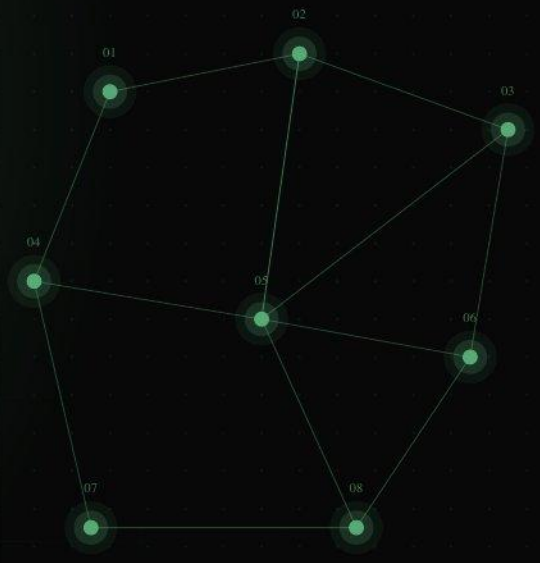
natoma



ENTERPRISE FRAMEWORK · 2026

8 Decisions Every Enterprise Must Make Before Deploying AI Agents at Scale

A practical governance framework for CIOs, CISOs, and AI steering committees. Grounded in real enterprise conversations. Protocol-agnostic. Immediately actionable.

THE FRAMEWORK

Eight decisions. *Not optional.*

Your engineers are already connecting AI agents to enterprise systems. Claude Code queries production databases. ChatGPT pulls CRM data. Cursor touches cloud infrastructure. The productivity is real. So is the risk.

Agents inherit user access, act without real-time oversight, and are configured in files that never reach an IT inventory. These eight decisions are what every AI steering committee must make before adoption reaches critical mass.

DECISION 01 ◆

Agent Inventory

Do you know what is running in your environment?

DECISION 02 ◆

Approved Registry

What is sanctioned versus what is shadow?

DECISION 03 ◆

Identity Model

How are agents authenticated?

DECISION 04 ◆

Authorization Model

Who can access what, with which tools?

DECISION 05 ◆

Audit Trail

Can you explain every action an agent took?

DECISION 06 ◆

Data Policy

What data can agents access, and what must be blocked?

DECISION 07 ◆

Lifecycle Management

How are agents provisioned, updated, retired?

DECISION 08 ◆

Incident Response

What happens when something goes wrong?

WHERE TO START

Inventory. You cannot govern what you cannot see. Everything else follows.

Agent Inventory

Do you know what is running in your environment?

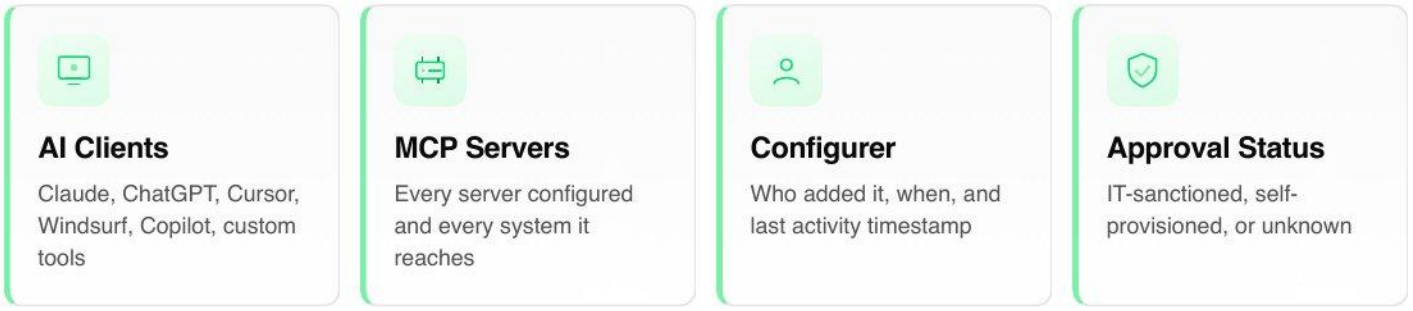
> THE QUESTION YOUR BOARD WILL ASK
>
> How many AI agents are connected to our systems, and who authorized them?

WHY THIS IS DECISION 1

You cannot govern what you cannot see. Every other decision depends on a complete view of AI agent activity. MCP servers live in local config files (`~/.cursor/mcp.json` , `claude_desktop_config.json`), do not generate network signatures, and do not appear in SaaS management platforms. Traditional discovery tools miss them.

What your inventory must capture:

FIG. 1.1 — WHAT YOUR INVENTORY MUST CAPTURE



- Every AI client in use (ChatGPT, Claude, Cursor, Windsurf, Copilot, custom)
- Every MCP server configured per client, and the systems it reaches
- Which user configured each connection, and when it was last active
- Whether each connection was IT-approved or self-provisioned

How to build it:

- **Endpoint scanning** via existing EDR or MDM (CrowdStrike, Jamf, Intune) to read MCP config files from managed devices. No new agent deployment.
- **Network analysis** to detect MCP traffic patterns between AI clients and internal APIs.
- **Config monitoring** to track when new MCP servers are added, modified, or shared.

COMMON MISTAKES TO AVOID

- Waiting until sprawl justifies the investment. By then, it is unmanageable.
- Relying on self-reporting. Engineers do not file tickets to add an MCP server.
- Inventorying only known connections. The risk lives in what you do not know.

Approved Registry

What is sanctioned versus what is shadow?

> THE QUESTION YOUR SECURITY TEAM WILL ASK
>
> Which of these agents are approved, and what is our process for evaluating new ones?

WHY THIS MATTERS

Not all agents carry the same risk. A read-only connection to public docs is not the same as a write-enabled connection to your production database. Your registry classifies every connection into one of three tiers, backed by technical enforcement, not policy alone.

FIG. 2.1 — THE THREE-TIER REGISTRY MODEL

The approval workflow: A user requests a new MCP server (or one is discovered). Security evaluates data sensitivity, scope, author trust, and compliance implications. Decision: approve, deny, or approve with restrictions. Target turnaround: 48 hours for standard requests.

COMMON MISTAKES TO AVOID

- Making approval too slow. If governed takes three weeks and shadow takes five minutes, shadow wins.
- Applying the same review rigor to every server. Tier by data sensitivity.
- Building the registry without enforcement. Policy documents do not block connections.

Identity Model

How are agents authenticated?

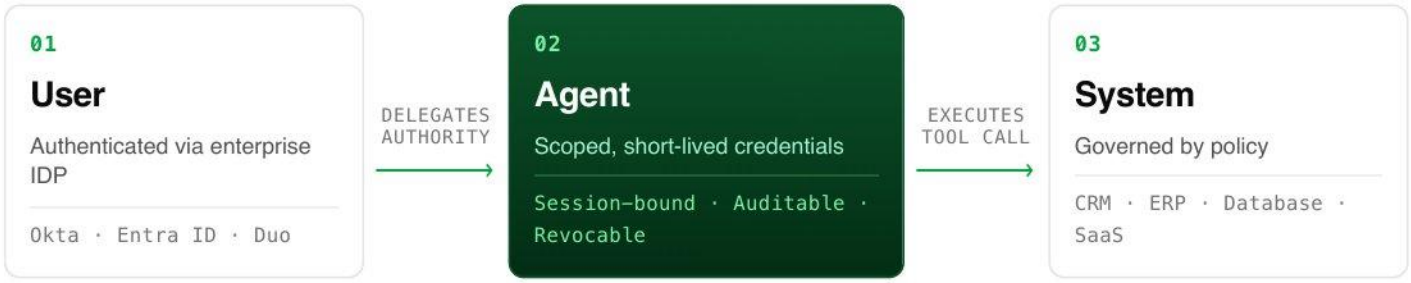
> THE QUESTION YOUR IDENTITY ARCHITECT WILL ASK
>
> When an agent makes a tool call on behalf of an employee, how do we know who the user is, which agent is acting, and whether it is authorized?

WHY THIS IS HARD

Traditional identity was built for humans logging into applications. Agents act on behalf of a user but are not the user. They inherit access, behave non-deterministically, and can run while the user is offline.

Key decisions:

FIG. 3.1 — THE IDENTITY CHAIN FROM USER TO ENTERPRISE SYSTEM



Every link in this chain must be traceable. An auditor asking "who authorized this tool call?" should trace the call back to the MCP server, back to the agent, back to the user, back to the authentication event.

- **Credentials.** Short-lived, scoped tokens tied to a specific agent session. Never static API keys. Never shared credentials in config files.
- **On-behalf-of.** The agent inherits the user's access, nothing more. Token-level enforcement, not configuration-level trust.
- **Heterogeneous systems.** OAuth token exchange for modern apps. User-consented credential mapping for legacy SSO gaps.
- **Autonomous agents.** Treat as non-human identities with their own credentials, permissions, and audit trail. Authorize more restrictively than human-delegated agents.

COMMON MISTAKES TO AVOID

- Sharing browser-session credentials with the agent. Agent credentials must be separately issued and scoped.
- Storing long-lived API keys in MCP config files. Every key in a file is a key that can leak.
- Treating all agents the same regardless of autonomy level.

Authorization Model

Who can access what, with which tools, under what conditions?

> THE QUESTION YOUR ENTERPRISE ARCHITECT WILL ASK
>
> How do I make sure the sales team's assistant can pull their pipeline but cannot export the entire customer database?

WHY API-LEVEL AUTHORIZATION IS NOT ENOUGH

Agent authorization must operate at the tool-call level: this specific operation, these parameters, this user, right now. Access to a CRM server does not equal access to every CRM operation.

FIG. 4.1 — AUTHORIZATION LAYERS, FROM BROADEST SCOPE TO MOST SPECIFIC

LAYER 1	**Agent-level**	Which agents can run at all. Registered in the approved inventory.
LAYER 2	**Server-level**	Which MCP servers each agent can connect to.
LAYER 3	**Tool-level**	Which specific operations within a server are allowed.
LAYER 4	**Data-level**	What rows, fields, or records the operation can return.
LAYER 5	**Context-level**	Under what conditions (time, risk score, user state) the operation is allowed.

Policy example:

```
Engineering agents:
  - Read production databases: allow
  - Write to production: block (requires human approval)
  - Write to staging: allow

Sales agents:
  - CRM access: own accounts + team pipeline only
  - Bulk export: max 100 records per session
  - Contact PII: masked unless manager role

Autonomous agents:
  - All write operations: require human approval
  - Data access: scoped to the workflow
```

Policy engines. OPA (flexibility, complex logic) or Cedar (speed, formal verification). Some enterprises use both.

COMMON MISTAKES TO AVOID

- Granting blanket MCP-server access and trusting the app's native permissions. Native permissions were built for humans, not agents at machine speed.
- Putting authorization in the AI app layer. Every new agent then needs new auth code. Put it in the infrastructure.
- Ignoring context. "Can access financial data" is not the same as "can access financial data via an autonomous agent at 3am while the user is offline."

Audit Trail

Can you explain every action an agent took?

> THE QUESTION YOUR COMPLIANCE OFFICER WILL ASK
>
> If a regulator requests every AI interaction with customer data in the last 90 days, can you produce it?

What to capture for every tool call: timestamp, user, AI client, agent identity, MCP server, tool called, parameters, result, policy evaluation, duration.

Logging allowed actions is half the picture. The complete audit record captures three things:

FIG. 5.1 — THE THREE-PART AUDIT RECORD

01

What the agent **did**

Successful tool calls. Parameters. Results. Every call, not sampled.

02

What it **tried** to do

Tool calls denied by policy. The reason for denial. Attempted parameters.

03

What it did **instead**

How the agent adapted after denial. Retries. Alternatives. Or gave up.

Security buyers evaluate governance by this three-part record. If your system captures only successful actions, you cannot demonstrate that your policies are working.

Retention and integration. Retain per your compliance framework (typically 1 to 7 years). Deliver to your SIEM (Splunk, Sentinel, Sumo Logic) in structured format. Enable real-time streaming for monitoring and historical query for investigations.

COMMON MISTAKES TO AVOID

- Logging only at the API level. Agent audit trails require tool-call granularity.
- Not logging denials. Denials prove your policies are working.
- Storing logs in a format your SIEM cannot ingest. If security cannot query alongside other telemetry, the trail is invisible.

Data Policy

What data can flow through the AI layer?

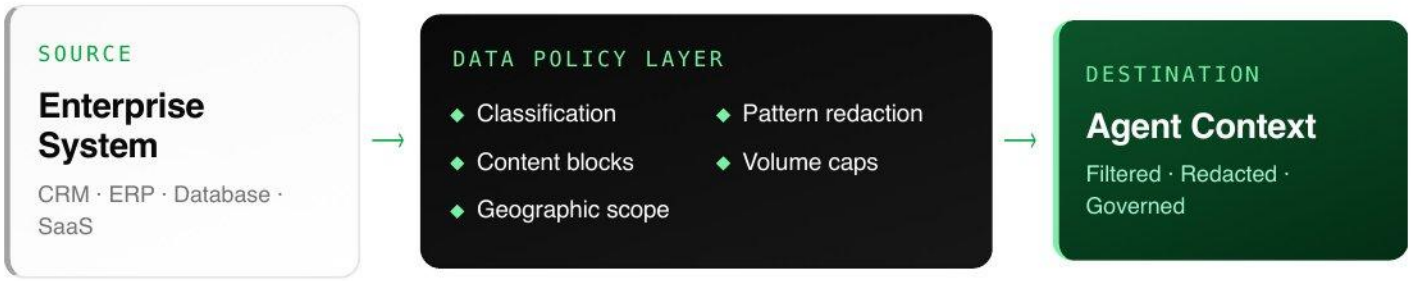
> **THE QUESTION YOUR CISO WILL ASK**
>
> If an agent pulls customer emails into a prompt, does that violate our data classification policy?

WHY AI AGENTS CREATE A NEW CLASS OF DATA RISK

When an agent accesses data, that data flows through an LLM context window, potentially gets logged, cached, or influences responses to other users. Data policy for agents is not only "who can access what," but "what data can flow through the AI layer at all."

The data policy layers:

FIG. 6.1 — HOW DATA FLOWS THROUGH THE AI LAYER



- **Classification.** Confidential or Restricted data cannot be passed to agents. Honor your existing Purview, Google DLP, or Varonis labels.
- **Pattern redaction.** SSNs, credit cards, API keys, passwords stripped automatically before reaching the model.
- **Content blocks.** Medical records, compensation data, legal privilege materials blocked entirely.
- **Volume controls.** Caps on records returned per session to prevent bulk extraction.
- **Geographic scope.** EU data cannot route through US-hosted models. PHI never leaves your VPC without a BAA.

COMMON MISTAKES TO AVOID

- Relying on the target system's access controls. Those govern access. Data policy governs what happens after retrieval.
- Ignoring the LLM context window. Data in a prompt may persist in conversation history, provider logs, or future responses.
- Building AI data policies separate from your existing classification. Inconsistency breaks governance.

Lifecycle Management

Who owns each agent, and when is it retired?

THE QUESTION YOUR IT GOVERNANCE LEAD WILL ASK

When someone creates an agent, who is responsible for it? When it stops being used, who decommissions it?

WHY THIS MATTERS

Agents get created, shared, abandoned. Without lifecycle governance, you recreate the orphan-resource problem cloud created a decade ago — resources nobody owns, permissions nobody reviews, costs nobody tracks.

Ownership and accountability: Every agent needs a named owner, not a team. The owner is accountable for behavior, scope, and quarterly review. Security evaluates new requests. IT governance sets standards. The AI steering committee approves high-sensitivity use cases.

FIG. 7.1 — THE AGENT LIFECYCLE

Review cycles back into the lifecycle. Every agent gets re-evaluated.

COMMON MISTAKES TO AVOID

- Not requiring an owner. Agents without owners become orphans. Orphans are the #1 source of governance gaps.
- Treating provisioning as one-and-done. Permissions must be reviewed periodically, like human access.
- Making decommissioning optional. Disabled agents left in the registry create reactivation risk.

Incident Response

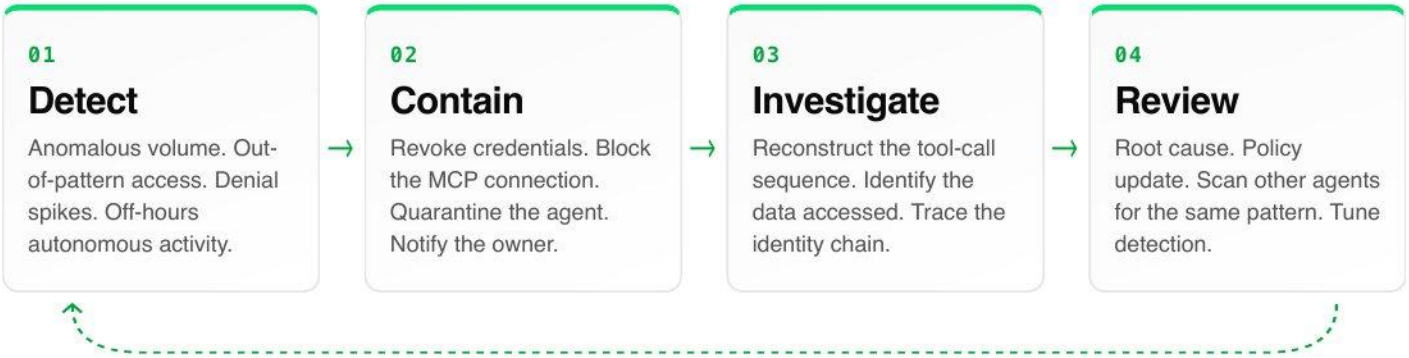
What happens when something goes wrong?

> **THE QUESTION YOUR SOC WILL ASK**
>
> If an agent accesses data it should not have, how quickly can we detect it, contain it, and explain it?

WHY THIS IS DIFFERENT FROM TRADITIONAL IR

Agent incidents are a new category. An agent might pull a bulk query at 3am that is technically allowed but contextually wrong. It might be manipulated by prompt injection. It might act autonomously while the delegating user is offline. Your existing IR playbook does not cover these patterns.

FIG. 8.1 — THE INCIDENT RESPONSE CYCLE FOR AI AGENTS



COMMON MISTAKES TO AVOID

- Leaving agent incidents out of the IR playbook. The SOC will misclassify or miss them.
- Relying on LLM-provider logs. They capture prompts, not tool-call activity, policy evaluations, or enterprise system interactions.
- Treating every anomaly as an incident. Agents are non-deterministic. Detection needs calibration.

SUMMARY

Putting It All Together

How the 8 decisions compose into a working governance program.

These eight decisions are not sequential gates. They are concurrent workstreams that reinforce each other. Inventory enables the registry. The registry informs identity. Identity powers authorization. Authorization generates the audit trail. The audit trail feeds incident response.

The maturity path:

FIG. S.1 — THE GOVERNANCE MATURITY PATH

Most enterprises start with visibility. Discovery is the lowest-friction entry point, and the inventory it produces is the prerequisite for everything else.

Start with your inventory. The rest follows.

Published by Natoma. Natoma connects any AI to every enterprise application, database, and tool, securely. One platform, every environment, full governance.

For a conversation about how this framework applies to your specific environment: natoma.ai/book-a-demo

natoma

WHERE TO START

Start with your inventory. The rest follows.

Natoma's discovery scan connects to your existing EDR and MDM systems and produces a complete MCP server inventory in hours, not weeks. From there, the other seven decisions become configurable, not greenfield projects.

Book a discovery scan natoma.ai

Natoma connects any AI to every enterprise application, database, and tool, securely. One platform. Every environment. Full governance.