The Execution Governance Doctrine

A field guide for building authority into agentic systems.

Authority. Evidence. Escalation. At the moment of action.

Public Field Guide | Version 1.3 | March 2026
judgementspine.com | founder@judgementspine.com

Read this first

This doctrine is operational. It names a failure mode, defines shared language, and gives you a repeatable control system for AI that can act.

So what?

Status quo: Policies, review boards, and audit logs assume humans initiate and pace actions.

AI shift: Agents and automations compress the window from proposal to consequence to seconds.

Judgement Spine impact: Execution governance binds authority and proof to the last safe moment, before impact.

How to use this guide

- If you only read one section, read Part II (The Doctrine).
- If you are building adapters and services, read Part III (The System).
- If you are responsible for risk, read Part V (Maturity + 30-60-90).
- Use the templates in the appendices directly. Keep them short and enforceable.

Non-goals

This is not a prompt handbook, a model evaluation guide, or a compliance checklist. It is the missing layer between capability and permission: execution governance.

Map

What is in this document (and why).

- **Part I - The Shift:** assistants became actors; wrong actions replaced wrong answers.
- **Part II - The Doctrine:** authority envelopes, evidence gates, escalation ladders, deterministic safe degrade.
- **Part III - The System:** traces, authority graphs, reference architecture, operating model.
- **Part IV - Application:** enterprise control surfaces and where to start.
- **Part V - Implementation:** maturity model, 30-60-90 plan, what good looks like.
- **Appendices:** copyable templates, checklists, glossary.

One sentence definition

Judgement Spine is an execution-time authority control plane that governs the moment before consequence - binding actions (human or agentic) to explicit authority, enforcing escalation under uncertainty, and emitting audit-grade evidence packs.

Part I - The Shift

We are entering the age of action. The failure mode is no longer wrong answers. It is wrong actions.

Three things changed at once

- Outputs became transactions: agents are wired to tools (identity, cloud, finance, ticketing, customer channels).
- Risk became non-linear: a small error can cascade through automation, privileges, and integrations.
- Accountability became architectural: you cannot policy your way out of actions you cannot trace.

The control gap

Zero Trust answered: who can access what.

Agentic systems introduce the next boundary: who can act, under what escalation, once access exists.

Execution governance is that missing layer.

Chapter 1 - The world changed

For the last decade, systems were judged on recommendation quality. That era is ending.

When systems act, the unit of risk is no longer a sentence. It is a side effect in the world: a permission grant, a production change, a payment release, a message sent to customers.

So what?

Status quo: Teams treat governance as review before work and audit after work.

AI shift: Action is now initiated by agents, triggered by signals, and executed by integrations.

Judgement Spine impact: Governance must move into the runtime - the last safe moment before impact.

What this makes possible (and dangerous)

- Safe speed: faster operations without expanding ambiguity.
- Automation under uncertainty: systems must know when not to act.
- Proof before consequence: evidence becomes a gate, not a log.

Chapter 2 - Failure mode: execution-time authority drift

Most incidents do not happen because a system was capable. They happen because capability was not constrained by authority.

- Agents silently expanding scope ("while I am here, I will also...").
- Privilege accretion: temporary access becomes permanent through convenience.
- Tool chaining that bypasses human intent.
- Unclear ownership: no single authority owner for the outcome.
- No deterministic fallback when signals conflict.

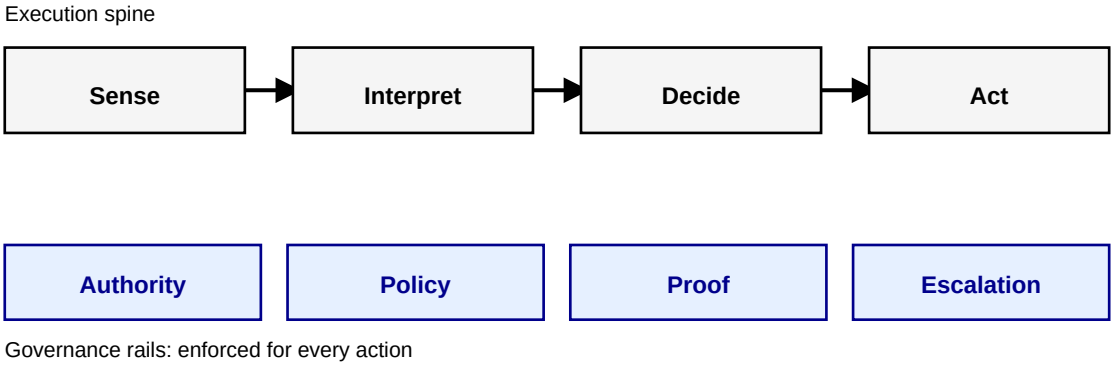
Category claim

The next security frontier is not model safety. It is execution governance - the authority layer at the moment of action.

Part II - The Doctrine

Judgement Spine is a control system: authority boundaries, evidence gates, and escalation logic across every action.

Judgement Spine model: execution + governance rails



- **Authority is not a permission list.** It is a bounded envelope tied to intent.
- **Evidence is a gate.** Proof is required before high-impact actions.
- **Uncertainty triggers contraction.** Less autonomy, more escalation.
- **Reversibility is a design constraint.** If it cannot be undone, it is not autonomous.

Chapter 3 - The language

If the market cannot name the problem, it cannot buy the solution.

Term	Definition
Action	A change to system state with side effects (create/modify/delete/approve/grant/deploy/message).
Authority	Formally approved scope of actions permitted under explicit conditions.
Judgement	Knowing when not to act: wait, ask, escalate, fail closed.
Authority envelope	Boundary between capability and permitted execution defined by scope, risk, reversibility.
Execution trace	Evidence record proving what happened, why, under which policy, with which approvals.
Degradation	Deterministic fallback that reduces autonomy when uncertainty increases.
Exploitable execution path	Sequence of legitimate tool calls that could produce illegitimate outcomes.

Repeat the terminology. Categories are built on language.

Chapter 4 - Ten axioms of execution governance

These rules are intentionally blunt. They prevent improvisation at machine speed.

- **Authority precedes capability.** If you cannot define the authority boundary, do not automate the action.
- **Every action must be attributable.** Name the authority owner and escalation target.
- **Evidence is a gate, not a log.** Proof is required before high-impact actions.
- **Uncertainty triggers contraction.** When confidence drops, autonomy reduces and escalation increases.
- **Reversibility is mandatory.** If the action cannot be undone, it is human-only.
- **Least privilege is dynamic.** Temporary permissions expire and must be re-justified.
- **Policies must be executable.** Rules must be machine-checkable at runtime.
- **Traceability is non-negotiable.** If you cannot reconstruct why, you cannot operate safely.
- **Attack paths become execution paths.** Tool graphs define what can be abused.
- **Governance must be testable.** Red team the agent like you red team your environment.

Example: the difference between policy and execution governance

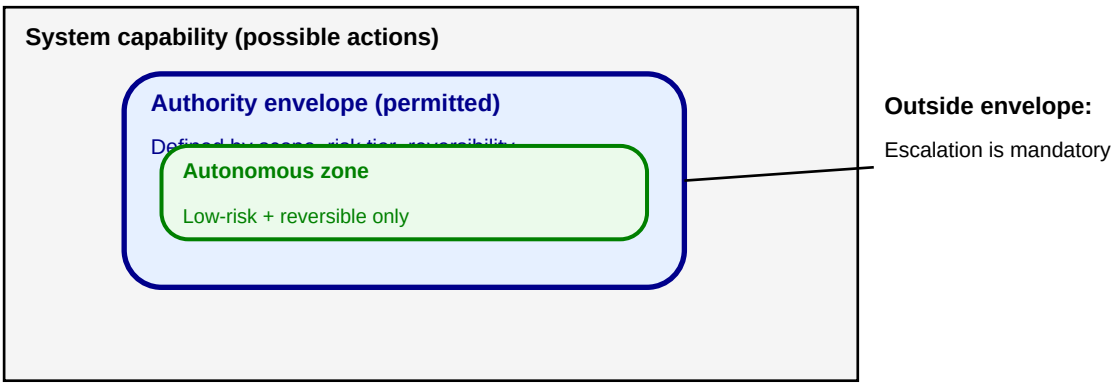
Policy says: "Only Finance can approve refunds over $X."

Execution governance ensures: the register cannot issue the refund without an authority token - and mints proof when it is allowed, bounded, degraded, or blocked.

Chapter 6 - Authority envelope

Most teams treat authority as a static permission list. In execution governance, authority is an envelope tied to intent, risk, and reversibility.

Authority Envelope: capability is larger than permission



So what?

Status quo: Agents can access tools; boundaries are implied by convention.

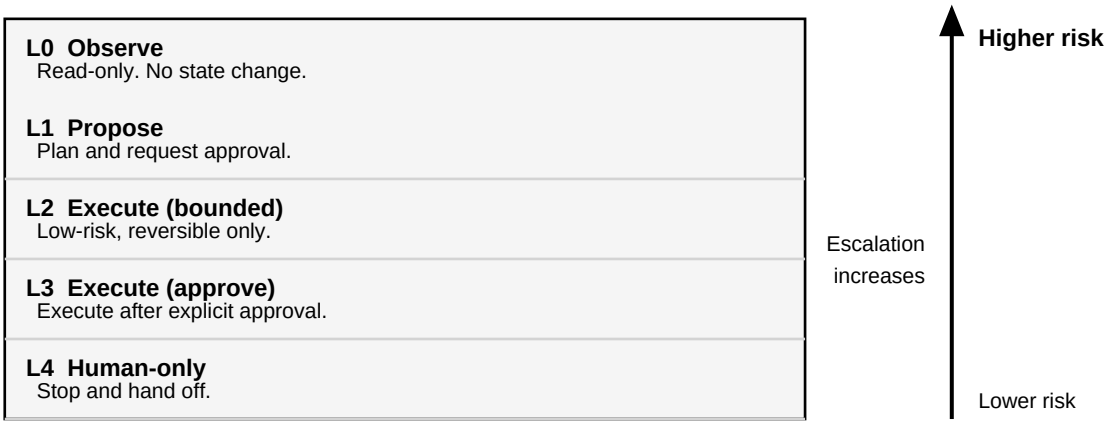
AI shift: Agents chain tools mid-run and expand scope by momentum.

Judgement Spine impact: The envelope is enforced at runtime. Outside envelope -> escalation is mandatory.

Chapter 7 - Escalation ladder

Escalation is not failure. It is a feature. The ladder makes escalation deterministic.

Escalation Ladder: uncertainty rises -> authority contracts -> escalation increases



Example: pressure override (banking edge)

A caseworker wants to override a fraud hold to release $250k.

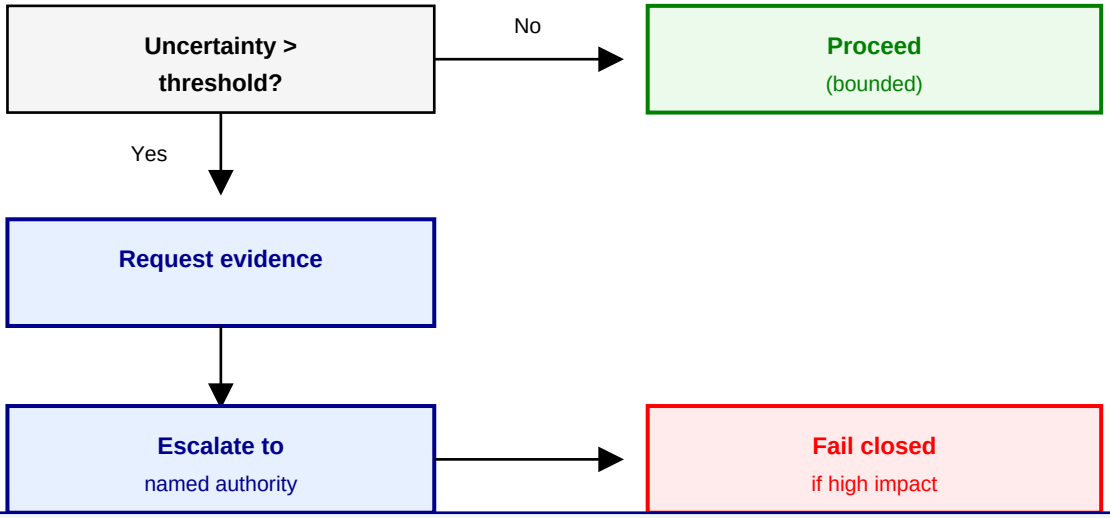
Missing step-up verification -> escalate to dual control.

If approved -> allow with bounds (scope + TTL + reversible).

Chapter 8 - Degradation protocol

Confidence is volatile. Degradation keeps you safe when inputs shift.

Degradation protocol: deterministic fallback under uncertainty



Example: cyber containment under thin corroboration

SOAR wants to isolate 400 endpoints.

Corroboration is thin -> allow with bounds (isolate 1 host) and wire escalation to incident commander.

Proof is minted before disruption.

Chapter 9 - Execution trace

Logs tell you what happened. A trace tells you why it happened, under which authority, and what evidence was used.

- **Intent:** objective + constraints.
- **Policy:** rules evaluated at runtime (versioned).
- **Evidence:** state snapshots, tickets, approvals, signals.
- **Plan:** explicit steps and expected outcomes.
- **Action record:** tool calls, parameters, results.
- **Reversal plan:** how to undo or mitigate.
- **Escalations:** who approved what, when.

Evidence Pack anatomy (portable proof object)

Authority contract (who can act, under what bounds).

Decision record (why this outcome was selected).

Enforcement receipt (what was actually applied).

Integrity (signed manifest, timestamps, hashes).

Example 1 - Platform rollout

An AI-assisted workflow wants to route 100% of checkout traffic to a new payment path.

So what?

Status quo: Feature flags and change tickets exist, but ramps are executed under pressure and bounds drift.

AI shift: Automation can flip traffic faster than humans can observe regression.

Judgement Spine impact: Execution-time bounds make shipping fast while keeping blast radius survivable.

Outcome (runtime)

ALLOW WITH BOUNDS: 1% canary, health checks mandatory, abort thresholds explicit, rollback armed.

If metrics regress, abort is permitted instantly without humans racing the incident.

The evidence pack includes the rollback anchor, policy versions, on-call path, and the exact bounds written into the rollout mechanism.

Example 2 - Cyber response

A playbook wants to isolate 400 endpoints based on anomaly signals.

So what?

Status quo: SOAR optimizes for speed; mass actions can cause self-inflicted downtime.

AI shift: Detection + automation compress the decision window to seconds.

Judgement Spine impact: Bounded containment prevents disruption by momentum and preserves escalation discipline.

Outcome (runtime)

ALLOW WITH BOUNDS: isolate 1 host as a controlled probe, shorten tokens, block org-wide lockout.

Escalation to incident commander is pre-wired with explicit widening thresholds.

The evidence pack captures signals relied upon, why scope was bounded, and what would have escalated.

Example 3 - Banking core payment

An $18M wire to a new beneficiary is about to settle.

So what?

Status quo: Screening and KYC checks exist, but ambiguity is often discovered after release.

AI shift: Payment operations accelerate; once settled, governance becomes remediation.

Judgement Spine impact: Sanctions ambiguity + missing dual control triggers a clean block before commit.

Outcome (runtime)

BLOCK: sanctions screen is fuzzy and dual control is not satisfied - hold with time-boxed dual approval.

Re-screening is triggered; release only occurs under a signed authority chain.

The regulator view lists sources, timestamps, screening outputs, approvals, and integrity proof.

Example 4 - Reliance governance

A model emits a high-risk score and an agent wants to lock an account.

So what?

Status quo: Organizations govern models, then rely on outputs informally in workflows.

AI shift: Probabilistic output becomes deterministic action through agents and automation.

Judgement Spine impact: Reliance itself is governed: who can rely, under what confidence, with what fallback.

Outcome (runtime)

ALLOW WITH BOUNDS: draft recommendation, attach provenance, require named human confirmation.

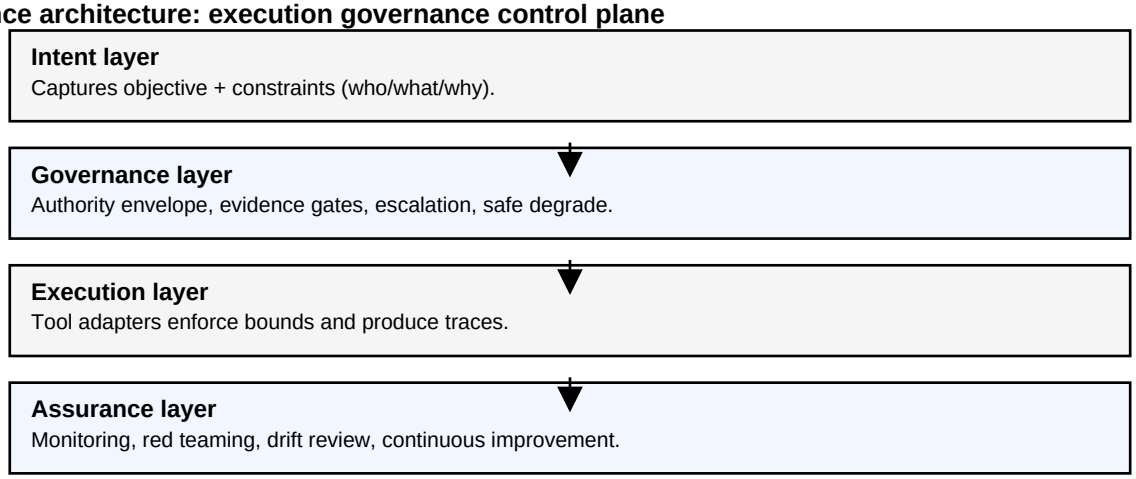
Disable auto-lockout; allow reversible mitigations; mint a dispute-ready evidence pack.

This is the premium, cross-cutting layer: model output -> reliance -> action.

Part III - The System

A doctrine becomes real when it is measurable: traces, architecture, and operating model.

Reference architecture: execution governance control plane



- Execution governance is a control plane that sits between the agent and the world.
- Agents operate through the plane. The plane enforces authority, proof, trace, escalation.
- Identity and privilege are where authority becomes enforceable.

Chapter 9 (continued) - Trace example

A trace is a portable explanation, not a pile of events.

Example trace (simplified)

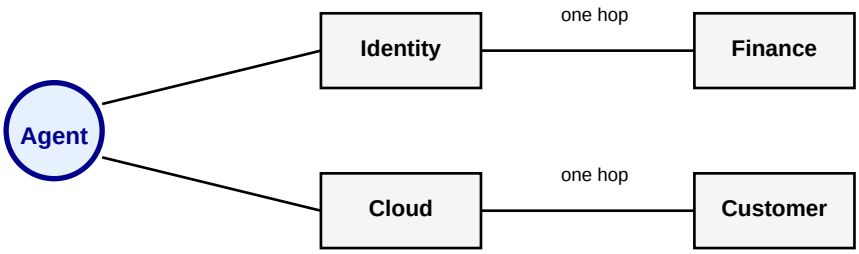
```
intent: action: "SOAR isolate endpoints" scope_requested: 400 requester:
"SOC Automation" mtm: id: "CYBER-MASS-CONTAIN" doctrine_version: "v1.3"
decision: "ALLOW_WITH_BOUNDS" bounds: isolate_scope: 1 token_ttl_minutes:
15 block_org_wide_lockout: true escalation: target: "Incident Commander"
widen_scope_requires: "corroboration >= 2 independent sources" integrity:
manifest_sha256: "<hash>"
```

The point is not the format. The point is that a reviewer can answer: who owned it, what was allowed, what was enforced, and why - without reconstruction.

Chapter 10 - Authority graph and exploitable execution paths

Risk is encoded in graphs: identities, privileges, groups, roles, tokens, and tool integrations.

- Map what an agent can do today given its privileges.
- Map what is one hop away (role assumption, group membership, token exchange).
- Find approvals that exist only as convention, not as enforced gates.
- Reduce exploitable execution paths systematically.



Execution governance extends attack-path thinking to legitimate tool chains that can produce illegitimate outcomes.

Chapter 12 - Operating model

Governance fails when ownership is vague. Assign roles explicitly.

Minimum viable operating model

- **Authority owner:** accountable for authority envelopes and risk tiers.
- **Execution owner:** accountable for how agents act and how traces are produced.
- **Escalation targets:** named humans for each risk tier and domain.
- **Assurance lead:** runs red team tests, monitors drift, manages incidents.

Weekly ritual (15 minutes)

Review top actions by impact, top escalations, and any authority drift incidents.

If you cannot reconstruct an action within 15 minutes, you are not enterprise-ready.

Part IV - Application

Start where authority and blast radius are highest.

- **Identity and privilege:** group membership, role changes, privileged tokens.
- **Production change:** feature flags, rollouts, config, CI/CD gates.
- **Finance and commitments:** payments, refunds, spend, terms.
- **Trust and safety:** lockouts, takedowns, identity state changes.

Where to start (fast path)

Pick one high-impact control surface.

Define the authority envelope on one page.

Enforce L0-L1 by default (observe + propose).

Mint evidence packs for every attempted action (allow, bound, escalate, block, degrade).

World packs (monetizable modules)

World packs extend the core platform with domain doctrine, MTM families, safe degrade patterns, and evidence templates.

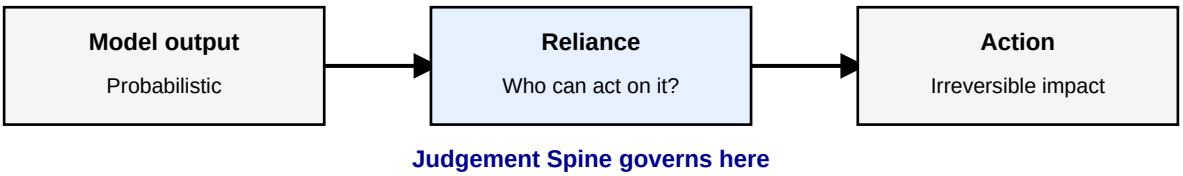
Product line	World pack
Platform Runtime Control	Platform Execution Governance
Platform Runtime Control	Agentic Execution Governance
Security and Integrity	Cyber Response Governance
Security and Integrity	Digital Identity and Representation Governance
Security and Integrity	Compliance Assurance Governance
Financial and Commitment Authority	Sales Authority Governance
Financial and Commitment Authority	Banking Edge Governance
Financial and Commitment Authority	Banking Core Payment Governance
Financial and Commitment Authority	Capital and Spend Authority Governance
Human and Physical Safety	Clinical Care Governance
Human and Physical Safety	Critical Infrastructure and Industrial Autonomy Governance
Operational Edge Governance	Retail Edge Governance
Operational Edge Governance	Retail Core Transaction Governance
Premium Layer	Reliance Governance Layer (Model-to-Action Boundary)

World packs do not change the platform. They extend it.

Premium cross-cutting layer: Reliance governance

This is not a vertical world. It governs the decisive AI boundary: model output -> reliance -> action.

Reliance boundary: model output -> reliance -> action



- **Human reliance:** drafted clause, clinical recommendation, risk score.
- **Automated reliance:** an agent executes directly from output.
- **Prompt and provenance governance:** who authored/approved prompts; guardrail overrides.
- **Multi-agent reliance chains:** Agent A relies, Agent B executes, Agent C approves fallback.

Part V - Implementation

The goal is not full autonomy. The goal is governed execution.

Level	What it looks like
M0 Ad hoc	Agents can act with minimal controls. Authority is implicit. Traces incomplete.
M1 Policy	Policies exist but are not executable. Approvals manual and inconsistent.
M2 Instrumented	Traces exist. Evidence gates enforced for some actions. Drift measured.
M3 Governed	Authority envelopes, escalation ladders, degradation protocols are standard. Paths reduced.
M4 Assured	Continuous red teaming, automated path reduction, audited governance across all actions.

Most organizations should target M3 as the near-term standard: governed authority everywhere, with deterministic escalation.

30-60-90 plan

Treat execution governance like identity security: map, reduce, enforce, monitor.

First 30 days - Constrain

- Name authority owners and escalation targets.
- Create the first authority envelope for one high-leverage surface (identity or production).
- Mandate an execution trace for any automated action.
- Default to L0-L1: observe and propose before executing.

Next 60 days - Enforce

- Add evidence gates for privileged actions (permission grants, deletion, payment release).
- Implement degradation protocol and measure triggers.
- Baseline exploitable execution paths and start path reduction work.

Next 90 days - Assure

- Run an agent abuse red team exercise focused on tool chaining.
- Automate checks for authority drift and privilege accretion.
- Publish a monthly benchmark report to stakeholders.

What good looks like

This is the moment you become safe to scale.

- We can state the authority envelope for every agent on one page.
- No high-impact action executes without evidence and a trace.
- Escalation targets are named and on-call.
- Uncertainty causes deterministic degradation (no improvisation).
- Privileges used by agents are time-bound and re-justified.
- We can reconstruct any action within 15 minutes, end-to-end.
- We run agent-focused red team tests quarterly.
- High-impact exploitable paths reduce month over month.

Fast path: first MTM in production (7 days)

Day 1: pick a commit boundary (rollout, isolation, payment release).

Day 2: define envelope + escalation ladder on one page.

Day 3: wire adapter hook at the last safe moment.

Day 4: enforce bounds + safe degrade; mint evidence pack.

Day 5: run a test scenario and replay the decision.

Day 6: run integrity verification; publish board/regulator views.

Day 7: go live in bounded mode (L1-L2) and review escalations.

Appendix - Template: Authority envelope charter

Use this to define the boundary between capability and permission. Keep it enforceable.

One-page charter

Agent name: _____

Business intent: _____

In-scope systems: _____

Out-of-scope systems: _____

Autonomous zone: low-risk, reversible actions only (define explicitly).

Evidence required: state snapshots, ticket ID, approval ID, etc.

Default ladder: L1 (propose) unless explicitly granted.

High-impact actions: require L3 or L4.

Reversal plan: how to undo actions.

Owners: Authority owner / Execution owner / Assurance lead.

Review cadence: weekly for 4 weeks, then monthly.

Appendix - Templates: Escalation matrix + degradation policy

Make escalation deterministic. Prevent improvisation under uncertainty.

Escalation matrix (example)

Action type	Risk tier	Default ladder	Escalation target
Add user to privileged group	High	L3	IAM On-call
Reset password (standard user)	Medium	L2	Service Desk Lead
Disable account	Medium	L2	SOC
Delete data / irreversible	Critical	L4	Data Owner

Degradation policy triggers (example)

- Confidence below threshold -> move from L2 to L1.
- Unknown system or new integration detected -> move to L1.
- Conflicting state evidence -> request evidence, then escalate.
- High-impact action requested -> require explicit plan + approval (L3).
- Repeated failures or abnormal responses -> fail closed and alert.

Appendix - Incident runbook (authority drift) + glossary

When an agent crosses its envelope, treat it like a security incident: contain, trace, learn, reduce the path.

- **Contain:** revoke agent tokens; lock tool adapters; switch to L0.
- **Trace:** pull the execution trace; identify intent, policy, evidence, actions executed.
- **Assess impact:** what changed, what persisted, what could be abused next.
- **Reduce path:** remove unnecessary privileges; add gates; adjust envelope and escalation.
- **Prevent recurrence:** add a red team test case; monitor for similar patterns.

Glossary (core terms)

- **Authority contract:** formal statement of what an agent can do, with conditions.
- **Evidence gate:** runtime check that blocks execution until required proof exists.
- **Intent capture:** turning a request into explicit objective + constraints.
- **Privilege accretion:** temporary access that becomes permanent via drift or convenience.
- **Safe speed:** operational acceleration without expanding risk or ambiguity.