



ENGINEERING SPECIFICATION

Kinship Graph Architecture v1.1

The operating graph for the agentic internet

Document property	Value
Status	Build specification for the August 10, 2026 initial release and beyond
Version	1.1 July 14, 2026
Audience	Engineering, product, security, operations, counsel, and Organization builders
Replaces	Kinship Graph Workflow
Authority	Kiduna Kit v2.1, July 14 architecture sync and graph update, plus the product-owner ratifications supplied July 14, 2026

The Kinship Graph is not primarily a question-answering knowledge graph. It is the authoritative operating context for accountable agency.

Architecture readiness is not legal authorization. Financial, legal, clinical, insurance, mortgage, and custody features remain subject to their stated deployment gates.

Document map

Use this specification as the normative engineering handoff. Historical source files remain evidence and precedent where they do not conflict with this document.

I. Reframe and invariants

Executive decisions

1. Why the prior workflow cannot be the build specification
2. Constitutional invariants
3. Canonical vocabulary and object model

II. Operating kernel

4. The Action architecture
5. Permission and policy engine
6. Organization and legal-form architecture
7. Compute, agency pricing, lineage, and the waterfall
8. Forums, proposals, Envoys, and Policy
9. Projects and real work

III. Implementation

10. Physical data architecture
11. Graph schema registry
12. Named query and agent tool surface
13. End-to-end pipelines
14. APIs and contracts
15. Migration from the existing workflow

IV. Release and operations

16. Scale and performance
17. Security, privacy, and safety
18. Observability and operations
19. August 10 build plan
20. Acceptance gates
21. Important graph and platform tools
22. Owner ratifications incorporated in v1.1

V. Appendices

- A. Minimum Action catalog
- B. Sample named Cypher patterns
- C. Source and authority map
- D. Final architectural position

Executive decisions

This specification makes the following implementation decisions.

1. **Replace the old center of gravity.** The graph is organized around **Personas, Allies, Organizations, and Actions**, with Memberships, Projects, Relationships, Communities, Alliances, Forums, Wisdom, Institutions, and Records as first-class supporting concepts. "Users, agents, and markets" is too generic and directs engineers toward an AI retrieval product rather than an operating system for coordinated work.
2. **Reserve Member for legal Organization membership.** Product and schema use Visitor for an unidentified person, Guest for an authenticated Persona without active Membership in the Organization in context, and Member only after admission under that Organization's terms. A Persona may be a Member of one Organization and a Guest of another. `Membership` is the authority-bearing object; account creation or payment alone never creates it.
3. **Use Persona as the generic principal.** `Persona` is the durable human identity behind an Account and the Source of an Ally. Builder, creator, founder, Catalyst, luminary, designee, and Curator are roles; they do not replace Persona, Guest, or Member.
4. **Make the Organization the accountability context of every Action.** Every Action must resolve to exactly one `organization_id`. When no Organization is supplied, only the explicit Guest/Public allowlist may default to Kinship Duna, with `context_defaulted=true` recorded. Defaulting context never creates Membership or authority.
5. **Treat DUNA as a legal-form adapter, not the universal ontology.** `Organization` is the system object. West Virginia DUNA is the first `LegalForm` and `LegalRegistration` adapter. This allows other jurisdictions and entity forms later without reworking the graph.
6. **Replace Market with Forum.** An Organization can have several Forums. A Forum is a governed decision space that can include debate, optional decision-market signals, and the signed enactment process. The old one-DUNA-to-one-Market `BECAME` relationship is removed.
7. **Model two agent families.** Every authenticated Persona has exactly one Ally identity; every non-Ally agent is an Actor. Sentinel, Envoy, Operator, Profiler, CodeManager, research agent, registrar, intake agent, and domain-specific agents are typed Actor kinds. Implementation workers such as queue consumers are services, not social agents.
8. **Model Actions as a subgraph, not as an edge.** `ActionDefinition`, `ActionRequest`, `AuthorizationDecision`, `Command`, `ActionRun`, `ExternalOperation`, and `Record` are distinct. Edges express who requested, authorized, performed, received, and was affected. This is the minimum structure that preserves intent, permission, retries, confirmation, governance, settlement, and provenance.

9. **Keep one deterministic boundary.** The Graph Command Service is the only component allowed to authorize or commit protected reads and changes. Models propose. The service resolves identity and context, evaluates policy, validates state, commits local changes, starts external operations, and appends Records.
10. **Promote Projects and work to the launch core.** The graph must support engagements, work items, deliverables, milestones, agreements, split rules, invoices, credentials, and domain resources. Retrieval and conversational answers are supporting capabilities, not the product's center.
11. **Rename and normalize agent configuration.** Inform/Wisdom, Instruct/Stance, Empower/Connection, Enable/Automation, and Impart/Skill are explicit configuration relationships. A Wisdom Drop is a permissioned namespace containing Items and vector chunks.
12. **Make money auditable but do not use the graph as an accounting ledger.** The graph models economic relationships and links to purchases, allocations, accounts, and settlements. Amounts and balances are enforced by an immutable double-entry ledger in PostgreSQL; on-chain state and regulated rails remain authoritative for their own settlement.
13. **Make the graph authoritative current state, not a loose projection of three databases.** The Graph Command Service updates typed graph state, command/record tables, ledger references, and a transactional outbox in one PostgreSQL transaction. External state is reconciled by idempotent workers. There is no uncontrolled multi-database dual write.
14. **Launch as a modular monolith with workers.** Use one tested PostgreSQL plus Apache AGE and pgvector deployment, one Graph Command Service, and isolated workers for orchestration, ingestion, outbox delivery, external reconciliation, and chain/payment adapters. Split services only at measured security or scaling boundaries.
15. **Implement the full ontology now, activate risky economics by policy and feature gate.** The schema supports Compute issuance, liquidity, lineage, launch campaigns, DEX settlement, market positions, Curator allocation, and Sponsor distribution. Transferable/DEX Compute, market positions, lineage payouts, Curator allocation, and Sponsor distribution are disabled at launch until legal, accounting, tax, custody, rail, and exchange reviews approve them.
16. **Separate legitimacy from prediction.** One eligible Member, one equal governance ballot determines Organization authority under its governing principles. A USDC-funded decision-market position is information only unless a later, explicit Organization policy delegates a narrowly specified advisory effect. It never silently becomes a vote.
17. **Publish a machine-readable command authority matrix.** Every `DomainCommand` declares one required authority class: Public, Guest, Member, Role, Policy, Forum, Treasury, GovernanceMarket, ExternalAgreement, or RegulatedCredential. The registry, not prose or model judgment, is authoritative.

18. **Keep protocol invariants hard and economic policy versioned.** Identity, principal types, Source precedence, command execution, and Records are stable protocol. Liquidity floors, pricing, waterfalls, allocations, and Organization economics are versioned policies. Regulated capabilities activate independently.
19. **Make every identity portable but homed.** Every Persona, Ally, Actor, Organization, Institution, and Ecosystem has a permanent protocol ID, a Home Ecosystem, and signed federation metadata. Migration changes the authoritative home without changing the permanent ID or erasing history.
20. **Bootstrap the network through a defined Genesis sequence.** The Kiduna Genesis Ecosystem, Genesis Ally Ki, and limited Mage account exist before the first invited Persona. Ki is immediately useful through a shared versioned AllyDefinition, while every Persona receives a segregated Ally identity, private state, alias/handle, and Source boundary.

1. Why the prior workflow cannot be the build specification

The original workflow contains useful engineering work: Apache AGE, anchored graph tools, hybrid vector plus graph retrieval, query guarding, audit logging, wallets, Kinship Codes, lineage, and provenance. These should be retained. The problem is the workflow's theory of the product.

1.1 It describes an assistant, not an agentic internet

Its main pipeline is: user asks a question, a supervisor model chooses one of seven retrieval tools, a Cypher query runs, and the model writes an answer. That pipeline is valid for a read operation, but it is only one small branch of the system. It does not answer the defining engineering questions:

- What consequence is the Persona trying to cause?
- Which Organization is responsible for the consequence?
- Is the Persona, Member, Ally, Actor, role, or Forum permitted to cause it?
- What human confirmation is required?
- Which deterministic command changes local state?
- Which external system must settle the effect?
- What happens when an external operation times out, partially succeeds, or is reversed?
- Which Record proves authorization and outcome?

The replacement pipeline is therefore: **prompt or trigger -> intent -> typed ActionRequest -> authority resolution -> policy decision -> confirmation or Forum -> command -> local commit -> external operation -> Record -> Field update.**

1.2 Its nouns encode the wrong system

User treats the human as an account. **Presence** treats the Ally as a created avatar. **Worker** makes agent architecture look like a supervisor tree. **Market** collapses an Organization's legal existence into one on-chain counterpart. **KnowledgeBase** hides ownership, permission, provenance, and the meaning of Wisdom. **Connection** erases the information-bearing Relationship. The model has no Project and almost no representation of real work.

The result is a graph optimized for answering "what do I own, who referred me, and what did I earn?" rather than for operating a law practice, solar installation, clinical service, mortgage workflow, game club, festival, research collaboration, or mutual insurer.

1.3 It confuses current state, evidence, and telemetry

The workflow proposes Event nodes for onboarding status and considers projecting every skill execution into the graph. Those are different kinds of data:

- **Current graph state** answers what is true and permitted now.
- **Records** prove consequential reads, decisions, signatures, commands, results, and settlements.
- **Internal events and telemetry** explain how software moved between states and how it performed.

Only current state and durable, persona-relevant Records belong in the operating graph. High-volume spans, model tokens, retries, queue leases, and low-level skill execution details belong in partitioned tables and observability storage, with graph links only when a durable consequence exists.

1.4 It makes the graph a derived reporting layer

The prior workflow says three relational databases remain sources of truth and the graph is their combined projection. That guarantees lag, duplicate identity, cross-database inconsistency, and ambiguous authorization. A permission decision cannot rely on an eventually synchronized copy.

The new rule is narrower and stronger: the Graph Command Service owns authoritative organizational state and authorization facts. Relational tables in the same PostgreSQL transaction own strict control-plane and ledger constraints. External rails own their settlements. Projections can be rebuilt from Records and events, but protected decisions never depend on a stale analytical graph.

1.5 It represents legal and economic relationships inaccurately

An Organization does not "become" a Market. It remains a real legal association and may operate several Forums. An Offering purchase does not pay lineage; only a Compute purchase can do so under the stated policy. An Institution is not a Persona with a larger subscription. It is an external legal

principal that can enter agreements, sponsor Personas or Members, appear in lineage, and have progeny, but cannot acquire Membership or human sovereignty merely by paying.

1.6 What remains worth keeping

- Apache AGE as the selected graph engine, subject to version pinning and release testing.
- PostgreSQL and pgvector co-location for transactional graph, relational controls, and semantic retrieval.
- Stable named graph queries with bounded traversal, limits, and caller scope.
- Closure-bound or token-bound caller context as defense in depth.
- Graph query auditing and prevention of unbounded traversal.
- Wallet, Code, lineage, provenance, document/chunk/entity, and external-tool concepts after renaming and normalization.
- Hybrid retrieval: semantic candidates first, permission-safe graph expansion second.

From Persona intent to accountable consequence

Visitor, Guest, and Member authority is resolved before every consequential command

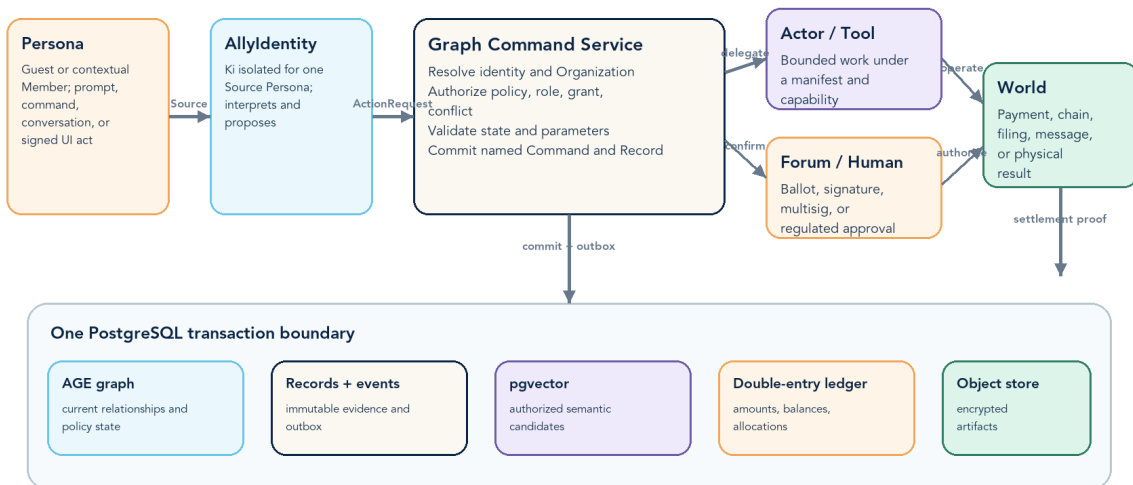


Figure 1. The revised operating architecture

2. Constitutional invariants

These invariants are implementation tests. A release that violates any P0 invariant is not a partial success; it is the wrong system.

2.1 Authority

- An authenticated Persona is the only free-form Source of instructions to their Ally.
- Messages from any other Persona, Member, website, email, agent, Tool, or Ecosystem are context or a request, never binding instruction.
- A Visitor has no Ally and no durable authority. A Guest may use only the Public/Guest allowlist. Member authority exists only through an active Membership in the exact Organization in context.
- An Ally never self-authorizes, votes, signs, widens a grant, changes its own budget, or alters its own command allowlist.
- An Actor has only the reads, Tools, commands, budget, duration, and Organization contexts declared in its immutable versioned manifest.
- Every consequential Action resolves to one Organization and one authority class.
- Default deny applies when identity, context, policy, state, or authority is ambiguous.

2.2 Privacy and access

- Every protected resource has one of four access levels: public, private, secret, or personal.
- Personal is visible only to the owning Persona and their Ally and is never grantable.
- Secret is not discoverable through search, counts, errors, timing, cached Scenes, or federation.
- Private may be discoverable but is retrievable only under a valid grant, Code, agreement, or policy.
- Authorization occurs before graph retrieval and before vector retrieval. "Retrieve then filter" is forbidden.
- Credentials and raw secrets never live in graph properties, vector metadata, model prompts, or ordinary logs.

2.3 Actions and Records

- Language captures intent; it does not mutate state.
- Every mutation uses a registered named Command with a versioned parameter schema.
- Every command has an idempotency rule, authority class, preconditions, effects, receipt renderer, error taxonomy, and tests.
- Receipt language is generated from exact command parameters. No independent prose field may describe the consequence.
- A local transaction can be atomic; an external filing, payment, title transfer, email, chain transaction, or physical-world act is a saga with explicit pending and terminal states.
- Corrections supersede Records; they do not rewrite history.

2.4 Organizations and work

- Organization is jurisdiction-neutral. A DUNA is one verified legal form.
- Every Project has exactly one accountable Organization.
- Communities, Alliances, Relationships, and Projects exist in Organization context; Personas and agents may participate across many Organizations, but Member-only authority is always resolved from the active Membership for that Organization.
- Organizational status is derived from verified registration and lifecycle state, not a manually edited label.
- Engagement, Project, Alliance, and Institution are separate concepts even when linked.

2.5 Money and deciding

- Compute is described and recorded as prepaid usage credits that power intelligent agents.
- Only a completed Compute purchase can trigger a lineage/waterfall calculation. Offerings, work income, votes, grants, and treasury distributions cannot.
- Every waterfall calculation binds to the exact version of the WaterfallPolicy and a frozen lineage snapshot.
- Governance ballots and decision-market positions are separate objects. One eligible Member has one equal ballot. A USDC-funded position is a distinct information signal and must never silently become a wealth-weighted legal vote.
- Financial balances come from a double-entry ledger or verified external rail, not from summing graph edges.
- The platform submits pre-authorized instructions and reconciles results; it does not improvise where funds go.

2.6 Simulation and observability

- Simulation resources are structurally incapable of reaching real payment, chain, filing, messaging, or regulated Tools.
- Operational logs contain IDs, reason codes, versions, latency, cost, and error class, not raw personal prompts or secret content.
- No Persona receives a hidden global score, inferred reputation rank, productivity score, or behavioral rank. Directional trust declarations may exist on a Relationship, but they are not transitive authority and are never aggregated into a secret social score.

2.7 Identity, invitation, and trust

- A pre-registration invite never creates a Persona, Guest, Member, or Relationship. It creates a purpose-limited [InvitationProspect](#) and [RelationshipIntent](#) owned by the inviter.
- Claims about an invitee remain attributed to the inviter, expire under policy, and are not promoted into the invitee's Profile without review and consent.
- Person-specific first-entry Codes are single-use and expire after 15 minutes by default. Generalized Codes may use a different explicit expiry but default the recipient to untrusted and carry no prebuilt Profile.
- The platform never sends outbound communication to an unregistered Visitor. The inviter delivers the Code out of band. Registered Personas may receive signed in-system invitations.
- Trust is directional and scoped to a Relationship: low, medium, or high, with author, basis, visibility, and timestamp. It never grants command authority by itself.
- Registration/verification and trust are different. Verification states who or what vouched for an identity; trust states one principal's declared confidence in another.
- No trust level is public by default. A Persona may explicitly disclose their own directional declaration; the counterparty's declaration remains separately controlled.

3. Canonical vocabulary and object model

3.1 The four centers and human states

Persona. The durable human principal behind an Account. Persona is the generic term when Visitor/Guest/Member status is unknown, varies by Organization, or is irrelevant. An authenticated Persona is the Source of exactly one Ally identity across the network.

Visitor. An unauthenticated or unidentified person. A Visitor may use public reads and begin account creation but has no durable principal authority, Ally, Membership, wallet authority, or protected access.

Guest. An authenticated Persona who does not have an active Membership in the Organization in context. Guest authority is restricted to the explicit Guest command allowlist. For Kinship Duna at launch, a Guest may fund at least 10 USDC of Compute without becoming a Member.

Member. A Persona with an active, legally effective [Membership](#) in the Organization in context, admitted under that Organization's governing terms. A Persona may simultaneously be a Member of Kinship Duna and a Guest of another Organization. Payment is at most one admission precondition; it is never sufficient by itself.

Ally. The authenticated Persona's single agentic counterpart. The product presents the shared canonical Ally Ki, but the graph stores a separate [AllyIdentity](#) per Persona bound to one versioned

AllyDefinition. This preserves the "one Ally" product experience without creating a cross-persona privacy or authorization singleton. Each Ally may have a Persona-selected display name and globally unique handle. It is personalized through Stance, Wisdom access, Connections, Automations, Skills, grants, and Organization context. It interprets, explains, proposes, coordinates, and delegates, but cannot cross the deterministic boundary on persuasion alone.

Organization. The accountability container in which Actions occur. At launch, the primary verified legal form is a West Virginia DUNA. An Organization has purpose, functions, members, agents, policies, Projects, Forums, accounts, Compute configuration, Wisdom, Stance, Connections, Automations, and Skills.

Action. A typed, consequence-bearing request and its accountable execution history. Actions are how Personas, Members, and Organizations communicate, create, collaborate, coordinate, build, decide, transact, and affect the world.

3.2 Supporting social and work objects

Relationship. A first-class node joining exactly two authenticated Personas in exactly one Organization context. The same two Personas can have different Relationships in different Organizations. Each side authors its own directional trust declaration, grants, and statements. Relationship existence does not imply Membership.

Community. Three or more Personas gathered around a purpose or sharing scope. It has no treasury authority by default.

Alliance. A formal working group with membership, charter, roles, and optionally a shared wallet inside an Organization. It is not a Project; it may own or sponsor many Projects.

Project. A collection of Personas, Members, agents, resources, agreements, Actions, work items, and Records organized to accomplish a purpose inside one Organization. Project is Studio's central organizing primitive.

Engagement. A commercial or professional agreement among an Organization and one or more parties. It defines scope, terms, milestones, compensation, credentials, conflicts, and acceptance. An Engagement may fund one or more Projects.

Institution. A first-class outside legal principal, such as a university, corporation, government agency, nonprofit, LLC, or church. It has its own identity, legal metadata, verification evidence, Agreements, delegates, wallets, authority chain, and optional lineage. It is not an Organization, Persona, Guest, or Member and has no Member ballot merely by paying or signing an Agreement.

Forum. A governed decision space owned by an Organization. An Organization may have multiple Forums for different domains, cadences, or risk levels.

Proposal. A versioned request for one or more Organization Commands. Before opening, the proposal must render an honest receipt from those exact parameters and disclose conflicts and dependencies.

Policy. The authoritative result of an enacted proposal or other valid policy command. Policy is queryable state with effective dates, scope, version, and source Record.

Offering. A paid product or service other than Compute or raw model consumption. An Offering purchase never triggers lineage.

3.3 Agent model

The schema uses one **Agent** identity contract with two families:

- **Agent:Ally**: exactly one Source Persona, one active Ally identity per authenticated Persona, bound to the shared canonical Ki AllyDefinition.
- **Agent:Actor**: no human Source and no sovereignty; owned or commissioned by an Organization, Alliance, Project, Institution, or Ecosystem.

Actor kinds include:

- **sentinel**: observes interaction health, detects manipulation/coercion or unsafe context, forces an explicit warning/confirmation where policy allows continuation, and escalates to a named human or role. It cannot waive a hard constitutional denial.
- **envoy**: carries a Member's typed Forum or Alliance instruction and may cast an authorized equal ballot or place a separately authorized market position after the required confirmation.
- **operator**: administers Forum mechanics and execution without choosing Member intent.
- **profiler**: collects inviter-authored prospect claims, public research allowed by policy, and onboarding questions without treating the prospect as an existing Persona.
- **code_manager**: creates, scopes, signs, expires, redeems, and revokes Codes for Ecosystem entry, Relationship proposals, Membership, Projects, Alliances, Communities, Wisdom, and other registered purposes.
- **registrar**: assembles and monitors official registration evidence.
- **researcher**: gathers sources and creates draft Wisdom.
- **project_steward**: derives work state and offers next Actions.
- **domain**: an Organization-defined kind created through a registered DomainPackage.

Queue consumers, indexers, reconcilers, and schedulers are **ServicePrincipal** records, not Actors. They do not appear socially in the Field. A **MagePrincipal** is a separate, tightly limited bootstrap administrator for an Ecosystem; it is never a Persona, Member, Ally, or general superuser.

3.4 Agent configuration: verb and noun

Verb	Noun	Graph meaning	Runtime rule
Inform	Wisdom	Grant an agent permission to retrieve one or more Wisdom Drops	Retrieval remains subject to item access, purpose, Organization context, and conflicts
Instruct	Stance	Apply versioned instructions, tone, disclosure, response preferences, and channel policy	System and Organization policy outrank Stance; only the Source can alter personal Ally Stance
Empower	Connection	Bind a scoped account or external capability to an agent	The graph stores a credential reference and scopes, never the credential
Enable	Automation	Activate a trigger-to-Action rule	Automation creates an ActionRequest; it never bypasses authorization or confirmation
Impart	Skill	Attach a versioned procedure or package to an agent	Skills can propose commands only from the agent's closed allowlist

3.5 Wisdom

A **WisdomDrop** is the permission and namespace boundary for material placed in the vector store. It is owned by a Persona, Organization, Alliance, Project, or Institution and credited to the Persona or principal that supplied it. It contains Items, Documents, Artifacts, and derived chunks.

Required WisdomDrop properties:

Property	Requirement
<code>id, protocol_id</code>	Stable external identity; never expose an AGE internal ID
<code>name, description, handle</code>	Human and agent discoverability
<code>owner_principal_id, credited_person_id</code>	Control and attribution are distinct
<code>organization_id</code>	Accountability context; Kinship Duna only when legitimately defaulted
<code>access_level</code>	public, private, secret, or personal
<code>purpose_tags</code>	Allowed uses such as research, client matter, health support, public education
<code>namespace_key</code>	Vector partition/namespace reference, not a secret
<code>retention_policy_id, legal_class</code>	Retention, privilege, health, financial, minor, employment, or general

Property	Requirement
version, status, created_at, superseded_by	Immutable versions and lifecycle

Private access may be unlocked by Code, Relationship grant, role, Agreement, Alliance, Organization policy, or a specific principal grant. Secret material is not listed before authorization. Personal material cannot be unlocked at all. Access changes generate Records and immediately invalidate cached reachability and vector-scope keys.

The operating graph

Personas and agents cross containers; Membership authority is Organization-specific

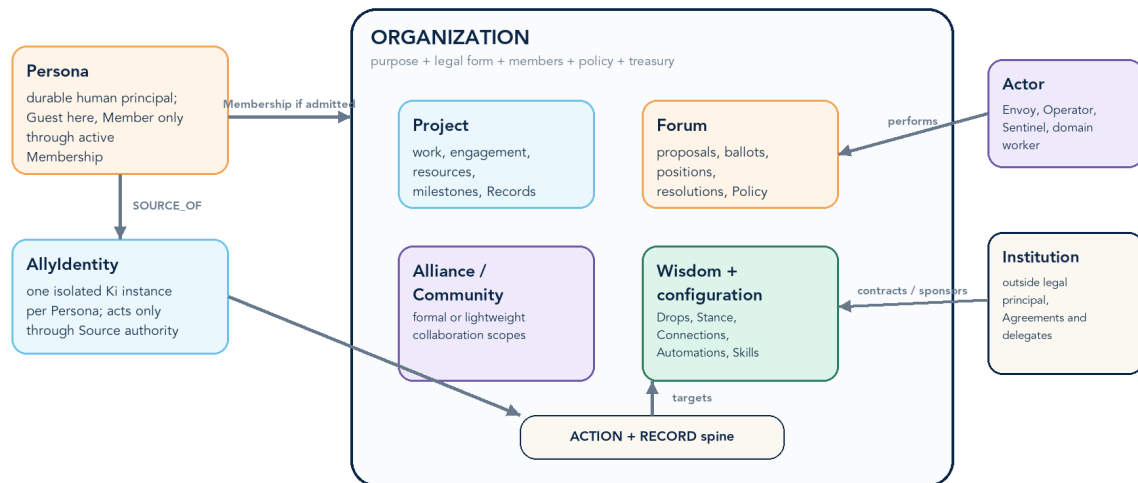


Figure 2. Core domain graph

4. The Action architecture

4.1 Why Actions are nodes and edges

An edge is appropriate for a compact current relationship such as `OWNS`, `IN_PROJECT`, or `HAS_ACTIVE_MEMBERSHIP`. An Action is not merely a relationship. It has a definition, parameters, requester, Source, Organization context, authority requirement, confirmation, state machine, attempts, side effects, errors, results, and evidence. It must therefore be reified.

The design uses nodes for the durable nouns and edges for their semantic roles:

```

(Persona) -[:SOURCE_OF]->(Ally)
(Ally) -[:CREATED_REQUEST]->(ActionRequest)
(ActionRequest) -[:USES_DEFINITION]->(ActionDefinition)
(ActionRequest) -[:IN_ORGANIZATION]->(Organization)
  
```

```
(ActionRequest)-[:TARGETS]->(Project|Resource|Principal)
(ActionRequest)-[:PROPOSES]->(Command)
(AuthorizationDecision)-[:DECIDES]->(ActionRequest)
(ActionRun)-[:ATTEMPTS]->(Command)
(Actor|ServicePrincipal)-[:PERFORMED]->(ActionRun)
(ActionRun)-[:STARTED]->(ExternalOperation)
(ActionRun)-[:PRODUCED]->(Record)
```

4.2 ActionDefinition

An ActionDefinition is a versioned registry entry. It answers "what kind of thing can be done?"

Required fields:

- **key** and semantic version, for example `project.work.assign@1.2.0`.
- family: converse, wisdom, relationship, membership, organization, project, work, governance, treasury, compute, commerce, identity, security, domain.
- JSON Schema for parameters and result.
- allowed requester principal types and allowed performer types.
- authority class and required roles, grants, policy predicates, credentials, and conflict checks.
- Organization and Project context requirements.
- risk class: read, reversible, consequential, financial, sovereign, regulated, emergency.
- confirmation mode: none, acknowledge, explicit confirm, fresh authentication, press-and-hold signature, multisig, or Forum.
- idempotency scope and concurrency key.
- command mapping and handler version.
- machine receipt renderer and redaction rules.
- data use, retention, and telemetry policy.
- compensating command, if one exists.
- simulation eligibility and hard prohibition on real rails when `sim=true`.

4.3 ActionRequest

An ActionRequest is a specific desired consequence. It is created from a Persona prompt, a signed UI control, an Automation trigger, an Actor output, a Forum result, an external event, or another completed Action.

Key fields:

- `action_definition_id`, `definition_version`, and validated parameters.
- `source_persona_id` when the request originates from a Persona's Ally.

- `requester_principal_id` and authenticated session/proof reference.
- `ally_id` and optional delegated Actor.
- `organization_id`, optional `project_id`, and target object IDs.
- `origin`: field, studio, express, channel, automation, forum, external webhook, federation.
- `authority_class`, requested deadline, and priority.
- `intent_record_id` referencing the original message or structured UI event under its access rules.
- `status`: proposed, awaiting_context, awaiting_authorization, awaiting_confirmation, queued, executing, waiting_external, completed, failed, denied, deferred, cancelled, expired.
- `expected_versions` for optimistic concurrency.
- `idempotency_key` bound to caller, definition, and canonical payload hash.

4.4 Machine-readable authority classes

Every DomainCommand registry entry declares exactly one primary authority class. Risk, confirmation, credential, multisig, and emergency requirements are additional overlays; they do not create ambiguous alternate classes.

Class	Example	Required basis
Public	Read public Organization purpose; begin Account registration	Public resource, safe bounded query, no protected consequence
Guest	Accept invitation; purchase qualifying Membership package; verify identity; create personal draft workspace	Authenticated Guest, Guest allowlist, exact Organization context, confirmation where required
Member	Join a Member-only Community; cast an eligible ordinary ballot	Active Membership in the exact Organization and current eligibility
Role	Assign permitted Project work; accept a deliverable	Active Membership or authorized delegate plus current role/grant and resource state
Policy	Run an approved Automation or recurring operational command	Active versioned Policy whose predicates and limits match current state
Forum	Enact constitutional change; issue Compute; approve a major Organization command	Passed Proposal containing the exact Command and parameters
Treasury	Authorize or execute a treasury movement	Treasury Policy, account authority, threshold approvals, budget, and fresh confirmation

Class	Example	Required basis
GovernanceMarket	Place or settle an optional decision-market position	Activated market policy, eligible Envoy instruction, limits, disclosures, and approved rail
ExternalAgreement	Act for an Institution or counterparty under an Agreement	Verified Institution, active Agreement, named delegate/agent, scope, and counterparty conditions
RegulatedCredential	File legal document; approve underwriting; clinical sign-off	Current license/credential, jurisdiction, Engagement, conflict check, and required human signature

`confirmation_mode` separately declares none, acknowledge, allow-once, allow-always-policy-create, explicit confirm, fresh authentication, press-and-hold signature, multisig, Forum, or credentialed-human. `risk_class` separately declares read, reversible, consequential, financial, sovereign, regulated, or emergency. Hard constitutional denials always outrank all three fields.

4.5 AuthorizationDecision

Every protected read and every ActionRequest receives a deterministic decision object. The decision stores no secret content; it stores the evaluated fact IDs and outcome.

```
{
  "decision_id": "kid:authz:01...",
  "principal": "kid:ally:01...",
  "source_persona": "kid:persona:01...",
  "action": "project.work.assign@1.2.0",
  "resource": "kid:project:01...",
  "organization": "kid:org:628407",
  "result": "allow",
  "basis": ["role:project_lead", "policy:work_assignment@3", "grant:project_scope@7"],
  "conflicts_checked": ["institution", "financial", "family", "engagement"],
  "requires_confirmation": "explicit",
  "policy_bundle": "sha256:...",
  "evaluated_at": "2026-07-14T...Z"
}
```

Decision precedence:

1. Hard constitutional deny.
2. Explicit forbid or conflict/recusal.
3. Resource state and legal/credential requirements.
4. Active Policy.
5. Specific grant or Code claim.
6. Role and Membership.
7. Public/read default where permitted.
8. Default deny.

4.6 Command

A Command is the deterministic instruction that can change Kiduna state. Commands never accept arbitrary Cypher or generic CRUD. A command registry entry defines:

- canonical name and version;
- parameter and result schemas;
- authority class;
- precondition query names;
- exact local writes;
- external operation templates;
- expected aggregate versions;
- idempotency and locking rules;
- receipt renderer;
- persona-facing error mapping;
- privacy and retention behavior;
- tests and migration compatibility.

Examples: `project.create`, `work.assign`, `wisdom.inform_agent`, `membership.admit`, `forum.proposal.open`, `forum.ballot.cast`, `compute.purchase.record`, `compute.waterfall.allocate`, `organization.registration.submit`, `treasury.payment.authorize`.

4.7 ActionRun and ExternalOperation

`ActionRun` records an attempt to execute a Command. Retries create new runs under the same `ActionRequest`, never overwrite the earlier attempt. A local Command either commits its current graph state, relational control rows, ledger references, `Record`, and outbox entry in one transaction or commits nothing.

An `ExternalOperation` represents a side effect outside that transaction: payment, chain submission, email, legal filing, DEX interaction, external account call, physical dispatch, or federated request.

Required states: `authorized`, `queued`, `submitted`, `acknowledged`, `awaiting_external`, `settled`, `failed_retryable`, `failed_terminal`, `compensating`, `compensated`, `manually_resolved`. The UI and Ally must never say "complete" while the operation is only submitted.

4.8 Record

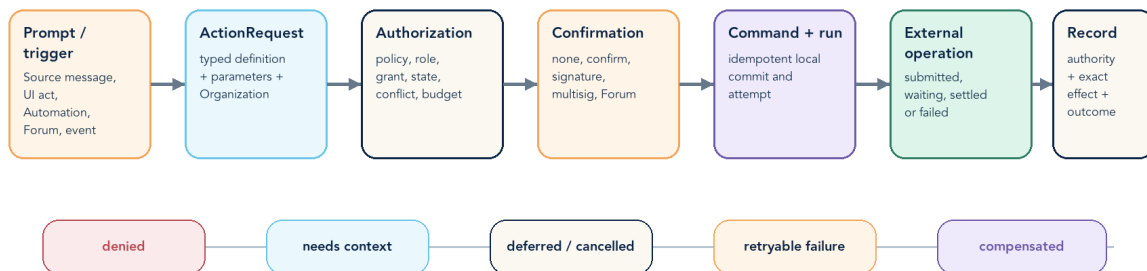
A Record is immutable persona-relevant evidence. It links:

- the intent source and ActionRequest;
- the identity and authority proof;
- exact Command and parameters, with protected fields hashed or redacted;
- policy, role, grant, conflict, and confirmation basis;
- ActionRun, performer, tool/model/agent versions;
- local state version before and after;
- external operation references and settlement proof;
- artifacts, results, corrections, and superseding Records.

The Record graph is the foundation for the Vigil, audit, accounting evidence, organizational memory, and provenance.

Action lifecycle

Intent, authority, execution, and settlement are separate states



Every transition appends an internal event. Consequential transitions append or link a Persona-visible Record.
Retries create new ActionRuns; they never overwrite a prior attempt.

Figure 3. Action lifecycle

5. Permission and policy engine

5.1 The authorization tuple

Every decision evaluates:

principal + Source + requested action + resource + Organization + Project + purpose + time + state + grants + role + Policy + credential + conflict + budget + device/session assurance.

The four access levels are a persona-facing disclosure model, not the full policy model. Engineers must not compress authorization into a single `visibility` property.

5.2 Principals

- Public/Visitor subject with no durable principal and Public authority only.
- Authenticated Persona acting as Guest or through one or more Organization Memberships.
- Member authority set derived from a specific active Membership, never from a global label.
- Ally acting for its authenticated Source.
- Actor instance under an immutable manifest.
- InstitutionDelegate acting under an Institution Agreement and role.
- Organization or Alliance authority set acting through a valid resolution.
- ServicePrincipal with a narrow deterministic capability.
- Ecosystem peer under a KAP relationship and signed envelope.

5.3 Grants

A `Grant` is a node because it can carry scope, purpose, fields, tools, terms, expiry, revocation, provenance, and author. Relationship grants are directional: each Persona controls their own grant to the other.

Grant fields include:

- grantor, grantee, Organization, optional Project or Relationship;
- resource selectors or collection;
- permitted ActionDefinitions and fields;
- allowed purposes and prohibited uses;
- access level ceiling;
- tool scopes and transaction limits;
- start, expiry, use count, revocation;
- whether further delegation is prohibited or narrowly allowed;
- source Code, Agreement, Policy, or sovereign confirmation Record.

5.4 Source-only instruction

The orchestrator classifies every inbound message before context assembly:

- Authenticated Source message: may create instruction intent.

- Other Persona or Member with a valid grant: request or scoped context.
- Code holder: context within Code claims.
- Actor/Tool/web content: untrusted input and potential prompt injection.
- Unknown origin: signal only; no instruction status.

The classification is carried as signed request metadata to the Graph Command Service. The model cannot promote a message to Source authority.

5.5 Conflict and recusal

Conflict evaluation is a named query over Institution affiliations, Organization roles, paid Engagements, family/Relationship disclosures, financial interests, and counterparties. A conflict can:

- disclose and allow;
- require a different human signer;
- prohibit a ballot or position;
- require Forum or multisig authority;
- block access to privileged Wisdom;
- route to compliance review.

Institution recusal is a hard launch rule: a Member enrolled or delegated by an Institution cannot cast the Organization ballot on that Institution's agreement. The decision-market position, if allowed at all, must be separately governed and disclosed.

5.6 Query safety

- Agents call named queries and named commands only.
- No LLM-generated Cypher reaches production execution.
- Every named query declares anchor types, maximum depth, result cap, time budget, allowed labels/edges, required policy context, and redaction shape.
- The query guard rejects unbounded variable paths, dynamic labels, user-controlled property expressions, unsupported functions, and results without a limit.
- Database statement timeout, cost monitoring, and per-principal quotas provide a second boundary.
- PostgreSQL row-level security applies to relational control, vector, and ledger tables as defense in depth; default deny applies when no policy matches.

5.7 Public and Guest launch allowlist

The Kinship Duna default context may be used only for the following registered commands. Any command not listed is denied until the Persona has the required Membership, role, Policy, Agreement, Forum result, or credential.

Authority	Launch commands
Public/Visitor	<code>public.organization.inspect</code> , <code>public.action_catalog.list</code> , <code>account.registration.begin</code> , <code>invitation.preview_public</code>
Guest	<code>account.complete</code> , <code>invitation.accept</code> , <code>identity.verify.request</code> , <code>relationship.proposal.accept</code> , <code>relationship.proposal.decline</code> , <code>membership.application.create</code> , <code>membership.package.purchase</code> , <code>compute.guest_purchase.intent</code> , <code>compute.guest_purchase.confirm</code> , <code>workspace.personal_draft.create</code> , <code>ally.rename</code> , <code>ally.stance.update</code> , <code>wisdom.personal.add</code> , <code>connection.add</code> , <code>organization.public_inspect</code>

Guest commands cannot vote, join a Member-only Community, bind the Organization, create or spend Organization treasury funds, access Member-only Wisdom, create legal Membership, issue transferable Compute, place a market position, receive lineage, or invoke a regulated rail. `membership.admit` is a separate Member/Role/Policy/Forum command after all admission conditions are verified.

5.8 Confirmation policy

The first consequential use of an ActionDefinition requires the confirmation mode registered for that Action. A rendered control may offer **Allow once** or **Allow every time**. Allow every time does not disable authorization; it creates a narrow, versioned Persona Policy bound to ActionDefinition, Organization, resource selector, parameter ceilings, device assurance, expiry, and revocation. Any material parameter, risk, policy, conflict, credential, or context change forces a new preview and confirmation.

6. Organization and legal-form architecture

6.1 Organization is not a draft label

Use three separate objects:

- `OrganizationPlan`: proposed name, purpose, Catalyst, Sponsoring Organization, designee, and intended legal form.
- `LegalRegistration`: evidence and lifecycle for a jurisdiction-specific filing.
- `Organization`: the active accountability container created or activated only when the legal adapter's requirements are satisfied.

For West Virginia DUNAs, the adapter must model governing principles, mutual consent, member threshold, registered agent, designee/contact package, filing reference, Org ID, status, verification source, and periodic re-verification. Current West Virginia Code defines a DUNA as at least 100 mutually consenting members; the system must not equate receipt of an Org ID with every substantive lifecycle requirement.

Every `LegalFormAdapter` exposes: legal identifier, jurisdiction, current standing, formation requirements, minimum and continuing Membership rules, dissolution status, authority model, verification source, verification timestamp, next verification deadline, and command gates affected by status. Status is refreshed continuously or on a defined schedule; an expired verification cannot authorize a new legal or financial Action.

6.2 Registration Offering

Organization registration is purchased as an Offering because it is a paid service outside model Compute.

`RegistrationOffering` includes provider/registered-agent principal, legal form, jurisdiction, base price, service-level option such as 2 hours, 4 hours, or 1 day, required fields, taxes/fees, expiry, and refund/cancellation policy.

The registration flow:

1. Catalyst creates `OrganizationPlan` with name and proposed designee.
2. Designee Persona accepts the role through a sovereign Action and supplies name, address, phone, and email under secret/legal access; where the legal form requires Membership, that condition is validated separately.
3. Authorized Persona purchases the `RegistrationOffering`.
4. Registrar Actor assembles a filing package; credentialed human registered agent reviews and files.
5. `ExternalOperation` tracks submission, acknowledgement, corrections, and completion.
6. `LegalRegistration` records the Org ID and verified source evidence.
7. The Graph Command Service activates the Organization and creates its default policy/configuration objects.

6.3 Organization configuration

Before Launchpad publication, an Organization requires:

- name, handle, purpose/mission, description, website, legal registration, Catalyst, Sponsor, and designee;
- at least one Stance bundle for its host/agent behavior;

- baseline Wisdom Drops and source/credit metadata;
- Connection policy and any configured accounts;
- Automations with budgets and stop conditions;
- Skills and DomainPackages;
- MembershipPolicy, ForumPolicy, ComputePolicy, WaterfallPolicy, AgencyPricingPolicy, TreasuryPolicy, ConflictPolicy, and RecordsPolicy;
- at least one contactable organizational Actor or host presence;
- LaunchCampaign terms and machine-rendered consequences.

6.4 Launchpad lifecycle

[LaunchCampaign](#) states: draft, scheduled, reservation_open, purchase_open, minimum_met, closed_success, closed_failed, issuing, settled, cancelled.

Validation rules from the product-owner specification:

- minimum target must be greater than or equal to the configured legal/product floor; stated launch rule is more than 1,000 USDC;
- maximum target must not exceed 10,000,000 USDC;
- close date must be after open date and any reservation period;
- reservations do not issue Compute and do not enter the waterfall until converted to settled Compute purchases;
- each purchase binds to a Kinship Code where the Organization requires one;
- the Sponsoring Organization may purchase Compute and later distribute it to its Members;
- a failed campaign follows explicit refund/release instructions and creates no Compute issuance.

Organization formation and launch

Legal formation, configuration, campaign, issuance, and operation have separate gates

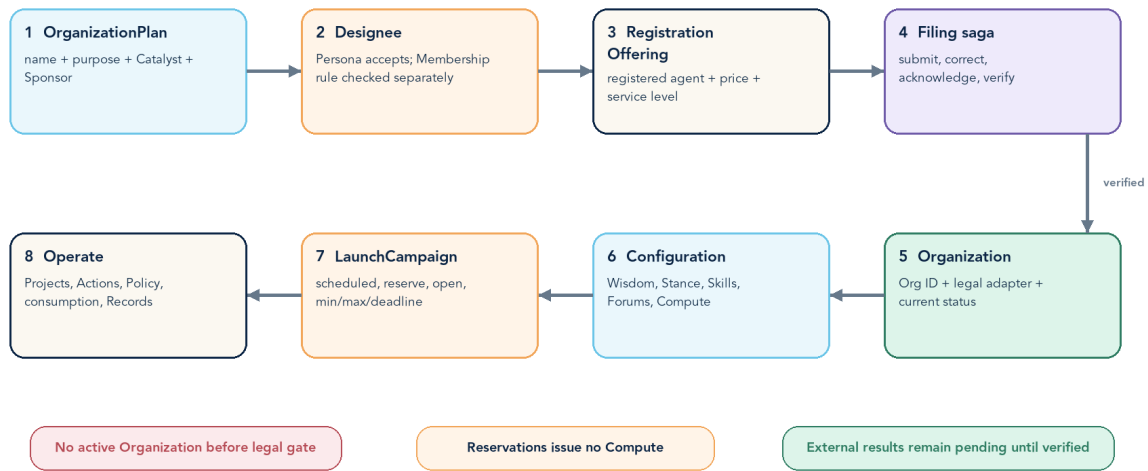


Figure 4. Organization formation and launch lifecycle

6.5 Genesis Ecosystem bootstrap

Deployment bootstrap is infrastructure, not an ordinary Persona Action. The install migration creates the Genesis objects in a fixed order before the first Account can register:

1. Create the **Ecosystem** with permanent protocol ID, unique handle **kiduna**, DID/verification methods, Home Ecosystem policy, federation defaults, and root key references.
2. Create the shared **AllyDefinition** for **Ki** (display name **Kinship Intelligence**) with base Stance, required Wisdom, Skills, Tool manifests, safety policies, and version.
3. Create the limited **MagePrincipal** and account handle convention `<ecosystem_handle>_mage`. Mage may configure bootstrap objects and issue the first Genesis Code; it cannot impersonate a Persona, become a Member, vote, sign, spend Organization funds, read Personal Wisdom, or use general domain commands.
4. Verify that Kinship Duna and the minimum Public/Guest policy bundle exist. The "every Action has an Organization" invariant begins at the Persona-facing boundary; install migrations are not Actions and cannot be invoked through Ally or Actor tools.
5. Issue a single-use Genesis Code under the Mage's narrow manifest and invite the first Persona.

The product may describe Ki as one Ally, but runtime state is tenant-separated. **AllyDefinition** is shared; **AllyIdentity**, Source link, alias, handle, Stance overlay, Wisdom reachability, Connections, Automations, Skills, memory, budgets, and Records are Persona-specific. No query may retrieve another Persona's Ally partition merely because both use Ki.

6.6 Onboarding surfaces and trust boundary

The same signed onboarding state machine may be rendered by Kiduna Live, Kiduna Studio, Kiduna Express, Kiduna TV, an approved embedded website flow, or a future surface. Surfaces never implement membership, pricing, Code, or wallet rules themselves; they call the Graph Command Service through versioned APIs.

The canonical financial origin is `kiduna.ai`; the canonical API origin is `api.kiduna.ai`. Wallet recovery/key-share architecture is represented through `Wallet`, `KeyShareReference`, `CustodyPolicy`, and recovery Records. No raw share or payment credential enters the graph. Alternate domains and embedded surfaces must display and cryptographically verify the canonical financial origin before redirecting a Persona.

The first authenticated Ki session should minimize time to first value. Ki may offer to complete Profile fields, inspect Organizations, apply for Membership, create or join an Alliance, create a Project, add Wisdom, or connect an account. Every offered option comes from `list_available_actions`; the model does not invent a capability.

6.7 Invitation, prospect, and Relationship lifecycle

The invitation pipeline is consent-preserving and graph-native:

1. Inviter Persona tells their Ally to invite a person and supplies the intended target scope: Ecosystem, Organization, Membership, Project, Alliance, Community, Relationship, or Wisdom.
2. Ally creates a draft `ActionRequest` and invokes a Profiler Actor only after the inviter confirms the purpose and supplies at least one contact route.
3. Profiler creates `InvitationProspect`, not Persona. It records inviter-authored claims, source, uncertainty, purpose, access level, and retention deadline. Optional public research is separately attributed.
4. Inviter chooses a directional trust declaration: low, medium, or high, plus optional rationale and visibility. Trust does not authorize code execution, file acceptance, payment, or protected access.
5. CodeManager checks whether the target resolves to an existing Persona by protocol ID, verified handle, contact proof, or privacy-preserving identity challenge.
6. For an existing Persona, the system creates an in-system `RelationshipProposal` or scoped invitation. Optional passphrase/challenge proof may confirm the intended recipient.
7. For an unregistered person, CodeManager creates a signed, single-use, recipient-bound Code with a 15-minute default expiry. The platform returns it to the inviter and performs no outbound delivery.

8. A generalized Code may be multi-recipient only when its registered type permits it. It carries no prospect Profile, defaults trust to untrusted, and declares target, maximum redemptions, expiry, issuer, and claims.
9. On redemption, the service verifies signature, issuer, intended recipient proof where present, expiry, use count, revocation, target status, and replay nonce before showing protected details.
10. The Visitor creates an Account and becomes an authenticated Guest Persona. The invitee reviews all inviter-supplied Profile claims; none become self-authored facts without acceptance.
11. The invitee accepts or declines each Relationship, target participation, data grant, and trust visibility independently. An invitation to a Project never silently admits Membership to its Organization.
12. Accepted Relationship creates one Relationship node with two independent trust declarations and grants. Declined/expired claims are retained only as restricted Records under the retention policy.
13. Any Membership application then runs the Organization's separate admission workflow and legal-form adapter.
14. Ki assembles the new Persona's environment from accepted claims, permitted Communities/Projects, public Wisdom, and active grants; it does not expose the inviter's private notes.
15. Acceptance, decline, expiry, revocation, and every Code redemption attempt create Records and invalidate relevant caches.

Invitation, identity, Relationship, and Membership

Each boundary requires its own consent and Record

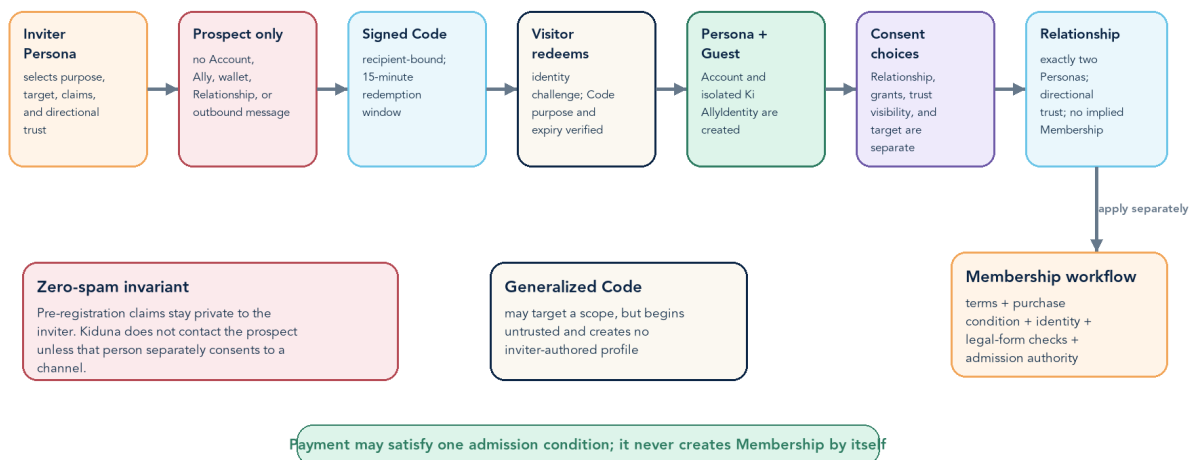


Figure 5. Invitation and onboarding lifecycle

7. Compute, agency pricing, lineage, and the waterfall

7.1 Compute objects

The model separates:

- **ComputeDefinition**: Organization issuer, denomination, usage purpose, transfer policy, decimals, settlement/chain references, status.
- **ComputeAccount**: balance owner, Organization, custody reference, restrictions.
- **ComputeIssuance**: amount, authority, Forum Proposal/Policy, mint or ledger reference.
- **ComputePurchase**: purchaser principal, USDC amount, Compute amount, price, Code, campaign, settlement state.
- **ComputeGrant**: distribution from an Institution or Sponsoring Organization to a Persona under an Agreement or to one of its Members under Organization Policy.
- **ComputeConsumption**: model/provider usage, input/output units, provider cost, Organization charge, multiple, premium, Project and Action.
- **WaterfallAllocation**: one destination amount under a specific WaterfallPolicy version.

7.2 Membership purchase policy

Each Organization may make a minimum initial Compute purchase one condition of Membership, but payment never makes a Persona a Member. Kinship Duna's launch policy is explicit:

- Guest access: minimum 10 USDC of Compute; remains Guest and receives only Guest authority.
- Persona seeking Membership: minimum 100 USDC of Compute plus accepted governing terms, admission decision, identity checks, and every applicable legal-form requirement.
- Institution participation: minimum 1,000 USDC of Compute plus an effective Institution Agreement; this never creates human Membership or a ballot.

`membership.admit` validates all conditions and creates the active Membership and Record.

`compute.purchase` records the economic event separately. Reversals, expiration, suspension, withdrawal, or expulsion change Membership only through their own Commands.

7.3 Agency pricing

For each model/tool consumption, the selected versioned schedule calculates:

```
provider_cost = verified input + output + tool cost
member_charge = pricing_rule(provider_cost, organization_multiple, minimums, caps)
agency_premium = member_charge - provider_cost
```

Kinship Duna's initial disclosed schedule uses a 7x Member multiplier on verified provider/tool cost. A Guest pays 2x the corresponding Member charge, producing an effective 14x provider-cost charge before any disclosed fixed tool charge. The receipt must show provider cost, applicable status, base multiple, Guest factor, fixed charges, final Compute charge, and Organization premium. The policy also declares a maximum markup and rounding rule; no hidden dynamic markup is permitted at launch.

The Organization receives the agency premium under its current [AgencyPricingPolicy](#). This is consumption revenue, not a Compute purchase and therefore does not trigger lineage or the launch waterfall. Future pricing changes are prospective, versioned, disclosed before confirmation, and cannot reprice completed consumption.

7.4 Default Compute purchase waterfall

The product-owner default is represented as a versioned policy:

Destination	Default	Rule
Lineage generation 1	20%	Current parent in the frozen Organization lineage snapshot
Lineage generation 2	5%	Second ancestor
Lineage generation 3	3%	Third ancestor
Lineage generation 4	2%	Fourth ancestor
Liquidity	20%	Default minimum under the current versioned Kiduna Organization policy; not a protocol invariant
Kiduna Club license	3%	Licensing allocation under the applicable Agreement
Curator	2%	Member or Institution that introduced the Catalyst, if eligible
Treasury	45%	Residual under the default policy

Policy requirements:

- percentages must sum to 100 after all explicit fallbacks;
- changes are prospective, versioned, Forum-authorized where policy requires, and never rewrite settled purchases;
- a missing or ineligible ancestor is skipped upward to the next qualified ancestor in the frozen lineage; any remaining unfilled generation allocation after the traversal limit goes to the policy's named treasury/reserve account, never to a fabricated participant;
- the Curator may also be an ancestor, but the two allocations remain distinct and disclosed;

- an Institution can occupy any eligible lineage position and can have Member or Institution progeny;
- the purchasing principal may be a Member, Institution, Organization, or another principal explicitly permitted by Policy; a Guest purchase does not receive lineage without a separate lawful eligibility rule;
- no allocation is paid until the purchase settlement is verified;
- fiat/onramp purchases may create pending allocations that settle only after verified bridge/rail settlement;
- Offerings never invoke this policy.

7.5 Lineage model

Do not store generation numbers as a set of permanent direct edges from the child to every ancestor. Store one current parent enrollment per Organization and derive/materialize ancestry.

Core objects:

- **LineageParticipant** role on a Persona with active qualifying Membership, or on an Institution; it is not a separate human identity label.
- **LineageEnrollment**: child, parent, Organization, Code, sponsoring context, effective time, status, provenance.
- **LineageSnapshot**: frozen four-generation chain created for a ComputePurchase.
- **WaterfallPolicy**: schedule and fallbacks.
- **WaterfallAllocation**: calculated destination, amount, reason, state, settlement reference.

This prevents later lineage changes or policy edits from changing historical payouts.

7.6 Founding Alliance policy

An Organization may configure a monthly transfer to its Founding Alliance as:

- fixed amount;
- percentage of settled Compute purchase proceeds;
- minimum and maximum amount;
- treasury sufficiency floor;
- start/end dates;
- approval and pause rules.

The Automation creates a recurring ActionRequest. It does not transfer funds by itself. The applicable Policy authorizes the Command, and the ledger plus external rail Records the result.

7.7 Architecture-level pushback

The schema supports transferable Compute, DEX trading, multilevel lineage, launch campaigns, and variable agency multiples. It should not hard-code their production activation. Current official guidance makes the implemented facts important: transferability and secondary-market expectations can affect digital-asset analysis; compensation tied to participant/downline purchases receives fact-specific scrutiny; and payment-transmission status depends on actual control of funds. Each risky feature therefore has a deployment gate tied to counsel approval, implemented custody/settlement facts, disclosures, monitoring, and the ability to disable new activity without corrupting historical Records.

7.8 Launch feature activation

The schema and simulation harness implement the full economic model, but activation is independent per capability. At August 10 launch, the following are **disabled for live value** pending legal, accounting, tax, custody, rail, and exchange review: transferable or DEX-traded Compute, decision-market positions, lineage payouts, Curator allocations, Sponsor distributions, and automated liquidity deployment. Their commands may run only in a non-settling simulation tenant with conspicuous test labeling.

Non-transferable Compute purchase and consumption may be active only after ledger, pricing-disclosure, refund, custody, reconciliation, and operator sign-off gates pass. A feature gate is evaluated by deterministic code and cannot be bypassed by an Ally, Actor, Forum result, or Mage.

8. Forums, proposals, Envoys, and Policy

8.1 Multiple Forums per Organization

Examples:

- Constitutional Forum: charter, membership rules, dissolution, major changes.
- Treasury Forum: budgets, grants, payments, liquidity.
- Program Forum: agent budgets, Skills, Automations, DomainPackages.
- Project Forum: large Project commitments above delegated thresholds.
- Domain Forum: claims, clinical standards, festival safety, game rules, underwriting policy.

Each Forum has a [ForumPolicy](#) defining eligibility, quorum, duration, proposal categories, discussion rules, market signal settings, position limits, conflict rules, operators, execution thresholds, appeal/reconsideration, and emergency procedure.

8.2 Separate the two pass/fail mechanisms

The canon contains both equal Member deciding and USDC-funded pass/fail trading. They must be separate in data and interface:

Governance ballot. One eligible Member, one free equal pass/fail ballot. This determines enactment when the Organization's governing principles say it does.

Decision-market position. An optional USDC-funded pass/fail position placed by an Envoy under Member instruction. It is a forecast or conviction signal with its own settlement logic and limits. It does not become a ballot and does not buy greater legal voting power.

If a future Organization chooses a different lawful governance algorithm, it must register a new `DecisionMethod` with explicit semantics, not overload `Forum`, `Vote`, or `Trade`.

8.3 Proposal pipeline

1. Member tells Ally the organizational intent.
2. Ally and Configuration Drafter map intent to one or more exact Commands.
3. Graph Command Service validates parameter completeness, authority route, dependencies, conflicts, and whether a Forum is required.
4. Receipt renderer creates the consequence sentence from the exact parameters.
5. Proposal opens in a specific Forum; discussion and evidence attach as Records/Wisdom.
6. Members give sovereign ballot instructions; Envoys may place allowed positions under separate instructions.
7. Operator closes according to ForumPolicy; deterministic tally/resolution service records the result.
8. Passed Commands execute locally or begin ExternalOperations.
9. Policy is created with effective scope and status. External effects remain pending until verified.
10. Every affected Persona can inspect the permission-filtered receipt chain; every eligible Member can inspect the Organization governance record required by its rules.

8.4 Proposal state

```
draft -> command_complete -> receipt_complete -> submitted -> deliberating ->
open -> closed -> passed|failed|withdrawn -> executing -> pending_external|
partially_settled -> enacted|execution_failed|superseded.
```

An authorization outcome and a physical-world outcome are not the same. A passed proposal can authorize a land purchase; title transfer becomes true only when the relevant external evidence settles.

Forum to Policy

Equal ballots and optional USDC-funded market positions remain distinct

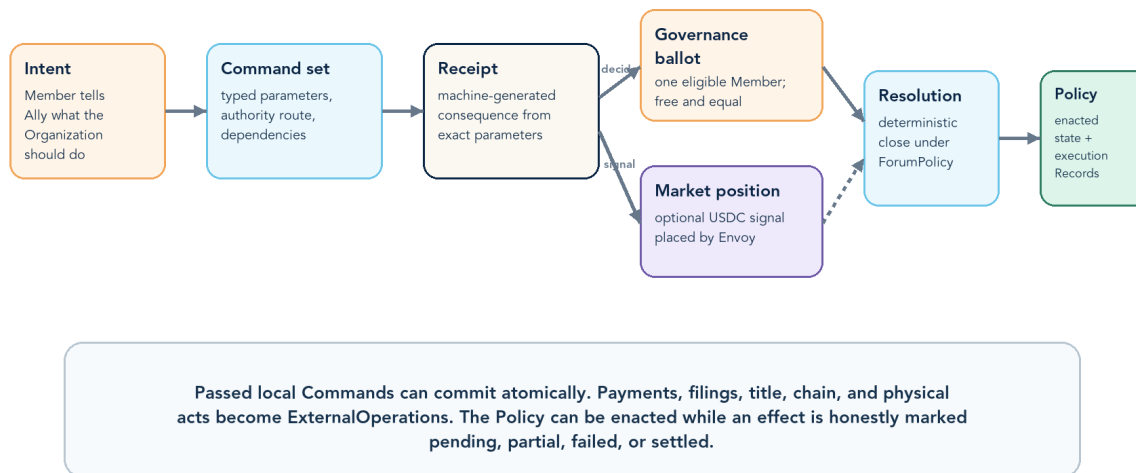


Figure 6. Forum to Policy pipeline

9. Projects and real work

9.1 Project graph

Every Project links:

- accountable Organization;
- purpose and success criteria;
- participating Personas, exact Organization Memberships where required, and Project roles;
- sponsoring Alliance or Institution;
- Engagements and counterparties;
- Wisdom Drops and Artifacts;
- Allies and Actor instances;
- Skills, Connections, and Automations;
- WorkItems, milestones, dependencies, and deliverables;
- Agreements, terms, split rules, invoices, and payouts;
- Actions, Records, conflicts, credentials, and legal holds;
- stable Field/Scene address and derived status.

Project status is derived from its graph: proposed, active work, waiting on an Action, blocked by external operation, under review, complete, archived. Personas do not type a cosmetic status detached from facts.

9.2 WorkItem and Contribution

[WorkItem](#) defines an outcome, not employee supervision. It can carry requested result, constraints, due date, required credential, acceptance criteria, budget, dependencies, and current assignee.

[Contribution](#) records what a Persona or Actor actually supplied, with provenance and acceptance status.

[assign_work](#) is an Action. [accept_deliverable](#) is a different, often sovereign or credentialed Action. [pay_split](#) is a financial Action triggered only after the agreed condition and settlement rules.

9.3 Engagement and compensation

An Engagement records the parties, Organization, scope, classification review facts, rate/split formula, expenses, tax forms, milestones, changes, acceptance, invoicing, payment trigger, and dispute terms. Worker classification must not be hard-coded as "1099"; the system stores the actual control and independence facts and routes exceptions for review.

Split rules are versioned and bound at acceptance. When client payment settles, the ledger creates the exact distributions. The graph links each distribution to the Engagement, Contribution, acceptance Record, and recipient.

9.4 DomainPackage extension model

Do not put every industry's noun into the global kernel. A [DomainPackage](#) supplies:

- ResourceType schemas and labels;
- ActionDefinitions and CommandDefinitions;
- role and credential definitions;
- policy templates and conflict rules;
- Skills and Actor manifests;
- Wisdom sources and retention classes;
- Field projections;
- acceptance and safety tests;
- migration and version compatibility.

The kernel remains stable while domains grow.

9.5 Required domain coverage

DUNA law firm. Matter, Client, Engagement, ConflictCheck, Credential, Filing, Deadline, AdviceDraft, HumanApproval, Invoice, OperatingPayment, and LegalHold. Secret and privileged Wisdom is isolated by matter. Agents may research and draft; credentialed lawyers sign. Client trust/IOLTA is an external restricted account and the automation refuses to split funds while they remain in trust.

Insurance company. CoverageProduct, Application, UnderwritingCase, RiskEvidence, Claim, Reserve, AdjusterAssignment, ActuarialModel, RegulatoryFiling, PaymentAuthorization. Licensing and phase gates are explicit. No Actor can promise coverage or approve a regulated act outside the configured credential/policy route.

Solar installer. Site, Assessment, Design, Permit, Equipment, Vendor Institution, Installer Credential, Financing Agreement, WorkOrder, Inspection, Interconnection, ProductionTelemetry, Maintenance. Sensor events remain time-series data; the graph stores assets, latest state, exceptions, and Records.

Mortgage company. Applicant, Property, Application, Consent, DocumentChecklist, Disclosure, Lender Institution, LoanProduct, UnderwritingDecision, Appraisal, Condition, Closing, ServicingHandoff. Sensitive documents stay in encrypted object storage; decisions cite policies and required licensed humans.

Psychotherapy and health. Client, CareEngagement, Consent, ProviderCredential, Appointment, CarePlan, SessionRecord pointer, Referral, SafetyEscalation. Clinical notes use separate encryption and strict purpose boundaries. Model inference cannot become diagnosis or treatment authority. Emergency routing is explicit and human-owned.

Table games and video games. Game, RuleSet, Session, Table/Lobby, PlayerRole, Team, Match, Result, Tournament, Asset, ModerationAction, SafetyRule, CreatorRoyalty. A game action can be simulated or real; the [sim](#) flag controls rails, not whether the Record exists.

Parties and festivals. Event, Venue, Permit, Ticket/Code, Vendor, Performer, Crew, Shift, SafetyPlan, Capacity, Incident, Settlement, RightsLicense. Actors coordinate schedules and communications; credentialed humans approve safety, permits, and payments.

These examples prove the schema by variation. They are not separate platform products.

9.6 August 10 DomainPackage activation matrix

Status	Packages/capabilities	Launch rule
Active when P0 gates pass	identity/account, Ki/Ally, invitation/Code, Relationship, Organization context, Membership, Wisdom, Project/WorkItem/Contribution, communication, equal governance ballot, non-transferable Compute consumption, audit/Vigil, games, and event coordination	Named commands only; no hidden mutation; exact authority and confirmation
Preview	Organization registration, Offering/Launchpad, Institution Agreements, legal work, insurance, mortgage, health/care, and regulated solar steps	Read, draft, checklist, simulation, and human handoff only; no representation that an external or regulated effect completed
Disabled for live value	transferable/DEX Compute, decision-market positions, lineage settlement, Curator allocation, Sponsor distribution, automated liquidity, custody, lending, securities/investment advice, payroll, insurance binding/claim payment, clinical diagnosis/treatment, and autonomous legal filing	Schema/test fixtures may exist; production command gate always denies until separately approved

Package status is stored by Ecosystem, Organization, version, jurisdiction, and command. Enabling a package never enables every command inside it. Regulated commands additionally require [RegulatedCredential](#) or [ExternalAgreement](#) authority and current legal-adaptor status.

10. Physical data architecture

10.1 Launch stack

Recommended launch deployment:

- PostgreSQL with a version-pinned Apache AGE extension for the typed property graph.
- pgvector in the same cluster for permission-scoped semantic candidates.
- Relational schemas for command registry, object registry, Records, events, outbox, external operations, double-entry ledger, identity/session, and migrations.
- S3-compatible encrypted object storage for files and large artifacts.

- Secrets manager/HSM or secure enclave references for credentials and root keys.
- Orchestration workers for Allies and Actors.
- Ingestion workers for files, metadata, chunks, embeddings, entities, and provenance.
- Outbox/reconciliation workers for payments, chain, messaging, filings, and federation.
- OpenTelemetry-compatible traces, metrics, and logs with privacy-safe attributes.

Apache AGE remains a sound launch fit because it places graph queries inside PostgreSQL's transaction and storage system and supports openCypher plus SQL. As of this specification, Apache lists AGE 1.7.0 for PostgreSQL 18 and 1.6.0 for PostgreSQL 16/17. Engineering must pin the pair already proven by migration, backup, load, and extension tests; "latest" is not a release strategy.

10.2 One graph per Ecosystem

Use one AGE graph namespace per Ecosystem, not one graph per Organization. Cross-Organization Relationships, Personas, Memberships, Institutions, lineage, and shared Wisdom require traversal. Organization boundaries are enforced by properties, policy, grants, named queries, and RLS defense in depth, not by creating hundreds of isolated graph databases.

10.3 Three forms of state

State	Store	Purpose
Current operating state	AGE graph plus typed object registry	What exists, how it is related, current policy/reachability, derived work state
Immutable evidence and transitions	PostgreSQL Record/event tables, selected Record nodes	What happened, under what authority, in what order
Semantic candidates	pgvector plus authorized Item metadata	What text or artifact may be relevant; never authority

Financial entries use a fourth specialized form: immutable double-entry ledger rows plus verified external references. Time-series telemetry such as solar output or gameplay events uses a fifth optimized store when volume warrants it; graph nodes hold durable assets, summaries, alerts, and provenance.

10.4 Stable IDs

- Use UUIDv7 or ULID database IDs and a portable `protocol_id` such as `kap://eco.kiduna.net/org/628407/project/01...`.
- Never expose AGE vertex IDs as API identity.
- IDs are immutable across migration and federation.

- Every mutable aggregate has one authoritative home Ecosystem and a monotonically increasing version.
- Cross-Ecosystem references include protocol ID, home Ecosystem, issuer, key/version, and last verified Record.

10.5 Universal node properties

Every first-class node carries where applicable:

`id`, `protocol_id`, `kind`, `schema_version`, `ecosystem_id`, `home_ecosystem_id`, `organization_id`, `controller_principal_id`, `access_level`, `legal_class`, `status`, `version`, `valid_from`, `valid_to`, `created_at`, `updated_at`, `origin_ecosystem_id`, `provenance_record_id`, `superseded_by`.

Not every node has an Organization, but every ActionRequest does.

10.6 Universal edge properties

Durable edges carry:

`id`, `edge_type`, `organization_id`, `access_level`, `purpose`, `status`, `valid_from`, `valid_to`, `version`, `created_by_record_id`, `superseded_by`.

Do not put a changing historical narrative only on an edge. When the relationship needs independent lifecycle, authorship, terms, or evidence, reify it as a node.

10.7 Relational control tables

Minimum tables:

- `object_registry`: global unique IDs, kind, home, controller, version, graph label, lifecycle.
- `command_definition`, `action_definition`, `named_query_definition`: versioned registries and digests.
- `command_instance`, `authorization_decision`, `action_run`.
- `record`, `record_link`, `event_log`, `outbox`.
- `external_operation`, `external_callback`, `reconciliation_case`.
- `ledger_account`, `ledger_transaction`, `ledger_entry`, `settlement_reference`.
- `embedding_chunk`, `embedding_model`, `ingestion_job`.
- `credential_reference`, `key_registry`, `session`, `code_redemption`.
- `schema_migration`, `projection_checkpoint`, `policy_bundle`.

The SQL registry supplies uniqueness, foreign keys, check constraints, and high-value invariants that AGE does not enforce as a complete application schema. The Graph Command Service writes the registry and graph in one transaction.

10.8 Transaction and outbox

For a local command:

1. Begin PostgreSQL transaction.
2. Lock aggregate/version rows.
3. Re-evaluate authorization against current state.
4. Validate parameters and preconditions.
5. Write AGE graph changes.
6. Write control/ledger rows.
7. Append Record and internal event.
8. Insert outbox messages.
9. Commit.
10. Workers deliver messages and reconcile external operations idempotently.

No API response says complete until the relevant local commit or external settlement state justifies it.

10.9 Signed federation and migration

Every portable object has one Home Ecosystem that is authoritative for mutable state. Federation peers exchange signed resource manifests and Records containing protocol ID, schema version, home, issuer key/version, digest, audience, expiration, and capability scope. A local registration proves provenance and routability; it never grants access.

A Persona migration is a sovereign Action. It freezes conflicting writes, exports encrypted authorized state, verifies the destination, records a signed handoff from old and new homes, advances the home epoch, updates routing, and preserves the permanent ID and complete history. Memberships remain governed by their Organizations and are revalidated rather than silently moved. Recovery and dispute procedures must work even when the former Home Ecosystem is unavailable.

Agent packages expose a signed [kiduna.md](#) manifest derived from the canonical package registry. Compatibility adapters may also emit [agents.md](#) or [CLAUDE.md](#), but filenames are not trust. The validated digest, issuer, permissions, declared Tools, Skills, Wisdom scopes, and command allowlist are the authority-bearing facts.

State, evidence, meaning, and settlement

One system, four different guarantees

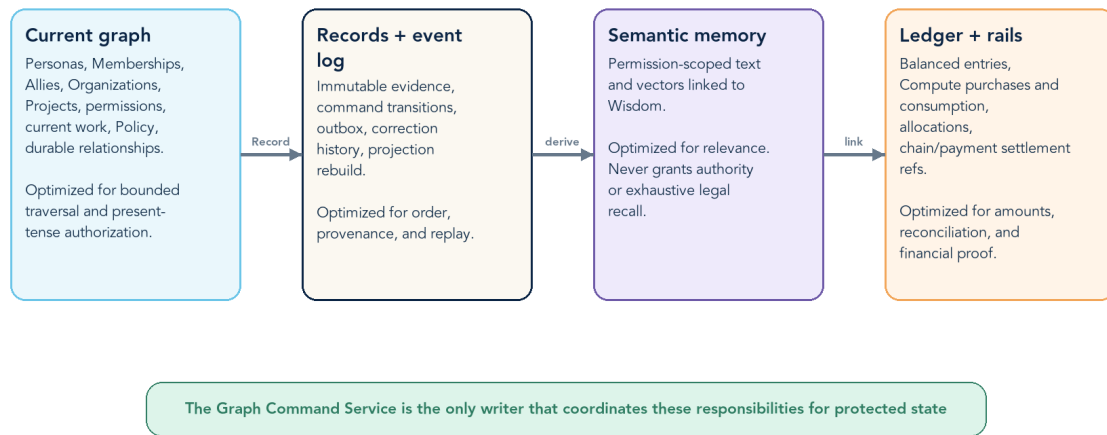


Figure 7. State, evidence, meaning, and settlement

11. Graph schema registry

11.1 Core labels

Domain	Node labels	Notes
Federation	Network, Ecosystem, EcosystemRelationship, RegistryEntry, HomeMigration	No private content at Network root; every portable object has one current home epoch
Identity	Account, Persona, Credential, Wallet, Code, Invitation, InvitationProspect, VerificationAttestation	Visitor is a session state, not a stored person; Guest/Member are contextual authority states, not permanent identity labels
Agents	AllyDefinition, AllyIdentity, Agent:Actor, ActorManifest, MagePrincipal, Stance, Connection, Automation, Skill	Ki is the canonical AllyDefinition; each Persona has a tenant-isolated AllyIdentity; Actor kinds are properties/secondary labels
Organization	OrganizationPlan, Organization, LegalForm, LegalRegistration, Membership, RoleDefinition, RoleAssignment	Organization exists as active legal container only after adapter gate

Domain	Node labels	Notes
Social	Relationship, RelationshipIntent, TrustDeclaration, Grant, Community, Alliance, Institution, InstitutionAgreement	Relationship exactly two consenting Personas per Organization; Membership is separate
Work	Project, Engagement, WorkItem, Milestone, Deliverable, Contribution, Agreement, SplitRule	DomainPackage may extend ResourceTypes
Action	ActionDefinition, ActionRequest, AuthorizationDecision, Command, ActionRun, ExternalOperation, Record	The accountable action spine
Wisdom	WisdomDrop, Item, Document, Artifact, Entity	Chunks live primarily in relational/vector store
Governance	Forum, ForumPolicy, Proposal, BallotInstruction, Ballot, MarketPosition, Resolution, Policy	Ballots and positions are distinct
Economy	Offering, Purchase, ComputeDefinition, ComputeAccount, ComputePurchase, ComputeIssuance, ComputeConsumption, WaterfallPolicy, LineageEnrollment, LineageSnapshot, WaterfallAllocation, LaunchCampaign, Reservation	Offering Purchase is not ComputePurchase
Place	FieldAddress, Scene	Projection identity, not authorization

11.2 Key relationships

Relationship	From -> To	Cardinality/invariant
SOURCE_OF	Persona -> AllyIdentity	Exactly one active AllyIdentity per authenticated Persona; no Visitor Ally
INSTANCE_OF	AllyIdentity -> AllyDefinition	Every Persona's Ally is an isolated instance of canonical Ki or an approved successor definition

Relationship	From -> To	Cardinality/invariant
HAS_MEMBERSHIP	Persona -> Membership	One current Membership lifecycle per Persona/Organization; Institution participation uses Agreement, not Membership
IN_ORGANIZATION	Membership/Project/Alliance/Forum/ActionRequest -> Organization	Every ActionRequest exactly one
PARTICIPANT_IN	Persona -> Relationship	Exactly two distinct consenting Personas per Relationship
AUTHORED_TRUST	Persona -> TrustDeclaration	Directional, scoped, versioned; never global authority
AUTHORED_GRANT	Persona/Policy -> Grant	A Persona cannot edit the other Persona's directional grant
GRANTS_TO	Grant -> Principal	Personal resources fail validation
INTENDS_RELATIONS HIP	Persona/InvitationProspect -> RelationshipIntent	Intent does not create Relationship or Membership
REDEEMS	Persona -> Invitation	Recipient-bound or generalized Code rules apply
OWNS_PROJECT	Organization -> Project	Exactly one accountable Organization
SPONSORS	Alliance/Institution/Organization -> Project/OrganizationPlan/ComputePurchase	Scoped by Agreement/Policy
HAS_WISDOM	Principal/Container -> WisdomDrop	Attribution separately linked
INFORMED_BY	Agent -> WisdomDrop	Access decision still evaluated on every retrieval
STANCE_FOR	Stance -> Agent/Organization	Versioned and layered
EMPOWERS	Connection -> Agent	Credential reference only
ENABLES	Automation -> Agent/Container	Produces ActionRequests
IMPARTS	Skill -> Agent/Container	Immutable version
USES_DEFINITION	ActionRequest -> ActionDefinition	Exact version
PROPOSES_COMMAND	ActionRequest/Proposal -> Command	Exact parameters
DECIDES	AuthorizationDecision -> ActionRequest	One or more decisions across lifecycle; latest current
REQUESTED_BY	ActionRequest -> Persona/Organization/Institution	Human source is a Persona; contextual Member authority is proven separately

Relationship	From -> To	Cardinality/invariant
PERFORMED	Agent/ServicePrincipal -> ActionRun	Source Persona still separately linked
PRODUCED	ActionRun/ExternalOperation -> Record	Immutable evidence
OWNS_FORUM	Organization -> Forum	One-to-many
IN_FORUM	Proposal -> Forum	Exactly one Forum
ENACTS	Resolution -> Policy	Only valid enactment path
CHILD_ENROLLMENT	LineageParticipant -> LineageEnrollment	Exactly one current parent per Organization when lineage enabled
PARENT_PARTICIPANT	LineageEnrollment -> LineageParticipant	Qualifying Member through Persona/Membership, or Institution
USES_SNAPSHOT	ComputePurchase -> LineageSnapshot	Frozen at purchase
ALLOCATES	ComputePurchase -> WaterfallAllocation	Amounts from exact policy version

11.3 Constraints

- Unique Persona and Institution handles within the declared network namespace; reserved system handles cannot be claimed.
- One active AllyIdentity per Persona; one Source Persona per AllyIdentity; every AllyIdentity is tenant-isolated even when all present as Ki.
- Exactly two distinct consenting Personas on a Relationship; one current Relationship per unordered pair plus Organization unless policy permits versions.
- One active Membership per principal and Organization.
- No Membership exists without accepted terms, admission authority, effective dates, Record, and current legal-adaptor validation where required.
- Visitor is never persisted as a Persona merely because invite/profile claims exist; InvitationProspect cannot own an Ally, wallet, Membership, or Relationship.
- One current lineage parent per participant and Organization; no cycles; maximum payable depth four at launch.
- LaunchCampaign minimum, maximum, dates, and policy version valid before opening.

- Waterfall percentages and explicit fallbacks total 100; the 20 percent liquidity minimum is enforced by the current versioned Organization policy, not by the protocol kernel.
- ComputePurchase and Offering Purchase use distinct labels and commands.
- Ballot uniqueness per eligible Member/Proposal; MarketPosition has separate limits.
- Every Proposal command has a registered receipt renderer before opening.
- Every ActionRequest has Organization, definition version, canonical payload hash, and idempotency key.
- Every external side effect has an ExternalOperation before submission.
- No secret/personal Item enters a public embedding partition, cache, Scene, or event payload.

11.4 Indexes

Create and test indexes for:

- protocol ID, external IDs, handle, Organization Org ID, wallet address, Code digest.
- node `organization_id`, `status`, `kind`, `controller_principal_id`, `home_ecosystem_id`.
- membership principal plus Organization plus current status.
- Relationship normalized pair key plus Organization.
- ActionRequest Organization/status/deadline; Command idempotency; ExternalOperation state/next attempt.
- Proposal Forum/status/deadline; Policy scope/effective time.
- Project Organization/status; WorkItem Project/status/assignee.
- Lineage child/Organization/current; ComputePurchase purchaser/Organization/settlement time.
- Record subject/time, ActionRequest, Organization, and previous hash.
- event/outbox partition key and availability time.
- vector `model_id`, `wisdom_drop_id`, `access_scope_hash`, and HNSW index appropriate to the embedding model.

Use exact vector search for small permission-filtered candidate sets. For broad public scopes, pgvector HNSW can be used with iterative scans and measured recall. Never depend on approximate search to find every legally required or conflict-related fact.

12. Named query and agent tool surface

The previous seven tools become a bounded operating toolset. Tools are not raw database access; they are stable contracts backed by named queries and commands.

12.1 Context and work reads

- `resolve_current_context()` - Persona, Visitor/Guest/Member state for the exact Organization, AllyIdentity, Project, roles, grants, conflicts, Compute state.
- `list_available_actions(object_id)` - permitted ActionDefinitions, missing requirements, confirmation class.
- `get_action_status(action_request_id)` - exact current state, attempts, external waits, Records.
- `list_my_work(filters)` - Projects, WorkItems, Actions requiring the Persona, exact Membership/role basis, deadlines, compensation terms.
- `get_project_state(project_id)` - purpose, people, work, dependencies, packages, Records, money, next Actions.
- `get_organization_state(org_id)` - verified legal status, membership, policies, Forums, Projects, Compute and treasury summaries.
- `get_forum_docket(forum_id)` - proposals, receipts, discussion evidence, conflicts, deadlines, ballot/position status.
- `get_relationship(persona_id, org_id)` - the viewer's permitted Relationship facts, directional trust, and grants.
- `get_lineage(org_id)` - permitted current ancestry/progeny and policy, never unrelated private details.
- `get_compute_activity(org_id)` - purchases, consumption, pending allocations, and settlement links.
- `get_invitation_status(invitation_id)` - purpose, target, expiration, redemption, consent, and safe next action without exposing private inviter claims.
- `get_home_and_federation(protocol_id)` - current Home Ecosystem, home epoch, signed routing metadata, and verification status.

12.2 Wisdom reads

- `search_wisdom(query, purpose, org_id, project_id).`
- `get_wisdom_provenance(item_id).`
- `list_wisdom_drops(scope).`
- `explain_why_context_was_used(record_id).`

Safe hybrid flow:

1. Resolve authorized Item/WisdomDrop IDs and purpose before retrieval.

2. Query pgvector only within that set, using exact search when the set is small.
3. Return scored candidate IDs and minimal authorized text.
4. Expand named graph relationships for provenance and context.
5. Assemble the model context with access labels, source weight, attribution, and conflict boundaries.

12.3 Action writes

- `create_action_request(definition_key, parameters, context).`
- `preview_action(action_request_id).`
- `confirm_action(action_request_id, proof).`
- `defer_or_cancel_action(action_request_id).`
- `create_project, assign_work, accept_deliverable.`
- `create_organization_plan, accept_designee_role, purchase_registration_offering.`
- `configure_organization, publish_launch_campaign, reserve_launch_purchase.`
- `record_compute_purchase_intent, confirm_compute_purchase, distribute_compute_grant.`
- `submit_proposal, give_ballot_instruction, give_position_instruction.`
- `add_wisdom_drop, set_access, inform_agent, set_stance, connect_account, enable_automation, impart_skill.`
- `create_invitation, redeem_invitation, accept_relationship, declare_trust, apply_for_membership.`
- `request_home_migration, confirm_home_migration, verify_package_manifest.`

The Ally can draft and preview any Action in scope. Confirmation and authority determine whether it can be committed.

13. End-to-end pipelines

13.1 Persona prompt to completed Action

1. Channel adapter authenticates session/device and labels the message source.
2. The Persona's AllyIdentity receives the prompt as Source instruction or non-Source context.
3. Orchestrator resolves Organization and Project context; allowed missing context defaults to Kinship Duna and is disclosed.

4. Ally identifies a candidate ActionDefinition or asks a narrow question.
5. `create_action_request` validates parameters and freezes the definition version.
6. Graph Command Service evaluates access, role, grant, Policy, conflict, budget, credential, state, and expected versions.
7. Service returns denied, needs information, needs confirmation, needs multisig, needs Forum, or authorized.
8. Field renders exact consequence and required act.
9. Persona, Member ballot, Forum, Role, Policy, Agreement, or credential authority supplies the required proof.
10. Command commits local graph/control/Record/outbox transaction.
11. Actor or service executes permitted work; external operations reconcile.
12. Ally reports honest status and presents resulting Actions.

13.2 Ally delegates to Actor

1. Ally creates ActionRequest with Source and Organization context.
2. Policy selects an Actor kind and immutable manifest version.
3. Runtime issues a short-lived capability containing allowed reads, Tools, commands, budget, deadline, and target IDs.
4. Actor receives permission-filtered context and executes within the package.
5. Every Tool call re-enters authorization; a capability cannot be widened by the Actor.
6. Actor returns draft Artifacts/Records and terminal status.
7. Consequential output is reviewed or signed before promotion to authoritative Organization state.

13.3 Wisdom ingestion

1. Persona or authorized principal adds file, text, link, or connected source and selects/access-confirms a WisdomDrop.
2. Artifact service stores encrypted content, hashes it, scans it, and appends an ingestion Record.
3. Ingestion worker extracts metadata, text, chunks, entities, citations, and provenance.
4. Authorization metadata is written before any vector row.
5. Embeddings are generated under declared provider/data policy.
6. Chunks reference Item and WisdomDrop; graph stores durable document/entity/provenance links.
7. Draft derived facts remain candidates until accepted if they would change authoritative state.

13.4 Organization formation and launch

1. OrganizationPlan and Sponsor/Catalyst relationships are created.
2. Designee accepts and RegistrationOffering is purchased.
3. Registrar plus registered agent complete the filing saga.
4. Legal adapter verifies evidence and Membership/governing-principle requirements.
5. Organization activates with default policy bundle.
6. Catalyst configures mission, Wisdom, Stance, Connections, Automations, Skills, Forums, Compute, waterfall, agency multiple, membership, and founding alliance.
7. LaunchCampaign validates and opens reservations or purchases only for enabled capabilities.
8. Settled Compute purchases create frozen lineage snapshots and simulated waterfall allocations while live settlement is gated.
9. Minimum reached before deadline closes success; enabled issuance settles with Records, while disabled allocations remain non-payable simulation output.
10. Organization becomes operational; new Compute issuance requires the applicable Forum command.

13.5 Forum proposal to Policy

Use the pipeline in section 8.3. The key implementation rule is that ballot, optional market signal, resolution, local command authorization, and external settlement are separately stateful and separately inspectable.

13.6 Offering purchase

1. Buyer accepts exact Offering terms.
2. Payment instruction is fixed before funds arrive.
3. External payment operation settles through the configured rail.
4. Offering entitlement or Engagement activates.
5. Revenue split executes under recorded terms.
6. No lineage/waterfall command is callable because the purchase type is `OfferingPurchase`, not `ComputePurchase`.

13.7 Model consumption

1. ActionRun records provider/model/tool usage IDs.
2. Metering worker verifies billable units and provider cost.

3. AgencyPricingPolicy calculates Compute charge and premium.
4. Double-entry ledger debits Persona/Project Compute account, credits provider-cost and Organization-premium accounts.
5. Graph links consumption to Action, Project, Organization, and Record.
6. Low balance may trigger an Automation-created reload Action; it never auto-purchases without an active policy and consent.

14. APIs and contracts

14.1 KAP/client endpoints

- `POST /v1/auth/challenges`
- `POST /v1/auth/sessions`
- `POST /v1/accounts`
- `POST /v1/codes/verify|reserve|redeem`
- `POST /v1/invitations`
- `POST /v1/invitations/{id}/accept|decline`
- `POST /v1/relationships/{id}/accept|end`
- `POST /v1/memberships/apply`
- `POST /v1/queries/{name}`
- `POST /v1/actions`
- `GET /v1/actions/{id}`
- `POST /v1/actions/{id}/confirm|defer|cancel`
- `POST /v1/commands` for trusted internal command clients only
- `GET /v1/records/{id}`
- `GET /v1/objects/{protocol_id}`
- `GET /v1/events?cursor=...`
- `POST /v1/packages`
- `GET /v1/manifests/{protocol_id}`
- `POST /v1/migrations/home/preview|confirm`
- `POST /kap/v1/handshake|envelopes|objects/resolve|receipts/resolve|codes/introspect|events/pull`

14.2 Command envelope

```
{
  "command": "project.work.assign@1.2.0",
  "idempotency_key": "01J...",
  "principal_id": "kid:ally:01...",
  "source_persona_id": "kid:persona:01...",
  "organization_id": "kid:org:628407",
  "project_id": "kid:project:01...",
  "expected_versions": {"project": 18, "work_item": 3},
  "parameters": {"work_item_id": "kid:work:01...", "assignee_id": "kid:persona:02..."},
  "proofs": ["session:...", "confirmation:..."],
  "sim": false
}
```

14.3 Response honesty

Responses distinguish:

- **committed**: local state and Record committed.
- **awaiting_confirmation**.
- **waiting_external**: submitted or acknowledged, not settled.
- **completed**: all required terminal conditions verified.
- **denied**: includes safe reason code and remediation if disclosure is permitted.
- **state_changed**: expected version stale; requires a fresh preview.
- **idempotency_conflict**: same key, different canonical payload.

15. Migration from the existing workflow

15.1 Label and concept map

Existing	Replacement	Migration note
User	Account plus Persona	Preserve IDs; Account authenticates and Persona is the durable human principal. Create Membership only where legal admission evidence exists
Presence	AllyIdentity linked to Ki AllyDefinition	Bind one Source Persona; existing custom prompt becomes initial Persona Stance and never a shared tenant prompt
Worker/Bot	Actor or ServicePrincipal	Product-visible functional agents become Actors; infrastructure workers do not
DUNA/Context	OrganizationPlan, Organization, LegalRegistration	Verify legal status and do not create active Organization from name alone
Market	Forum	Remove BECAME ; one Organization may have many Forums

Existing	Replacement	Migration note
Elector	Envoy Actor	Convert permissions and Persona instruction plus exact Membership/Forum authority provenance
KnowledgeBase	WisdomDrop	Add owner, credit, Organization, access, purpose, retention, namespace
Prompt	Stance	Split system/Organization/Persona layers and version
ToolAccount	Connection	Replace credential fields with secret-store refs and scopes
SkillExecution	ActionRun only when consequential	Low-level runs stay in event/telemetry tables
CONNECTED edge	Relationship node plus Grants	Create per-Organization dyad; preserve history
MEMBER_OF edge	Membership node	Capture agreement, status, roles, payment, effective dates
Offering reward	Remove	Only ComputePurchase may call waterfall allocation
Referral ancestry edges	LineageEnrollment plus snapshot	Preserve current parent; rebuild four-generation materialization and history

15.2 Migration procedure

1. Freeze the canonical glossary, labels, IDs, enums, and command registry.
2. Inventory every existing source table, graph label/edge, producer, and consumer.
3. Establish ID crosswalk and `object_registry` without changing runtime behavior.
4. Backfill Accounts and Personas, create isolated AllyIdentities bound to Ki, and backfill Organizations, LegalRegistrations, WisdomDrops, Connections, and lineage.
5. Backfill Memberships only where accepted terms, admission evidence, effective dates, and the applicable Organization can be proven. Classify all other authenticated Personas as Guests in that Organization context; never invent legal Membership to preserve an old product label.
6. Create Forums from Markets, but require manual review where old one-to-one assumptions lose meaning.
7. Convert social edges to Relationship/Grant nodes with conservative access; old invitations do not become accepted Relationships.
8. Introduce Graph Command Service shadow decisions and compare with existing paths.
9. Dual-read for a short measured period; do not long-term dual-write.
10. Cut writes domain by domain to named Commands and transactional outbox.
11. Recompute projections, run integrity queries, and reconcile money/chain references.

12. Remove legacy mutation endpoints and revoke database credentials.
13. Preserve migration Records and crosswalks for audit.

15.3 Integrity queries

- Personas with zero or more than one active AllyIdentity; AllyIdentity with a shared tenant boundary or more than one Source.
- Persona labeled Member without a current Membership in the exact Organization context.
- InvitationProspect with Account, AllyIdentity, wallet authority, Membership, or Relationship.
- Actions without Organization context.
- Membership without agreement/consent Record.
- active Organization without current legal verification.
- Relationship with not exactly two distinct consenting Personas.
- lineage cycle, multiple current parent, or depth beyond policy.
- OfferingPurchase linked to WaterfallAllocation.
- Waterfall sum not equal to purchase amount.
- Proposal opened without receipt renderer or exact command version.
- secret/personal Item in public vector scope.
- external settlement reported complete without verified reference.
- graph object missing object_registry row or version mismatch.

16. Scale and performance

16.1 Launch service objectives

- Named graph read p95 at or below 300 ms, excluding model generation and remote federation.
- Local command commit p95 at or below 700 ms, excluding fresh-auth ceremony and external operations.
- Field semantic first render within 2 seconds on a supported mid-range phone.
- Identity, Graph Command Service, and account web availability target 99.9% monthly during beta.
- Recovery point objective 5 minutes or less; ordinary recovery time objective 4 hours or less.
- Receipt parameter round-trip 100 percent; no independent consequence prose.

16.2 Scaling rules

- Keep authorization writes and current-state reads on the primary. Never authorize from a lagging read replica.
- Use read replicas only for public/low-risk analytics and historical Records where staleness is disclosed.
- Partition event, Record-link, ActionRun, outbox, callback, and ledger tables by time and optionally Organization at measured volume.
- Keep detailed model/tool spans out of AGE. Graph holds durable consequence nodes and current materialized relationships.
- Cache named-query results only with aggregate version, policy bundle hash, viewer scope hash, and short expiry.
- Invalidate reachability and vector-scope caches on access, role, Membership, Relationship, Policy, or home-Ecosystem change.
- Bound traversals by label, Organization, depth, cardinality, statement timeout, and result count.
- Precompute only high-value derived edges such as current four-generation lineage or Project next Action; all materializations include source version and are rebuildable.
- Move time-series and analytical graph workloads to separate stores or replicas before they threaten the command boundary.

16.3 Portability

AGE-specific Cypher stays behind a repository and named-query registry. Domain services depend on typed query/command contracts, not graph-driver details. Maintain a conformance dataset and benchmark suite that can run against a second property graph engine if AGE limitations become material. Do not attempt dual-engine production operation at launch.

17. Security, privacy, and safety

17.1 Threats to test

- Non-Source prompt injection through web, email, Tool output, Wisdom, Actor, or peer Ecosystem.
- Inference of secret existence through count, timing, error, cache, vector, or Scene behavior.
- Actor capability escalation or self-modified manifest.
- Stale permission cache after grant revocation.
- Cross-Organization context bleed inside one Ally.

- Duplicate payment or chain submission after timeout/restart.
- Forged webhook, replayed KAP envelope, Code reuse, or expired signature.
- Lineage cycle, self-referral, collusion, and purchase-type substitution.
- Receipt/parameter mismatch.
- Registered artifact used as a false trust signal.
- Malicious file/package, credential exfiltration, or model-provider boundary violation.
- Compromised Mage, Persona, Organization, Code, payment, or KAP key.
- Invite spam, unauthorized profiling, forged inviter claims, generalized-Code scraping, or trust score used as authority.
- Cross-Persona memory, Stance, Tool credential, or Wisdom leakage through the shared Ki definition.

17.2 Controls

- Passkeys and proof-bound short sessions; fresh authentication for sovereign/high-risk Actions.
- Separate Persona, Organization, Ecosystem, KAP, and service keys with independent rotation/recovery.
- Envelope encryption per artifact; field-level encryption for legal/clinical/financial classes.
- Credential vault references and narrow OAuth/MCP scopes.
- Package quarantine, malware scan, declared manifest, no raw credentials, human review before authoritative promotion.
- Signed KAP envelopes, nonce/timestamp/audience, replay cache, capability narrowing, peer circuit breakers.
- Append-only Records with hash chaining and verifiable key history.
- Policy and authorization negative tests generated from every ActionDefinition.
- Rate limits, budgets, stop conditions, and kill switches by Actor, Tool, Organization, and external operation.
- Zero-notification pre-registration prospect records; inviter delivers the Code out of band unless the prospect separately consents to a channel.
- Signed package manifests with digest and declared authority; `kiduna.md`, `agents.md`, or `CLAUDE.md` text never becomes trusted merely because of its filename.

17.3 Legal and policy gates

This specification is not legal advice. Before activation, counsel and responsible operators must approve the implemented facts for:

- DUNA formation, governing principles, membership threshold, registration, and continuing status;
- Compute issuance, transferability, secondary markets, launch communications, and AgencyPricing;
- lineage and Curator compensation, especially initial purchases and Institution lineage;
- payment flows, custody, money transmission, refunds, and automated splits;
- worker classification, tax forms, and engagement controls;
- legal trust accounts, insurance, mortgage, clinical/health, minors, and other regulated domains;
- Profiler, ambient channel data, privacy notices, retention, deletion, model providers, and international access.

18. Observability and operations

18.1 Required telemetry

- request, ActionRequest, Command, Record, Organization, Project, and external-operation IDs;
- authorization outcome and reason-code class, not secret facts;
- named query/command version and policy bundle hash;
- duration, rows/nodes touched, traversal depth, lock wait, retry count;
- outbox lag, oldest external operation, callback/reconciliation errors;
- model/tool version, token/cost totals, failure class, safety refusal;
- KAP peer health and signature/replay failures;
- ledger imbalance attempts and settlement mismatches.

OpenTelemetry semantic conventions should be used for standard HTTP, database, messaging, exception, and service attributes, with Kiduna-specific attributes under a controlled namespace.

18.2 Persona-visible Vigil

For any selected Action, show a permission-filtered chain:

intent source -> context and Organization -> authority basis -> confirmation -> Actor/Tool/model versions -> exact Command -> local Record -> external status -> result/correction.

The Vigil is an accountability product, not raw debug logs.

18.3 Runbooks before beta

- key compromise by key type;
- data incident and access-scope analysis;
- graph/registry version divergence;
- ledger imbalance or settlement mismatch;
- payment, chain, filing, or DEX outage;
- outbox backlog and poison message;
- model/provider outage or quality regression;
- access-level bug and emergency secret cache invalidation;
- legal registration lapse or Organization dissolution;
- federation abuse, partition, and peer revocation;
- backup restore and projection rebuild.

19. August 10 build plan

The launch should prove the constitutional kernel, not pretend to finish every domain. The schema and registries must be future-complete enough that new DomainPackages do not require changing authority, Action, Record, Organization, or Project fundamentals.

July 14-17: freeze and foundations

- Freeze this ratified glossary, source precedence, invariants, Action model, Organization/legal-form separation, Forum dual mechanism, and Compute feature gates.
- Freeze IDs, schemas, enums, command envelope, error taxonomy, Record and receipt format.
- Pin and test PostgreSQL/AGE/pgvector versions; create migration and backup baseline.
- Build object registry, command/action/query registries, policy bundle, event/outbox, and double-entry ledger skeleton.
- Produce migration inventory and crosswalk.

July 18-24: command boundary and core graph

- Implement Graph Command Service resolve/authorize/validate/commit/record loop.
- Implement Source-only instruction proof, four access levels including secret, Grants, Membership, Relationship, Persona/AllyIdentity, Visitor/Guest/Member context resolution, and the Organization default allowlist.

- Implement invitation prospects, recipient-bound Codes, account creation, consent-separated Relationship acceptance, and separate Membership application.
- Implement ActionRequest/ActionRun/ExternalOperation/Record state machines and Field-facing API.
- Implement WisdomDrop ingestion and authorization-before-vector search.
- Backfill core identity and agent data; shadow named queries.

July 25-31: work, Organization, Forum, and economics

- Implement Project, WorkItem, Engagement, Contribution, acceptance, and one real work loop.
- Implement OrganizationPlan, designee, RegistrationOffering, LegalRegistration adapter, Organization activation/configuration.
- Implement multiple Forums, Proposal, exact command receipts, equal ballot, optional separate market position behind feature flag, Resolution and Policy.
- Implement Compute account/purchase/consumption, AgencyPricing, lineage snapshot, waterfall calculation, and double-entry postings in simulation/shadow mode first.
- Implement Kinship Duna pricing, Guest/Member admission defaults, and Sponsor/Institution distribution path in simulation where live settlement is gated.

August 1-6: hardening and rehearsal

- Run migration twice from a production-like snapshot; reconcile all integrity queries.
- Run authorization, prompt-injection, secret non-discovery, idempotency, external timeout, outbox, ledger, and receipt tests.
- Load-test named reads and command writes; tune indexes and traversal caps.
- Complete backup restore, key rotation, external outage, and rollback drills.
- Obtain counsel/product sign-off for activated financial and governance features; leave unapproved features disabled but schema-compatible.

August 7-9: release cut

- Freeze schemas and command registry.
- Final migration and reconciliation rehearsal.
- Cut legacy writes; verify no generic graph mutation path remains.
- Verify Field/Studio render the same Action and Record IDs.
- Publish operator dashboards, runbooks, support path, known limits, and provenance.

August 10: release

Release only if every P0 gate in section 20 is green. A narrower honest launch is preferable to an architecture that can silently perform unauthorized or misrepresented Actions.

20. Acceptance gates

20.1 P0 release gates

- No non-Source input becomes an Ally instruction.
- Every ActionRequest has exactly one Organization; defaulting is allowlisted and recorded.
- Secret is absent from unauthorized graph, vector, count, cache, error, timing, and Scene paths.
- Personal is ungrantable.
- No protected resource is retrieved then filtered.
- Every mutation is a versioned named Command.
- Every consequential Action has an AuthorizationDecision and Record.
- Receipts are generated from exact parameters and round-trip in tests.
- Same idempotency key and payload returns original result; same key with different payload hard-fails.
- Stale aggregate version returns `state_changed` and requires fresh preview.
- External operations never report settlement before verification.
- Offerings cannot call lineage/waterfall commands.
- Waterfall uses a frozen lineage snapshot and exact policy version; allocations balance.
- Ballot and market position are distinct and cannot be substituted.
- active Organization requires current legal adapter status.
- Mage/service/Actor cannot impersonate Member, vote, or sign.
- Visitor has no Persona/Ally; Guest can invoke only the frozen Guest allowlist; Member authority always cites one current Membership in the exact Organization.
- A pre-registration invite creates no Persona, Ally, Membership, Relationship, or outbound message; redemption and each consent are independently recorded.
- Every live-value capability matches the August 10 activation matrix; disabled commands deny even after model, Role, or Forum request.
- Ledger transactions balance to zero and chain/payment callbacks are replay-safe.
- Backup restore and graph projection rebuild pass.

20.2 End-to-end scenarios

1. Persona who is a Member of the Organization prompts Ki to create a Project; exact Membership authority, Organization, and consequence preview appear; the Persona confirms; Project and Record share IDs in Studio and Live.
2. Non-Source website tells Ally to transfer funds; becomes context only; command is never created.
3. Persona adds secret Wisdom; another Persona, including a Member of the same Organization, cannot infer it without the exact grant; an authorized Project Actor retrieves only in scope.
4. Catalyst buys RegistrationOffering, designee accepts, filing times out and resumes, Org ID verifies, Organization activates exactly once.
5. Scheduled LaunchCampaign accepts reservations, opens, settles an enabled non-transferable Compute purchase, freezes lineage including an Institution ancestor, balances simulated waterfall allocations, pays none while gated, and creates no allocation for a separate Offering purchase.
6. Proposal renders exact Commands; Member ballot and Envoy market position are distinct; conflict-recused Member cannot ballot; passed local command and pending external effect render honestly.
7. Law-firm Engagement runs conflict check, creates secret matter Wisdom, lets Actor draft, requires lawyer signature, and refuses to split trust funds.
8. Solar Project coordinates Site, installer Institution, permit, milestones, and inspection; telemetry creates exception Action without flooding the graph.
9. Actor exceeds budget or asks for a Tool outside its manifest; authorization denies and Record explains safely.
10. Duplicate payment callback and service restart settle one ledger transaction and one final Record.
11. Access is revoked during a long Actor run; next Tool/read call fails and cached scope is invalidated.
12. Peer Ecosystem request carries a registered resource but no grant; registration provides provenance, not permission.
13. Inviter creates a medium-trust invitation for an unknown person; only a prospect and RelationshipIntent exist. The Visitor redeems within 15 minutes, becomes an authenticated Persona/Guest, independently accepts the Relationship, and remains a Guest until the Organization admits a separate Membership application.
14. A Persona is a Member of Kinship Duna and a Guest of another Organization in the same session; the same command is allowed in the first context and denied in the second without state leakage.

15. Home Ecosystem migration preserves permanent Persona and Ally protocol IDs, revalidates Memberships, rejects stale-home writes, and retains signed old/new home Records.

21. Important graph and platform tools

Launch-critical

- **PostgreSQL:** transactional control plane, Records, outbox, ledger, RLS, backups, replication.
- **Apache AGE:** property graph and bounded openCypher traversal inside PostgreSQL.
- **pgvector:** permission-scoped semantic candidates; exact and HNSW search with measured recall.
- **AGE Viewer:** engineering visualization and query inspection in non-production or scrubbed environments.
- **Schema migration tool:** Flyway, Liquibase, or the team's existing equivalent; one migration ledger for SQL, AGE labels/indexes, registries, and policy bundles.
- **OpenTelemetry collector and SDKs:** correlated traces, metrics, and logs without personal prompt capture.
- **S3-compatible object storage plus malware scanning:** encrypted artifacts, hashes, retention, quarantine.
- **Secret manager/HSM:** key and credential references.

Strongly recommended

- JSON Schema or Protobuf registry for Commands, Actions, Records, KAP envelopes, Actor manifests, Grants, and Codes.
- Contract/conformance test harness that runs every named query and command against fixed graph fixtures and adversarial permission cases.
- Property-based testing for lineage, waterfall balancing, idempotency, state transitions, and receipt round-trip.
- PostgreSQL outbox worker for launch; adopt a durable workflow engine only when external saga volume and duration justify it.
- Infrastructure-as-code, signed containers, SBOM, dependency scanning, point-in-time recovery, and restore automation.
- A graph benchmark suite covering deep but bounded lineage, Relationship grants, Project state, Forum docket, access revocation, and high-cardinality public traversal.

Later, at measured need

- CDC/event streaming for independent projections and analytics.
- Dedicated workflow engine for long-running filings, claims, closings, and multi-rail settlement.
- Separate analytics graph or graph data science engine for permitted aggregate analysis.
- Time-series database for device, gameplay, or energy telemetry.
- Search engine for public full-text discovery, still gated by graph authorization for protected resources.
- A second property graph implementation in CI for portability conformance, not dual production writes.

22. Owner ratifications incorporated in v1.1

1. **Human state:** Member is reserved for a Persona who has legally joined the Organization under its terms; authenticated non-members are Guests; unknown or logged-out people are Visitors; Persona is the generic term.
2. **Guest authority:** launch with the restrictive Public/Guest allowlist in section 5.7 and expand only through a versioned command-registry change.
3. **Institution:** first-class outside legal principal with metadata, Agreements, delegates, wallets, verification, authority chain, and optional lineage; never pretend it is a Persona, Member, or Organization.
4. **Governance:** one eligible Member has one equal ballot. A decision-market position is information only and never directly governs unless a future explicit lawful delegation says exactly how.
5. **Command authority:** every DomainCommand declares exactly one required authority class from the machine-readable registry, with risk/confirmation overlays evaluated separately.
6. **Economic launch gates:** transferable/DEX Compute, market positions, lineage payouts, Curator allocation, and Sponsor distribution remain independently disabled until legal, accounting, tax, custody, rail, and exchange review approves the implemented facts.
7. **Missing lineage:** skip upward to the next qualified ancestor; after the traversal limit, route the remainder to the named treasury/reserve account; never fabricate an ancestor.
8. **Liquidity:** 20 percent is a default minimum in versioned Organization policy, not an immutable protocol invariant.
9. **Agency pricing:** start with a simple disclosed schedule, maximum markup, and line-item receipt; Kinship defaults to 7x for Members and 2x the Member charge for Guests.

10. **Legal adapter:** continuously verify legal ID, jurisdiction, standing, formation and continuing requirements including any 100-member requirement, dissolution, authority model, verification source, timestamp, and next review.
11. **Home Ecosystem:** every portable identity/object has one authoritative home and signed federation metadata; migration preserves permanent ID and history.
12. **August 10 packages:** freeze launch activation by DomainPackage and command; regulated finance, securities, lending, custody, payroll, investment advice, insurance, clinical, and autonomous legal effects remain Preview or Disabled as stated in section 9.6.

Appendix A. Minimum Action catalog

Persona, Guest, Member, and Ally

- `account.create`, `persona.profile.update`, `ally.rename`, `ally.stance.update`, `ally.inspect`.
- `wisdom.drop.create`, `wisdom.item.add`, `wisdom.access.set`, `wisdom.inform_agent`, `wisdom.access.request`.
- `connection.add`, `connection.scope.change`, `automation.enable|pause`, `skill.impart|revoke`.
- `invitation.create|accept|decline`, `relationship.propose|accept|end`, `trust.declare|withdraw`, `grant.create|narrow|revoke`, `code.issue|redeem|revoke`.

Organization and membership

- `organization.plan.create`, `organization.designee.accept`, `organization.registration.purchase|submit|verify`, `organization.activate`.
- `membership.apply|admit|suspend|restore|withdraw|expel`, `role.assign|revoke`.
- `community.create`, `alliance.create|merge|dissolve`, `institution.agreement.create|terminate`.

Project and work

- `project.create|archive`, `work.create|assign|accept|reject`, `milestone.complete`, `deliverable.submit|accept`.
- `engagement.offer|accept|change|close`, `split_rule.set`, `invoice.issue`, `payout.authorize`.

- `package.compose|dispatch|recall|return|accept|reject`.

Forum and Policy

- `forum.create|configure, proposal.draft|submit|open|withdraw, ballot.instruct|cast, position.instruct|place|close`.
- `resolution.close, policy.enact|supersede|repeal`.

Compute and treasury

- `compute.define|issue|retire, compute.purchase.intent|confirm, compute.grant.distribute, compute.consume`.
- `waterfall.calculate|allocate|settle, agency_pricing.change, liquidity.add|remove`.
- `treasury.payment.authorize|submit|reconcile, recurring_payment.create|pause`.

Domain

- `legal.matter.open|conflict_check|filing.sign|filing.submit`.
- `insurance.application.review|claim.open|claim.decide|payment.authorize`.
- `solar.site.assess|permit.submit|work.inspect|interconnection.confirm`.
- `mortgage.application.open|disclosure.sign|underwriting.decide|closing.confirm`.
- `care.consent.record|appointment.schedule|care_plan.approve|safety.escalate`.
- `game.session.create|result.record|moderation.act, event.ticket.issue|vendor.approve|safety_plan.sign`.

Appendix B. Sample named Cypher patterns

The exact AGE syntax must be validated against the pinned version. These patterns show intent; production uses parameterized named queries only.

Resolve current Project Actions

```
MATCH (p:Persona {id: $persona_id})-[:SOURCE_OF]->(a:AllyIdentity)
MATCH (ar:ActionRequest)-[:IN_ORGANIZATION]->(o:Organization {id: $org_id})
MATCH (ar)-[:IN_PROJECT]->(p:Project {id: $project_id})
WHERE ar.status IN ['awaiting_confirmation', 'queued', 'waiting_external']
```

```

AND ar.visibility_scope_key IN $authorized_scope_keys
RETURN ar, p, o
ORDER BY ar.priority DESC, ar.created_at ASC
LIMIT 50

```

Check current four-generation lineage

```

MATCH (child:LineageParticipant {id: $participant_id})
MATCH path=(child)-[:CURRENT_PARENT*1..4]->(ancestor:LineageParticipant)
WHERE ALL(rel IN relationships(path) WHERE rel.organization_id = $org_id AND rel.status = 'active')
RETURN length(path) AS generation, ancestor.id AS participant_id
ORDER BY generation

```

The production calculator must also check for cycles, freeze a LineageSnapshot, and bind the WaterfallPolicy version before creating allocations.

Permission-scoped Wisdom provenance expansion

```

MATCH (i:Item)-[:IN_DROP]->(d:WisdomDrop)
WHERE i.id IN $authorized_vector_result_ids
MATCH (d)-[:OWNED_BY]->(owner)
OPTIONAL MATCH (i)-[:MENTIONS]->(e:Entity)
RETURN i.id, d.id, owner.id, collect(e.name)[0..20]
LIMIT 100

```

Appendix C. Source and authority map

Package sources

- [kinship-graph-flow.html](#): prior workflow, seven tools, query guard, current node/edge inventory, graph-as-projection assumption.
- [kinship-graph-gaps.html](#): thirteen canon gaps and retained AGE decision.
- [skill-updates/cofounder-canon-2026-07-13.md](#): major nodes, Field, canonical agent types, Organization-scoped Action accountability.
- [foundation.md](#) sections 1-7: graph boundary, access levels, Relationships/grants, memory, money, Action nodes.
- [orchestration.md](#) sections 1-8: Ki, Source-only instruction, personas, contextual Actions, observability.
- [protocol.md](#) sections 1-4: minimal chain boundary, Commands, receipts, registration, Codes.
- [actions.md](#) sections 1-4: baseline action inventory and extension rule.
- [organizations.md](#) sections 1-4: Organization variance, 31 concrete domains, Compute and lineage defaults.
- [real-work-real-money.md](#) sections 2-5: Projects/engagements, real work, splits, lineage, law practice.

- [institutions.md](#) sections 1-4: outside entity mechanics and launch institutions.
- [legal.md](#) sections 1-4: securities language, payment control, organizer compensation, classification, trust, insurance, conflict.
- [architecture.md](#) sections 1-7 and [kiduna-agentic-internet-spec-v0.1.html](#) sections 3, 5, 6, 10-15: deterministic boundary, canonical distinctions, command loop, storage, development and release gates.

Supplied papers

- [genesis_nightpaper_v2.docx.pdf](#): sense-decide-act, Source/Ally, deterministic checkpoint, real Organizations, Compute language, mesh, Field, Sentinel.
- [Market Forum MetaDAO Proposal Policy Types.pdf](#): domain Command expansion beyond treasury/token proposals.
- [Kiduna __ Techneural Architecture Sync - 2026_07_14 08_30 EDT - Notes by Gemini.pdf](#): Genesis sequence, canonical Ki, onboarding surfaces, Guest/Member economics, Profiler and CodeManager, invitations, directional trust, package manifests, and launch priorities.
- [Agentic Internet graph update.pdf](#): July 14 architecture review context and corrections.
- Product-owner ratification answers supplied July 14, 2026: restrictive Guest authority, Institution ontology, equal ballots, authority classes, economic feature gates, lineage fallback, liquidity policy, pricing disclosure, legal adapters, Home Ecosystem migration, and August 10 package status.

Current official references checked July 14, 2026

- Apache AGE overview and downloads: <https://age.apache.org/overview/> and <https://age.apache.org/download/>
- pgvector indexing, filtering, HNSW, and hybrid search: <https://github.com/pgvector/pgvector>
- PostgreSQL row-level security: <https://www.postgresql.org/docs/18/sql-createpolicy.html>
- West Virginia DUNA Act, especially sections 36-13-2 and 36-13-14: <https://code.wvlegislature.gov/36-13/>
- SEC 2026 crypto-asset interpretation: <https://www.sec.gov/files/rules/interp/2026/33-11412.pdf>
- FTC business guidance concerning multilevel marketing: <https://www.ftc.gov/business-guidance/resources/business-guidance-concerning-multi-level-marketing>
- IRS common-law worker classification: <https://www.irs.gov/businesses/small-businesses-self-employed/employee-common-law-employee>
- OpenTelemetry semantic conventions: <https://opentelemetry.io/docs/specs/semconv/>

Appendix D. Final architectural position

The Kinship Graph should be built as the operating constitution of an agentic network, not as an AI feature database. Its central query is not "what answer should the model give?" It is:

Who may cause this exact consequence, for whom, inside which Organization, using which authority, through which agent or Tool, with what human or Forum confirmation, and which Record will prove both authorization and outcome?

If the system can answer and enforce that question for ordinary Project work, professional practice, organizational formation, governance, Compute consumption, and external settlement, it can support the first release and extend into the wide range of Organizations envisioned. If it cannot, more nodes and more retrieval tools will only make a more articulate assistant. They will not make the agentic internet.