

Enterprise Agentic Pattern Catalogue (starter set)

Version 0.1 · CC BY-NC 4.0 — attribute to Contextual Agentic, contextualagentic.com

Ten recurring shapes in enterprise agent systems, each with the problem it answers, what it costs, and — the part usually missing — when not to use it.

Scope. This is a starter set of ten patterns written for this site. It is not Appendix A of the book, which holds 25 pattern cards with their failure modes, controls and evidence; the two lists are different. The full set is in the book.

Drafted for the companion material of The Contextual Agentic Enterprise. These are descriptions of recurring designs, not a standard, and nothing here is certified by anybody.

Entries

CAP-01 — Semantic Ground Truth

Problem — The agent infers the business from whatever documents retrieval happened to return, so two runs can disagree about what a customer is entitled to.

Approach — Give the agent a queryable model of the organisation's own definitions, entitlements and records, and make retrieval a fallback rather than the primary source.

Evidence — The query and the version of the model it ran against, recorded with the decision.

When not to — When no semantic model exists and building one is the whole programme. Say so rather than pretending a vector store is one.

Related — CAP-02, CAP-07

CAP-02 — Bounded Context Window

Problem — Everything the agent can see, it can act on and can leak. An over-broad context is an over-broad blast radius.

Approach — Assemble context per task from what the task needs, scoped to the principal on whose behalf the agent is acting, rather than handing it a corpus.

Evidence — A manifest of what was assembled, for whom, and why each item was included.

When not to — Exploratory research tasks where the point is breadth and nothing is written back.

Related — CAP-01, CAP-04

CAP-03 — Explicit Authority Grant

Problem — What the agent may do is implied by its prompt and its credentials, which means nobody can answer "who authorised this" afterwards.

Approach — Model authority as a first-class object — principal, scope, resource, limit, expiry — granted deliberately and revocable in one action.

Evidence — The grant that was relied on, its scope, and who issued it.

When not to — Read-only agents with no write path at all, where the grant is the identity itself.

Related — CAP-04, CAP-05

CAP-04 — Enforcement Point

Problem — Controls implemented inside the agent's reasoning can be reasoned around. A model asked nicely enough will talk itself past its own instructions.

Approach — Put the decision in a component that never sees the argument: it receives an action and a grant, and it executes or refuses.

Evidence — The decision, the inputs to it, and the rule that produced it.

When not to — Never, for any action that touches a system of record. For read paths it may be disproportionate.

Related — CAP-03, CAP-09

CAP-05 — Escalation with Hold

Problem — Approval implemented as a notification means the action proceeds and someone is told about it, which is not approval.

Approach — Hold the action in a durable state until a named approver decides. An unanswered escalation stays held; it does not time out into a yes.

Evidence — Who was asked, when, what they decided, and what was held in the meantime.

When not to — Where latency genuinely cannot tolerate a human, in which case the honest answer is to narrow the agent's authority rather than to skip the approval.

Related — CAP-03, CAP-06

CAP-06 — Fail-Closed Control

Problem — A control that cannot be reached is frequently treated as a control that said yes. This is the failure mode behind most agentic incidents worth the name.

Approach — Treat unavailable as refused, everywhere, and make the refusal visible rather than silent so the outage is noticed.

Evidence — The attempted call, the failure, and the refusal it produced.

When not to — Never for an action with external effect. For enrichment steps, degrading is fine — say which is which.

Related — CAP-04, CAP-05

CAP-07 — Write-Path Review

Problem — Retrieval gets the design attention and the write path gets a function call. The expensive failures are all on the write path.

Approach — Treat every mutation as a designed interface: preconditions, idempotency, a compensating action, and a review of it separate from the read design.

Evidence — The precondition that held, the idempotency key, and what would undo it.

When not to — Read-only systems. If you think that is you, check what the agent's tools can do.

Related — CAP-01, CAP-04

CAP-08 — Evidence as By-product

Problem — Assurance is assembled afterwards from logs that were designed for debugging, which is archaeology, and an auditor has no reason to trust it.

Approach — Emit the record at the point of decision, on every path including refusals, in a form built to be read by a reviewer rather than grepped by an engineer.

Evidence — Itself. That is the pattern.

When not to — Where the volume genuinely cannot be afforded — then sample deliberately and record the sampling rule, rather than recording nothing and calling it pragmatism.

Related — CAP-06, CAP-10

CAP-09 — Two Independent Boundaries

Problem — One control plane means one bug between the agent and production.

Approach — Enforce the same intent twice through mechanisms that fail differently — an application-level check and a platform-level one — so a defect in either is contained.

Evidence — Both decisions, and an alert when they disagree, because disagreement is information.

When not to — Low-consequence actions where the second boundary costs more than the risk it removes.

Related — CAP-04, CAP-06

CAP-10 — Deterministic Replay

Problem — A record produced by a process that would answer differently next Tuesday supports no argument about what the system does.

Approach — Make the control chain a pure function of recorded inputs, so a decision can be re-evaluated later and must come out the same.

Evidence — The inputs, the version of the policy set, and the outcome — enough to re-run it.

When not to — The agent's own reasoning is not replayable and should not be claimed to be. Replay the controls, not the model.

Purpose and professional-use notice

This book, course and related articles are designed to help readers understand, discuss and apply practical approaches to real enterprise challenges involving artificial intelligence, architecture, governance, security, data and operating models.

The material is educational. It is intended to support informed thinking, better questions, stronger design decisions and meaningful professional work. It is not a ready-made solution for every organisation or situation.

Adapt what you learn to your own context. Before applying any idea, framework, pattern, checklist or technical approach, consider your organisation's objectives, people, systems, data, risk appetite, contractual commitments, laws, regulatory obligations and governance requirements. Obtain the appropriate internal approvals and independent professional advice where needed.

The material does not constitute legal, financial, investment, tax, regulatory, privacy, cyber-security, safety, employment, procurement, audit, medical or other professional advice. It should not be relied on as a substitute for advice tailored to your particular circumstances.

The views, frameworks and examples are the author's independent professional perspectives. They do not represent the views, policies, strategies, recommendations or endorsement of any employer, client, partner, vendor, regulator or other organisation.

Application requires organisation-specific architecture, security, privacy, legal, regulatory, operational, safety, accessibility, commercial and governance review, together with appropriate approvals. Full disclaimer: <https://contextualagenticenterprise.com/disclaimer/>