

ENGINEERING ARCHITECTURE

# Kiduna Protocol and Stack

A simple implementation baseline for Agency Servers, Agency Clients, identity, permissions, agents, data, governance, and on-chain accountability.

**ARCHITECTURE RULE** The graph service is the sole policy and command boundary. Models, personas, surfaces, channel adapters, and integrations never authorize themselves.

Version 1.0 · 10 July 2026

Derived from Kiduna Kit v1.1 (spec v5.2 + Design R6)

*Status: engineering baseline; unresolved implementation choices are identified explicitly*

## How to use this document

This document is the implementation map for engineers working on the Kiduna protocol and stack. It states the boundaries that must remain invariant, assigns responsibilities to components, and identifies choices the current specification has not yet settled. It is intentionally narrower than the complete Kiduna Kit: product narrative and visual design remain in the source specification.

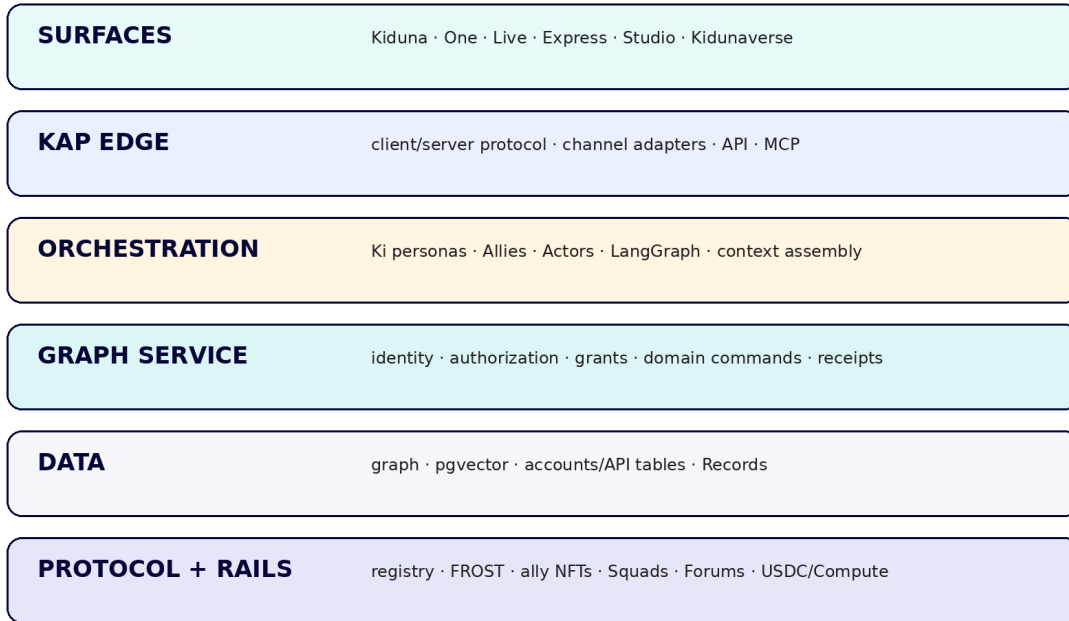
**NORMATIVE PRECEDENCE** Where this document conflicts with the Kiduna Kit, the Kit controls. Where the Kit is silent, this document marks the matter as an engineering decision rather than inventing a rule.

## Contents

- 1. Architecture at a glance
- 2. Non-negotiable invariants
- 3. System layers and responsibilities
- 4. Domain and data architecture
- 5. Identity, authority, and access
- 6. Command execution model
- 7. KAP, registry, and on-chain boundary
- 8. Agent orchestration
- 9. Surfaces and integrations
- 10. Observability and the Sentinel
- 11. Deployment and ecosystem interoperability
- 12. Engineering sequence and acceptance gates
- 13. Decisions still required
- Appendix A. Source map

## 1. Architecture at a glance

Kiduna is a federated agentic system with one shared policy model. Members interact through Agency Clients and external channels. A single orchestration system presents Ki as personalized Allies and runs scoped Actors. Every meaningful read, write, tool call, vote, payment instruction, and organizational action passes through the graph service. The graph service evaluates identity, access, grants, codes, roles, policies, and conflicts before executing a named domain command. Off-chain stores hold context and organizational state; the chain holds only the identities and transactions whose legal traceability requires it.



One policy boundary: all reads, writes, and tool calls pass through the graph service.

Figure 1. Logical architecture. The graph service is the enforcement seam between probabilistic agents and deterministic state changes.

Specification basis: *foundation.md* §§1–6; *orchestration.md* §§1–8; *protocol.md* §§1–5; *integrations.md* §§1–3.

## 2. Non-negotiable invariants

- **One policy boundary.** Every command and protected read is authorized in the graph service. Prompts, models, personas, clients, adapters, and integrations are not enforcement points.
- **Source-only instruction.** A member is the Source of their Ally. Only the Source may instruct it. Everyone else contributes context or a request within an existing grant. A co-owned Ally takes instruction through the Squads wallet holders' vote.
- **Context is shared; authority is not.** Facts about a subject are stored once with provenance and access. Personas receive a permission-filtered view; they never receive duplicated private stores that can drift.
- **Four access levels.** public, private, secret, and personal are one enum used across information and social containers. Personal is never grantable. Secret requires a code even for discovery.
- **Vectors are meaning, never authority.** pgvector proposes relevant material. The graph service decides whether the caller may traverse to or use it.
- **Named commands, not raw CRUD.** All state changes resolve to registered, deterministic domain commands with role, policy, permission, and state validation.
- **Receipts cannot lie.** Human-readable receipt sentences are generated from the exact command parameters. A command without an honest renderer displays its raw name.
- **Minimal chain.** Put an object on-chain only when legal accountability requires traceability. Chat, context, relationships, grants, wisdom, skills, and training remain off-chain.

- **Registered, not trusted.** Registration proves traceability to a member, Ally, alliance or organization, and legal entity. It is not a security or truth verdict.
- **Simulations cannot touch real rails.** Sim-flagged Moves use structurally separate simulated wallets and Compute; no real financial edge is reachable.
- **Integrations are tools.** An integration has grants, not standing: it cannot vote, hold sovereignty, or widen its own scope.

*Specification basis: foundation.md §§1–7; orchestration.md §§1–4; protocol.md §§1–5; actions.md §§1–4; integrations.md §3.*

### 3. System layers and responsibilities

| Layer          | Responsibility                                                                                                                                   | Primary implementation                                              |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|
| Agency Clients | Render Chat/Live/account/browser/build experiences; collect human intent and signatures; never decide authorization.                             | Flutter/Flame first-party surfaces; third-party KAP clients         |
| KAP edge       | Authenticate clients and servers; carry identity, context, requests, results, receipts, and registrations between ecosystems.                    | KAP protocol; thin channel adapters; API/MCP edge                   |
| Orchestration  | Assemble permission-filtered context; route to the correct Ally persona or Actor; choose candidate actions; execute only through graph commands. | LangChain/LangGraph; LangSmith tracing                              |
| Graph service  | Own identity resolution, access, grants, roles, policies, conflicts, command validation, receipt generation, and transactional state changes.    | Authoritative application boundary                                  |
| Data           | Store graph relationships, context metadata, Records, semantic vectors, and ordinary account/API records.                                        | Graph engine + pgvector + Postgres tables                           |
| Protocol/rails | Anchor accountable identities and settlements; manage wallets, Compute, Forums, and the decentralized registry.                                  | Solana-facing components; FROST; Squads; MetaDAO base; USDC/Compute |
| Integrations   | Expose outside capabilities under scoped Tool grants and expose Kiduna commands to authorized outside callers.                                   | MCP, APIs, local package protocol, Telegram/Bluesky/email/browser   |

*Specification basis: surfaces.md §§1–6; orchestration.md; foundation.md; protocol.md; integrations.md.*

## 4. Domain and data architecture

### 4.1 Core domain model

The graph is the organizing model for people, agents, relationships, organizations, authority, information, actions, and Records. The primary organizational progression is Ecosystem → Organization → Alliance → Guild, with Members and their Allies participating across those containers. Institutions are a special type of Alliance outside the DUNA hierarchy and have no vote.

| Node               | Engineering meaning                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------|
| Ecosystem          | One server-side installation, created from a Genesis Profile; interoperates over KAP.                     |
| Member / Source    | A person with one FROST wallet and one Ally across all organizations.                                     |
| Ally               | Ki personalized to a Source; anchored as an NFT in the Source wallet; the only member-bound agent family. |
| Actor              | A scoped, non-sovereign worker: operator, elector, performer, background worker, or Sentinel.             |
| Relationship       | First-class member↔member bond containing independently assigned grants and history.                      |
| Guild              | Named sharing scope with no wallet and no chain footprint.                                                |
| Alliance           | Working group with a Squads wallet, purpose, membership, and organization context.                        |
| Organization       | Registered DUNA with WV Org ID, treasury, Forum, policies, and a Squads wallet.                           |
| Institution        | Outside legal entity represented as a non-voting special Alliance with enrollment and sponsorship rules.  |
| Code / Tool / Item | Portable signed authority; an operable external capability; and any shareable information or artifact.    |
| Action / Record    | Registered capability and the provenance-carrying account of what happened.                               |

Specification basis: *foundation.md §2*; *orchestration.md §1*; *protocol.md §§1a–2*.

## 4.2 Storage responsibilities

| Store           | Owns                                                                                                                    |
|-----------------|-------------------------------------------------------------------------------------------------------------------------|
| Graph           | Authoritative relationships, scopes, grants, roles, policies, Actions, Records, organizational truth, and provenance.   |
| pgvector        | Embeddings for permission-scoped semantic retrieval. Search results are candidates; authorization remains graph-native. |
| Postgres tables | Accounts and ordinary API serving. They must not become a parallel authorization or relationship model.                 |
| Object storage  | Large files and media may be referenced by graph Items/Records; exact implementation is not specified in the Kit.       |
| Chain           | Member wallet, Ally NFT, Alliance wallet, Organization wallet+WV binding, Compute movement, and vote/settlement trace.  |
| Observability   | LangSmith traces for persona actions, rendering, and grant decisions, linked back to graph state.                       |

**IMPLEMENTATION DECISION** The Foundation specifies an in-house graph; the detailed architecture proposal names Apache AGE on Postgres. Select and record the graph engine, but preserve one graph model, one graph-service API, and structural authorization.

*Specification basis: foundation.md §§1, 5–6; kiduna-architecture.html §§1–6; orchestration.md §8.*

### 4.3 Context and retrieval

1. Resolve the caller, Source, acting persona, container, and requested action.
2. Compute the caller's reachable subgraph from access level, Relationship grants, Code claims, roles, and policies.
3. Run semantic retrieval only within that authorized scope.
4. Traverse from matching Items/Records to subjects, owners, provenance, containers, and related policies.
5. Assemble the minimum context required for the task, preserving source weights and citations.
6. Answer or propose an Action. Any write or tool call re-enters the command path; retrieval never grants execution authority.
7. Write resulting Records and embeddings with access, provenance, and subject links.

*Specification basis: foundation.md §§3–5; orchestration.md §§2, 6; kiduna-architecture.html §§5–6.*

## 5. Identity, authority, and access

### 5.1 Identity chain

The protocol's traceability chain is: Member FROST wallet → Ally NFT → Alliance or Organization Squads wallet → registered DUNA → WV Secretary of State Org ID. Kinship Codes carry the relevant addresses as signed JWT claims so the chain can travel through ordinary web and messaging infrastructure.

**MEMBER-FACING VOCABULARY** Render registered with its trace, or unregistered — a stranger, not a threat. Never render trusted or accountable as a verdict.

*Specification basis: protocol.md §§2, 4; foundation.md §2.*

### 5.2 Access decision

| Level    | Required behavior                                                     |
|----------|-----------------------------------------------------------------------|
| public   | Discoverable and readable without permission.                         |
| private  | Existence is visible; content requires an author grant or valid Code. |
| secret   | Not discoverable; a valid Code is required even to find it.           |
| personal | Member and their Ally only; cannot be granted or shared.              |

Every protected operation evaluates, in order: authenticated identity; Source relationship; acting container; access level; Relationship or container grants; Code claims and expiry; Tool scope; role/capability; organization policy; state preconditions; and structural conflicts such as Institution refusal. A denial returns no protected data and produces no side effect.

*Specification basis: foundation.md §§3–4; protocol.md §2; integrations.md §§1–3.*

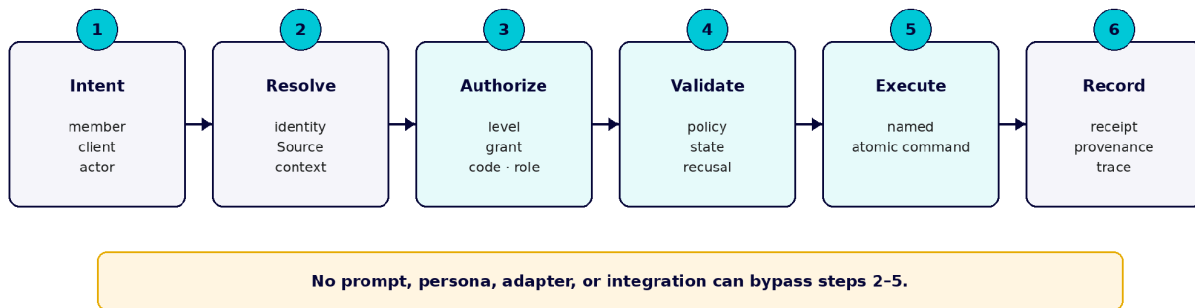
### 5.3 Kinship Codes

A Kinship Code is both invitation and portable credential. At minimum, the current specification requires issuer, scope, access level, expiry, uses, binding, and claims referencing the issuer Member wallet, Ally NFT, and relevant Alliance or Organization address. Verification must cover signature, expiry/use limits, binding, registry resolution, current grant/policy state, and revocation. The Kit does not yet define the JWT profile, algorithms, key rotation format, or revocation transport; these remain protocol decisions.

*Specification basis: foundation.md §2; protocol.md §4.*

## 6. Command execution model

Kiduna exposes capabilities as registered Actions that resolve to named graph-service commands. Natural language may select and parameterize an Action, but only deterministic code authorizes and commits it.



*Figure 2. The command path. Authorization and validation occur again for every state change and tool call.*

### 6.1 Required command properties

- **Named.** A stable domain verb such as `transfer_usdc`, `admit_member`, `create_alliance`, or `rotate_keys`—not arbitrary record mutation.
- **Typed.** Parameters are schema-validated, identity-bound, and sufficient to generate an honest receipt.
- **Authorized.** The graph service evaluates identity, Source, grants, codes, roles, policy, and conflicts.
- **Deterministic.** The same accepted parameters and state produce a predictable set of effects.

- **Atomic.** A proposal's command set either completes as the specified organizational act or fails without partial organizational truth.
- **Traceable.** The result creates a Record with actor, authority basis, parameters, outcome, provenance, and external settlement references.
- **Render-bound.** The human sentence is generated from command parameters, not supplied as independent prose.

*Specification basis: protocol.md §3; foundation.md §§2, 6; actions.md §§1–4.*

## 6.2 Read, act, and settle

Separate three concepts even when one member experience spans them:

- **Read:** permission-scoped graph/vector retrieval with no side effect.
- **Act:** a graph-service command changes Kiduna state and creates a Record.
- **Settle:** a web or chain rail finalizes money, Compute, wallet, or governance effects and returns a verifiable reference.

The exact asynchronous settlement, retry, idempotency, and reconciliation design is not specified in the Kit and must be defined before financial commands ship.

## 7. KAP, registry, and on-chain boundary

### 7.1 KAP

KAP is the protocol connecting Agency Servers to other Agency Servers and to Agency Clients. One installed server-side stack is an Ecosystem. The Genesis Ecosystem is Kiduna; other ecosystems are peers, not subordinates. KAP must preserve the same identity, authority, access, registration, command, receipt, and trace semantics across implementations.

*Specification basis: protocol.md §1a; foundation.md §2.*

### 7.2 Placement rule

| Placement     | Scope                                                                                                                                                             |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| On-chain      | Member FROST wallet; Ally NFT; Alliance Squads wallet; Organization Squads wallet + WV Org ID binding; Compute movement; votes/proposal outcomes with settlement. |
| Off-chain     | Chat; context; relationships; grants; Guilds; wisdom; skills; lineage detail; Items; most Actions and Records; simulations.                                       |
| Decision test | If legal accountability requires independent traceability, consider the chain. Otherwise keep it off-chain.                                                       |

*Specification basis: protocol.md §§1–2, 5; foundation.md §§2, 6–7.*



## 7.3 Governance

Each Organization has a Forum. A proposal carries the exact duna commands that will execute if it passes. Members see a generated receipt, discuss, and set one equal, free pass/fail token. Passed proposals execute their command set and become Policy nodes. The graph service enforces Institution conflict recusal at the vote command.

**GOVERNANCE BOUNDARY** Deciding is never fundraising. Holdings do not change vote weight, role, badge, or standing.

*Specification basis: protocol.md §§2–3; foundation.md §6; actions.md §2.*

## 8. Agent orchestration

### 8.1 One system, many personas

Ki is the Genesis Ally. Each member's Ally is Ki personalized with the Source's permitted context, identity, Contract, handle, voice, and current container grounding. Organization hosts and Move roles are additional personas of the same orchestration system, not separately sovereign agents. Personas are cheap; authority is not.

*Specification basis: orchestration.md §1; foundation.md §2.*

### 8.2 Two agent families

| Family | Constraint                                                                                                                                   |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Allies | Member-bound; take instruction only from their Source; one Ally per member across organizations; can communicate Ally-to-Ally within grants. |
| Actors | No member and no sovereignty; run scoped work such as Forum operations, background labor, Move performance, and Sentinel regulation.         |

### 8.3 Runtime flow

8. Channel adapter authenticates the speaker and forwards the exchange with channel metadata.
9. Router resolves the member, Source, Ally persona, conversation, and acting container.
10. Context assembly retrieves permitted graph state and semantic Records, with source weights and citations.
11. Planner produces an answer and candidate Actions; the Source's Contract and graph policy filter them.
12. Pure answers render directly. Every tool call or state change goes through a graph-service command.

13. Result, receipt, citations, provenance, usage, and trace are recorded and rendered on the active surface.

*Specification basis: orchestration.md §§2–8; actions.md §§1–2.*

## 8.4 Channels

Each outside channel has one system presence, not one bot per member. Thin adapters handle transport; the middle tier owns identity, routing, source weight, permission, persona selection, and command access. A conversation between two people follows Source 1 → Ally 1 → Ally 2 → Source 2.

*Specification basis: orchestration.md §§1, 3; integrations.md §1.*

## 9. Surfaces and integrations

| Surface        | Architecture responsibility                                                                                          |
|----------------|----------------------------------------------------------------------------------------------------------------------|
| Kiduna         | Primary iOS/Android/web KAP client; Chat and Live.                                                                   |
| Kiduna One     | Chat-only entry to the same Ally and graph.                                                                          |
| Kiduna Live    | Live-only entry; hosts authored Moves.                                                                               |
| Kiduna Express | Browser extension; registration visibility and agentic web action.                                                   |
| Kidunaverse    | Account, identity, permissions, wallets, web payments, directories, protocol browser, publishing.                    |
| Kiduna Studio  | Build/create environment; context, relationships, organizations, skills, automations, MCP, and local-agent packages. |

All first-party and third-party clients use the same commands and policy checks. Payment execution remains on the web surface. Mobile and Live may initiate intent and display results, but they do not execute payment rails directly.

*Specification basis: surfaces.md §§1–6; actions.md §2; integrations.md §2.*

### 9.1 Integration contract

Inbound integrations become Tool nodes scoped by grants to a Relationship, Guild, Alliance, Organization, or member. Outbound API and MCP surfaces expose registered Kiduna Actions. Local Claude Code/Codex interoperability uses self-describing packages containing context, ask, constraints, and return location; returned work is unpacked into the graph as a recorded artifact.

**NO PRIVILEGED PATH** The Flutter app, an MCP client, a local coding agent, and a third-party application are all checked by the same graph service.

*Specification basis: integrations.md §§1–3; surfaces.md §5.*

## 10. Observability and the Sentinel

### 10.1 Operational observability

Every persona action, rendering, citation, and grant decision must trace with the graph state and authority basis it used. Members can inspect behavior through the Vigil; inspection is a product capability, not only a developer console. Usage is metered against Compute in plain language.

*Specification basis: orchestration.md §8; actions.md §2.*

### 10.2 Sentinel architecture

The Sentinel is an Actor attached to the orchestration context stream. It reads interaction fields, never scores people; readings carry provenance and decay. It acts only through a closed set of registered commands and cannot vote, touch money, sanction members, or expose internal meters. Members experience its effects and, when required, plain-language disclosures using the word vibe.

| Tier   | Purpose                                                   |
|--------|-----------------------------------------------------------|
| S-tier | Cheap continuous seven-signal read on exchanges.          |
| M-tier | Periodic context synthesis, decay, and provenance rollup. |
| L-tier | Escalation candidates and human convenings only.          |

Ship order is mandatory: observe-only measurement first; human escalation path second; ambient correction last. Hard cases—risk of harm, minors, privilege, coercion or exploitation—go directly to humans.

*Specification basis: sentinel.md §§1–7.*

## 11. Deployment and ecosystem interoperability

An Agency Server installation is an Ecosystem created from a Genesis Profile. Each ecosystem operates its own server-side stack and interoperates as a peer over KAP. The Kiduna implementation is the Genesis Ecosystem, not a control plane over other ecosystems. The stack is intended to ship under Apache 2.0; marks and brand remain separately licensed.

### 11.1 Minimum deployable services

- **KAP/API edge.** Client/server and server/server endpoints, authentication, request envelopes, and rate controls.
- **Identity and registry service.** Wallet/Ally/Alliance/Organization resolution and registration proofs.
- **Graph service.** Queries, access evaluation, domain commands, receipts, policies, Records, and reconciliation state.

- **Orchestration workers.** Persona routing, context assembly, skills, Actors, channel processing, and model calls.
- **Data services.** Graph engine, pgvector, relational account/API tables, and referenced artifact storage.
- **Protocol adapters.** FROST, Ally NFT, Squads, Forum/MetaDAO, Compute, USDC, and on-ramp integration.
- **Observability.** LangSmith traces plus service logs, command audit, and member-visible Vigil data.

The Kit does not specify service boundaries, cloud topology, queue technology, consensus, cache design, backup targets, or disaster-recovery objectives. Treat the list above as logical responsibilities, not a mandated microservice decomposition.

*Specification basis: protocol.md §1a; foundation.md §1; orchestration.md §8; integrations.md.*

## 12. Engineering sequence and acceptance gates

| Phase | Outcome                  | Acceptance focus                                                                                                                                |
|-------|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| 1     | Freeze contracts         | Define IDs, access enum including secret, Code profile, Action/command schemas, Record/provenance schema, receipt renderers, and KAP envelopes. |
| 2     | Harden graph service     | Make it the only read/write/tool boundary; add structural grants, Source checks, role/policy evaluation, and recusal.                           |
| 3     | Complete data plane      | One subject-keyed context model; pgvector scoped retrieval; graph traversal; provenance; migration away from parallel authority stores.         |
| 4     | Implement identity chain | FROST member wallet, Ally NFT, Squads identities, WV Org ID validation, registry publication, Code verification.                                |
| 5     | Ship core command loop   | Baseline Actions; atomic commands; generated receipts; Records; idempotent external settlement and reconciliation.                              |
| 6     | Connect orchestration    | Ki personas, Source routing, Contracts, candidate Actions, channel adapters, LangSmith traces.                                                  |
| 7     | Connect surfaces         | Account/payment web flow first; then Kiduna Chat; then Express/Studio/Live according to product cut.                                            |
| 8     | Add Sentinel safely      | Observe-only; human escalation; ambient correction after evidence and review.                                                                   |
| 9     | Prove federation         | Second ecosystem interop test over KAP with independent registry and command verification.                                                      |

### 12.1 Release gates

- No protected object is retrieved and filtered after the fact; unauthorized paths are not traversed.
- The secret access level exists end-to-end before code-gated content ships.
- No non-Source message becomes an instruction.

- Every external side effect has an authorization basis, idempotency strategy, terminal status, and reconciliation path.
- Every proposal receipt is derived from executable parameters and round-trips in tests.
- Institution recusal is enforced by the vote command, not by UI convention.
- Sim-flagged environments cannot resolve production wallet or settlement commands.
- Third-party and first-party clients pass the same authorization suite.
- Registration UI never implies trustworthiness.
- Sentinel hard boundaries and human escalation are tested before ambient intervention is enabled.

## 13. Decisions still required

| Decision                | What must be specified                                                                                                            |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| Graph engine            | Foundation says in-house graph; engineering proposal says Apache AGE. Choose engine and migration path.                           |
| KAP wire protocol       | Transport, authentication, schemas, versioning, discovery, federation errors, and compatibility are not yet defined.              |
| Kinship Code profile    | JWT algorithms, key discovery/rotation, audience/binding semantics, revocation, replay control, and canonical claims.             |
| Command contract        | Canonical envelope, idempotency key, optimistic concurrency, atomicity boundary, error taxonomy, and compensation rules.          |
| Settlement architecture | Queues/workers, confirmations, retry windows, bridge timing, reconciliation, and partial external failure behavior.               |
| Data operations         | Graph schema evolution, object storage, retention/deletion, backups, RPO/RTO, encryption, regional placement, and privacy export. |
| Model/runtime policy    | Model selection, tool sandboxing, prompt/version rollout, evaluation suites, fallback behavior, and cost budgets.                 |
| Ecosystem operations    | Genesis Profile schema, server bootstrap, registry namespace, software update governance, and peer discovery.                     |
| Security model          | Threat model, signing custody, account recovery, admin/break-glass powers, rate limits, abuse response, and incident process.     |

Design Round 6 also contains ten product/UI decisions. They should be resolved in the design queue unless they change an API or data contract; only then should they enter the engineering architecture decision record.

*Specification basis: open-questions.md; design-r6/OPEN-QUESTIONS-R6.md; gaps explicitly absent from the cited architecture tracks.*

## Appendix A. Source map

| Source                    | Used for                                                                      |
|---------------------------|-------------------------------------------------------------------------------|
| foundation.md             | Graph, nodes, levels, grants, memory, Records, money state, simulations       |
| protocol.md               | KAP, ecosystem model, chain boundary, governance, registry, Kinship Codes     |
| orchestration.md          | Ki, Sources, personas, Allies/Actors, channels, primitives, runtime           |
| actions.md                | Baseline Actions, sovereign acts, command extension rule, Chat/Live rendering |
| surfaces.md               | Six products, payment boundary, protocol browser, Studio package protocol     |
| integrations.md           | Inbound Tools, outbound API/MCP, local agents, no-privileged-path rule        |
| sentinel.md               | Interaction-health Actor, tiering, storage, commands, human boundaries        |
| kiduna-architecture.html  | Detailed graph/pgvector proposal; Apache AGE option                           |
| open-questions.md         | Current decision queue and resolved rules                                     |
| START-HERE.md / AGENTS.md | Terminology, precedence, public vocabulary, archive rules                     |

*Source snapshot: Kiduna Kit v1.1, dated 10 July 2026; specification v5.2; Design R6.*