LiyaEngine

ENTERPRISE TECHNICAL OVERVIEW

Architecture and integration guide for engineering and security teams

How LiyaEngine fits into your architecture: model providers, runtime, knowledge, governance, deployment, security and pricing, with what is available today stated plainly.

Version 1.1

October 2026

Audience

Engineering, security, procurement

Contact

liyaengine.ai/contact

EXECUTIVE SUMMARY

LiyaEngine in one page

What it is

A managed platform for building, grounding, governing and observing production AI capabilities behind one stable API. Applications call a named, schema-bound intent, and models, knowledge, tools and policy sit behind it.

The problems it solves

Provider lock-in and coupling, the engineering cost of orchestration, retrieval and policy, releases with no evidence behind them, and no shared view of cost, quality and behaviour.

How organizations integrate it

From an existing backend, through one API call, keeping their own authentication, databases and model providers (BYOK). Managed cloud today, with region-pinned inference, and private deployment scoped per engagement.

What they gain

Faster delivery of AI features, independence from any one model provider, evaluation evidence tied to releases, and a trace and cost record for every request.

Know this before you evaluate

We state limits plainly, so your review starts from an accurate picture.

Release gates cover promotion

Evaluation gates apply when promoting between projects, not to direct edits inside one, and have no cost or latency thresholds (Section 7.3).

Retrieval has no per-user permissions

Use separate domains and collections for audiences with different access (Section 5.6).

Identity and certification

SSO and SCIM are in development. SOC 2 is a goal, and no certification is claimed (Sections 9 and 11).

Some retention is a default, not a guarantee

Audit logs are removed on schedule. The other categories are built and enabled per deployment, and report what they would remove until then (Section 9.3).

Where to read next

IF YOU ARE	START WITH
Engineering or architecture	Sections 2 to 7: platform architecture, API, model routing, knowledge, agents, governance.
Security or compliance	Section 9, then Section 11 for what is and is not available.
Procurement or finance	Sections 10, 11 and 13: pricing, availability and next steps.

CONTENTS

What is in this document

This document is self-contained. Each section describes behaviour you need to evaluate LiyaEngine without switching between web pages. Reference documentation is at liyaengine.ai/docs.

01 Why organizations adopt LiyaEngine

02 Platform architecture

03 API architecture

04 Model providers and routing

05 Knowledge and retrieval

06 Agents and workflows

07 Governance, evaluation and observability

08 Deployment and hosting

09 Security and data handling

10 Pricing and plans

11 Availability and roadmap

12 Enterprise technical FAQ

13 Contact and next steps

How to read the status labels

LABEL	MEANING
AVAILABLE	Generally available and in production use.
LIMITED AVAILABILITY	Available with stated constraints. The constraints are described next to it.
ENTERPRISE CUSTOM	Delivered through an Enterprise engagement and scoped per customer. Not self-service.
IN DEVELOPMENT	Being built. Do not rely on it for a purchase decision without confirming timing with us.
PLANNED	A goal. No delivery commitment is implied.
NOT AVAILABLE	Not currently supported. Where a workaround exists, it is described.

SECTION 1

Why organizations adopt LiyaEngine

Shipping production AI means assembling provider integrations, orchestration, retrieval, policy enforcement, evaluation and tracing, and then operating all of it. LiyaEngine provides these as one managed platform behind a stable API, so your teams build product instead of AI infrastructure.

1.1 What you would otherwise build and run

CONCERN	TYPICALLY BUILT IN HOUSE	IN LIYAENGINE
Provider integration	Per-provider SDKs, retries, key handling, fallbacks	Model router with BYOK for seven provider types and ordered failover
Orchestration	Agent loops, workflow state, approvals, schedulers	Agent runtime and workflow engine with approvals, triggers and run history
Grounding	Ingestion pipelines, a vector store, retrieval tuning	Managed knowledge collections with hybrid retrieval and analytics
Policy	PII, injection and schema checks repeated in each service	Guardrail policies enforced in the request path
Evaluation	Scripts and spreadsheets, disconnected from releases	Datasets, suites, judges, baselines and release gates
Observability	Custom tracing and cost accounting	Execution traces, cost explorer and an audit log

1.2 Three ways organizations adopt it

MODEL	HOW YOU USE LIYAENGINE	WHAT YOU KEEP	STATUS
Build on LiyaEngine	Configure domains, intents, knowledge, agents and workflows, and integrate through the API without building the orchestration layer.	Your applications, user authentication and systems of record.	AVAILABLE
Integrate with existing systems	Keep your applications, databases, business services and model providers. LiyaEngine supplies AI execution, retrieval, governance and observability through API integration.	Provider accounts and agreements (through BYOK), data stores, application logic.	AVAILABLE
Customer-infrastructure solutions	For stricter requirements, solution design and a deployment arrangement that uses customer-managed infrastructure, subject to technical discovery.	Infrastructure and data location, as set by the agreement.	ENTERPRISE CUSTOM

> **Evaluation can be adopted on its own.** Evaluation Studio scores outputs produced anywhere, through a standalone API, so a team can adopt evaluation and release governance without moving its runtime to LiyaEngine (Section 7.4).

1.3 Where it fits alongside other tools

These are architectural comparisons of typical roles. They are not claims of feature parity, and many teams use these tools together with LiyaEngine.

TOOL	TYPICAL ROLE	WHERE LIYAENGINE FITS
LangChain	An application framework you run inside your own services.	A managed runtime called through an API, with no framework to host.
LangGraph	A library for stateful agent orchestration.	Managed agent and workflow execution with governance attached.
LangSmith, Langfuse	Tracing, evaluation and prompt-management tools.	Tracing and evaluation tied to the runtime that produced them, with policy enforcement and release gates.
AWS Bedrock, Azure OpenAI	Managed model access.	A provider-independent layer above them. Both are supported providers.
Custom AI infrastructure	An in-house platform.	Less platform engineering and operations to staff.

1.4 Who owns what

LIYAENGINE ASSUMES	YOU KEEP
Orchestration runtime, retrieval, policy enforcement, tracing, evaluation and the managed knowledge store.	Your application, and authenticating and authorising your end users.
Provider routing, failover and credential storage.	Your model provider relationships, quotas and, with BYOK, inference billing.
Platform operations for the managed cloud service.	Data classification, and the retention and consent settings you choose.

SECTION 2

Platform architecture

LiyaEngine is a multi-tenant platform. Applications integrate with a stable runtime contract, while the configuration behind it (models, knowledge, policies, workflows) is owned and changed by your teams.

2.1 Core concepts

CONCEPT	WHAT IT IS
Tenant	Your organisation's isolated workspace. Credentials, data, configuration and usage are scoped to it.
Domain	A business or product boundary that owns shared behaviour, linked knowledge and enabled tools.
Intent	A named, schema-bound AI action inside a domain: an input contract, an output contract, instructions, tools and guardrail policies. This is what applications call.
Agent	A configured entity with its own capabilities, tools and memory, for multi-step work.

CONCEPT	WHAT IT IS
Workflow	A chain of intents, agent runs, tool calls and human approvals, started by API call, signed webhook, schedule or event.
Knowledge collection	Documents organised for retrieval, linked to domains and scoped per collection.
Guardrail policy	Reusable input and output controls, attached to the capabilities that need them.
Evaluation	Datasets and suites that score an intent's real behaviour before and after changes.

2.2 System architecture

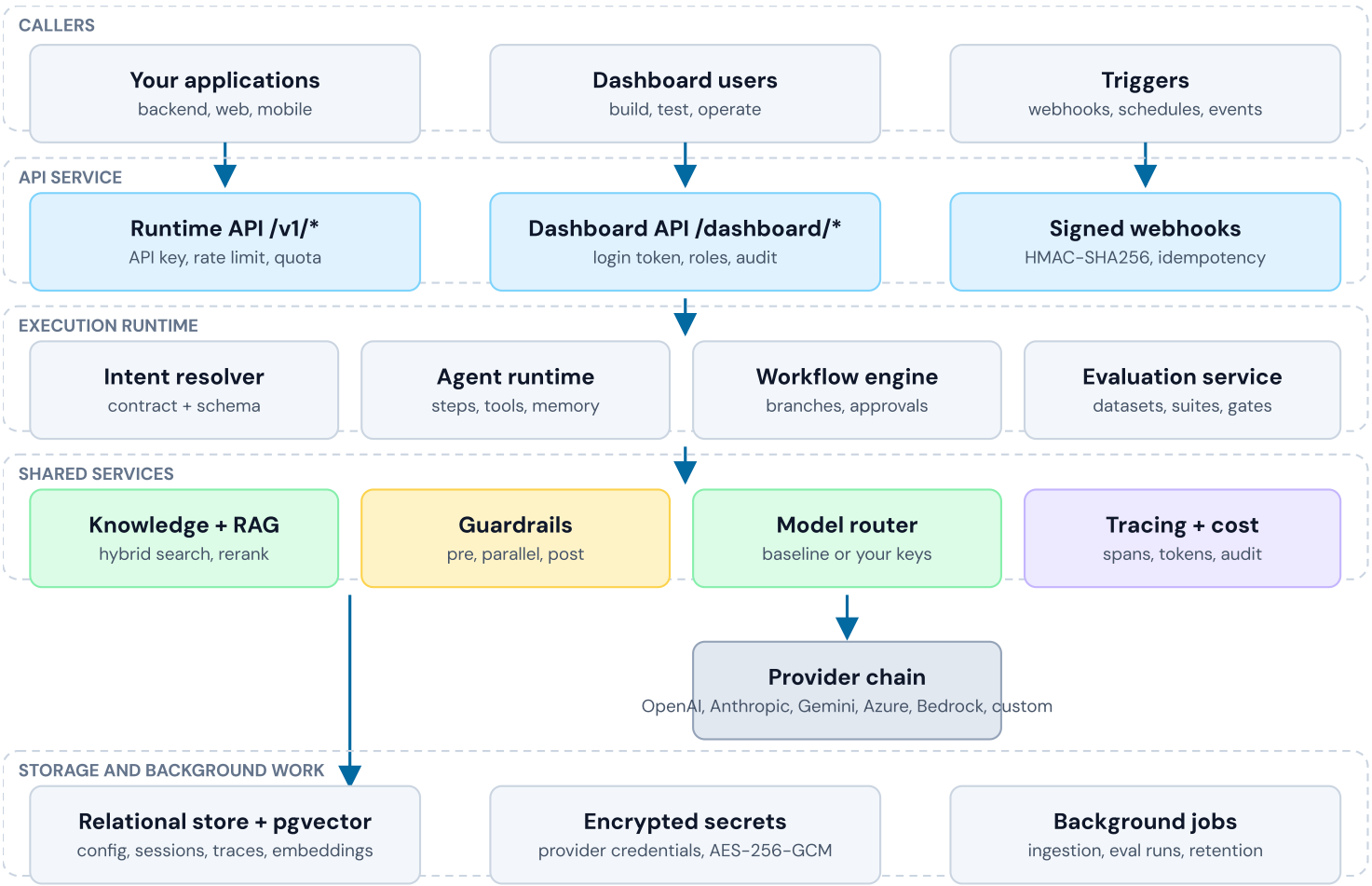

Figure 1. LiyaEngine components and where each runs. Callers enter through the API service, the execution runtime coordinates shared services, and state sits in the storage layer.

2.3 Component responsibilities

COMPONENT	RESPONSIBILITY	WHERE STATE LIVES
API service	Authenticates the caller, applies rate limits and quotas, and hands the request to the runtime. Runtime API calls and dashboard calls are separate surfaces.	Rate-limit counters, held in a shared cache so limits hold across instances.
Execution runtime	Resolves intents, runs agents and workflows, assembles context, and calls the model router and guardrails.	Run state, sessions and messages in the relational store.

COMPONENT	RESPONSIBILITY	WHERE STATE LIVES
Relational store	Tenant configuration, sessions and messages, traces, usage, audit logs and evaluation data. Embeddings are stored natively with pgvector.	Managed database, tenant-scoped.
Secrets	Provider credentials are encrypted with AES-256-GCM using per-tenant keys and are never returned by any API response.	Encrypted columns, decrypted only to call your provider.
Background jobs	Knowledge ingestion, connector syncs, evaluation runs, and the nightly retention sweep. A sweep recovers jobs stalled for more than 30 minutes.	Job records in the relational store.
Failure handling	Provider failover inside a request, SDK retries with backoff on 429 and 5xx, idempotency keys on workflow calls, cooperative cancel, and resume of a failed evaluation run without re-billing scored cases.	Per request and per run.

SECTION 3

API architecture

Application code calls a domain-owned intent. Retrieval, tools, model selection, policy checks and tracing are assembled behind it from configuration.

3.1 Request lifecycle

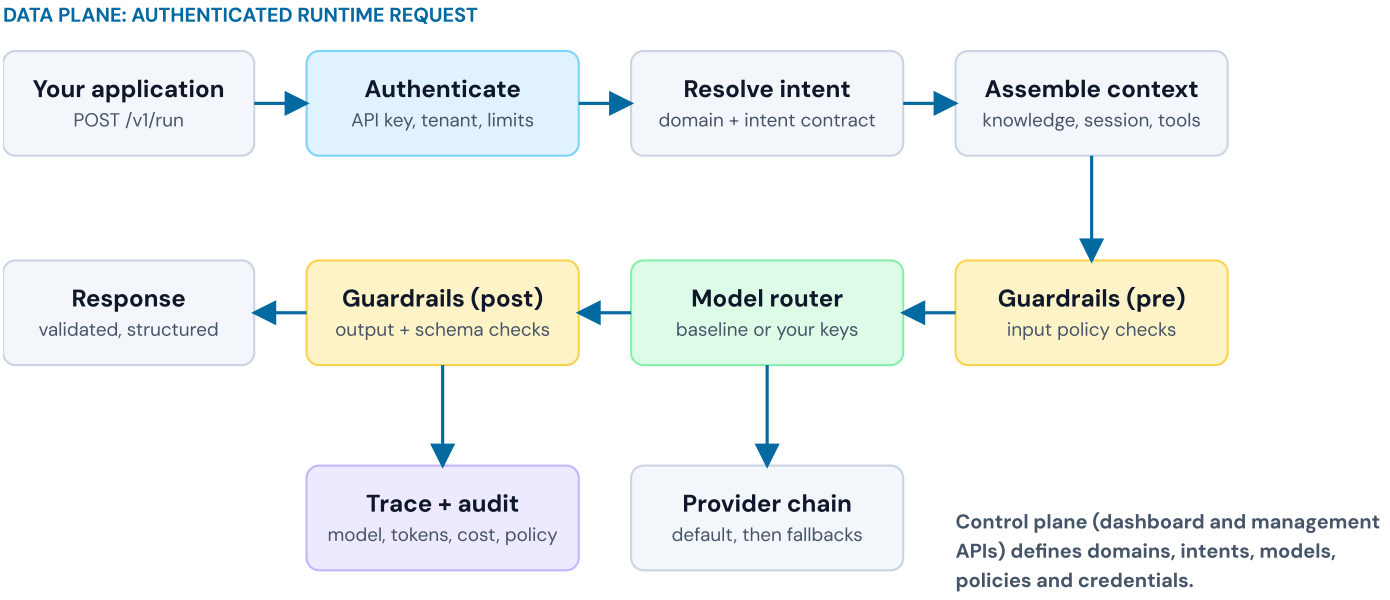

Figure 2. How a runtime request is processed. The control plane defines the configuration the data plane executes.

1. **Authenticate and resolve.** The tenant credential, domain, intent and request contract are validated, and the tenant's rate limit and quota are applied.
2. **Assemble context.** Shared behaviour, the intent schema, linked knowledge, session memory, tools and model settings are loaded.
3. **Execute and govern.** Input policy checks run, then the model router calls a provider (or an agent runs its steps), then output and schema checks run.
4. **Return and record.** A validated response is returned and the trace, usage and cost are recorded.

3.2 Calling the API

ITEM	DETAIL
Base URL	`https://api.liyaengine.ai`
Authentication	Tenant API key as a bearer token: `Authorization: Bearer liya_...`
Content type	`application/json`
Primary endpoint	`POST /v1/run`, with `domain`, `intent` and `input` in the body
Equivalent path form	`POST /v1/{domain}/{intent}`, with only `{ "input": ... }` in the body
Required input field	`input.user_id`, an identifier for your end user. Calls without it are refused with `400 INVALID_INPUT`
Conversation continuity	Optional top-level `session_id`, returned by the first call and passed on follow-ups
Response format	JSON envelope with a `success` flag, then `data` and `metadata` on success, or an `error` object with a stable `code` on failure
Rate-limit headers	`X-RateLimit-Limit` and `X-RateLimit-Remaining` on every response; `Retry-After` on `429`

```
POST https://api.liyaengine.ai/v1/run
Authorization: Bearer liya_...
Content-Type: application/json

{
  "domain": "support",
  "intent": "answer_question",
  "input": {
    "user_id": "usr_123",
    "message": "How do I reset my password?"
  }
}
```

```
{
  "success": true,
  "data": { "output": { ... }, "session_id":
"ses_abc", "message_id": "msg_xyz" },
  "metadata": {
    "intent": "answer_question", "domain": "support",
    "model_used": "claude-sonnet-5", "tokens_used":
1240,
    "cost_usd": 0.00062, "latency_ms": 340, "cached":
false
  }
}
```

3.3 Integration example: healthcare member support

An illustrative scenario. A health plan's member portal lets a member ask whether a service is covered. The portal calls one intent, and LiyaEngine does the rest.

1. LiyaEngine authenticates the portal's API key and resolves the configured intent.
2. It retrieves from the benefits collections linked to the `member-support` domain.
3. Input guardrails redact identifiers before the model call, and the model router calls the provider the organisation configured, such as its own Azure OpenAI deployment.
4. Output checks validate the response against the intent schema and check grounding, then the structured answer returns with execution metadata.

```
POST /v1/run
{
  "domain": "member-support",
  "intent": "answer_benefits_question",
  "input": {
    "user_id": "member_48213",
    "message": "Is a knee MRI covered under my plan?",
    "plan_id": "PLN-2026-GOLD"
  }
}

→ output: { "answer": "...", "citations": ["benefits-guide-2026#p14"], "escalate": false }
→ metadata: model, tokens, cost, latency, and a message_id that opens the full trace
```

Your application owns member authorisation. LiyaEngine authenticates your application, not your member, and has no knowledge of a member's entitlements. Your application authenticates the member, decides which domain and intent the call may reach, and supplies `user_id` so sessions, memory and usage attach to the right person. Retrieval is scoped by tenant, domain and collection (Section 5.6), so audiences with different data access belong in different domains.

3.4 Integration example: adding AI to an existing SaaS platform

An illustrative scenario. A project-management SaaS wants to add "summarise this account and draft a follow-up" without changing its backend, authentication or databases.

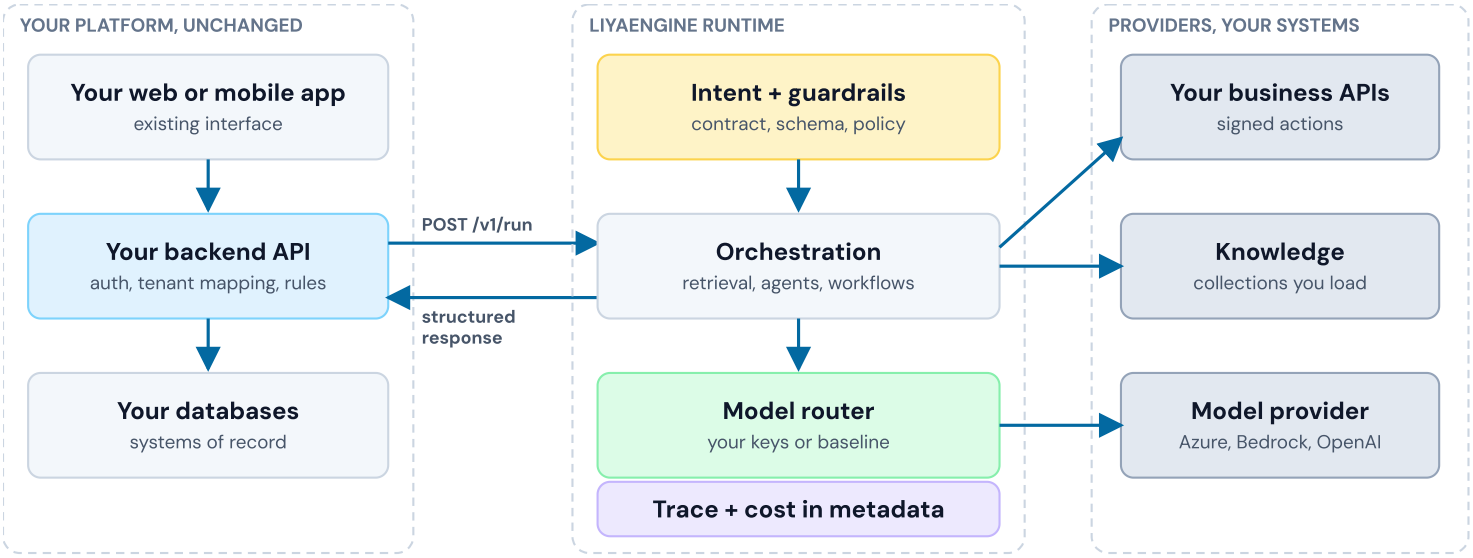

Figure 3. Introducing AI into an existing platform. Everything on the left stays as it is; the platform adds one backend call to the LiyaEngine runtime.

1. The user clicks the feature in the existing interface. The request goes to the platform's existing backend, which authenticates the user and applies its own authorisation, as it does for every other feature.
2. The backend loads the records that user may see and calls `POST /v1/run`, with its own user id as `input.user_id` and the context as input fields.
3. LiyaEngine resolves the configured intent, retrieves from the linked knowledge collections, runs guardrails, calls the model through the provider the company configured, and can call the company's own business APIs through signed

actions.

4. A structured, schema-validated response returns to the backend with execution metadata. The backend applies its own rules and returns it to the interface.

```
app.post("/api/accounts/:id/summary", requireUser, async (req, res) => {
  const account = await db.accounts.findForUser(req.params.id, req.user);  // your authorisation
  const r = await fetch("https://api.liyaengine.ai/v1/run", {
    method: "POST",
    headers: { Authorization: `Bearer ${process.env.LIYA_API_KEY}`, "Content-Type": "application/json" },
    body: JSON.stringify({ domain: "crm", intent: "summarise_account",
      input: { user_id: req.user.id, account: account.publicFields() } }),
  });
  res.json((await r.json()).data.output);  // typed fields: summary, next_steps, draft_email
});
```

WHAT YOU DO NOT CHANGE	WHAT YOU ADD
Your backend, authentication, databases and business services.	One intent definition, a knowledge collection, and one call from your backend.
Your model provider relationships, through BYOK.	Optional signed actions that let the model call your own APIs.

> **Keep the key on your backend.** The API key is a server-side secret, so call LiyaEngine from your server and never ship it to a browser; use an embed token for any browser widget. Give the service a runtime-scoped key, so that if it leaks it can run intents but cannot change configuration or read traces. If your platform serves many customers of its own, keep each customer's knowledge in its own collection and domain and route to it in your backend, because there are no per-document permissions inside a collection (Section 5.6).

3.5 Credentials, identity and access

CREDENTIAL	SCOPE	USED FOR
API key (`liya_...`)	Issued per environment. The full key is shown once, and rotating a key invalidates the old one immediately. Each key has a scope (below).	Your backend calling the runtime API.
Embed token	Restricted to a list of allowed intents. Public by design, so it cannot read traces, parse files or call secret-key endpoints.	A browser or mobile widget.
Dashboard login token	A signed-in user with a role (owner, admin, supervisor, member). Expires after 7 days by default.	The dashboard and the Dashboard API.

API key scopes

SCOPE	WHAT THE KEY CAN DO
`full`	Everything. The environment's own key, and every key created before scopes existed.
`runtime`	Execute only: run intents, agents and workflows, the standalone guardrails check, knowledge query and file parsing. It cannot change configuration, list resources or read traces.
`read`	GET and HEAD only, on any /v1 resource, including traces. It cannot run intents, so it cannot spend model tokens, and it cannot change anything.

A request outside a key's scope returns `403 API_KEY_SCOPE_DENIED`. Scopes are deny-by-default, so a route not listed for `runtime` is refused to a runtime key. Owners create, list and revoke scoped keys through the Dashboard API, and rotating the environment's key replaces only its full-access key.

> **What scopes do not do.** A scope limits what a key can do, not which domains or resources it can reach. A runtime key can run every intent your tenant has enabled. To keep an audience away from a domain, call LiyaEngine through your own backend, and use embed tokens, restricted to allowed intents, for anything that runs in a browser. Give each service its own key and rotate on any exposure.

3.6 Error codes

CODE	HTTP	MEANING
`MISSING_API_KEY`	401	No Authorization header was sent.
`INVALID_API_KEY`	401	The key is not recognised.
`INVALID_API_KEY_FORMAT`	401	The key is malformed.
`API_KEY_REVOKED`	401	The key was rotated or revoked.
`DOMAIN_NOT_ENABLED`	403	The domain is not enabled for the tenant.
`INTENT_NOT_PERMITTED`	403	The intent is disabled for the tenant.
`INVALID_INPUT`	400	A required input field, such as input.user_id, is missing or invalid.
`RATE_LIMIT_EXCEEDED`	429	The tenant's rate limit was exceeded. See Retry-After.

3.7 Rate limits and quotas

Runtime requests are rate-limited per tenant over a sliding 60-second window. The limit is a per-tenant setting rather than a fixed platform cap: new workspaces start at 60 requests per minute and higher throughput is configured for your account on request. Enterprise plans carry committed capacity. Chat and evaluation scoring apply an additional lower ceiling because each call generates significant model spend.

Every response carries `X-RateLimit-Limit` and `X-RateLimit-Remaining`, so clients should read the limit from the headers rather than hard-coding a number. An exceeded limit returns `429` with `Retry-After` and `X-RateLimit-Reset`. The official SDKs retry `429` and `5xx` responses automatically. Monthly request and token quotas are separate and depend on plan (Section 10).

3.8 Sessions and memory

Sessions are persistent conversation threads tied to a `user_id` and a domain. Omit `session_id` on the first call and the engine returns one; pass it on follow-up calls to continue the conversation. Sessions expire after inactivity under the retention settings in Section 9.3, and consent flags can disable memory per request.

3.9 API surfaces

Runtime capabilities, knowledge, evaluation and trace access are available with a tenant API key. Account management and reporting are available through the Dashboard API, which uses a dashboard login token.

SURFACE	ACCESS	CAPABILITY
Intents	API key	Invoke a structured capability. Responses conform to the intent's output schema.

SURFACE	ACCESS	CAPABILITY
Agents	API key	Create, configure, deploy and run standalone agents (`/v1/agents`), with tools, retries and execution state.
Workflows	API key	Create, deploy and run multi-step automations, started by API call or signed webhook (HMAC-SHA256). An Idempotency-Key header stops a repeated call from running twice.
Collections and documents	API key	Create and configure knowledge collections, upload documents, scope retrieval and attach collections to domains.
Domains and guardrail policies	API key	Manage custom domains and intent definitions, and create and update guardrail policies.
Evaluations	API key	Datasets, suites, runs and reviews that score your intents or your own outputs.
Guardrails check	API key	Check content against a policy standalone, with no intent involved.
Traces	API key	List, filter and search traces, fetch the full trace for any request, open agent and workflow run traces, and read trace statistics (`/v1/traces`). Keys with the full or read scope. Every `/v1/run` response also returns a trace summary in its `metadata` .
Trace collaboration and replay	Dashboard API	Comment on and bookmark traces, and replay a model call. These belong to a signed-in user, and a replay is a metered model call.
Usage, cost and audit	Dashboard API	Usage and quota, cost breakdown by model, domain and intent, sessions, and the tenant audit log with filters (`/dashboard/costs` , `/dashboard/audit`).
Account and credentials	Dashboard API	API keys per environment, provider credentials, model configuration, users and configuration promotion.

SECTION 4

Model providers and routing

Model routing runs inside the engine on every call. Your application never names a provider or model, so you can change either without changing code.

4.1 Two ways to run models

	LIYAENGINE BASELINE MODELS	BRING YOUR OWN KEYS (BYOK)
Provider account	Not needed	Your own accounts
Availability	Every plan, starting with Standard models on Free	Every plan, including Free
Model access	Tier rises with plan: Standard on Free, Advanced on Core, Premium on Pro and Teams, Frontier on Enterprise	Not limited by plan tier
Usage and cost	Counts against your monthly token budget; top-ups available	Bypasses the platform token pool. Inference cost goes to your provider account
Failover	Handled by LiyaEngine	Your ordered provider chain

4.2 Supported providers for BYOK

PROVIDER	CREDENTIAL YOU SUPPLY	NOTES
OpenAI	API key	Direct
Anthropic	API key	Direct
Google Gemini	API key	Direct
Azure OpenAI	API key, endpoint, deployment ID, API version	Pin inference to the Azure region you deploy to
AWS Bedrock	Region plus access keys, or a cross-account IAM role ARN	Role assumption avoids storing long-lived AWS secrets
OpenRouter	API key	One key for many hosted models, including Mistral and Llama families
Custom endpoint	API key and HTTPS base URL	Any OpenAI-compatible server, including self-hosted or fine-tuned models

4.3 How routing works

- **Strategy.** `hybrid` sends standard intents to a lighter model and premium intents to a stronger one; `all-mini` and `all-standard` use one model for everything. A specific model can be pinned to a specific intent.
- **Health and latency.** A live health check can be run on any provider on demand. The router also tracks recent latency and can route away from a primary that is slowing down before it fails.
- **Attribution.** Each response records the provider and model used, token count and cost, and whether the platform or your own key served the call.

4.4 Failover behaviour and its boundaries

Failover is deliberately scoped, and you should design your provider chain with its current behaviour in mind.

BEHAVIOUR	DETAIL
Which providers can serve a call	For a BYOK tenant, only the providers in your own ordered chain. LiyaEngine does not route a BYOK call to a provider you have not configured.
What triggers the next provider	Any error from the current provider, including an authentication failure. The chain does not currently separate transient failures (an outage, a timeout) from permanent ones (a rejected credential).
When every provider fails	If all attempts fail with an authentication error, the call returns a clear BYOK_KEY_INVALID error so a bad credential is surfaced, not hidden.
Residency and workload rules	There is no per-request residency or workload restriction on failover today. Put in the chain only providers approved for every workload that uses it, and use separate environments or domains where workloads differ.
Failure-type-aware failover policies	`IN DEVELOPMENT`

4.5 Integrating with your existing AI infrastructure

Adopting LiyaEngine does not require replacing your model providers. You can keep your credentials, agreements and infrastructure, and use LiyaEngine as the execution contract, routing, governance and observability layer above them.

SCENARIO	HOW IT WORKS
A regulated firm standardised on Azure OpenAI	Connect your Azure OpenAI resource as the default provider. Inference runs in your Azure resource and region, under your agreement. LiyaEngine adds the intent contract, guardrails, routing and tracing.
An AWS shop using Bedrock	Connect Bedrock with a cross-account IAM role. LiyaEngine assumes the role at request time, so no long-lived AWS secret is stored with us.
Self-hosted or fine-tuned models	Register an OpenAI-compatible HTTPS endpoint as a custom provider and route intents to it.
Existing provider contracts and quotas	Keep them. With BYOK, inference is billed by your provider, and you keep its rate limits and enterprise terms.

> **Boundary to be clear about.** Pointing inference at your own region does not move LiyaEngine's control plane, execution traces, knowledge storage or logs into that region. Section 8 describes exactly what stays where.

4.6 Cost controls

CONTROL	DEFAULT	BEHAVIOUR
Per request	$0.10	A request whose estimated cost exceeds the cap is rejected before it spends.
Per session	$1.00	New messages are rejected once the session reaches the cap.
Per day (tenant)	$100.00	A daily ceiling across the tenant.
Output tokens per request	4,000	Limit on generated tokens per call.

4.7 Credential handling

Provider keys are validated against the provider before they are saved, encrypted with AES-256-GCM using per-tenant keys, and never returned by any API response; only a masked preview is shown. Saving and removing a key are recorded in the audit log.

SECTION 5

Knowledge and retrieval

Retrieval-augmented generation keeps answers grounded in your own documents. LiyaEngine offers a managed knowledge base service (KBaaS) on every plan, and Enterprise plans can host the knowledge base in the tenant's own environment.

5.1 Where the knowledge base lives

OPTION	PLANS	WHERE YOUR DATA LIVES	CAPACITY
Managed knowledge base (KBaaS)	All plans	LiyaEngine-managed, tenant-scoped storage. Nothing to deploy or operate.	Set by plan (Section 5.2)
Tenant-hosted knowledge base	ENTERPRISE CUSTOM	Hosted in the tenant's own environment, so documents and embeddings stay under your control.	Negotiated, with the engagement scoped during technical discovery

Both options use the same collections, retrieval settings and domain links, so an application can move from managed to tenant-hosted without changing how it calls an intent.

5.2 Capacity

KBaaS storage and monthly knowledge queries are set by plan: 5MB and 1,000 queries on Free, 1GB and 50,000 on Core, 10GB and 500,000 on Pro and Teams, and negotiated on Enterprise. Usage is metered against the allowance and shown in the dashboard alongside request and token usage.

5.3 Ingestion and synchronisation

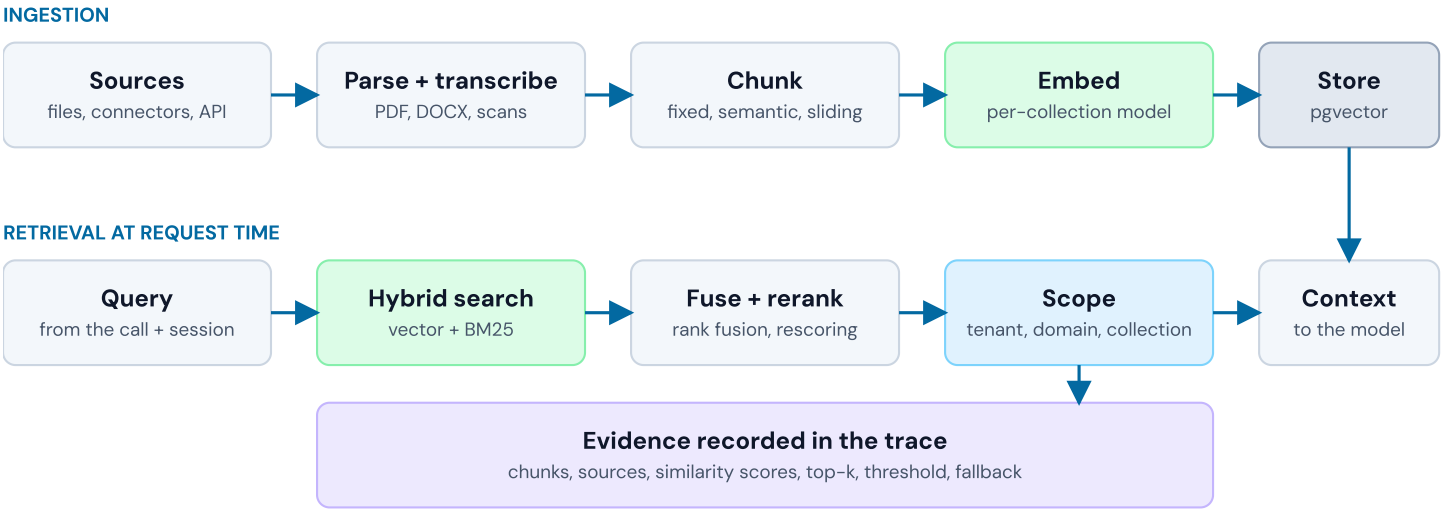

Figure 4. Knowledge ingestion and retrieval. Retrieval evidence is recorded in the trace for every call.

- **Documents:** PDF (text layer, labelled by page), DOCX, TXT, Markdown, HTML and JSON. Scanned pages and images are transcribed by a vision-capable model through your own model routing, when enabled.
- **One source, many domains:** a document attached to a collection is retrievable by every domain the collection is connected to. Update the source once and every domain sees the change.
- **Async jobs:** large files and URL crawls run as background ingestion jobs with status, cancel, and recovery of a job that stalls.

CONNECTOR	HOW IT SYNCHRONISES	STATUS
Google Drive, Confluence, Notion, SharePoint	A sync re-reads the folders, spaces or pages you selected and ingests their content. Removing a connector removes the content it ingested.	LIMITED AVAILABILITY

CONNECTOR	HOW IT SYNCHRONISES	STATUS
File upload, REST API ingestion	Content is ingested when you submit it. Re-pushing the same source id re-syncs it instead of duplicating it.	AVAILABLE

Synchronisation limits. Connector sync is a re-read of the selected sources, not a change feed. Deletions at the source are not propagated automatically, and source-system permissions and permission changes are not mirrored into LiyaEngine. If a source document must disappear or change audience, remove or update it in LiyaEngine as well.

5.4 Retrieval pipeline

STAGE	WHAT IT DOES
Semantic chunking	Documents are split at natural content boundaries such as headers and paragraphs, with fixed and sliding-window strategies also available.
Hybrid search	Dense vector search and BM25 keyword search run in parallel and are combined by reciprocal rank fusion.
Re-ranking	Top candidates are re-scored against the query before they reach the model.
Metadata filtering	Retrieval can be scoped by collection, with a priority boost and a defined fallback when nothing matches.
Storage	Embeddings are stored and searched natively in Postgres with pgvector HNSW indexes, so there is no separate vector database to run.

5.5 Per-collection settings

SETTING	OPTIONS	DEFAULT
Chunking strategy	fixed, semantic (sentence boundaries), sliding_window (overlapping)	fixed
Chunk size	100 to 2,000 tokens	512
Embedding model	text-embedding-3-small, text-embedding-3-large	text-embedding-3-small
Retrieval scope	Per collection, with priority boost and an empty-result fallback	collection_only

Changing the embedding model after ingestion requires re-embedding existing documents. Retrieval is enabled per tenant through feature configuration.

5.6 Access boundaries in retrieval

BOUNDARY	HOW IT IS ENFORCED	STATUS
Tenant	Knowledge, embeddings and queries are scoped to the owning tenant.	AVAILABLE
Domain	An intent retrieves only from the collections linked to its domain.	AVAILABLE
Collection	A collection can be marked restricted to limit who can change it. This controls who edits the content, not who can retrieve it.	LIMITED AVAILABILITY
End user, department or document	Retrieval does not filter by the end user, role or department of the person asking, and documents carry no per-document access rules.	NOT AVAILABLE

> **Designing for audience separation.** If different departments must not see each other's documents, give each audience its own domain and collections, and have your application route each user to the domain they are entitled to. Tenant isolation alone does not provide per-user document permissions, and role-based output policies (Section 7.1) filter what a response may contain; they do not restrict which documents are retrieved.

5.7 Retrieval quality and memory

- **Measure it before you ship it:** evaluation scores faithfulness, context relevance and answer relevance whenever a call retrieved knowledge, so a change to documents, chunking or embeddings can be judged on evidence before it reaches production (Section 7.2).
- **Retrieval analytics:** each collection reports its retrieval hit-rate and top queries. Every call records the chunks retrieved, their sources and similarity scores, the top-k and threshold applied, and whether a boost or fallback fired.
- **Memory tiers:** a short-term buffer for the active conversation, episodic memory that summarises prior sessions for multi-session agents, and the semantic store for knowledge-intensive work.

SECTION 6

Agents and workflows

Agents handle multi-step reasoning with tools. Workflows coordinate intents, agents, tool actions and human approvals into an automation with a visible state at every step.

HOW A WORKFLOW RUN EXECUTES

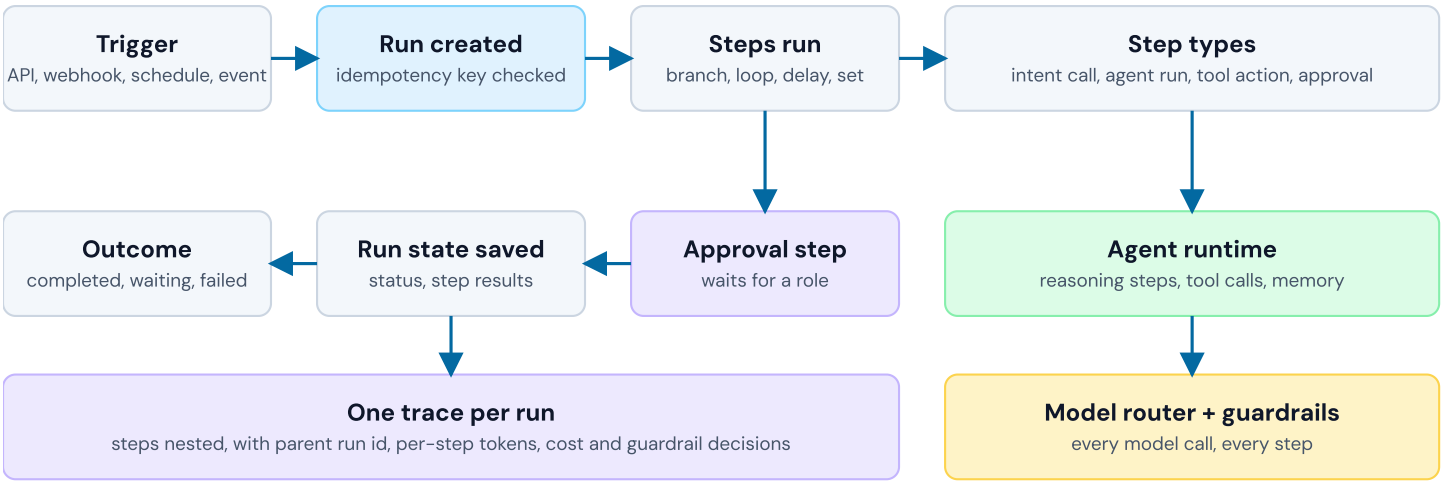

Figure 5. How a workflow run executes. Every step that calls a model passes through the model router and guardrails, and the whole run is one trace.

CAPABILITY	DETAIL
Triggers	API call, signed webhook (HMAC-SHA256 over the timestamp and body), schedule, or event.
Step types	Intent call, agent run, tool action, conditional branch, switch, delay, loop, set-fields, state, and human approval.
Human approval	An approval step keeps a role instead of a named person, so it travels safely between environments. The run waits in awaiting_approval until that role decides.

CAPABILITY	DETAIL
Run states	completed, awaiting_approval, waiting, needs_input, failed, rejected. A run that is waiting can be cancelled.
Repeat-call safety	An Idempotency-Key header stops a repeated call from running twice, and a workflow can limit how many of its runs execute at once.
Authoring and testing	A visual builder with a readiness check before deploy, test runs from a previous run's input, and versions with restore.
Tools and actions	Tool actions call your systems through configured actions. Guardrails can restrict which actions a model may invoke, and action sanitisation can redact PII from an action's parameters (opt-in).

Both agents and workflows are reachable from the dashboard and by API key. They run under the same guardrails and model routing as intents, and each run records a trace that nests the calls it made (Section 7.5).

SECTION 7

Governance, evaluation and observability

Governance is built into the request path, not added around it. Policies run on every call, evaluations test real behaviour before changes ship, and every request leaves evidence your engineering, risk and audit teams can inspect.

7.1 Guardrails

Every tenant has a default guardrail policy that applies to every intent, agent and action call. Named, reusable policies can be created and attached to specific intents, agents and actions, and the dashboard shows which policy applies where and what each one did over the last 7 days.

CONTROL	WHAT IT DOES
PII detection and redaction	Detects patterns such as emails, phone numbers, national IDs, card numbers and dates of birth in inputs and outputs, and masks, hashes or replaces them with typed placeholders. Healthcare identifiers and addresses are covered by an optional setting.
Content policy	Prompt-injection detection, profanity and violence filters, and your own blocked topics, layered on a platform safety floor (illegal activity, weapons, explicit content) that a tenant cannot loosen.
Bias and fairness filters	Detects and flags outputs showing demographic bias, with pre-built detectors for hiring, lending and clinical settings.
Role-based output policies	Defines what each user role may see, ask and receive, enforced at runtime. These shape what a response may contain. They do not decide which documents are retrieved, so they are not document-level access control (Section 5.6).
Schema and action validation	Enforces the intent's output schema and restricts which of your actions the model may invoke.
Hallucination and grounding checks	Flags or blocks claims that lack support in retrieved context, with a defined behaviour when an answer cannot be grounded.
Action sanitisation (opt-in)	Redacts PII from an action's outbound parameters or inbound response.
Audit logging	Records inputs, outputs and every guardrail decision.

Three-phase enforcement. Fast synchronous pre-filter checks run before the model call. Heavier validators, such as PII redaction, bias and grounding, run in parallel with the model response. A post-filter does the final review, including role-based masking and compliance metadata, before the response returns.

Standalone Guardrails API. `POST /v1/guardrails/check` runs your policy directly, with no intent, agent or domain involved, to screen content from your own systems. A `pre_llm` check covers input-side stages and a `post_llm` check covers output-side stages. The response reports whether the content passed, the content after any redaction, the issues found with their severity and action taken, and, for a blocked input, a user-safe fallback message. It is the same pipeline `/v1/run` uses, so the verdict matches what would happen in a real call.

7.2 Evaluation

Evaluation Studio scores your intents against real datasets so that a change in prompt, model or configuration can be judged on evidence. Everything is also available by API key and through the SDKs.

CAPABILITY	DETAIL
Datasets and cases	Each case holds an input shaped like your intent's input, an optional expected output for a deterministic assertion, and optional judge context. Import and export as JSON or CSV, and clone from starter templates covering support triage, RAG grounding and structured-output accuracy.
Suites	A suite binds one dataset to one domain and intent, and is what actually gets run and scored.
Deterministic gate	Expected-output matching and an optional custom scorer run before the judge. A failing check marks the case failed and skips the judge call. A scorer is either a safe expression over input, output and expected (comparisons, boolean logic, arithmetic, no arbitrary code), or your own HMAC-signed webhook.
LLM judge	Scores 1 to 5 at temperature 0 on coherence and actionability for any custom intent. When a call retrieved knowledge, it also scores faithfulness, context relevance and answer relevance.
Runs	Asynchronous, with status polling, cooperative cancel, and resume of a failed run in which already-scored cases are skipped and not billed again. Judge spend is capped per run.
Model comparison	Run one suite through two models and compare them with a statistical test (t-test and Cohen's d). Your own BYOK credentials are used first, so a model only your own key supports compares the same as a built-in one.
Baseline and trend	Pin one completed run as the suite's baseline. Every later run is compared to it automatically.
Pairwise judging	Ask the judge to pick a winner for each shared case between two runs, instead of two independent scores.
Human review	Reviewers agree with, override or flag any judge-scored result, with multiple reviewers per result for consensus.

7.3 Release governance

The point of evaluation is a release decision: is this version safe enough to deploy? LiyaEngine connects the evidence to the promotion step.

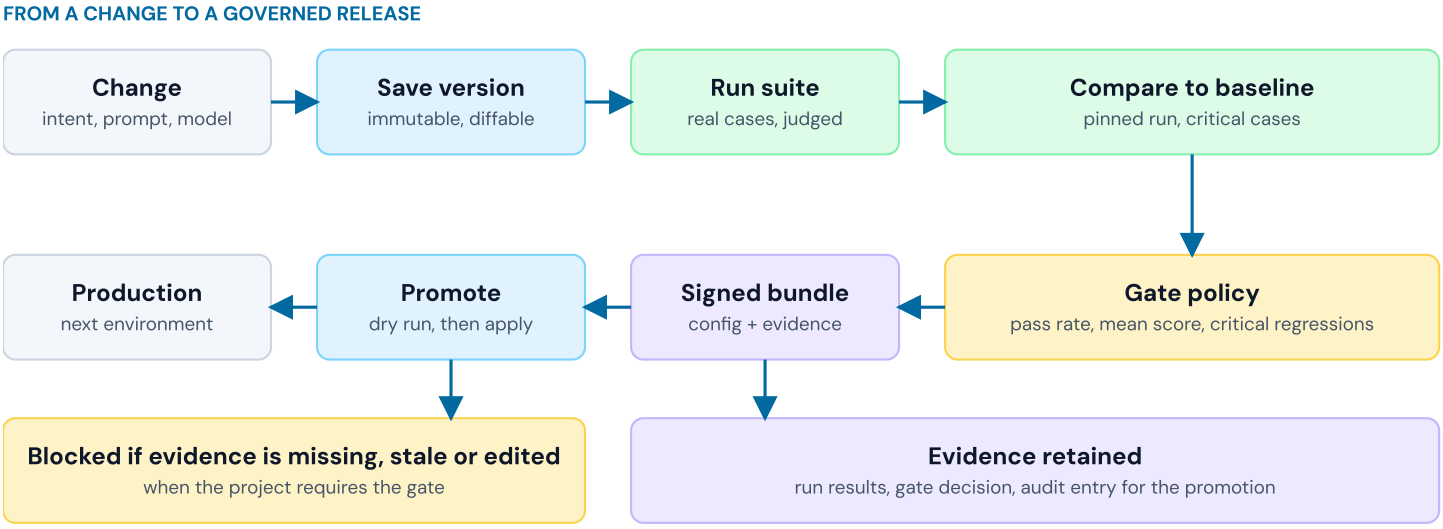

Figure 6. From a change to a governed release. Evidence travels with the configuration in a signed bundle.

Gate policy

SETTING	WHAT IT ENFORCES
Require a passed outcome	A failed, errored or inconclusive evaluation blocks the gate.
Minimum pass rate	The share of cases that must pass.
Minimum mean score	The average judge score, on the 1 to 5 scale, that must be reached.
Block critical regressions	Requires a pinned baseline, and blocks when a case marked critical passed in the baseline but does not pass in this run.
Fail fast on a critical failure	Stops the run at the first critical failure instead of finishing it.
Judge quorum for critical cases	Runs three independent judge attempts for each critical case and uses the majority decision.

How enforcement works today

- **A per-project switch.** Each project can require evaluation evidence for promotion. Only an owner can change the setting, and promoting requires an admin.
- **What is checked.** When the switch is on, an import is blocked unless every intent in the bundle passed its gate recently and has not been edited since.
- **Tamper evidence.** The evidence is only trusted in a bundle exactly as it was exported, because the bundle is signed. A bundle that was changed after export, or not produced by the platform, is blocked.
- **Dry run.** The check step writes nothing and shows every change as create, update or unchanged, with warnings. The plan carries a hash, so the import refuses if anything changed since you reviewed it.
- **Audit.** Every promotion is recorded in the audit log.

CONTROL	STATUS
Mandatory evaluation gate on promotion between projects, owner-controlled	AVAILABLE
Gate thresholds on pass rate, mean score and critical regressions	AVAILABLE

CONTROL	STATUS
Gate thresholds on cost or latency	NOT AVAILABLE
A required human approval step inside the release gate	NOT AVAILABLE
Requiring a passing security evaluation as a separate gate	NOT AVAILABLE
Gating edits made directly inside a project, outside promotion	NOT AVAILABLE

Human approval is available inside workflows (Section 6) and as reviewer sign-off on evaluation results, but it is not yet a required step of the release gate itself.

7.4 Evaluate without replacing your runtime

The Evaluation API scores any (input, output) pair directly, with no domain or intent required. Teams that run their AI elsewhere can send outputs to LiyaEngine for scoring, datasets, regression suites, baselines and review, and adopt the rest of the platform later or not at all.

7.5 Observability and unified tracing

Every request that reaches execution produces a trace automatically, with no instrumentation in your application. Traces are available in the dashboard and by API key (Section 3.9), and retention is set by plan (Section 10).

WHAT LINKS A TRACE TOGETHER	DETAIL
Identifiers	A trace id (the message_id that /v1/run returns), a request id, a session id, and for work done inside a run, the parent run id and kind (agent or workflow).
Nesting	An agent or workflow run is itself a trace, with the intent calls, model calls and tool calls it made nested inside. One run id opens the whole execution.
Per step	Every model call, retrieval and guardrail check is recorded as a timed span, with tokens and cost for model calls, and the guardrail issues found with their severity and action.
Retrieval evidence	Chunks retrieved with source and similarity score, plus the retrieval method, top-k and threshold applied.
Billing link	The trace joins to the usage and billing record for the same request.
Configuration version	The trace records a hash of the system prompt used, but not a reference to the exact intent configuration version. IN DEVELOPMENT

Requests rejected before execution (authentication, rate limit or quota) return an error and do not produce a trace. A workspace set to minimal capture records the structure and metrics of every trace but no message text.

CAPABILITY	DETAIL
Trace Explorer	Navigate a request's steps as a tree, a timeline or a graph, and inspect any one step. Open it as a side panel over the table, or as its own page.
Search, filter and API access	Search, sort, filter and choose columns in the Traces table, and share any view as a link. The same filters (domain, intent, issue kind, model, user, session, latency, time window) are available through the Traces API with a secret API key.
Collaboration and replay	Comment on and bookmark traces in the dashboard, and replay a model call to see how a different model would have answered.

CAPABILITY	DETAIL
Cost explorer	Real serving model and real cost on every request, sliced by provider and model, by domain, and by intent, with a live projection of the end-of-month bill.

7.6 Audit log

A tenant-wide audit log records dashboard and API actions, such as key rotation, provider credential changes, configuration updates, trace comments, replays and promotions. Entries carry the actor (user, system or API), the resource, a severity and a flagged marker, and can be filtered by severity, actor and flag or searched in free text. Access requires the admin role.

7.7 Change control

- **Intent versions:** every explicit save creates an immutable version capturing the prompt, schemas, guardrails and execution configuration. Compare any two versions with a field-level diff and restore an earlier one.
- **Workflow versions:** restoring one puts back its steps and connections and is itself saved as a new version.
- **Promotion:** move guardrail policies, actions, domains with their intents, agents and workflows between projects, such as development to UAT to production. A bundle never contains secrets, people, knowledge documents, API keys, model keys or billing, and references use names and keys, never database ids.

SECTION 8

Deployment and hosting

LiyaEngine is delivered as a managed cloud service. Teams with stricter data-location or isolation requirements can pin model inference to their own region today, and can scope a private deployment with us. These are different things, and the boundaries below show exactly what moves in each.

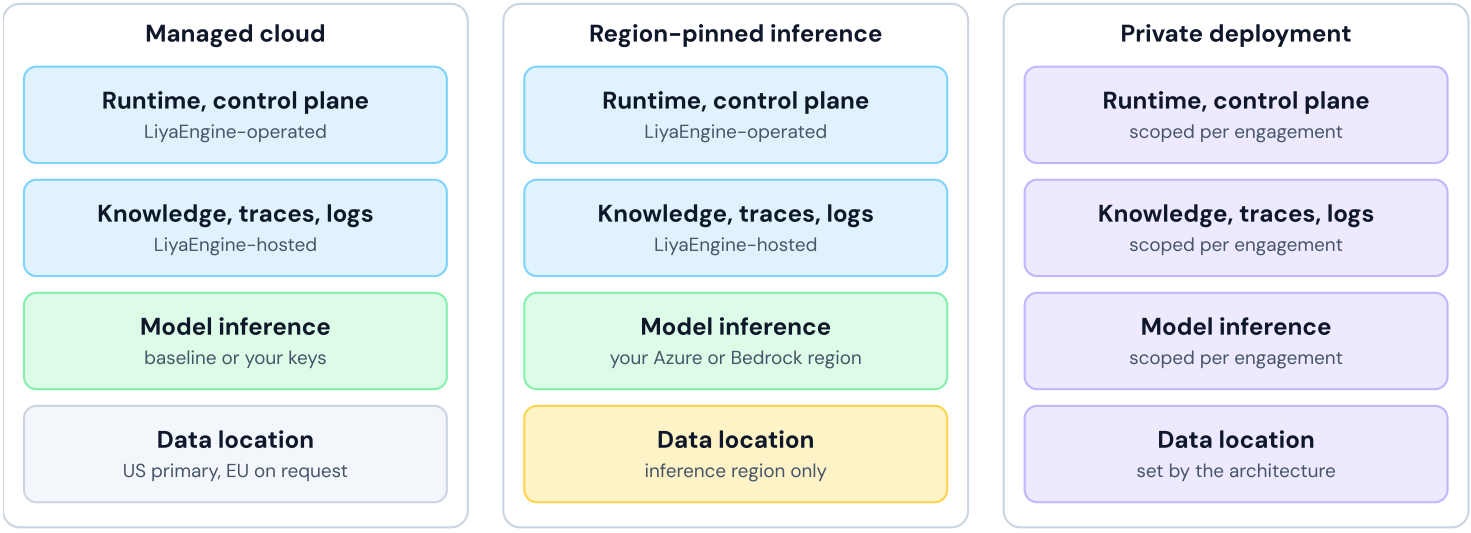

Figure 7. Deployment boundaries. Region-pinned inference changes where model calls run, not where LiyaEngine's own services and storage run.

COMPONENT	MANAGED CLOUD	REGION-PINNED INFERENCE	PRIVATE DEPLOYMENT
LiyaEngine runtime and control plane	LiyaEngine-operated	LiyaEngine-operated	Scoped per engagement
Model inference	Baseline models or your configured providers	Your Azure OpenAI or Bedrock region, or your own endpoint	Scoped per engagement
Knowledge storage	LiyaEngine-managed, tenant-scoped	LiyaEngine-managed, unless separately arranged	Scoped per engagement (tenant-hosted is an Enterprise option)
Execution traces and logs	LiyaEngine-hosted	LiyaEngine-hosted	Scoped per engagement
Operations	LiyaEngine	LiyaEngine, with your provider operating inference	Defined by the agreement
Data location	US primary, EU on request	Inference region only; everything else as managed cloud	Set by the architecture
Status	AVAILABLE	AVAILABLE	ENTERPRISE CUSTOM

> **Region-pinned inference is not regional data residency.** If you connect Azure OpenAI in a specific region, model calls run there. Your prompts and responses are still processed by the LiyaEngine runtime, and traces, knowledge and logs are stored by LiyaEngine in its hosting region. Choose private deployment, or a tenant-hosted knowledge base, where residency of those components matters.

8.1 Private deployment

Private deployment arrangements are designed per engagement during technical discovery. Architectures we discuss with customers include customer-managed cloud accounts, a dedicated network boundary, single-tenant managed deployment and hybrid arrangements that keep selected components, such as the knowledge base, in the customer's environment. These are starting points for scoping. None is a self-service option, and none should be assumed available until it is agreed in an engagement.

8.2 Environments

API keys are issued per environment, so development, staging and production traffic can be separated and each key rotated independently. Configuration moves between environments through promotion (Section 7.3).

8.3 Enterprise adoption path

- **Discover:** map the use case, systems, data classes, owners and production requirements.
- **Design:** define the domain, intent contract, knowledge boundary and policy path.
- **Validate:** exercise realistic requests, review evidence and complete stakeholder controls.
- **Expand:** standardise the pattern across product areas, teams and environments.

> **Availability and SLAs.** Contractual service levels and committed capacity are agreed per Enterprise engagement. This document does not state an uptime commitment.

SECTION 9

Security and data handling

Platform controls enforce boundaries inside a clear tenant boundary. You remain responsible for application design, authorisation, data classification and safe integration. This section separates what the platform enforces from what is a contractual commitment, and states limits plainly.

9.1 Encryption and credentials

CONTROL	BASIS	DETAIL
Encryption in transit	Contract commitment	TLS 1.2 or higher, as stated in the Data Processing Agreement.
Encryption at rest	Contract commitment	AES-256 for customer data, as stated in the Data Processing Agreement.
Provider credentials	Platform control	AES-256-GCM with per-tenant keys. Never returned by any API response; only a masked preview is shown.
API keys	Platform control	Shown once at creation or rotation. Rotation invalidates the old key immediately.

9.2 Tenant isolation and access control

- Domains, knowledge, configuration, sessions and operational data are scoped to the owning tenant.
- Runtime access uses API keys issued per environment, each with a scope of full, runtime (execute only) or read (GET only). Dashboard access uses authenticated users with roles, and trace content is shown to the supervisor role and above.
- Key scopes limit what a key can do, not which domains it can reach. Restricting an audience to part of a tenant is done in your own backend (Section 3.5).
- Internal administrative access follows least privilege, with multi-factor authentication required, and access to personal data is audit logged, as committed in the Data Processing Agreement.

9.3 Retention

These are configured defaults for every workspace. A default is a guarantee of deletion only where the status shows Available; the other rows state what is and is not automated.

DATA	DEFAULT	ENFORCEMENT	STATUS
Audit logs	730 days	A nightly job applies each workspace's window. Flagged and error or critical entries are kept regardless of age.	AVAILABLE
Sessions	30 days	A nightly sweep removes idle sessions and their messages. Pro and Teams keep them at least a year, because a stored message is also its trace. Until it is enabled for the deployment, the sweep reports what it would remove and removes nothing.	LIMITED AVAILABILITY
Messages	90 days	As for sessions.	LIMITED AVAILABILITY
Artifacts	365 days	Files and generated content attached to sessions are removed after the window. Until it is enabled for the deployment, the sweep reports what it would remove and removes nothing.	LIMITED AVAILABILITY

DATA	DEFAULT	ENFORCEMENT	STATUS
Conversation embeddings	365 days	Vectors derived from conversations (session summaries) are removed after the window. Knowledge base content is never expired by a retention window; it is removed when you delete it. Until it is enabled for the deployment, the sweep reports what it would remove and removes nothing.	LIMITED AVAILABILITY
PII-bearing text	7 days	Where PII was detected but not redacted (redaction is on by default), the message text is replaced with its redacted form after the window. The message itself stays. Until it is enabled for the deployment, the sweep reports what it would remove and removes nothing.	LIMITED AVAILABILITY
Traces, by plan	7 days, 30 days, 1 year	A trace past the plan's window is not shown in the Trace Explorer or the Traces API. Enterprise has no plan maximum.	AVAILABLE
Trace detail on Free and Core	7 and 30 days	Beyond the plan's window, timed steps, retrieved chunks, agent steps and guardrail issues are removed, and the message, tokens and cost stay. Until it is enabled for the deployment, the sweep reports what it would remove and removes nothing.	LIMITED AVAILABILITY

- **Conversation windows and trace windows.** A stored message is also a trace, so a conversation window never shortens a plan's trace window: Pro and Teams keep stored messages and runs at least a year, even if a shorter window is configured. Shorter plans show no traces beyond their window. To store no message text at all, use minimal capture or per-request consent flags (Section 9.4).
- **Deletion.** You can delete documents, collections, sessions and connector content on request or through the API. Removing a connector removes the content it ingested.
- **Backups.** Retention windows above apply to live data. Backup retention and deletion follow the hosting provider's backup cycle, and details are available on request.

9.4 PII and consent

Automatic PII redaction is on by default. Detection is pattern-based, for example emails, phone numbers, national IDs, card numbers and dates of birth, with an optional healthcare identifier and address setting, and an ML detector for names in guardrail checks. It can miss identifiers it does not recognise, so treat it as one layer of protection and not as the only control.

WHERE	COVERED
Input to the model	Redacted before the model call by the pre-filter guardrail. AVAILABLE
Stored message text	Redacted before storage when automatic redaction is on (the default). AVAILABLE
Stored span text in traces	Redacted under the same setting. AVAILABLE
Agent tool-call parameters	Redacted only when action sanitisation is turned on for the policy. LIMITED AVAILABILITY
Structured outputs, retrieved knowledge chunks, agent step data	Stored as produced under full capture, not redacted by this setting. NOT AVAILABLE
Evaluation datasets and judge output	Stored as you supply and as the judge returns them. NOT AVAILABLE

For workloads where no content should be stored, use minimal capture, which records structure and metrics but no message text, and per-request consent flags that disable data storage, session memory or embeddings, keep analytics, or set a shorter retention period for that end user.

9.5 Use of customer data

Data submitted through the API is not used to train, fine-tune or improve LiyaEngine's models or any third-party model. This is committed in the Data Processing Agreement.

9.6 Legal and compliance commitments

ITEM	DETAIL	STATUS
Data Processing Agreement	Published at liyaengine.ai/dpa, with sub-processor categories and EU Standard Contractual Clauses for international transfers.	AVAILABLE
Sub-processors	Cloud infrastructure (US primary, EU on request), web hosting, error monitoring and payment processing. Specific names on request, with 14 days' notice of material changes.	AVAILABLE
Breach notification	Without undue delay and within 72 hours of becoming aware of a personal-data breach.	AVAILABLE
HIPAA BAA	A Business Associate Agreement is available for teams processing protected health information.	ENTERPRISE CUSTOM
Customer audit rights	Information to demonstrate compliance, with 30 days' notice.	AVAILABLE
Data export and deletion	Export in a standard format within 30 days of a request on termination; deletion after export or after the 90-day post-termination window.	AVAILABLE
SOC 2	A goal on our enterprise roadmap. No certification is claimed today.	PLANNED

9.7 Vulnerability disclosure and shared responsibility

Report suspected vulnerabilities to security@liyaengine.ai with reproduction steps, expected impact and supporting evidence, and avoid accessing customer data or disrupting production systems. LiyaEngine supplies the platform controls above. Customers are responsible for application authorisation, classifying the data they send, choosing retention and consent settings for their workload, and safe handling of their own API keys.

> **Security reviews.** We support security review with architecture walkthroughs, answers to security questionnaires and the Data Processing Agreement. Request them through liyaengine.ai/contact.

SECTION 10

Pricing and plans

Plans scale with requests, tokens and operational depth. Every plan includes the same stable API model, and BYOK is included on all of them.

	FREE	CORE	PRO	TEAMS	ENTERPRISE
Price	$0	$79 / month	$199 / month	$499 / month	Custom
Monthly requests	1,000	50,000	150,000	150,000	Committed
Monthly token budget	250K	8M	20M	20M	Committed
Model access	Standard (1×)	+ Advanced (15×)	+ Premium (30×)	Same as Pro	+ Frontier, negotiated
Bring your own key (BYOK)	Included	Included	Included	Included	Included
Knowledge base (KBaaS)	5MB · 1K queries/mo	1GB · 50K queries/mo	10GB · 500K queries/mo	10GB · 500K queries/mo	Negotiated
Evals run-credit	—	$30/mo	$225/mo	$225/mo	Negotiated
Agentic execution	—	Included	Included	Included	Included
Traces retention	7 days	30 days	1 year	1 year	No plan maximum
Security and governance	Standard	Standard	Standard	Roles and audit trail. SSO in development	Roles and audit trail. SSO and SCIM in development
Support	Community	Email · 48h SLA	Email · 48h SLA	Priority	Dedicated + SLA
Deployment options	Cloud	Cloud	Cloud	Cloud	Cloud, or private by engagement

"No plan maximum" means Enterprise has no plan-imposed limit on how long traces can be retained, with the window agreed in your contract. It does not mean data is kept forever regardless of your retention and deletion requirements.

10.1 What counts toward usage

ACTIVITY	HOW IT IS COUNTED
An intent call	One request, through either the unified endpoint or the path form.
Tokens	Model tokens used by the call count against the monthly token budget, unless a BYOK key served it.
Knowledge queries	Count against the knowledge query allowance, and stored documents count against the storage allowance.
Evaluation runs	Judge calls draw on the evaluation run-credit, in dollars, and are capped per run. They do not use the request allowance.
Agent and workflow runs	Metered through the intent and model calls they make. We provide a worked cost estimate for your design during technical discovery.
Standalone guardrails checks	Recorded in usage like other calls.

10.2 Overages and top-ups

Subscriptions include allowances for requests, tokens, knowledge storage and queries, and evaluation spend. Paid plans can exceed them with metered overage, and token top-ups add capacity without changing plan. Overage rates are confirmed when you subscribe or in your Enterprise agreement.

TOP-UP BUNDLE	PRICE	EFFECTIVE RATE
500K tokens	$5	$0.01 / 1K tokens
1.2M tokens	$10	$0.0083 / 1K tokens
3.5M tokens	$25	$0.0071 / 1K tokens

10.3 Choosing between Pro and Teams

Pro and Teams share the same request, token, model and knowledge allowances. Teams adds the governance layer: roles, an audit trail and priority support. SSO is in development, so do not buy Teams for SSO until timing is confirmed with us.

BYOK requests bypass the token pool and model-tier limits on every plan, and you pay your provider directly for inference. Enterprise adds custom domain packs (up to 10), negotiated frontier access, SLAs and private deployment by engagement. Current pricing is at liyaengine.ai/pricing.

SECTION 11

Availability and roadmap

What is generally available, and what is not, using the labels defined at the start of this document.

CAPABILITY	STATUS	NOTES
Runtime API, intents, agents, workflows, collections, evaluations, traces	AVAILABLE	Available on all plans, subject to plan limits.
Baseline models and BYOK for seven provider types	AVAILABLE	With ordered failover across your chain (Section 4.4).
Guardrails, evaluation, tracing, audit log, promotion with evaluation gate	AVAILABLE	Gate thresholds are pass rate, mean score and critical regressions (Section 7.3).
Encryption of provider credentials, PII redaction of stored message text	AVAILABLE	Coverage and limits in Section 9.4.
Audit-log retention enforcement	AVAILABLE	Per workspace.
Conversation retention (sessions, messages)	LIMITED AVAILABILITY	Built, with the plan trace window as a floor. Applied when enabled for the deployment (Section 9.3).
Connector synchronisation	LIMITED AVAILABILITY	Re-read of selected sources, with no deletion or permission propagation (Section 5.3).
JavaScript / TypeScript and Python SDKs	AVAILABLE	Early access. They cover most of the runtime and management API, including read-only traces.
Tenant-hosted knowledge base	ENTERPRISE CUSTOM	Scoped per engagement.
Private deployment	ENTERPRISE CUSTOM	Scoped per engagement during technical discovery.
SSO and SCIM	IN DEVELOPMENT	SAML 2.0 and OIDC integration is being built for Teams and Enterprise. Confirm timing with our team.

CAPABILITY	STATUS	NOTES
Retention of artifacts, conversation embeddings, PII text and Free and Core trace detail	LIMITED AVAILABILITY	Built. Applied per category when enabled for the deployment (Section 9.3). Knowledge base content is never expired.
Trace windows by plan (Free 7 days, Core 30, Pro and Teams 1 year)	AVAILABLE	Traces past the window are not shown. Pro and Teams rows are kept at least a year.
Exact configuration version recorded on each trace	IN DEVELOPMENT	Traces record a system-prompt hash today.
Failure-type-aware provider failover policies	IN DEVELOPMENT	Section 4.4.
Scoped API keys (runtime and read)	AVAILABLE	Limit what a key can do (Section 3.5).
API keys restricted to specific domains or resources	NOT AVAILABLE	Scopes limit actions, not which domains a key reaches (Section 3.5).
Per-user and per-document retrieval permissions	NOT AVAILABLE	Use separate domains and collections per audience (Section 5.6).
Cost, latency, human-approval and security gates in release governance	NOT AVAILABLE	Section 7.3.
SOC 2 certification	PLANNED	A goal. No certification is claimed today.

SECTION 12

Enterprise technical FAQ

QUESTION	ANSWER
Can we keep our own model provider agreements?	Yes. Bring your own keys for seven provider types, including your own Azure OpenAI resource or an AWS Bedrock role in your account. Inference is billed by your provider.
Does pointing inference at our region keep our data in that region?	Only inference. LiyaEngine's runtime, traces, knowledge storage and logs stay with LiyaEngine unless a private deployment or tenant-hosted knowledge base is agreed (Section 8).
What happens if our primary provider fails?	The router tries the next provider in your ordered chain inside the same request. It does not use providers you have not configured, and it does not yet distinguish transient from permanent failures (Section 4.4).
Can an API key be limited?	By action, yes: a key can be limited to running intents (runtime) or to reading (read). Not by domain or resource. Use embed tokens, restricted to allowed intents, for browsers (Section 3.5).
Can different departments be prevented from retrieving each other's documents?	Through separate domains and collections, routed by your application. There are no per-user or per-document permissions inside a collection (Section 5.6).
Can we enforce an evaluation gate on releases?	Yes, for promotion between projects. An owner can require passing evidence, and the import is blocked without it. Cost, latency, human-approval and security gates are not available yet (Section 7.3).
Can we use evaluation without moving our runtime?	Yes. The Evaluation API scores outputs from any system (Section 7.4).

QUESTION	ANSWER
What is stored when a request runs, and can we stop it?	Message text, spans, retrieved context and metrics are stored as traces. Use minimal capture for no message text, and consent flags per request (Section 9.4).
Do you offer SSO, and do you hold SOC 2 or ISO 27001?	SSO and SCIM are in development (Section 11). SOC 2 is a goal, and no certification is claimed today.

SECTION 13

Contact and next steps

We will review architecture, security, residency and procurement requirements with your team.

13.1 A suggested evaluation path

STEP	WHAT HAPPENS	WHAT WE NEED FROM YOU
1. Technical discovery	Map the use case, systems, data classes, owners and production requirements.	An architecture sketch, the first use case and the sensitivity of the data involved.
2. Security review	Walk through Section 9, the Data Processing Agreement and your questionnaire.	Your security questionnaire, compliance requirements and residency needs.
3. Proof of concept	One intent and one knowledge collection, evaluated against your own cases.	Real questions with expected outcomes, and a test environment.
4. Production readiness	Gate policy, promotion path, rate limits, committed capacity and a runbook.	Your release process, expected volumes and service-level needs.

13.2 Contacts

FOR	WHERE
Enterprise sales and technical discovery	liyaengine.ai/contact
Security questions and vulnerability reports	security@liyaengine.ai
Legal, DPA, BAA and privacy requests	legal@liyaengine.ai
Documentation and pricing	liyaengine.ai/docs and liyaengine.ai/pricing

LiyaEngine technical overview, version 1.1, October 2026. Describes the platform at publication and may change. Provided for evaluation; it is not a contract, and commercial and legal terms are set out in the applicable agreements.