CITADEL CLOUD MANAGEMENT

The Enterprise AI Agent Blueprint

How to deploy AI agents that clear a security review: workload identity per agent, three autonomy tiers, scoped ephemeral credentials, attributable audit trails, and a rollout sequence that earns trust one phase at a time.

Published by **Citadel Cloud Management** · citadelcloudmanagement.com · 2026 edition
The governance framework behind SIAS, Citadel's catalog of 425 governed AI agents across 8 domains.

CONTENTS

What's in this blueprint

01 Why agent projects stall in production

02 The governance model at a glance

03 Workload identity: one agent, one principal

04 Autonomy tiers: T1, T2, T3

05 Scoped, ephemeral credentials

06 The attributable audit trail

07 The phased rollout sequence

08 Before an agent touches money, people, or legal

09 The governance readiness checklist

10 Glossary and further reading

CHAPTER 1

Why agent projects stall in production

The demo works. The pilot works. Then the project sits in review for a quarter and quietly dies. The model was never the problem.

Here is the pattern we see across enterprise AI agent projects, regardless of industry or vendor. A team builds an agent — a support responder, a report generator, a code reviewer. It performs well in a sandbox. Stakeholders are impressed. Then someone asks the deployment question: *can this run against production systems with production data?* The request lands with the security team, and the security team asks five questions:

1. **Who is this agent?** Not "what does it do" — what identity does it run as? Can you distinguish its actions from every other workload's?
2. **What exactly can it access?** Which systems, which scopes, for how long, granted by whom?
3. **Who owns it?** If it misbehaves at 2am, whose phone rings?
4. **What did it do last Tuesday?** Show the log. Every action, attributed, with enough context to reverse it.
5. **Can we turn it off?** Right now, this minute, without redeploying anything?

Most agent projects cannot answer a single one of these. The agent runs under a shared service account that six other things also use. Its credentials are a static API key created eight months ago, scoped to everything, stored in a config file. There is no per-action log — just application logs that interleave the agent's activity with everything else on the box. Nobody is on the hook as owner. And the only kill switch is deleting the deployment.

So the security team does the only responsible thing available to them: they say no. Not because they are obstructing progress, but because approving an unaccountable, unauditable, unscoped autonomous system would be negligent. The project stalls, the sponsors lose interest, and the post-mortem blames "AI readiness" — when the actual gap was governance architecture that was never built.

The gap, in one table

Ungoverned versus governed agents — the difference a security review sees

Without governance	With governed agents
Shared service account	Per-agent workload identity
Broad standing permissions	Scoped, ephemeral, broker-issued credentials
No action log	Full, attributable audit trail
All-or-nothing autonomy	Tiered autonomy (T1 / T2 / T3), set per action
No named owner	Registry entry with a named human owner
Kill switch = redeploy	Revocation wired into the model gateway
Security team blocks it	Security team can approve it

For regulated industries — healthcare, financial services, government, energy — this is not optional hardening. It is the difference between an AI initiative that clears a SOC 2, ISO 27001, HIPAA, or FedRAMP-aligned review and one that never leaves the lab.

> THE CORE CLAIM OF THIS BLUEPRINT
>
> Governance is an **architecture property**, not a policy document. You cannot write a memo that makes an agent auditable. You build identity, credential scoping, tiered autonomy, and logging into the platform the agent runs on — and then any agent you deploy inherits all four. The rest of this document specifies each property and the order in which to build them.

CHAPTER 2

The governance model at a glance

A governed AI agent is an agent operated as a first-class principal. Four properties make that true.

Definition. A governed AI agent has its own attested workload identity, scoped and ephemeral credentials, an assigned autonomy tier, and a full, attributable audit trail of every action. In plain terms: the agent has a name the system recognizes, can only do what it is explicitly allowed to do, and every action it takes can be traced and reversed.

The four properties

1. **Identity** — each agent is its own principal with an attested workload identity (for example, SPIFFE/SPIRE), never a shared key. You always know *which* agent acted, and every identity has a named human owner. *Chapter 3*.
2. **Tiered autonomy** — how much the agent may do is set per agent, per action: T1 reads and reports, T2 writes under supervision, T3 drafts only behind named approvers. Autonomy is enforced by the runtime, not requested in the prompt. *Chapter 4*.
3. **Scoped, ephemeral credentials** — credentials are broker-issued, short-lived, and purpose-bound (OAuth token exchange, RFC 8693). There are no static, long-lived secrets to leak. *Chapter 5*.
4. **Full audit trail** — every action is logged, attributable to a specific agent identity, and designed to be reversible. Runtime detections watch for identity abuse, and a kill switch revokes an agent in seconds. *Chapter 6*.

How the properties reinforce each other

These are not four independent controls; each one depends on the previous. The audit trail is only *attributable* because every agent has its own identity — a shared service account makes attribution structurally impossible, no matter how much you log. Credentials can only be *scoped* because the identity registry records what each agent is allowed to touch. And autonomy tiers are only *enforceable* because the credential broker refuses to issue a write-capable token to a T1 agent. Pull one property out and the other three degrade into paperwork.

DESIGN RULE

Every enforcement decision lives in **policy, not prompt**. A prompt instruction ("do not modify records") is a suggestion to a language model. A policy decision point that declines to issue the credential is a guarantee. Whenever this blueprint says an agent "cannot" do something, it means the platform makes the action impossible — not that the agent was asked nicely.

One more framing note before the detail chapters. This model treats agents the way mature organizations already treat human employees: they are hired into a named role (registry entry), given badge access proportionate to that role (scoped credentials), supervised according to seniority (autonomy tiers), and subject to review (audit trail). None of these ideas is exotic. What is new is applying the full employment lifecycle to software that acts on its own initiative — and refusing to deploy software that will not submit to it.

CHAPTER 3

Workload identity: one agent, one principal

If you cannot say which agent acted, nothing downstream — scoping, auditing, revocation — can work. Identity is the control plane for everything else.

The problem with shared service accounts

The default failure mode in enterprise automation is the shared service account: one machine identity, one set of credentials, used by every job that needs to touch a system. It fails three ways, and agents make each failure worse.

- **Attribution is impossible.** When six workloads share one account, the audit log says the account acted — never which workload. With deterministic jobs that is an inconvenience; with agents that decide their own next action, it is disqualifying.
- **Blast radius is unbounded.** The shared account accumulates the union of every consumer's permissions. Compromise it — or let one agent misbehave under it — and you have exposed everything any of them could reach.
- **Rotation breaks everything at once.** Because rotating the shared key disrupts every consumer, rotation gets deferred, and the key quietly becomes a permanent, well-known secret.

The registry: agents as first-class principals

The fix is structural: every agent instance is its own principal, declared in a version-controlled registry. One entry per agent, reviewed like code, with a schema along these lines:

```
> identity/registry/invoice-triage-agent.yaml

id: invoice-triage-agent
owner: jane.doe@company.com        # a named human, not a team alias
risk_tier: T3                      # money path: draft-only, human-gated
description: Extracts and codes inbound invoices; drafts AP entries.
tools:
  - erp.invoices:read
  - erp.ap-drafts:write            # drafts only — posting is a human action
  - rag.search: [finance-policies, vendor-master]
prompt_pin: sha256:7f3c9a…        # the exact prompt version this identity runs
credential_ttl_ceiling: 15m
approvers: [ap-manager@company.com]
```

The registry gives you four things at once. A **name** for every agent that logs, policies, and dashboards can reference. A **named owner** accountable for its behavior. A **declared surface** — the complete list of tools and corpora it may touch, so a security reviewer reads one file instead of reverse-engineering the code. And a **pinned prompt**: the sha256 hash means a changed prompt is a changed diff in review, not a silent behavioral drift.

Attestation: proving the workload is who it claims

A registry entry is a claim; attestation makes it verifiable. Frameworks such as SPIFFE/SPIRE bind a running workload to its registry entry: the platform attests *where* the workload is running (node attestation) and *what* is running (workload attestation — image, service account, namespace), then issues a short-lived identity document only if both match the registration. An agent cannot borrow another agent's identity by copying a key, because there is no key to copy — identity is derived from the attested runtime, not from a secret in a file.

Binding agent identity to cloud IAM

Each major cloud has a native mechanism for federating workload identity into its IAM, so an attested agent exchanges its identity document for cloud permissions without any static cloud key existing anywhere:

Workload identity federation per cloud

Cloud	Mechanism	Effect
AWS	IRSA (IAM Roles for Service Accounts), IAM Roles Anywhere, STS token exchange	Pod identity maps to an IAM role; credentials are STS-issued and expire
Azure	Entra Workload ID, managed identities	Workload federates into Entra; no client secrets provisioned
GCP	Workload Identity Federation, service account impersonation	Attested workload impersonates a scoped service account with short-lived tokens

The lifecycle: joiner, mover, leaver — for machines

Human identity programs run a joiner-mover-leaver process: access is granted on hire, adjusted on role change, revoked on departure. Agents need the identical lifecycle, because agents multiply non-human identities faster than quarterly access reviews can catch. Concretely:

- **Joiner:** an agent exists only after its registry entry merges — creating the entry *is* the provisioning event.
- **Mover:** changing an agent's tools, tier, or corpora is a registry diff that goes through review, and the credential broker picks up the new scope on the next issuance.
- **Leaver:** retiring an agent deletes the registry entry, which invalidates attestation — the identity cannot be re-issued, and orphaned-credential sweeps confirm nothing survives it.

Two continuous jobs keep the estate honest: **discovery** (find non-human identities that exist in cloud IAM but not in the registry — shadow agents) and **ownership attribution** (every identity resolves to a current, named human; when the owner leaves the company, the agent's identity is flagged, not forgotten). Identity drift detection compares what the registry says an agent should be against what is actually deployed, and pages the owner on divergence.

> WHAT THIS CHAPTER BUYS YOU
>
> When the security review asks "who is this agent and who owns it?", the answer is one file: the registry entry, with an attested runtime binding and a named owner. That single artifact converts the review's hardest question into its easiest one.

CHAPTER 4

Autonomy tiers: T1, T2, T3

"How autonomous should the agent be?" is the wrong question. Autonomy is not a property of the agent — it is a property of each action the agent takes.

The three tiers

T1 — Autonomous

Read and report

The agent observes, analyzes, and recommends. It changes nothing. It can run unattended on a schedule because the worst case of a wrong answer is a wrong report a human reads — recoverable by definition. Examples: log analysis, security alert triage, spend analysis, research synthesis, weekly status reports.

T2 — Supervised

Writes to systems of record, within limits

The agent may write — but only within scoped credentials and, where the target system has operational risk, declared change windows. Every write is attributable to the agent's identity and designed to be reversible. Examples: CRM record updates, ticket routing and updates, knowledge base drafts published to staging, network configuration changes inside a change window.

T3 — Human-gated

Draft-only behind named approvers

Anything touching money, people, or legal. The agent prepares the artifact — the payment batch, the offer letter, the contract clause — and a named human approver reviews and executes. The agent never holds the credential that completes the action. Examples: accounts payable, payroll queries that lead to changes, hiring decisions, contract commitments, anything that moves funds.

THE MONEY-PEOPLE-LEGAL RULE

If an action moves money, affects a person's employment or record, or creates a legal obligation, it is T3 — regardless of how accurate the agent has been, how small the amount is, or how much the workflow owner wants full automation. This rule is not negotiable per project, and that is precisely what makes it useful: nobody has to re-litigate it under delivery pressure.

Assigning tiers: per action, not per agent

A single agent usually spans tiers. An invoice-processing agent *reads* the ERP (T1), *writes* draft journal entries (T2 mechanics, but on a money path — so gated), and *posts* nothing, ever, because posting is the human approver's act (T3 boundary). Assign a tier to each action the agent can take by asking three questions in order:

1. **Is it reversible?** If the action cannot be cleanly undone — sent emails, executed payments, deleted records without soft-delete — it cannot be autonomous. Irreversible actions start at T3 and argue their way down.
2. **What is the blast radius?** A wrong CRM note touches one record. A wrong firewall rule touches a datacenter. Same "write" verb, different tier: bounded and reversible writes can be T2; wide-blast-radius writes need change windows, canary scopes, or a human gate.
3. **Is there a regulatory surface?** Actions inside HIPAA, SOX, employment law, or export-control scope inherit the strictest treatment their framework implies, independent of technical risk.

Worked examples from a production catalog

Tier assignments Citadel applies across its 425-agent catalog

Domain	Policy
Security operations	Tier-1 alert triage runs at T1. Anything that isolates a host or disables an account requires a human gate — containment is high-blast-radius even when it is the right call.
Network operations	Config-writing agents are T2 with change-window enforcement: the credential is only issuable inside the approved window.
Finance	Anything that moves money is T3: draft-and-approve only, full audit trail, four-eyes enforcement on the approval itself.
Sales operations	CRM-writing agents are T2; pricing approvals stay human-owned regardless of deal size.
Data and analytics	SQL-generating agents run against read-only replicas with row-level security intact — T1 by construction, because the credential cannot write.
Industrial / OT	OT-adjacent agents are strictly read-only against historians and SCADA exports. Nothing writes to a control system, at any tier.
HR	Screening and review agents carry bias-audit requirements in their evaluation suites as a deployment precondition, and all decisions are T3.

Enforcement: the tier lives in the runtime

The tier is recorded in the agent's registry entry and enforced at three points, none of which is the prompt:

- **The credential broker** will not issue a write-capable token to a T1 identity, or any token outside a T2 agent's change window. The wrong action fails authentication, not judgment.
- **The orchestrator** inserts mandatory human-approval gates as edges in the execution graph for T2+ actions that require them. The agent's plan literally cannot route around the approval node, and step budgets cap how much work an agent does before a checkpoint.
- **The policy decision point** (Chapter 5) re-evaluates every tool call against the tier policy — per call, not per session — so a mid-session escalation attempt is denied at the moment it happens.

Graduation between tiers is evidence-based, and Chapter 7 describes the sequence: agents earn T2 scope by performing at T1 with measured accuracy, and the T3 boundary does not move at all. Autonomy is granted the way credit limits are — slowly, on the record, and revocably.

CHAPTER 5

Scoped, ephemeral credentials

The safest secret is one that does not exist. Every credential an agent holds should be issued moments before use, scoped to the task, and expired minutes later.

Why static keys and agents don't mix

A static API key in an agent's environment is a standing grant: everything the key allows, the agent can do at any time, forever, and so can anyone who obtains the key. Agents raise the stakes on all three clauses. They handle untrusted input (retrieved documents, user messages) that can attempt to steer them; they run continuously; and they multiply — a platform that starts with three agents will run fifty. Fifty long-lived, broadly-scoped keys is not a secret-management problem, it is an incident scheduled for later.

The replacement model has three properties, each doing distinct work:

- **Broker-issued:** agents never hold root credentials. A central broker — the only component trusted to mint credentials — issues them against the agent's attested identity and registry entry.
- **Short-lived:** every credential carries a TTL with a ceiling set per risk tier. Illustratively: minutes-scale for T2/T3 actions, an hour at most for T1 read scopes. A leaked token is a shrinking asset instead of a permanent one.
- **Purpose-bound:** the token encodes what it is for — this tool, this scope, this task — so a credential minted to read a knowledge base is cryptographically useless against the ERP.

Token exchange and the delegation chain

The standards machinery for this exists and is boring in the best way. OAuth 2.0 token exchange (RFC 8693) lets the platform trade an agent's identity token — plus, when acting for a user, that user's token — for a narrowly-scoped access token for one downstream system. Applied to agents, it produces a verifiable **delegation chain**: every action carries who asked (the human principal), who acted (the agent identity), and under what grant (the scoped token), all the way down the call graph.

Three constraints keep delegation from becoming a loophole:

1. **On-behalf-of semantics.** An agent acting for a user is capped at the *intersection* of its own grants and the user's entitlements. An agent can never do for a user what the user could not do themselves — this is the structural defense against the confused-deputy problem.
2. **Purpose and consent binding.** The grant records why the access exists and, where consent is required, points at the consent record. Access "because the integration needed it" stops being an acceptable

answer.

3. **Delegation depth limits.** Agents can call agents, and each hop dilutes accountability. A per-tier depth limit caps the chain, and the chain itself is recorded in the session ledger.

Authorization at the moment of the call

Scoping the credential is necessary but not sufficient, because the decision "may this agent make this call" depends on live context: the change window, the current tier policy, whether the kill switch fired two minutes ago. So tool-grant policies are evaluated at a **policy decision point** (OPA or Cedar are the common engines) on *every call, not at session start*. A session that begins legitimately and drifts — the tool-scope-creep pattern — hits a denial at the first out-of-policy call, and that denial is itself a logged, alertable event.

Governing the non-human identity estate

Agents create identities faster than quarterly reviews can catch, so the controls have to be continuous jobs, not calendar events:

Continuous NHI governance controls

Control	What it does
Discovery and inventory	Continuously reconciles cloud IAM (AWS, Azure, GCP) and secret-manager contents against the registry; anything unregistered is a finding.
Orphaned credential sweeps	Finds credentials whose owning agent was retired or whose human owner left, and revokes them.
Secret sprawl detection	Scans code, config, and CI logs for static secrets that should not exist at all under this model.
Rotation enforcement	For the residual secrets that must exist (e.g., third-party API keys), enforces rotation schedules instead of trusting them.
Least-privilege right-sizing	Compares granted scopes against scopes actually used and proposes shrinking the difference.
Blast-radius scoring	Ranks every identity by what its compromise would expose, so review effort follows risk.

THE ONE-SENTENCE TEST

Ask your team: "if this agent's credentials leaked right now, what would the attacker hold, and for how long?" Under this model the answer is: a purpose-bound token for one tool, expiring in minutes, revocable in seconds, and useless without the attested identity that requested it. If your current answer is "an API key to the ERP, valid until someone notices," this chapter is your gap.

CHAPTER 6

The attributable audit trail

"What did the agent do?" must have an exact answer — per action, attributed to one identity, with enough context to reverse it. Logging that can't answer that is telemetry, not audit.

What to log: the session ledger

The unit of record is the **session ledger**: an append-only log, one entry per agent action, written by the platform (the gateway and orchestrator), not by the agent itself — an agent that writes its own audit log can also decline to. Each entry captures:

Session ledger — fields per action

Field	Why it's there
Session ID and step number	Reconstructs the full trajectory: what the agent was trying to do, in order.
Agent identity (registry ID)	The attribution anchor — resolves to a registry entry and a named owner.
Human principal (when acting on-behalf-of)	Who asked; the top of the delegation chain.
Delegation chain and token grant	Under what authority, at what depth, with which scopes.
Tool called, parameters, target system	What was actually done, specifically enough to replay or reverse.
Model, prompt pin (sha256), token counts, cost	Which exact behavior version produced this, and what it cost.
Guardrail verdicts	What the input/output screens decided (pass, redact, block) and why.
Policy decision	The PDP's allow/deny for this call, with the policy version.
Result and reversal pointer	Outcome, plus the idempotency key or compensating action that undoes it.

Attribution: the chain that survives an audit

Attribution means a complete chain with no gaps: *this action* was taken by *this agent identity*, attested against *this registry entry*, owned by *this person*, under *this grant*, on behalf of *this principal*. Every link exists because of the previous chapters — which is why bolting logging onto an ungoverned platform doesn't work. If the agent runs under a shared account, the first link is broken and the chain never starts.

Reversibility is designed, not hoped for

"Designed to be reversible" is a build-time discipline with three concrete practices. **Idempotency keys** on every write, so a replayed or duplicated action cannot double-apply. A **replayable event log** per session, so

an investigation can reconstruct state at any step. **Compensating actions** declared per tool — a write a T2 agent may perform must have a documented undo, and Chapter 7 makes rehearsing that undo a gate for granting write access in the first place. Where a target system offers soft-delete or drafts, agents write those; hard state changes stay with humans.

Watching the watchers: identity threat detection

The audit trail is also a sensor. Runtime detections read the session ledger and gateway logs alongside cloud audit trails (CloudTrail, Entra sign-in logs, Cloud Audit Logs) for the identity-abuse patterns specific to agents:

- **Impersonation** — activity claiming an agent identity that attestation didn't issue.
- **Token theft and replay** — a credential used outside the session, workload, or geography it was minted for.
- **Confused deputy** — an agent exercising its own broader grants in service of a less-privileged request.
- **Privilege escalation and scope creep** — repeated PDP denials, or a tool-use profile drifting from the registry's declared surface.
- **Anomalous delegation** — chains that are deeper, broader, or more frequent than the agent's baseline.

When a detection fires, the response is already wired: a **kill switch** at the model gateway revokes the agent's identity, invalidates its outstanding tokens, and halts its sessions — in seconds, without a deploy. Because every agent's traffic passes through the gateway chokepoint, revocation is total, and because the agent is one principal among many, revoking it disturbs nothing else.

What the auditor will ask

Compliance frameworks were written for human actors, but their questions map cleanly onto this architecture. Access reviews (who can touch what) are answered by the registry plus PDP policies. Change management maps to T2 change windows and approval gates. Segregation of duties maps to T3 draft-only boundaries and four-eyes approval. Log integrity maps to the append-only, platform-written ledger. Under SOC 2 or ISO 27001, the evidence is not a screenshot scramble — it is a query against artifacts the platform produces as a side effect of running. Two hygiene rules keep the trail itself compliant: scrub PII from telemetry before export, and set retention windows per corpus sensitivity so the audit log never becomes its own data-protection liability.

CHAPTER 7

The phased rollout sequence

Start with agents that cannot hurt you, measure them, and let the evidence — not the roadmap pressure — decide when autonomy grows.

Phase 0 — Foundations before the first agent

Before any agent runs, four platform pieces must exist, however minimally: the **identity registry** (even as a reviewed YAML directory), a **gateway chokepoint** all model traffic routes through (this is what makes audit, cost attribution, and the kill switch possible in one place), a **credential path** that issues scoped short-lived tokens rather than static keys, and **one prepared corpus** — a document set with known ownership, access rules, and freshness. Also decided now, on paper: the tier definitions and the money-people-legal rule, agreed with security and the business owner *before* there is delivery pressure to bend them.

Phase 1 — The first T1 agent: read and report

Choose the first agent by risk, not by value. The ideal candidate reads a well-understood corpus and produces a recurring artifact humans already want: a weekly spend summary, alert triage notes, a compliance-evidence digest, a research brief. It writes to no system of record. Deliberately boring — that is the point. This phase proves the plumbing: identity attestation works, the ledger captures every action, the report lands on schedule, and a human confirms it is worth reading.

> GRADUATION CRITERIA — PHASE 1 → 2
>
> The agent has run unattended for a sustained period (weeks, not days) with zero identity or policy findings; its output is accurate against spot-checks; and someone would notice — and complain — if the report stopped arriving. Usefulness is a criterion: an accurate agent nobody reads should be retired, not scaled.

Phase 2 — Measure and harden

Before any agent earns write access, build the measurement machinery on the agents you already run: **golden evaluation sets** (curated question-answer pairs per corpus, versioned like code) that gate every prompt, model, or retrieval change in CI; **guardrails** on input (injection screening) and output (PII redaction, groundedness checks — does the answer actually follow from the retrieved sources); and **red-team suites** that attempt prompt injection and data exfiltration on every guardrail change and on a schedule. The

operating rule from here forward: no prompt, model, or retrieval change ships without passing the eval gate. Behavior changes become diffs with test results, exactly like code.

Phase 3 — The first T2 agent: supervised writes

Now, and only now, an agent writes to a system of record. Constrain the first write workflow four ways: a **narrow scope** (one system, one record type, the smallest useful grant); a **rehearsed reversal** — before the agent performs its first real write, the team executes the compensating action end-to-end and timestamps the drill (if you cannot undo it in rehearsal, the agent does not get the scope); **change windows** where the target system carries operational risk, enforced by the credential broker; and **human review of early writes**, sampled down as measured accuracy accumulates. Graduation to broader T2 scope repeats the same loop per new scope: evidence at current scope, rehearsed reversal, then the grant.

Phase 4 — Scale the catalog; hold the T3 line

With the pattern proven, scaling is mostly registry work: a new agent is a new registry entry binding a persona, tool grants, a tier, and corpora — reviewed and deployed without new platform code. Portfolio-level practices keep growth governed: every new agent enters at T1 in production regardless of how it tested; the NHI controls from Chapter 5 run continuously as the estate grows; and cost, quality, and guardrail-trip metrics are tracked per agent, with regression triggering demotion — autonomy is revocable, and demoting an agent to T1 must be a routine operational act, not an admission of failure. Meanwhile the T3 boundary holds: money, people, and legal workflows get draft-generating agents behind named approvers, and the approval itself gets four-eyes treatment. The correct end state is not "everything autonomous eventually" — it is every workflow at the tier its risk profile supports, with the evidence on file.

Anti-patterns

- **Starting with the flagship workflow.** The highest-value workflow is usually the highest-risk one. Start where failure is cheap; arrive at the flagship with evidence.
- **Autonomy by prompt.** "You must never modify records" in a system prompt is not a control. If the tier is not enforced by credentials and gates, it is not enforced.
- **The shared "AI service account."** Provisioning one identity "for the AI" recreates, on day one, the exact anti-pattern this blueprint exists to remove.
- **Skipping the eval gate under deadline.** An unmeasured behavior change to a production agent is an incident with a delay on it.

- **Granting T2 because T1 was "fine."** "No complaints" is not evidence. Graduation requires measured accuracy and a rehearsed reversal, on the record.

CHAPTER 8

Before an agent touches money, people, or legal

Twelve questions to answer in writing — with named owners — before any agent operates near a T3 surface, even in draft-only mode.

1. **What is the exact action boundary?** Which artifacts may the agent draft, and which acts (posting, sending, signing, paying) are reserved to humans? Written as a list of verbs, not a paragraph of intent.
2. **Who are the named approvers?** Individuals, not team aliases. Who covers absence? Is four-eyes required, and above what threshold?
3. **Can the agent's credential complete the action?** If the agent's token could technically post the payment or send the offer, the boundary is fiction. The completing credential must belong to the approver alone.
4. **What does the approver actually see?** A review surface showing the draft, the sources it was grounded in, and the confidence signals — or just an "approve" button that trains people to click yes?
5. **How is approval fatigue handled?** If one person approves 200 drafts a day, the gate has already failed. What volume triggers adding approvers or narrowing the agent's scope?
6. **What is the reversal path?** For each artifact type: if a bad draft is approved, what undoes the consequence, who executes the undo, and has it been rehearsed?
7. **What regulatory scope applies?** SOX, HIPAA, employment law, export control — and has the owning compliance function reviewed the workflow, not just been informed of it?
8. **What data does the agent see, and should it?** Payroll, health, and legal corpora are the most sensitive in the estate. Are ACLs enforced at retrieval time, per chunk — not just at the application door?
9. **What are the eval and bias requirements?** For anything touching people (screening, reviews, cases): what bias audits run in the evaluation suite, and are they a deployment precondition or a good intention?
10. **What does the audit trail show an examiner?** Can you produce, for any given draft: the sources, the prompt version, the guardrail verdicts, the approver, and the timestamp — in one query?
11. **What is the kill criterion?** Which specific signals (error rate, guardrail trips, a detection firing) trigger suspension, and is the suspension automatic or does it wait for a meeting?
12. **Who owns the outcome?** When an approved draft causes harm anyway, accountability sits with a named human owner and approver chain. If that sentence makes stakeholders uncomfortable, the workflow is not ready.

A NOTE ON HONESTY

If a vendor tells you their agent can safely automate money, people, or legal decisions end-to-end, they are describing their risk appetite, not yours. The draft-behind-approver pattern is not a temporary limitation to be engineered away — it is the correct permanent design for actions whose failure modes land on humans.

CHAPTER 9

The governance readiness checklist

Print this page. When every box is checked for a given agent, it is ready for the security review — and likely to pass it.

Identity

- [ ] The agent has its own registry entry with a named human owner
- [ ] Identity is attested at runtime (workload attestation, not a shared key)
- [ ] The prompt version is pinned (sha256) and changes require review
- [ ] Cloud access uses workload identity federation — zero static cloud keys
- [ ] Discovery jobs reconcile cloud IAM against the registry on a schedule

Autonomy

- [ ] Every action the agent can take has an assigned tier (T1/T2/T3)
- [ ] The money-people-legal rule is applied with no project-level exceptions
- [ ] T2 writes have declared change windows where operationally relevant
- [ ] T3 actions are draft-only; the completing credential belongs to the approver
- [ ] Human-approval gates are enforced by the orchestrator, not the prompt

Credentials

- [ ] All credentials are broker-issued, short-lived, and purpose-bound

- [ ] TTL ceilings are set per risk tier
- [ ] On-behalf-of actions are capped at the user's own entitlements
- [ ] Tool calls are authorized at a policy decision point on every call
- [ ] Delegation depth is limited and the chain is recorded
- [ ] Orphaned-credential sweeps and rotation enforcement run continuously

Audit

- [ ] Every action lands in an append-only session ledger written by the platform
- [ ] Each entry resolves to: agent identity, owner, grant, principal, tool, result
- [ ] Writes carry idempotency keys and a declared compensating action
- [ ] The reversal path has been rehearsed and timestamped before first write
- [ ] ITDR detections (impersonation, replay, confused deputy, escalation) are live
- [ ] The kill switch revokes identity and tokens at the gateway in seconds
- [ ] PII is scrubbed from telemetry before export; retention windows are set

Process

- [ ] Golden eval sets exist and gate every prompt/model/retrieval change in CI
- [ ] Red-team suites run on guardrail changes and on a schedule
- [ ] The agent entered production at T1 regardless of test-environment results

- [ ] Graduation to T2 cited measured accuracy, not absence of complaints
- [ ] Per-agent cost, quality, and guardrail-trip metrics are tracked; demotion is routine

CHAPTER 10

Glossary and further reading

Terms used in this blueprint

Term	Meaning
Governed agent	An AI agent operated as a first-class principal: own identity, scoped ephemeral credentials, assigned autonomy tier, attributable audit trail.
Workload identity	A verifiable identity issued to a running workload based on attestation of where and what it is — not on possession of a secret.
SPIFFE / SPIRE	An open standard (and its reference implementation) for issuing attested workload identities.
Registry entry	The version-controlled declaration of an agent: owner, risk tier, tool grants, corpora, pinned prompt.
T1 / T2 / T3	Autonomy tiers: autonomous read-and-report; supervised writes within scoped credentials and change windows; human-gated draft-only for money, people, and legal.
Token exchange (RFC 8693)	The OAuth 2.0 mechanism for trading one token for a narrower, purpose-bound one — the basis of agent delegation chains.
Policy decision point (PDP)	The engine (e.g., OPA, Cedar) that evaluates whether a specific call is allowed, per call, against current policy.
Confused deputy	A privileged intermediary tricked into using its own authority on behalf of a less-privileged requester.
NHI	Non-human identity: the machine identities (agents included) that now outnumber human accounts in most estates.
Session ledger	The append-only, platform-written log of every agent action, with attribution and reversal pointers.
ITDR	Identity threat detection and response — runtime detection of impersonation, token replay, escalation, and anomalous delegation.
Kill switch	Gateway-level revocation of an agent's identity, tokens, and sessions, effective in seconds without a deploy.

Term	Meaning
Golden set	A versioned, curated evaluation dataset that gates behavioral changes in CI.

Where this framework comes from

This blueprint is the governance model behind **SIAS**, Citadel Cloud Management's catalog of 425 governed AI agents across 8 business domains — every one engineered around the four properties in this document, grounded in your own corpora, in your tenancy, with the autonomy tier set per agent, per action. Citadel runs the same model internally: the agents we sell are the agents that operate our own business, under the same registry, the same tiers, and the same ledger.

The natural next step

If you are planning a deployment, the highest-leverage 30 minutes is a scoping conversation: pick the highest-value, lowest-risk first agent — usually a T1 read-and-report agent against a corpus you already trust — and map it to the Phase 1 criteria in Chapter 7.

- Book a free discovery call: citadelcloudmanagement.com/book-a-call
- Explore the SIAS agent catalog: citadelcloudmanagement.com/ai-agents
- Background reading: What are governed AI agents? · Enterprise AI security