suite, platform, evidence, and residual risk.

4. **Connect.** Implement the participant-owned Hub adapter, negotiate an exact mutually supported Profile, exchange only the required responsibility evidence, and keep admission and verdict writing local.

This is not a wait-for-stability sequence. Inspect, Embed, and the first Prove loop should begin now against pinned alpha coordinates. Hub architecture and policy work should proceed in parallel. Cross-Hub production claims should wait for the exact interoperability evidence they require.

9.3 Roadmap and Gates

The near-term product gate is a low-friction time-to-first-cut experience and a complete cut-to-cut continuation across a fresh Agent context. The runtime gate is deterministic, recoverable behavior across supported host layers without hidden authority in a projection. The Hub gate is receiver-owned admission, capability negotiation, disclosure control, conflict preservation, and portable exit across at least two independently owned implementations.

Public package installation, stable compatibility, platform coverage, and any future certification program require their own release and governance evidence. A future Kungfu Cloud may implement the same KFD boundary, but it is not a prerequisite for adopting KFD or `libkungfu`, and this paper does not announce such a product.

9.4 Risks and Non-Claims

The primary risks are premature certainty and collapsed boundaries:

- treating a Project Cut as a replacement for Git, Atlas, Episode, policy, or decision authority;
- marketing runtime embedding as automatic KFD certification;
- rebuilding the runtime inside every Hub and creating incompatible local fact semantics;
- turning KFD into a central registry, cloud dependency, or mandatory universal ontology;
- exporting prompts, secrets, or private Episode bodies when rooted disclosure would suffice; and
- presenting first-party dogfood, reference adapters, or alpha suites as external adoption, endorsement, broad production readiness, or a multi-platform guarantee.

The safe alpha posture is ambitious in implementation and conservative in claims: integrate now, publish exact evidence, expose failure, and let stable promises follow independent qualification rather than precede it.

10 Conclusion

The market is moving from Agents that answer inside a chat to Agents that carry responsibility across real work. The decisive product question is no longer only how intelligent one session

can be. It is whether the work remains verifiable, recoverable, and continuable after that session disappears.

Kungfu's answer begins with the user: the first object is a Project Cut, not a transcript. The Cut states what the project formally became, what occurrence and decision evidence supports it, which uncertainty remains, and where bounded work may continue. A fresh Agent can verify that boundary instead of inheriting an unverifiable story.

The answer continues with builders: embed `libkungfu` rather than rebuilding the runtime. Keep the Hub, product experience, models, identity, policy, cloud, admission, and customer relationship. Build the KFD boundary in parallel so evidence and responsibility can move between independently owned systems without creating one central authority.

The answer remains open: KFD is independently implementable, receiver verdicts remain local, Buildchain owns release evidence rather than runtime truth, and Project Cut binds authorities without absorbing them. The architecture is designed for plural products, models, contexts, transports, and organizations.

The immediate alpha action is therefore clear:

> **Start now**
>
> Inspect the exact sources. Embed the runtime. Prove one complete Project Cut continuation. Build the Hub boundary. Connect only through evidence that both sides can verify.

Give your Agent verified context. Keep the work when the chat ends.

If the product succeeds, Kungfu is not merely another Agent tool. Its public KFD, Buildchain, trust, runtime, and Hub boundaries become an open Agent Supply Chain that other products can evaluate and implement without yielding their product authority. The current proof is intentionally narrower than that future ecosystem; keeping the distinction visible is part of the trust model.

References

[1] Kungfu Origin Technology Limited. Buildchain release evidence. `https://buildchain.libkungfu.dev`, 2026. Artifact, package, source, publication, and release provenance; not runtime or project authority.

[2] Kungfu Origin Technology Limited. For agent builders: Build your hub, do not rebuild the runtime. `https://kungfu.tech/agent-builders/`, 2026. Product architecture and source-pinned adoption boundary.

[3] Kungfu Origin Technology Limited. Kfd—kung fu decisions. `https://kfd.libkungfu.dev`, 2026. Open engineering standard, package 1.0.0-alpha.40, source commit ada402f671a878ff21e2f0d1c1b21c0458bac7a5.

[4] Kungfu Origin Technology Limited. Kfd agent hub profile. `https://github.com/kungfu-systems/kfd/blob/ada402f671a878ff21e2f0d1c1b21c0458bac7a5/protocols/agent-hub/README.md`, 2026. Transport-neutral profile 0.1.0-alpha.1; experimental and non-certifying.

[5] Kungfu Origin Technology Limited. Kungfu product layers and fact–episode runtime architecture. `https://github.com/kungfu-systems/kungfu/tree/`

4ae3fb33acea77055f73e2ccc882eb36bbeb9186/docs, 2026. Reference implementation architecture at source commit 4ae3fb33acea77055f73e2ccc882eb36bbeb9186.

[6] Kungfu Origin Technology Limited. Kungfu public dogfood evidence snapshot. https://libkungfu.dev/dogfood-evidence.json, 2026. Fixed observed snapshot kungfu-systems-2026-07-21T04:45:00Z; not live analytics or external adoption evidence.

[7] Kungfu Origin Technology Limited. Project cut-centered product loop. https://github.com/kungfu-systems/kungfu/blob/4ae3fb33acea77055f73e2ccc882eb36bbeb9186/docs/adr/ADR-0127-project-cut-centered-product-loop.md, 2026. Accepted architecture decision and linked qualification contract; public product-loop release qualification remains incomplete.