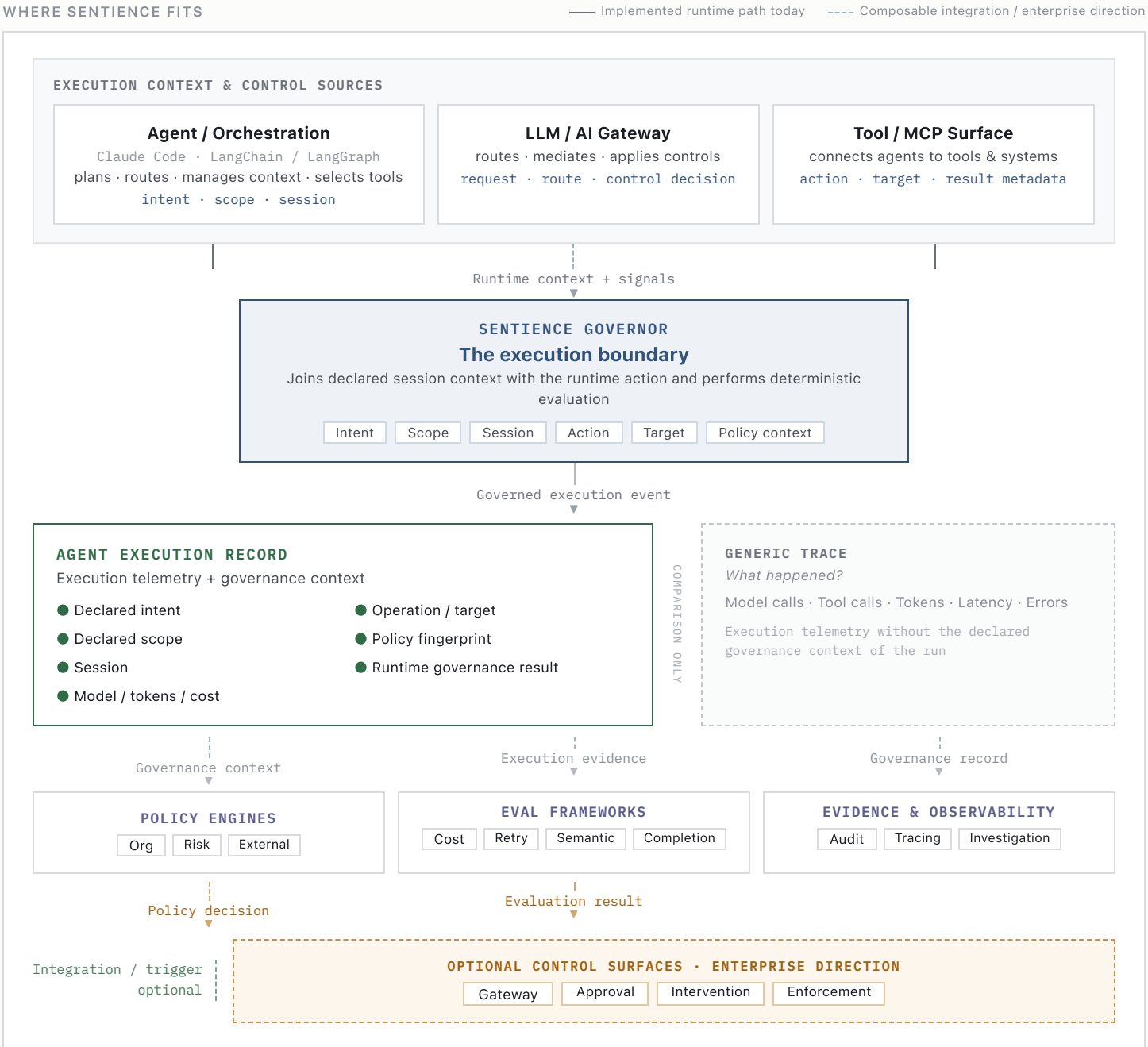
Sentience Governor

The execution boundary for AI agents

AI agents operate across orchestration, models, gateways, tools, APIs, data, and enterprise systems. Sentience sits at the execution boundary, where declared session context and the runtime action can be evaluated together.

WHERE SENTIENCE FITS



Sentience turns runtime activity into a governance-aware Agent Execution Record, binding execution telemetry to declared intent, scope, session, target, policy fingerprint, and runtime governance result.

That record can compose with the policy engines, eval frameworks, observability systems, and control surfaces an enterprise already uses.

COMPLEMENTARY CONTROLS

How Sentience complements existing controls

As companies give AI agents more autonomy, agents increasingly interact with files, APIs, databases, MCP servers and business systems. Sentience Governor independently evaluates agent actions against an operator-authored governance policy — as they happen.

DETERMINISTIC · LOCAL-FIRST · SESSION-SCOPED POLICY · FLAGS & RECORDS · NON-BLOCKING

WHAT EACH LAYER ANSWERS

Different controls answer different questions across an agent run. Sentience focuses on the execution boundary.

Model guardrails	Is this AI input or output safe and allowed?	e.g. AWS Bedrock Guardrails
Agent observability	What happened during the agent run?	e.g. Langfuse
Runtime governance	**Did the agent's runtime actions stay within the scope it declared?**	**Sentience Governor**

HOW THE LAYERS DIFFER

	AWS Guardrails	Langfuse	Sentience Governor
Primary focus	Model inputs & outputs	Traces & telemetry	**Agent runtime actions**
Core question	Is this content allowed?	What happened?	**Did execution stay within declared scope?**
Scope adherence	Not the primary purpose	Can expose activity for analysis	**Core governance signal**
Evidence of policy applied	Service / configuration dependent	Trace-based	**Policy fingerprint recorded with runtime evidence**
Local-first	No	Self-hosting available	**Yes, by design**

WHY RUNTIME GOVERNANCE MATTERS

A safe model response does not guarantee accountable agent execution. Between the model's answer and the completed task, an agent can still take an unexpected path, invoke the wrong system, or act outside its declared scope — and neither guardrails nor after-the-fact traces evaluate those actions against a governance boundary as they happen.

Sentience **complements** guardrails and observability rather than replacing them: guardrails keep the model interaction safe, observability records what ran, and Sentience evaluates runtime actions against the policy declared for the session.

CURRENT OPERATING MODEL

Sentience flags and records governance events. It does not block execution today.

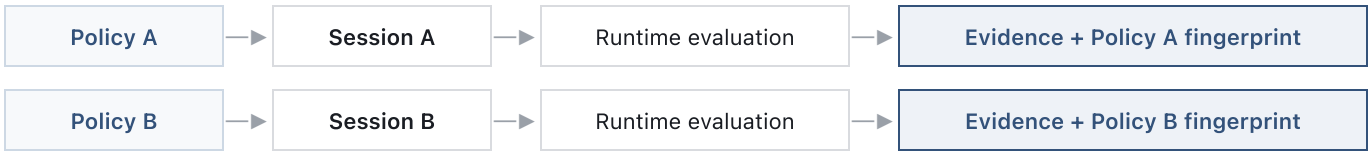
An advisory, evidence-producing layer, deliberately non-blocking.

POLICY FLEXIBILITY

Policy flexibility is built into the runtime

Sentience's runtime is built around **session-scoped policy** rather than a single global policy model. Policy is bound to each session, evaluated locally, and recorded in the resulting evidence with a fingerprint of the policy that applied.

VERIFIABLE PER-SESSION POLICY



Sentience can prove not only what an agent did, but which policy was applied when it did it.

The runtime is already designed to support differentiated policy across sessions.

CONFIGURABLE TODAY

Intent requirements
When a session must declare intent.

Task-boundary signals
Signals that mark the edges of the declared task.

High-consequence detection
Patterns that flag actions warranting attention.

DETERMINISTIC BY DESIGN

Evaluation happens against **structured runtime events**, not one AI model judging another.

Independent — evaluates the agent from outside its own reasoning.

Local-first — policy and evidence live close to the runtime.

Deterministic — the same events and policy yield the same result.

RUNTIME POLICY EVALUATION

Declared objective — Update subscription status for Customer A

Declared scope `database`

`Read Customer A`	✓ within scope
`Update Customer A`	✓ within scope
`Access credential store`	⚠ outside scope

Guardrails can protect the model interaction. Observability can show that these actions occurred. **Sentience evaluates those actions against the governance boundary declared for the run and creates independent runtime evidence of the result.**

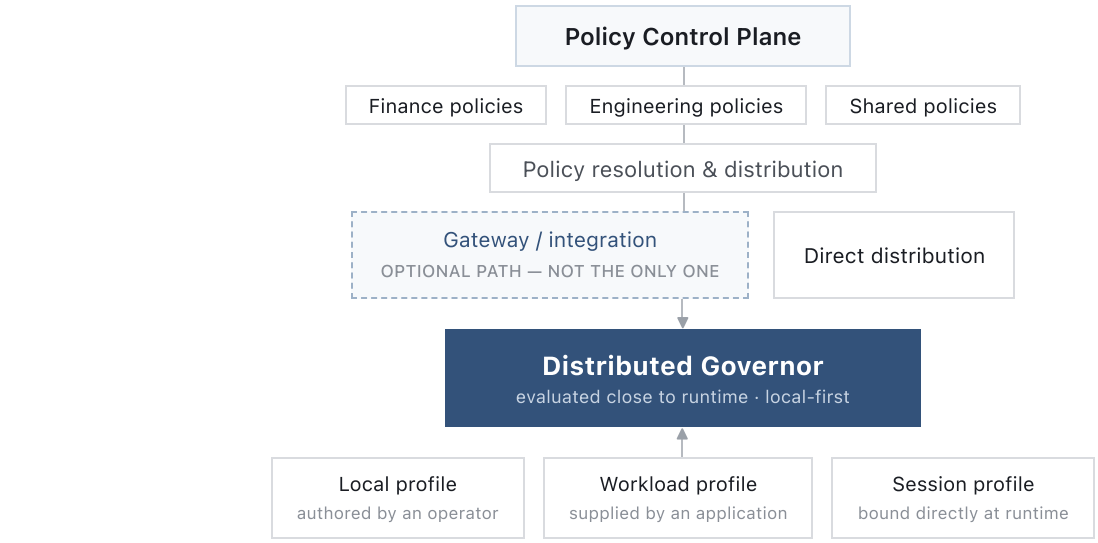
Sentience evaluates actions against declared scope. It does not yet judge whether an action inside an allowed scope serves the stated objective.

ENTERPRISE GOVERNANCE

Policy can originate centrally or locally

The runtime foundation already provides **session-scoped policy, local evaluation, and evidence of which policy applied**. Enterprise Sentience extends this with policy resolution, distribution and fleet-level management — without forcing one mandatory policy path.

ILLUSTRATIVE ENTERPRISE DIRECTION



A gateway can be one enterprise integration and policy-distribution point where that fits the customer's architecture, while direct distribution and locally supplied policy remain valid paths.

SHIPPING TODAY	ENTERPRISE DIRECTION
Session-scoped policy · deterministic runtime evaluation · local evidence · policy fingerprints · configurable governance signals · flags and advisories	Differentiated policy resolution · centralized management and distribution · gateway integration where appropriate · fleet-level visibility · direct local policy where the workload requires it

Policy can originate centrally or locally. Governance is bound to the execution session, evaluated close to runtime, and recorded in the evidence.

> Guardrails govern AI inputs and outputs.
> Observability records execution.
> **Sentience Governor holds runtime execution accountable to its declared governance boundary.**