KYE PROTOCOL™ · WHITEPAPER

Agentic Governance™ with Authority Finality™.

Every act has authority. Every authority has finality.

Agentic Governance™ with Authority Finality™.

Who is acting. On behalf of whom. Under what authority. With what proof. Six questions. One signed answer per call.

Maps to SOC 2, ISO 27001, ISO 42001, OSCAL, PSD2, DORA, NIS2, EU AI Act, GDPR, HIPAA.

PLAIN TAKE

Plain take

Your AI agent signs. KYE™ logs. Your auditor verifies from the public key alone. Every privileged call leaves a signed Evidence Pack™ behind.

- You sign. KYE™ logs. Your auditor checks the bytes, not your story.
- Your data stays with you.
- Your keys stay with you.
- The verifier is open source. Anyone can re-derive the verdict.

Whitepaper · Version 1.0 · April 2026 · github.com/KYE-Protocol · **Download PDF (2.3 MB)**

> **KYE Protocol™** proves *who or what is acting*, *on behalf of whom*, using *which capability*, under *what authority*, *in what state*, and with *what audit trail* — for every action a human, business, AI agent, service, model, tool or workflow takes. The technical and evidentiary foundation for the **KYE Chain of Authority™** — the linked, attenuating delegation chain over which **Authority Finality™** is asserted. For AI-agent systems: accountability, compliance, dispute resolution, legally defensible audit trails.

Contents

Abstract

The agentic stack — AI agents, models, tools, workflows acting autonomously on behalf of humans and businesses — is reaching production at a velocity that has outrun the identity, authorization and audit infrastructure beneath it. KYC verifies humans. KYB verifies businesses. KYA (just emerging in 2026 from Visa, Skyfire, Persona, Sumsub, Trulioo) verifies agents. Each layer is siloed; each verifies once, at registration; none answers "*is the answer still true 200 ms from now when the next call arrives?*"

KYE Protocol™ — *Know Your Entity™* — is the **Entity Authority Protocol for AI governance**. It proves *who or what is acting*, *on behalf of whom*, using *which capability*, under *what authority*, *in what state*, and with *what audit trail*. Every entity (human, business, agent, service, model, tool, workflow) shares one identifier — the **KYEID™**, in URN form `kye:<class>:<trust-domain>:<subclass>:<local>`. Every action is bound to a signed delegation chain with attenuable scope, gated at runtime by the **Authority Gate™** and admitted by an unrevoked **Purpose Permission™**. Every decision is projected as a signed **Decision Map™**, sealed under an **Execution Context Seal™**, exported in an **Evidence Pack™**, and re-verifiable as a **Replay Proof™**. Every call is logged to the **AI Call Ledger™**. Every revocation feeds the **Resilience Loop™** — *detect drift → revoke → re-grant → replay → improve*. The result: a complete **KYE Chain of Authority™** over which **Authority Finality™** holds — replayable proof for accountability, compliance, dispute resolution, and legally defensible audit trails in agentic systems. KYE™ does not replace legal agreements, signatures, or regulatory obligations; it provides the technical and evidentiary foundation for them.

0 · Canonical inventory

Five locked inventories define the whole surface area of KYE™: a **nine-element kernel**, **seven rails**, **eight surfaces**, **ten engines**, **eight public profiles**, and **seventeen schema namespaces**. Nothing in the protocol — not a schema, not an engine, not a page — lives outside these inventories. They are governed by `constitution/01-NAMING.md` and bind every downstream specification.

0.1 The kernel (9 elements)

Every KYE™ feature, decision, schema, engine and product surface reduces to one of these nine kernel concepts, in order:

Entity → Purpose → Authority → Scope → State → Rules → Decision → Evidence → Replay

Commercial restatement: *Define → Govern → Decide → Prove → Revoke → Replay → Improve.* The 9-element kernel is the **contract**; data structures, schemas and engines are how that contract lands on the wire. If a proposed feature does not advance one of these nine, it is secondary and does not get a top-level home.

0.2 The seven rails

#	RAIL	OWNS KERNEL	SUB-MODULES · CANONICAL SUB-ENGINES	
1	**Operating Model Rail**	Define	Operating Model™ · Authority Mo	
2	**Authority Rail**	Authority	Entity Registry · Agent Entity · Pri	
3	**Purpose & Scope Rail**	Purpose / Scope	Purpose Permission™ · Action Pur	
4	**Runtime Decision Rail**	Decide	Decision Engine · Rules & Obligat	
5	**Evidence & Replay Rail**	Evidence / Replay	Decision Record · Decision Map™	
6	**Directory & Assurance Rail**	Govern / Prove	KYE Directory™ · KYE Authority Se	
7	**Ecosystem Rail**	Improve	KYE Partner Engine™ · KYE Partne	

0.3 The eight surfaces

#	SURFACE	WHAT IT IS	
1	**KYE Protocol™**		The ope

2	**KYE Gateway™**	The runt
3	**KYE Cloud™**	Custome
4	**KYE Directory™**	Search &
5	**KYE Assurance™**	Evidenc
6	**KYE Counterparty Governance Rail™** *(renamed from KYE Partner Rail™)*	Runtime
7	**KYE Developer Tools™**	Integrat
8	**KYE Admin Console™**	**Owners**

0.4 The ten engines

#	ENGINE	OWNING RAIL	OWNS
1	**Entity Engine**	Authority Rail	Entity Registry, Agent Entit
2	**Authority Engine**	Authority Rail	Delegation, Grant, Graph, C
3	**Purpose & Scope Engine**	Purpose & Scope Rail	Purpose Permission™, Adm
4	**State Engine**	Runtime Decision Rail	Lifecycle states, six-dimens
5	**Rules Engine**	Runtime Decision Rail	Rights, Obligations, Stop Co
6	**Decision Engine**	Runtime Decision Rail	PDP / ePDP / sPDP, signal fu
7	**Evidence Engine**	Evidence & Replay Rail	Decision Map™, Evidence P
8	**Replay Engine**	Evidence & Replay Rail	Execution Context Seal™, P
9	**Directory Engine**	Directory & Assurance Rail	Search, indexing, blast radi
10	**Ecosystem Engine**	Ecosystem Rail	Partner, Connector, Widget

Engines **decide deterministically**; agents (Temporal-style workflows) **orchestrate** calls to engines. Continuity is a *module* inside Runtime Decision Engine + the Assurance surface, not a top-level engine. Discoverability is the Directory Engine's surface name. Ontology / Taxonomy / Metadata / Dictionary all fold into a single **Semantic Registry** supporting all engines — not a standalone product.

0.4.1 New sub-engines (2026-05-17)

The 10 top-level engines are locked by constitution -NAMING. New named sub-engines shipped 2026-05-17 under existing parents — preserving the lock while extending the runtime surface for compliance, search, agent memory, and self-reporting.
The construction of each sub-engine is proprietary and is not disclosed in this repository.

SUB-ENGINE	PARENT	OBSERVABLE CONTRACT
KYE Data Mapping Agent™	Directory Engine	Signed `kye.data_flow_graph.v1` envel
KYE Native Search Engine™	Directory Engine	Signed `kye.search_result.v1` envelop
KYE Memory Engine™	Decision Engine	Signed `kye.agent_memory.v1` records w
KYE Reporting Engine™	Evidence Engine	Signed `kye.report.v1` envelope per (t

0.5 The eight public profiles

The public profile catalogue is exactly eight. v1.0 ships **8 public profiles + sector extensions**. Sector profiles (Healthcare, Payments, Open Banking, Sovereign AI, Agent Purchasing, Custody, Treasury, Capital Markets, Insurance) are extensions of the eight, not equal to them.

#	PROFILE	WHAT IT NORMALISES
1	**KYE Core Profile™**	Entity model, lifecycle, delegation, scope, decisi
2	**KYE Agent Entity Profile™**	AI-agent / model / capability identity + attestatic
3	**KYE Operating Model Profile™**	Pre-runtime authority design + behaviour model
4	**KYE Authority Profile™**	Delegation chains, grants, graphs, revocation
5	**KYE Purpose & Scope Profile™**	Purpose Permission™, admissibility, scope, restri
6	**KYE Runtime Decision Profile™**	PDP shape, rules, obligations, continuity, commit
7	**KYE Evidence & Replay Profile™**	Decision record, Decision Map™, Evidence Pack™
8	**KYE Assurance Profile™**	Directory, conformance, certification, regulatory

0.6 The 17 schema namespaces

Every JSON schema, regardless of which profile authored it, MUST live under exactly one of these top-level families:

#	NAMESPACE	OWNS

1	`kye.entity.*`	Entity records, agent entity, KYEID™ resolution
2	`kye.operating_model.*`	Pre-runtime artefact, Authority Model™, Behaviour Model™
3	`kye.authority.*`	Grants, delegations, graphs, gate decisions, revocations
4	`kye.purpose.*`	Purpose Permission™ grants and bindings
5	`kye.scope.*`	Scope, restrictions, time windows, data classes
6	`kye.state.*`	Six-dimension state, composition matrix, transitions
7	`kye.rules.*`	Rules, obligations, stop conditions, policy bundles
8	`kye.decision.*`	Decision records, reason codes
9	`kye.evidence.*`	Decision Map™, Evidence Pack™, audit events, AI Call Ledge
10	`kye.replay.*`	Execution Context Seal™, Replay Proof™
11	`kye.directory.*`	Index descriptors, directory entries, query/result shapes
12	`kye.assurance.*`	Conformance reports, certifications, regulatory mappings
13	`kye.partner.*`	Partner entity, certification, deal, widget licence
14	`kye.connector.*`	Connector manifests, capability descriptors
15	`kye.developer.*`	SDK descriptors, MCP tool inventories, sandbox artefacts
16	`kye.billing.*`	Usage, metering, invoicing records
17	`kye.admin.*`	Owner-only operator records (tenants, key rotation logs)

1 · Problem

Modern agentic workflows generate three classes of pain that legacy identity stacks cannot resolve.

1.1 Fragmented identity

One agent typically holds three or four identities at once: a SPIFFE SVID for workload attestation, an OAuth client_id for the API gateway, a vendor-specific KYA passport for payment rails, and a model card for inference governance. Each format is reconcilable only by hand. Auditors reconstructing an incident traverse four logging systems and stitch traces by timestamp.

1.2 Static authorization

OAuth scopes and KYC files describe state at issuance. Neither propagates a revocation. When a delegated agent is compromised, the human delegator may not learn about it until the next compliance review. Stop signals (entity stopped, credential revoked, attestation stale) need to ripple through dependent grants in real time, recursively, with cryptographic guarantees they were applied.

1.3 Unprovable history

Audit logs are usually JSON lines in a search engine. They are searchable but not provable: a malicious operator can edit the past. Each audit event is durably ordered and publicly committed; the construction is proprietary and is not disclosed here.

2 · Prior art & gaps

LAYER	SOLVES	DOESN’T SOLVE
OAuth 2.1 / GNAP	Human authorization, token introspection	No agent identity, no delegation chain, no cascade
SPIFFE / SPIRE	Workload identity (SVIDs)	No first-class delegation, no decision vocabulary
Anthropic MCP	Agent ↔ tool communication	No identity, no policy, no audit
Google A2A / ADK	Agent registration metadata	No delegation, no cascade, no proof
OpenID AuthZEN	Standard decision API shape	No URN, no chain, no signals

OpenSSF SCITT	Transparency receipts No identity, no scope, no cascade
OpenID SSF / CAEP	Stop-event distribution No delegation chain, no Evidence Pack™
Visa Trusted Agent / Skyfire / Persona KYA	Agent passport (AID), spend caps (TXG) Vendor-specific, agent-only, no unified URN, no cascade

Each of these solves a slice. None composes into a single contract that an auditor can read end-to-end. KYE Protocol™ is that contract layer; it does not replace these, it composes them.

3 · Design principles

KYE Protocol™ stands on **16 protocol-core principles** grouped in three tiers: six runtime-governance semantics that define what KYE™ *decides*, eight protocol-design disciplines that define how KYE™ *classifies and ships*, and two developer-adoption disciplines that define how KYE™ *reaches developers*.

3.1 Tier A — Runtime governance (what KYE™ decides)

1. **Authority-first.** KYE™'s centre is authority, not identity. Every action answers *who or what is acting, on behalf of whom, using which capability, under what authority, in what state, with what audit trail*. Core records: `KYEAuthorityGrant`, `KYEAuthorityScope`, `KYEDelegation`, `KYEActingOnBehalfOf`, `KYEAuthorityState`, `KYEAuthorityDecision`, `KYEAuthorityRevocation`, `KYEAuthorityEvidence`.
2. **State-first.** Authority is meaningless without state. Every authorize call composes `entity_state`, `authority_state`, `delegation_state`, `credential_state`, `capability_state`, `payload_state`, `recovery_state`, `risk_state`. KYE™ is state-aware authority infrastructure.
3. **Decision-first.** Runtime systems need an answer. `POST /v1/runtime/authorize` returns one of `allow` / `allow_with_constraints` / `require_approval` / `require_step_up` / `require_human_review` / `require_recovery` / `quarantine` / `deny` — with reason code, matched policies, obligations, evidence refs — in milliseconds.
4. **Evidence-first.** KYE™ turns authority into evidence. Every decision produces a `policy_decision_id` binding to validation results, audit events, transparency receipts, signals, and a signed `evidence_pack` consumable by GRC, auditors, regulators, dispute panels.
5. **Audit-trail-first.** No authority without audit. No audit without evidence. Every material change emits an append-only, hash-linked, timestamped, reason-coded, actor-bound, policy-bound, decision-bound audit event — exportable, replayable, externally verifiable.

3.2 Tier B — Protocol design (how KYE™ classifies and ships)

6. **Schema-first.** JSON Schema 2020-12 with absolute `$id` at `kyeprotocol.com/schemas`. OpenAPI, SDKs, validators, docs, and conformance tests are derived from schemas, not hand-written.
7. **Dictionary-first.** Canonical vocabulary for entity types, all state dimensions, decisions, reason codes, capability kinds, side-effect levels, data classes, signal types, redaction fields, taxonomies, graph types.

8. **Taxonomy-first.** 16 V1 canonical taxonomies (entity_type, capability_type, action_type, resource_type, data_class, side_effect_level, risk_state, environment, decision, reason_code, evidence_type, compliance_framework, sector, jurisdiction + state taxonomies). Versioned, status-bound, mappable to framework controls.
9. **Metadata-first.** Every KYE™ object carries a normative metadata block — `labels`, `classifications`, `ownership`, `lineage`, `compliance`. Field values draw from registered taxonomies. Metadata *influences decisions*; the runtime exposes it to the policy layer as `request.{actor,capability,resource,authority}.metadata`.
10. **Graph-first.** Authority is relational. Every entity, delegation, capability, credential, policy, state, decision, and evidence object is a node; every authority relationship is a typed edge. Authority Graph™, Decision Map™, Evidence Graph™, Blast Radius Map™, Compliance Map™. Storage substrate is implementation choice.
11. **Profile-first.** Core stays small. **8 public profiles + sector extensions** add domain semantics; profiles never modify Core.
12. **Registry-first.** Every object is resolvable. `/.well-known/kye` advertises versions, profiles, crypto suites, JWKS, dictionaries, taxonomies, metadata schemas, registries.
13. **Conformance-first.** 129-fixture black-box pack that any conformant Gateway must pass. Vendor self-attestation via `conformance-report.json`.

3.3 Tier C — Developer adoption (how KYE™ reaches developers)

14. **API-first.** 550 OpenAPI operations across 87+ runtime endpoints across runtime, registry, taxonomy, metadata, and graph endpoints. `POST /v1/runtime/authorize` is the headline; `GET /v1/decisions/{id}/map` returns a Decision Map™.
15. **SDK-first.** TypeScript, Python, Go SDKs ship with schema types, local validators, decision clients, signing/verification helpers, policy adapters, audit emitters, evidence-pack builders, taxonomy resolvers, metadata classifiers, graph traversal clients, decision-map renderers.

Protocol-design corollaries that fall out of the 16 principles:

- *Delegation is a graph, not a flag.*
- *Scope must attenuate, not extend.*
- *Revocation and quarantine signals fan out before the next request.*
- *Profiles, not forks.*
- *Mechanism designs are proprietary* (decision evaluation, audit-record linking, signal propagation, scope attenuation, signing-suite construction, graph traversal).

4 · Conceptual model

KYE™ governs acting entities, principal entities, capability entities, resource entities, credential entities, and evidence artefacts — and keeps them strictly distinct. Payloads are *evidence artefacts*, never authority-bearing entities; the **KYE Payload Trust Profile™** binds signed payloads to entity authority, capability state, policy decisions, and replayable audit evidence.

KYE Protocol™ surfaces the 9-element kernel through these first-class on-the-wire records. The 9-element kernel chain — **Entity → Purpose → Authority → Scope → State → Rules → Decision → Evidence → Replay** — is the *contract*; the records below are how that contract lands on the wire.

- `Entity` — any actor or resource. Identified by a **KYEID™**. Has a status, a lifecycle state, an immutable block, classification, assurance, optional sectoral profile.
- `PurposePermission` — the **Purpose Permission™** grant: *which entity, for which purpose, against which data classes, under which restrictions, for which time window?* If any sub-question is `null`, the Decision Engine MUST deny. Admitted upstream of the Authority Gate™.
- `Delegation` — a record that *actor* may act for *subject*, granted by *delegator*, within *scope*, for *allowed_actions*.
- `Scope` — named bundle of constraints + obligations. Attenuable through `parent_scope_id`.
- `AccessRight` — fine-grained, resource-level grant.
- `Credential` — signed assertion about an entity. Issuer + holder + subject + claims; the signature suite is proprietary and is not disclosed here.
- `Attestation` — workload identity binding (SPIFFE / EAT / build provenance).
- `Signal` — reactive event on the bus. Stop / quarantine / revoke / cascade — the drift class that drives the **Resilience Loop™**.
- `PolicyDecision` — record of an authorize call: decision, reasons, obligations, stop_conditions. Projected as a signed **Decision Map™** (the graph of inputs → rules → obligations → output).
- `AuditEvent` — append-only entry written to the **AI Call Ledger™**. Audit events are durably ordered and publicly committed; the construction is proprietary and is not disclosed here.

Three derived records support proof, replay & observability:

- `EvidencePack` — **Evidence Pack™**: signed export of `(decision_map, inputs, signed_state, signals, signatures, execution_context_seal)` for one decision or one workflow. Verifiable by external auditors with public keys alone.
- `ExecutionContextSeal` — **Execution Context Seal™**: the cryptographic snapshot of all runtime inputs (model build, rule bundle, signal versions, time, RNG seed) that makes a decision deterministically replayable. Re-running the Execution Context Seal through the same rule bundle produces an identical Decision Map™ and yields a signed **Replay Proof™**.
- `TransparencyReceipt` — signed receipt that a statement is included in the public log at a given index.

4.1 Multi-dimensional state model

An entity is not a single state. KYE Protocol™ runtime evaluates authorization against multiple independent state dimensions covering lifecycle, authority, delegation, credential validity, recovery posture, and risk telemetry. The specific dimensions, their state alphabets, the per-dimension transition rules, and the composed-test construction are proprietary and are not disclosed in this repository. The canonical dimension and state-value vocabulary lives in the private conformance pack; conformant implementations consume it via the runtime contract, not by re-deriving it from the public surface.

4.2 Cascade revocation & the Resilience Loop™

When an entity is stopped, quarantined, revoked or marked compromised, downstream authorities are no longer usable. The propagation mechanism is proprietary and is not disclosed here. Dependent delegations, payment authorities, access rights, capability grants, and recovery decisions are all affected. What KYE™ *does* publish is the surrounding loop: **Resilience Loop™** is the closed-loop control surface — *detect drift → revoke → re-grant → replay → improve*. A governed deployment that can emit evidence but cannot close a loop on its own drift is **not** conformant. The Resilience Loop turns drift signals — model drift, purpose drift, principal drift, scope drift, agency_drift — into runtime control events with the same SLO discipline as an HTTP request.

4.3 · Implementation architecture

KYE Protocol™ stands on **16 protocol-core principles** in three tiers (see §3 above). Tier A — runtime governance — defines what KYE™ *decides*: *authority-first, state-first, decision-first, policy-bound, evidence-first, audit-trail-first*. Tier B — protocol design — defines how KYE™ *classifies and ships*: *schema-first, dictionary-first, taxonomy-first, metadata-first, graph-first, profile-first, registry-first, conformance-first*. Tier C — developer adoption — defines how KYE™ *reaches developers*: *API-first, SDK-first*. Canonical JSON Schemas drive OpenAPI, SDK code generation, validators, documentation, and conformance fixtures. Shared dictionaries (entity types, states, decisions, reason codes, capability kinds, side-effect levels, data classes, signal types, taxonomies, graph types) make implementations interoperable. Versioned taxonomies (16 V1 canonical taxonomies) and a normative metadata model (labels, classifications, ownership, lineage, compliance) drive policy decisions at runtime. Profiles keep Core small while letting domain extensions (Capability, Recovery, Payments, Payload Trust, Taxonomy & Metadata, Graph, Healthcare, EU AI Act, ISO 42001, sector overlays) compose without modifying it. Registries make every object resolvable. The runtime API is decisioning-led, with `POST /v1/runtime/authorize` as the headline endpoint and `GET /v1/decisions/{id}/map` returning a Decision Map™ for every adjudication. SDKs in TypeScript, Python, and Go ship with schema types, signing/verification helpers, evidence-pack builders, taxonomy resolvers, metadata classifiers, graph traversal clients. Every adjudicated action produces a signed `evidence_pack` a regulator can verify with public keys alone. A 129-fixture black-box conformance pack tests vendor and reference implementations byte-for-byte the same.

5 · Contract surface

v1.0 publishes **550 OpenAPI operations** across the published OpenAPI specs (87+ runtime endpoints in the KYE Reference Gateway™), spanning resource families: Entities, Delegations, Scopes, Access Rights, Credentials, Attestations, Capabilities, Capability Grants, Runtime (authorize / invoke), KYE Signal Bus™ (publish / subscribe / replay / webhook-endpoints / deliveries), Audit, Evidence Packs, Transparency, Federation, Recoveries, Break-Glass Grants, Compromise Reports, Keys, Conformance, plus the Payments families (Wallets, Payment Authorities, Payment Intents, Beneficiaries). **556 JSON Schemas** with absolute `$id` URIs at `https://kyeprotocol.com/schemas/`; **708 example payloads** pinned to schemas and validated in CI. **133 conformance fixtures** assert deterministic behaviour under happy paths and edge cases.

Every state-changing endpoint accepts an `Idempotency-Key` header and returns the original response on replay; conflicting bodies under the same key return HTTP 409. Every state-changing endpoint emits an audit event whose `correlation_id` matches the originating request.

6 · Reference runtime

The reference deployment ships:

- A **KYE Reference Gateway™** in Node.js (no Express, no external runtime dependency) that implements every endpoint and demonstrates conformant cascade behaviour.
- An **embedded PDP library** (`@kye/epdp`) for low-latency local decisions backed by signed bundles.
- An **Express PEP middleware** (`@kye/pep-express`) that fails closed for high-risk actions when the central PDP is unreachable.
- **Three SDKs** — TypeScript, Python, Go — each with idempotency-key support and webhook helpers.
- A **conformance runner** that executes the fixture pack against any candidate Gateway.
- A **Cloudflare Worker** + **wrangler.toml** deploy config for Cloudflare-native production patterns.

The reference is illustrative. It does not implement the proprietary decision algorithm; conformant production Gateways are expected to substitute that mechanism while preserving the open contract surface.

6.5 · Resilience Loop™

A governed deployment of KYE Protocol™ MUST run a **Resilience Loop™** as a continuous control. A system that can issue grants and emit evidence but cannot close a loop on its own drift is **not** conformant. The loop is the difference between an audit log and a control system.

The Resilience Loop™ is the closed-loop control surface that continuously **detects drift**, **revokes** the affected grants, **re-grants** under corrected scope, **replays** affected decisions against the new state (producing fresh **Replay Proofs™**), and emits an **improvement record** that proves the loop closed. The five drift classes are:

- **Model drift** — upstream provider rotates a model; tool-use distribution shifts.
- **Purpose drift** — a principal narrows the purposes they had previously authorised, or refuses a reconfirmation prompt against their **Purpose Permission™**.
- **Principal drift** — a delegating principal loses authority — director removed, subsidiary divested, agent re-assigned.
- **Scope drift** — an agent calls tools or touches data classes outside the scope it was admitted for.
- **Agency drift** (`agency_drift`) — the autonomy envelope shifts — a previously human-in-the-loop step is now agent-only, or vice-versa.

Each drift event is dispatched to closure with the same SLO discipline as an HTTP request. Detect-to-revoke, revoke-to-replay and replay-to-improve are time-bound floors the rail commits to. The loop closes against the **AI Call Ledger™** entries and Evidence Packs™ the runtime already produced; the improvement record updates the Operating Model™ rule bundle via the **Control Compiler™**.

7 · Profiles — the full v1.0 inventory

Profiles add vocabulary, schemas, endpoints, and conformance fixtures to Core. v1.0 ships **8 public profiles + sector extensions**; the eight are listed in the inventory at §0.5 — eight public profiles. Sector profiles (Healthcare, Payments, Open Banking, Sovereign AI, Agent Purchasing, Custody, Treasury, Capital Markets, Insurance) are extensions of the eight, not equal to them. The shipping inventory below lists Core, Gateway and the sector extensions alongside their owning rail:

Core + supporting profiles:

- **KYE-Core** — entity registration, delegations, scopes, credentials, attestations, runtime authorize / evaluate, audit chain, signal bus.
- **KYE-Gateway** — API surface, headers, idempotency, content types, signing, error envelopes; the contract any conformant implementation honours.
- **KYE-Federation** — cross-trust-domain entity import with attenuated scope and origin metadata; signed assertions; transferred entities preserve provenance.
- **KYE-Credentials** — issue / verify / present / revoke with Ed25519 detached signatures over canonical payloads.
- **KYE-Attestation** — SPIFFE / EAT / build-provenance bindings; explicit revocation; stale detection.
- **KYE-Signals** — pub/sub bus with subscribe / ack / replay; signed webhook delivery with replay-window enforcement and key rotation. The specific signing-suite vectors are part of the normative specification and are not published in this repository.
- **KYE-Transparency** — append-only statement log + signed inclusion receipts; verifiable with the gateway's public keys alone.
- **KYE-Conformance** — 129-fixture black-box pack + reporter; tests vendor stacks, internal stacks, and the KYE Reference Gateway™ with the same pack.
- **KYE-Treasury** — treasury authority chain (sweeps, rebalances, intercompany transfers, FX); reconciliation hooks bind state-changing intents back to authority + scope.
- **KYE-Custody** — asset custody chain of authority; workload attestation binds runtime, custody authority binds wallet, audit chain binds timeline.
- **KYE-Healthcare** — HIPAA-aligned overlay; binds the agent to consent credentials, attaches redaction obligations and external-send blocks.
- **KYE-Telemetry** — structured authorization-decision telemetry (policy, inputs, reason code, decision time, entity chain).

- **KYE-Capability** — skills / tools / MCP tools / functions / connectors / prompts / workflows / playbooks / runbooks / model_profiles; register, grant, invoke (allow / deny / require-approval / quarantine), supersede, revoke.
- **KYE-Recovery** — recovery requests, decisions, signed proofs, time-boxed break-glass grants, compromise reports; replaces ad-hoc admin reset.
- **KYE-Payments** — payment authorities, beneficiaries, intents; sPDP with currency / amount / rail / approval gating; PSD3-aligned obligations.
- **KYE-Payload-Trust** — payload artefacts as first-class evidence (not entities). Signed, hashed, replay-resistant request bytes carry state but never authority. 13 lifecycle + denial states; 10 deny reason codes.
- **KYE-Taxonomy-&-Metadata** — 16 V1 canonical taxonomies plus a normative metadata model (labels, classifications, ownership, lineage, compliance). Metadata is not decorative; it influences runtime decisions and binds to compliance-framework controls.
- **KYE-Graph** — the Authority Graph™ contract. Canonical node + edge schemas; Decision Map™, Evidence Graph™, Blast Radius Map™, Compliance Map™ as first-class projections. Storage substrate (Postgres, Neo4j, Neptune, Memgraph, TigerGraph, ArangoDB, RDF) is implementation choice.

Payment overlays (jurisdictional):

- **kye-payments-eu-1.0** — PSD2/PSD3 + DORA alignment.
- **kye-payments-card-1.0** — PCI DSS 4.0 alignment.
- **kye-payments-high-assurance-1.0** — ISO 20022 alignment for high-value flows.

Sector overlays beyond these (healthcare-clinical / payer / research, 42 CFR Part 2) are intentional placeholders for v1.1.

7.1 · Sector coverage

The eight public profiles and their sector extensions compose into nine sector-ready bundles:

- **Retail & commercial banking** — Core + Payments + Treasury + Federation + Capability + Recovery.
- **Payments & cards** — Core + Payments + Credentials + Signals + Telemetry.
- **Healthcare & life sciences** — Core + Healthcare + Credentials + Capability + Telemetry.
- **Capital markets & treasury** — Core + Treasury + Custody + Attestation + Transparency + Recovery.
- **Custody & digital-asset operators** — Core + Custody + Attestation + Credentials + Recovery + Capability.
- **Insurance & underwriting** — Core + Credentials + Federation + Capability + Telemetry.
- **AI labs & agent platforms** — Core + Capability + Attestation + Signals + Recovery + Telemetry.
- **Public sector & defence** — Core + Federation + Attestation + Transparency + Recovery.
- **Marketplaces & platforms** — Core + Federation + Capability + Signals + Telemetry.

7.2 · Compliance & control mapping

The compliance addendum ships **1,791 individually-mapped requirements** across **21 horizontal frameworks** (the framework-coverage bijection gate cross-checks every requirement against a canonical KYE™ artefact — schema, runtime engine, agent, worker, gateway, or verification gate; honest *enforced / designed / advisory / out-of-scope* per row; ~82% weighted enforced coverage at this revision). Each row points to the specific KYE™ artefact — endpoint, schema, profile section, runtime engine, or conformance fixture — that satisfies it. Horizontal frameworks covered:

- SOC 2 Trust Services Criteria 2017 (rev. 2022)
- ISO/IEC 27001:2022 Annex A
- PCI DSS 4.0
- PSD2 + PSD3 / PSR
- DORA (Digital Operational Resilience Act)
- NIS2
- EU AI Act (Title III high-risk system obligations)
- ISO/IEC 42001 (AI Management System)
- NIST AI Risk Management Framework
- NIST SP 800-207 (Zero-Trust Architecture)
- NIST Cybersecurity Framework 2.0
- GDPR / UK GDPR
- FedRAMP

Sector overlays bind on top of the horizontal mappings: HIPAA Privacy + Security Rule (US healthcare), MiCA (EU crypto-asset markets), FFIEC (US bank exam guidance), IEC 62443 (industrial cybersecurity), 42 CFR Part 2 (US substance-use confidentiality).

The mapping is the audit-evidence skeleton an enterprise GRC team uses to certify a deployment.

The **KYE Compliance Mapping Rail™** (`schemas/compliance-mapping.json`) binds each framework control to the specific KYE™ runtime events that produce evidence for it; the **KYE EU AI Act Profile™** (`kye-euaiact-v1.md`) is the first such mapping, covering the EU AI Act with 10 controls (KYE-EUAIACT-001 through KYE-EUAIACT-010): entity accountability, AI system registry, capability manifest + risk classification, human-oversight decision gates, runtime authority decision logs, technical documentation evidence pack, corrective action and revocation trail, provider/deployer/operator role mapping, high-risk workflow profile, and post-market monitoring evidence hooks. KYE™ EU AI Act does not replace conformity assessments, notified bodies, or fundamental-rights impact assessments — it provides the signed evidence those processes consume.

Sector profiles in v1.0 cover financial services, healthcare, custody, treasury, AI labs, public sector, marketplaces, defence, critical infrastructure, energy, manufacturing, oil & gas, mining, automotive, maritime & shipping, logistics, and aviation. Each composes with KYE™ EU AI Act when AI systems or AI agents participate.

7.3 · Conformance & certification

v1.0 ships a **129-fixture black-box conformance pack** covering registration, delegation, scope attenuation, lifecycle and stop-cascade, capability invoke (allow / deny-quarantined / require-approval), capability-grant cascade, recovery flow, state transitions (allowed / rejected), break-glass grants, compromise cascade, key rotation, point-in-time replay, payments allow / deny / approval, audit integrity, idempotency, federation transfer, transparency log append + receipt, signal cascade, webhook delivery, Evidence Pack™ export, workload attest. Fixtures speak only HTTP — any conformant implementation can be tested with the same pack regardless of language, runtime, or topology. The conformance reporter emits machine-readable evidence for the auditor (`conformance-report.json`, normative as of v1.0).

7.4 · Normative addenda (gap-closure register)

An independent v1.0 normative review identified 15 gaps across blockers, important and polish classes. Each is closed at the spec contract level; the underlying mechanisms remain proprietary.

Blockers (5 / 5 closed)

- **State-composition transition matrix** — the matrix publishes the 10 illegal compositions, 5 break-glass entry conditions and 3 terminal states across the 6-tuple state. Normative version ships in the procurement pack; the public conformance mirror exposes the test-fixture form.
- **Wire-protocol version negotiation** — Gateway spec: `Accept-Version` header, optional URN version segment, `/.well-known/kye::kye_supported_versions`, 18-month backward-compatibility floor.
- **Cascade atomicity contract** — Gateway spec: caller-visible end-state semantics. The contract shape, the on-the-wire summary, and the algorithm are proprietary and remain unpublished.
- **RFC 7807 problem+json error envelope** — Gateway spec + `schemas/problem-detail.json`, with status-code map and reason-code pinning.
- **MCP elicitation / sampling / versioning** — `mcp-agent-runtime` profile: gating per MCP operation, sampling budgets, tool versioning rules.

Important (5 / 5 closed)

- **Recovery m-of-n approval** — `evidence-replay` profile + `compliance-evidence` rule pack (`approval_quorum`, ordered/unordered, reject-wins, window-expiry escalation).
- **Conformance-report schema** — promoted from v1.1 placeholder to v1.0 in `schemas/conformance-report.json`.
- **Payments post-execution lifecycle** — `financial-services` rule pack + Financial Services sector pack: 10 states (hold, release, reversed, disputed, charged_back, dispute_resolved, settled, …), 9 signal types, ISO 20022 message-class alignment.
- **Telemetry redaction / sampling / export MUST** — `telemetry` dictionary + `compliance-evidence` rule pack: redaction field list, per-decision-class sampling floors, OTLP and CloudEvents 1.0 exports.
- **Quantitative SLA conformance tiers** — Gateway SLA spec: Tier-1 Bank / Tier-2 Mid-market / Tier-3 Reference targets for p50, p99, throughput, cascade latency.

Polish (5 / 5 closed)

- **Selective disclosure + GDPR right-to-erasure** — `credentials` dictionary + `compliance-evidence` rule pack (SD-JWT, BBS+, Article 17 erasure flow). DSAR evidence assembly lands in V1 via **KYE Data Governance Pack™** (constitution §31 — landing page ships when the §31 build lands per the lean V1 decision).
- **Cryptographic agility** — Gateway crypto-suite spec: `Accept-Crypto-Suite` negotiation; opaque suite names; algorithms remain proprietary.
- **Capability dependency + supply chain** — `mcp-agent-runtime` profile + `capability` dictionary: DAG resolution, supply-chain attestation MUST, `state=blocked_by_dependency` cascade.
- **Multi-region geo-replication + conflict resolution** — `federation` rule pack: replication topology metadata, 6 conflict-resolution rules, 18-month key archival floor.
- **Vocabulary completeness** — the 12 canonical dictionaries (taxonomy, signals, reason-codes, pricing, credentials, capability, inference, ontology, telemetry, tenant, tool, graph) carry every term referenced by the rule packs and sector packs.

7.5 · Continuity & Discoverability

Two new normative profiles ship in v1.0 alongside the sectoral inventory above. Both extend Core: **Continuity** preserves *alignment* across the chain, **Discoverability** makes the chain *operational*.

7.5.1 Continuity Module (under Runtime Decision Rail)

Continuity is a **module inside the Runtime Decision Rail**, not a top-level profile. Where Core records *who acted, on whose authority, in what state, with what evidence*, the Continuity Module records whether the action **remained continuous** from intent to execution. It introduces three new dimensions on top of the eight Core records: **intent** (declared goal + constraints + declared_by), **interpretation** (interpreted goal + confidence + material_drift_detected), and **incentive / pressure context** (optimisation goal, commercial / affiliate / commission flags, urgency / coercion / social-engineering signals). Six normative objects: `KYEContinuityContext`, `KYEIntentTrace`, `KYEContinuityDecision`, `KYEAgencyDriftEvent`, `KYEContinuityEvidencePack`. Decision values: `continuity_preserved` · `continuity_degraded` · `continuity_broken` · `require_human_review` · `require_reconfirmation` · `deny` · `quarantine`. Ten drift classes (`intent_drift`, `authority_drift`, `scope_drift`, `state_drift`, `capability_drift`, `execution_drift`, `incentive_drift`, `oversight_drift`, `evidence_drift`, `delegation_drift` — collectively `agency_drift`) each emit a signed `KYEAgencyDriftEvent` on KYE Signal Bus™ and feed the Resilience Loop™. Spec: `kye-continuity-v1.md`.

7.5.2 Directory & Assurance Rail — KYE Directory™ + KYE Evidence Finder™

Identity, authority, scope, state, decision, audit, evidence are what KYE™ *records*. The **Directory & Assurance Rail** turns the recorded graph into a **policy-filtered, audited, masked** discovery surface, exposed via four canonical surfaces: **KYE Directory™** (the lookup surface), **KYE Authority Search™** (Authority Graph™ traversal), **KYE Evidence Finder™** (search across Evidence Packs™ and Decision Maps™), and **KYE Investigation Console™** (the auditor/regulator workbench). Six discovery types (entity, authority, capability, evidence, risk, ecosystem); six discovery modes (`private_tenant`, `workspace`, `cross_workspace`, `federated`, `public_registry`, `certification_registry`); seven normative objects (`KYEDirectoryEntry`, `KYEDiscoveryQuery`, `KYEDiscoveryResult`, `KYEDiscoveryPolicy`, `KYEAuthorityPathDiscovery`, `KYEDiscoveryAuditEvent`, `KYEEvidenceFinderQuery`). Every query is purpose-bound (`security_review`, `audit`, `incident_response`, `compliance`, `procurement_review`, `operations`, `investigation`) and audit-required by default. The rail is explicit about what it never returns: raw credentials, private keys, secret_refs, personal data outside the requesting tenant, and proprietary mechanism content.

7.5.3 Positioning

The refined v1.0 thesis: **KYE™ makes delegated agency observable, governable, revocable, replayable — via a Resilience Loop™ that closes on drift, and a Directory & Assurance Rail that makes the closed loop discoverable.**

7.6 · Semantic Registry — semantic authority

A login session, an OAuth scope, a payment mandate, a legal delegation, a healthcare consent, an API permission and an AI-agent capability are not the same thing. Treating them as if they were is a category of security failure that does not surface as a permission bug; it surfaces as an authority bug. The **Semantic Registry** — a module under *Protocol → Foundations*, supporting every engine; not a standalone product — is the v1.0 layer that prevents it.

7.6.1 Where the layer sits

Schemas define structure. Dictionaries define controlled terms. Taxonomies define hierarchy. Ontology defines *meaning* and relationships. Live graph instances show concrete relationships at runtime. Policies enforce decisions. KYE™ needs all of them. **Schemas make data valid. Ontologies make data meaningful.** The **Semantic Registry** (a module under Protocol → Foundations — not a standalone product) sits between dictionaries / taxonomies and the live authority graph; it does not replace any of them.

7.6.2 Ontology domains

Every KYE™ term declares exactly one domain. The domain enumeration, the normative-object set, and the predicate dictionary are proprietary and are not disclosed in this repository. New predicates require an RFC against the private contract. Spec: `kye-ontology-v1.md` (private).

7.6.3 Mapping types between external systems and KYE™

External-system terms map into KYE™ with a declared `mapping_type`. The runtime enforces semantic non-equivalence: presenting an OAuth scope alone, without the companion authority + scope + state + policy-decision records, is denied. Asserting equivalence against a non-equivalent mapping is itself a policy violation. The specific mapping-type alphabet, the per-type enforcement rule, and the false-equivalence-prevention construction are proprietary and are not disclosed in this repository.

7.6.4 Semantic assertions in the audit chain

Every runtime decision emits a signed `KYESemanticAssertion` listing exactly which terms, mappings and predicates were consulted. The assertion hash-chains into the KYE™ audit chain and is included in the Continuity Evidence Pack™. A regulator with the published JWKS can re-derive what the decision *meant* — not just what it returned. The reason-code dictionary, the canonical-form rule used for the signed assertion, and the chain-binding construction are proprietary and are not disclosed in this repository.

7.6.5 Serializations

KYE™ is JSON-native at runtime, JSON-LD-ready for semantic interoperability, and graph-aware for authority discovery, continuity and evidence. JSON Schema is required at the API surface; a JSON-LD context is published at `https://kyeprotocol.com/schemas/jsonld-context.json` and recommended; RDF / OWL export is supported as an optional serialization for research, public-sector, regulator and knowledge-graph integrations.

7.6.6 Positioning

KYE™ makes authority semantically portable. The ontology contracts are open. The managed semantic engine is paid.

7.7 · Operating Model — readiness to runtime

Most AI-governance programs stop at policies, committees, training or review documents. The narrative form does not produce runtime-policy configuration; gates, commit boundaries, review paths and evidence requirements remain in slide decks. The runtime gateway therefore enforces nothing specific to the use case, and the evidence record produced at runtime cannot be replayed against the declared intent. **KYE Operating Model Profile™** is the v1.0 layer that closes this gap.

The pre-runtime artefact is the **Operating Model™**: *who acts, on whose authority, for what purpose, under which rules, with what evidence, revoked by whom*. It decomposes into two named subsets — the **Authority Model™** (delegation graph, grant types, gate criteria) and the **Behaviour Model™** (allowed actions, obligations, stop-conditions, escalation paths). Both are signed before promotion. The **Control Compiler™** is the deterministic transform that turns the signed Operating Model into runnable rule bundles, evidence triggers, webhook contracts and Purpose Permission™ classes. If a customer cannot show a signed, compiled, deployed Operating Model™ for a given agent / partner / workflow, the Decision Engine MUST refuse to emit an `allow`. There is no ambient runtime in KYE™.

7.7.1 Bounded journey

Every implementation enforces a canonical readiness-to-runtime journey and emits a signed event per stage onto KYE Signal Bus™. The stage enumeration, ordering, per-stage transition guards, and gates between stages are proprietary and are not disclosed in this repository. Spec reference: `kye-operating-model-v1.md` (private).

7.7.2 Ten normative objects

Profile descriptor + nine living governance records: `KYEUseCaseIntake`, `KYEReadinessAssessment`, `KYEEntityAuthorityRecord`, `KYEAuthorityGate`, `KYECommitBoundary`, `KYEReviewPath`, `KYETrainingRecord`, `KYEAdoptionEvidencePack`, `KYEGovernedCatalogEntry`. Each record has a public JSON Schema mirrored under `public/examples/`; the validator runs every example through ajv on every CI build.

7.7.3 Authority Gates™ + Commit Boundary™

Authority Gates™ enforce a closed set of high-risk action classes; the Commit Boundary™ separates a recommendation from a committed action across each gate class. The specific gate-type enumeration, the decision-policy alphabet, and the per-gate commit-boundary contract are proprietary and are not disclosed in this repository. Conformant runtimes consume the canonical list via the private operating-model contract.

7.7.4 Adoption evidence pack

The signed `KYEAdoptionEvidencePack` binds an entity's runtime governance state to a replayable evidence artefact. The pack's component composition, the canonical encoding rule, and the chain-binding construction are proprietary and are not disclosed in this repository. A third party with the published JWKS can re-derive decisions through the public verifier surface; the verifier consumes the canonical recipe via the private contract.

7.7.5 Reason codes

The operating-model reason-code dictionary is canonical and bounded; representative entries are mirrored under `public/vocabulary/reason-codes.md`. The full enumeration, the decision-tree binding for each code, and the routing rules between codes are proprietary and are not disclosed in this repository.

7.7.6 Positioning

Assess. Gate. Decide. Prove. Revoke. Replay. KYE™ turns AI-governance operating models into runtime authority decisions and replayable evidence — from use-case intake to runtime execution. The operating-model contracts are open. The managed readiness, gate, commit-boundary and portal workflows are paid.

7.8 · Assurance Card — system cards become executable

Defence-grade AI assurance work (Alan Turing Institute / Accenture / UK MoD) makes the case directly: model cards are insufficient where models are integrated with platforms, where assumptions change and where multiple models interact. Agentic systems need more than model cards — they need authority cards, runtime decisions and evidence trails. **KYE Assurance Card Profile™** is the v1.0 lifecycle assurance layer that turns KYE™ runtime evidence into a living assurance record per delegated entity.

7.8.1 Bounded lifecycle

Every implementation enforces a canonical lifecycle and emits a signed event per stage onto KYE Signal Bus™. The stage enumeration, ordering and transition guards are proprietary and are not disclosed in this repository. Spec reference: `kye-assurance-card-v1.md` (private).

7.8.2 Six normative objects

Profile descriptor + five living governance records: `KYEAssuranceCard`, `KYEHumanInvolvementPlan`, `KYEProvenanceEvidence`, `KYEAssuranceReviewCycle`, `KYEDecommissioningPlan`. Each record has a public JSON Schema mirrored under `public/examples/`.

7.8.3 Human Involvement Plan™ as a runtime policy gate

KYE Human Involvement Plan™ defines points where authorised, suitably qualified humans MUST exercise judgement to *influence*, *direct*, *limit*, *approve*, *override* or *review* an AI-enabled system. The plan is enforced as a runtime policy gate: a commit-level action that would bypass an involvement point is denied with reason code `bypass_of_human_involvement_attempted` and a signed `kye.human_involvement.bypass_denied` event is emitted onto Signal Bus™. This converts human-involvement requirements from documentation into operational architecture.

7.8.4 Provenance and supply-chain evidence

`KYEProvenanceEvidence` records model / dataset / tool / hardware / supplier / licence references with a verification status (`verified` / `partially_verified` / `unverified` / `failed`). Verification methods: `self_attested`, `third_party_attestation`, `signed_evidence`, `registry_check`. Verification failure emits `kye.provenance.verification_failed`; supplier and licence changes emit dedicated events that promote to a review trigger.

7.8.5 Review cycle

`KYEAssuranceReviewCycle` watches the authority graph + provenance + risk-state for governance-relevant changes and instantiates reviews without operator intervention. The specific trigger enumeration, the trigger-to-review routing, and the review-cycle state machine are proprietary and are not disclosed in this repository. Reviewer identity, decision, reason code and evidence are recorded per review and chain-bound into the audit ledger.

7.8.6 Decommissioning

`KYEDecommissioningPlan` orchestrates retention windows and a bounded off-boarding sequence under a declared cascade-revocation scope. The step sequence, the cascade-revocation scope alphabet, and the post-execution verification construction are proprietary and are not disclosed in this repository.

7.8.7 Positioning

System cards say what should be assured. KYE™ proves what happened. KYE™ makes assurance executable. The assurance-card contracts are open. The managed runtime assurance engine, sector packs and use-case library are paid.

7.9 · Formal Rules — rights, obligations, runtime

Authority is incomplete unless permissions, obligations, prohibitions, powers, exceptions and governance rules are explicit. KYE Formal Rules Profile™ is the v1.0 layer that models them as first-class, machine-readable authority objects so KYE Gateway™ can enforce them at runtime, the Obligation Ledger™ can track them across their lifecycle, the Rule Prover™ can test their consistency before deployment, and the Control Compiler™ can compile them into runtime policy bindings.

7.9.1 Six rule families + meta-governance

Every `KYEFormalRule` declares exactly one of: `permission` (`P` — "may"), `obligation` (`O` — "must"), `prohibition` (`F` — "must not"), `power` (`Pow` — "has authority to"), `immunity` (`Imm` — "cannot be altered by"), `exception` (`Ex` — "displaced by, in conditions"). A seventh family `meta_governance` is recorded under `KYEGovernanceRule`. Spec: `kye-formal-rules-v1.md`.

7.9.2 Normative objects

The formal-rules profile defines a bounded set of authority-object schemas (formal rule, permission, obligation, prohibition, power, exception, governance rule, conflict, proof, obligation state). The complete schema set, the obligation lifecycle alphabet, and the chain-binding construction are proprietary and are not disclosed in this repository.

7.9.3 Prohibition is structurally distinct from deny

Generic policy engines (XACML, OPA, Cerbos, ZenAuth) compile a flat set of allow/deny rules. KYE Formal Rules Profile™ distinguishes `deny` (action not allowed *this time*) from `prohibited` (this *class* of action is forbidden under the bound authority model). The reason codes are different (`permission_missing` vs `prohibited_action_requested`); the runtime artefacts are different (gate decision vs prohibition trigger event); the audit-chain entries are different.

7.9.4 Pre-runtime consistency — KYE Rule Prover™

Before a rule set is bound to runtime, the Rule Prover™ emits a signed `KYERuleProof` attesting a bounded set of soundness properties (conflict-freeness, satisfiability, commit-boundary coverage, evidence coverage, revocation reachability, acyclicity, boundedness, override governance). The specific property enumeration, the proof construction, and the conflict-resolution strategy are proprietary and are not disclosed in this repository.

7.9.5 Multi-target compilation — KYE Control Compiler™

A single formal rule does not compile into a single allow/deny policy. The Control Compiler™ emits multiple coordinated runtime artefacts; the compilation-target set, the per-target binding rule, and the compilation seal construction are proprietary and are not disclosed in this repository. The runtime gateway enforces the compiled artefacts; it never re-interprets the formal rule at decision time.

7.9.6 Positioning

A rule should not only be written. It should be enforceable, discoverable, revocable, provable and replayable. KYE™ provides that layer. Formal rules define the normative structure. KYE™ operationalises it at runtime. The formal-rule contracts are open. The managed reasoning engine, prover, compiler and obligation ledger are paid.

7.10 · Action Admissibility™ — before authority, before commit

Attribution is late. If an AI agent forms a harmful, unlawful, out-of-scope, coercive, misleading or structurally invalid action, the organisation may already carry risk even if the action is later denied. **KYE Action Admissibility Profile™** is the v1.0 upstream layer that prevents inadmissible actions from forming in the first place.

7.10.1 The pipeline shape

The Admission Gate™ sits UPSTREAM of the authority pipeline. Downstream surfaces are blocked from receiving a proposed action when admission decides against it. The specific pipeline stages, the per-stage signal contract, and the routing rules into downstream surfaces are proprietary and are not disclosed in this repository. Spec reference: `kye-action-admissibility-v1.md` (private).

7.10.2 Normative objects

The profile defines a bounded set of admissibility objects. The complete schema set, the per-object signing recipe, and the chain-binding construction are proprietary and are not disclosed in this repository. The evidence record lists every input and signal consulted, hash-chained into the KYE™ audit chain so a third party can re-derive *why* a proposal was admitted, rejected or quarantined.

7.10.3 Decision alphabet + inadmissibility taxonomy

The Admission Gate emits a decision from a closed alphabet and classifies inadmissibility against a closed taxonomy. The specific decision values, the inadmissibility-class enumeration, and the per-class detection rule are proprietary and are not disclosed in this repository.

7.10.4 Distinct from validation, classification and policy

Generic input-validation toolchains validate JSON shape but do not bind a signed admission decision into the audit chain. Prompt-injection / safety classifiers classify text but do not emit a runtime gate verdict tied to an authority pipeline. Generic policy engines (XACML, OPA, Cerbos) compile allow / deny over a single resource but do not perform the upstream admission gate before the authority decision step. Action Admissibility™ is structurally upstream and structurally signed.

7.10.5 Positioning

KYE™ does not only attribute delegated actions after they exist. It checks whether proposed actions are admissible before they enter the authority pipeline, then decides, proves, revokes and replays what happens next. The admission contracts are open. The managed Admissibility Engine™ is paid.

7.11 · Developer surface — MCP, SDK, Conformance

Anything an integrator touches to consume the KYE™ kernel — model agent, host app, partner backend, third-party runtime, regulator harness — MUST go through one of four canonical components. Bespoke clients and undocumented HTTP surfaces are constitutional violations.

- **KYE™ MCP Server** — the Model Context Protocol server exposing KYE Gateway™ calls as MCP tools. Owners-run, customer-attached. Transports: `stdio` for in-process agents (Claude Code, Cursor, Cline, Continue) and `streamable-http` for hosted agents (OpenAI Responses API, Vertex Agent Builder, Bedrock AgentCore). The locked tool inventory is the only sanctioned way for a model-side agent to call the kernel.
- **KYE™ MCP Gateway** — the edge-deployed front door for KYE™ MCP Server traffic. Authority Gate™ fire, Purpose Permission™ admit, rate caps, audit fan-out to the AI Call Ledger™. The Gateway enforces; the Server defines wire.
- **KYE SDK™** — first-party client libraries in TypeScript, Python and Go wrapping the Gateway REST + MCP surfaces. Typed surfaces for `authority`, `purpose`, `decide`, `evidence`, `directory`; retry/backoff; SDK-side telemetry.
- **KYE Conformance Pack™** — the shippable bundle (locked fixtures + harness + runner + signed report) that a third-party runtime delivers to claim KYE™ compliance. The pack is an assurance artefact, not a runtime.

7.12 · KYE Counterparty Governance Rail™

(renamed from KYE Partner Rail™ — same product, broader scope: partners + suppliers + 3rd-party AI agents)

The public Partner surface is **two engines sharing one Partner Authority Kernel**. Both engines bind to the same kernel (entity, delegation, scope, decision, evidence); they read and write the same Authority Graph™ from opposite sides and never duplicate state.

- **KYE Partner Engine™** — operator-side. Runs the partner program: registration, certification, scope, deal registration, widget licensing, revenue share.
- **KYE Partner Governance Rail™** — customer-side. Governs partners acting on the customer's behalf — same kernel, customer-side controls.
- **KYE Partner App™** — the Android-first mobile workspace for certified partners (case load, scope grants, evidence sign-off, deal registration).
- **KYE Widgets™** — the embeddable widget catalogue partners place in their own product to carry KYE™ authority into their UI.

7.13 · KYE GovernedUI™ — the visible human-control surface

Constitution §36 (locked 2026-05-19). KYE Protocol™ is the governance brain; **KYE GovernedUI™** is the human-facing layer through which operators review, approve, reject, edit, escalate, evidence, and audit consequential AI-agent actions *before* those agents touch systems, data, users, money, code, documents, workflows, or regulated processes. Every approval lands in the WORM (Write-Once-Read-Many) audit chain; every verdict is replay-proof from public-key material alone.

- **Eleven canonical modules** declared in `internal` across two suites — seven OPS modules (operator-facing) and four GTM modules (ecosystem-facing). The `governedui-manifest-alive` blocking reconciler enforces the module ↔ widget_path ↔ envelope_schema bijection on every push.
- **OPS suite**: Action Approval (pre-action human review), Entity Passport (at-a-glance identity), Authority Scope (can / cannot / needs-approval), Critical Point Review (two-person + two-person-with-legal modes), Evidence Timeline (replay-proof chain), Approval Queue (multi-reviewer routing), Authority Drift Detector (ten drift dimensions emitted by the `kye-drift-detector` Worker).
- **GTM suite**: Consultants surface, Trainers surface, Auditors (Replay-Proof verification console), Partners (registry + deal attribution).
- **Approval modes (LOCKED, §36 §6)**: `none`, `single_approver`, `two_person` (SR 11-7 four-eyes), `two_person_with_legal` (regulated data), `delegated`, `auto` (forbidden at `risk_level: high+`).
- **Meta-governance gate (§36 §9, LOCKED)**: any `action_proposal` whose `action_type` is `delegate_authority` or `modify_own_authority` where `actor_id == grantee_id` is rejected at the PDP gate with `reason_code: meta_governance_violation`. No human approval can override this; the operator must invoke the explicit, separately-audited break-glass flow.
- **Multi-channel**: dashboard + email shipped (signed, single-use email-action tokens with kid rotation; the token construction is proprietary and is not disclosed publicly); Slack, Teams, mobile in v1.1.
- **Critical-action catalogue (twenty classes)**: `send_email`, `send_message`, `delete_file`, `export_data`, `access_sensitive_data`, `update_crm`, `issue_refund`, `create_invoice`, `submit_form`, `deploy_code`, `switch_traffic`, `run_sql`, `modify_policy`, `approve_workflow`, `share_document`, `call_paid_api`, `trigger_payment`, `change_customer_record`, `delegate_authority`, `modify_own_authority`.

Schemas: `kye.governedui.action_proposal.v1`, `kye.governedui.approval.v1`, `kye.governedui.entity_passport.v1`, `kye.governedui.authority_scope.v1`, `kye.governedui.critical_point_review.v1`, `kye.governedui.evidence_timeline.v1`, plus `kye.agency_drift.event.v1` for the Authority Drift Detector. Pricing is published openly across five tiers from Pilot (£95–150k fixed fee) to National / Consortium (£5–15m+ / year) at /governed-ui/pricing.html.

8 · Security & threat model

The reference implementation defends against:

- **Replay attacks** — webhook signatures include a Unix timestamp and are rejected outside a 5-minute window. Idempotency keys cache responses for 24 hours.
- **Tampered audit events** — audit events are bound to their predecessors. The exact binding is proprietary and is not disclosed here. The verify endpoint detects breaks end-to-end.
- **Stale revocations** — downstream authorities are no longer usable. The propagation mechanism is proprietary and is not disclosed here.
- **Forged credentials** — audit events carry cryptographic signatures verifiable by a public key set. The signature suite and key publication mechanism are proprietary and are not disclosed here.
- **Approval timeout abuse** — pending approvals carry a `required_by` deadline; the runtime expires them and emits a deny signal.
- **Compromised actor** — compromise reports cause downstream-derived authorities to become unusable; re-activation is gated by a signed, time-boxed recovery flow. Cascade and recovery algorithms are proprietary.
- **Lost or rotated keys** — key rotation requires a valid `X-Break-Glass-Grant-Id`; appends a `key.rotated` entry to the audit chain; previous keys remain verifiable for a configurable retention window. The signing-suite construction is part of the normative specification and is not disclosed publicly.
- **Forensic back-dating** — point-in-time audit replay reconstructs entity / authority / state at any sequence or timestamp; investigators verify behaviour rather than reading current state.

Out of scope for the reference: HSM-backed key custody, multi-tenant gateway hardening, transport-level mTLS configuration. These are deployment concerns; production Gateways must address them.

9 · Governance

Vocabulary, schemas, OpenAPI specs and KYE Reference Gateway™ behaviours are published openly under Apache License 2.0 in github.com/KYE-Protocol. Specific mechanism designs (decision algorithms, hash-chain construction, cascade ordering, attenuation propagation) are intentional placeholders pre-filing in `internal` and are not disclosed publicly to preserve IP novelty. Conformant implementations may license the mechanism designs royalty-free for any conformant use; full terms are forthcoming with the Linux Foundation / OpenWallet Foundation track.

Trademark policy: KYE™, KYE Protocol™, and Know Your Entity™ refer to the protocol as published. Forks, modifications and unrelated projects must not use the marks to identify themselves.

10 · Roadmap

- **v1.1** — Sector overlays: `healthcare-clinical`, `healthcare-payer`, `healthcare-research` sector packs, 42 CFR Part 2. Extended signal-bus durability options. `conformance-report.json` + `conformance-fixture.json` schemas.
- **v1.2** — Conformance certification program; independent test-vector runners; vendor self-attestation portal.
- **v2.0** — Federation v2 with multi-hop attenuation and cross-jurisdiction proofs. Proprietary algorithms moved to royalty-free open standard.

References

Industry & protocol references

1. Visa. Trusted Agent Protocol. `developer.visa.com/use-cases/trusted-agent-protocol`
2. Persona. Know Your Agent (KYA). `withpersona.com/identity-glossary/know-your-agent-kya`
3. Sumsub. Agent Verification. `sumsub.com/verification`
4. Trulioo / PayOS. Digital Agent Passport. `trulioo.com/products/digital-agent-passports`
5. Anthropic. Model Context Protocol. `modelcontextprotocol.io`
6. OpenID Foundation. AuthZEN v1.0, SSF, CAEP. `openid.net/specs/openid-authzen-v1_0.html`
7. SPIFFE Project. SPIFFE Identity Specification. `spiffe.io`
8. IETF SCITT WG. Supply Chain Integrity, Transparency, and Trust. `datatracker.ietf.org/wg/scitt`
9. NIST. SP 800-207 Zero Trust Architecture. `csrc.nist.gov/publications/detail/sp/800-207/final`

Regulatory frameworks cited in this whitepaper

10. European Union. Regulation (EU) 2024/1689 — Artificial Intelligence Act. `eur-lex.europa.eu/eli/reg/2024/1689/oj`
11. European Union. Regulation (EU) 2016/679 — GDPR (plus UK GDPR equivalent). `eur-lex.europa.eu/eli/reg/2016/679/oj`
12. European Union. Regulation (EU) 2022/2554 — DORA (Digital Operational Resilience Act). `eur-lex.europa.eu/eli/reg/2022/2554/oj`
13. European Union. Directive (EU) 2022/2555 — NIS2. `eur-lex.europa.eu/eli/dir/2022/2555/oj`
14. European Union. Directive (EU) 2015/2366 — PSD2; PSD3 proposal: COM(2023) 367 — PSD3. `eur-lex.europa.eu`
15. ISO/IEC. ISO/IEC 27001:2022 — Information security management. `iso.org/standard/82875.html`

16. ISO/IEC. ISO/IEC 42001:2023 — AI management systems. `iso.org/standard/81230.html`
17. PCI Security Standards Council. PCI DSS v4.0. `pcisecuritystandards.org`
18. U.S. Department of Health & Human Services. HIPAA Security Rule — Technical Safeguards. `hhs.gov/hipaa/for-professionals/security`
19. U.S. Code of Federal Regulations. 42 CFR Part 2 — Substance-use-disorder confidentiality. `ecfr.gov/current/title-42/part-2`
20. AICPA. SOC 2 — Trust Services Criteria (2022). `aicpa-cima.com/topic/audit-assurance/audit-and-assurance-greater-than-soc-2`
21. IETF. RFC 8785 — JSON Canonicalization Scheme (JCS). `datatracker.ietf.org/doc/html/rfc8785`
22. IETF. RFC 8032 — Edwards-Curve Digital Signature Algorithm (Ed25519). `datatracker.ietf.org/doc/html/rfc8032`
23. IETF. RFC 7807 — Problem Details for HTTP APIs. `datatracker.ietf.org/doc/html/rfc7807`
24. NIST. OSCAL — Open Security Controls Assessment Language. `pages.nist.gov/OSCAL`

Appendix — verify a sample evidence pack live

The whitepaper's claims about offline-verifiable evidence packs are not abstract. Below is the same KYE Evidence Pack™ Viewer that ships at `widgets.html#evidence`: it verifies a signature against the publisher's JWKS, replays the bound decision, walks the audit chain, and projects to SOC 2 / ISO 27001 / EU AI Act / PSD3 / DORA. No signup, no install — the same flow your auditor will run on production packs.

> Cite as: KYE Protocol™ Project. *KYE Protocol™: an open trust layer for the agentic economy.* Whitepaper v1.0, April 2026. `https://kyeprotocol.com/whitepaper.html`

Ready to see your AI agents flagged?

Start in shadow mode. We’ll deliver your first Evidence Pack™ in 4–8 weeks.

Apply for pilot →

Try the sandbox →