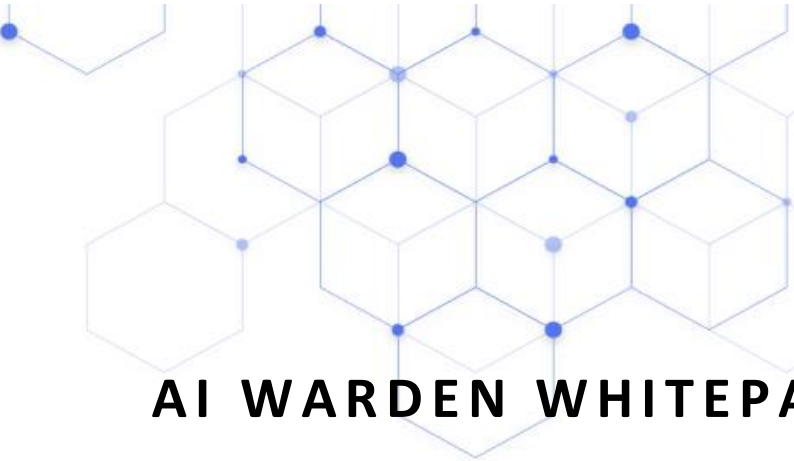




AI WARDEN WHITEPAPER

ENTERPRISE ARCHITECTURE

Governed Agentic Enterprise Architecture

Secure, compliant AI adoption starts with an architecture and governance model, whether enterprises build it themselves, use an enterprise governance platform, or combine both.

AT A GLANCE

Do not centralize all data.

Systems of record stay distributed. Governance intent, evidence, and standards must be coherent.

Federate enforcement.

Policy becomes real at LLM gateways, MCP gateways, retrieval services, workflow brokers, and SaaS hooks.

Govern tools, not just prompts.

The LLM call is only one risk. Tool execution, identity delegation, memory, and state mutation are often more dangerous.

Restrict opaque agents.

If an embedded agent cannot be centrally governed, constrain its data classes, action types, workloads, geographies, and exception expiry.

Contents

Executive summary	3
1. Problem statement	4
2. Architecture principle.....	5
3. Architecture concept.....	5
4. Governance boundary.....	6
5. Reference flow	7
6. Identity and policy	8
Policy decision semantics.....	8
7. Threat model	8
Design requirements	9
Failure modes and resilience.....	10
8. Decision framework for application-embedded agents	10
Embedded-agent maturity model.....	11
Risk-based limits for opaque embedded agents	12
Anti-patterns to avoid	12
9. Recommended operating model	13
10. Where AI Warden fits	14
11. First 30 days: practical next steps	14
Closing position	15

Executive summary

Every major enterprise application and data platform is adding embedded agentic capabilities. Salesforce Agentforce, ServiceNow AI Agents, Atlassian Rovo, SAP Joule, Microsoft Copilot and Copilot Studio agents, Workday AI capabilities, Snowflake Cortex Agents, developer copilots, productivity copilots, and domain-specific agents are all examples of the same pattern: vendors are placing agent runtimes directly beside the data, workflows, and users they already control.

The default path is attractive. Enable the agent where the data already lives. Let the business move quickly. Accept each vendor's guardrails, audit logs, model choices, and permission model.

That path is also how enterprises accidentally create a fragmented AI estate.

The highest-value agents do not stay inside one application. They cross customer data, contracts, tickets, policies, knowledge bases, transaction systems, analytics, approvals, communications, and operational workflows. If every application ships its own isolated agent, the enterprise gets many narrow assistants instead of a coherent agentic capability. Worse, each assistant becomes a separate governance domain with its own runtime, tool model, prompt controls, model routes, audit format, data-retention policy, and exception process.

The architectural answer is not to move all data into one AI platform. Systems of record will remain distributed. The answer is to separate where enterprise data lives from where agentic reasoning, tool execution, model routing, policy, approval, and evidence are governed.

CORE THESIS

The enterprise needs a defensible agentic AI architecture before embedded agents spread across every application. The required model is common governance standards, defined enforcement points, identity-aware execution, and evidence by design. The architecture should stand on its own, independent of any single product or vendor.

Centralize governance intent

- Inventory, policy authoring, risk tiering, approvals, and identity mapping
- Evidence schema, operational governance, and exception management

Distribute enforcement

- LLM gateways, MCP gateways, connector proxies, retrieval services, and workflow brokers
- SaaS policy hooks and embedded-agent adapters where vendors support them

This does not mean every application-embedded agent is forbidden. It means embedded agents should only be enabled broadly when they can meet enterprise governance requirements: central inventory, identity propagation, policy enforcement, data-flow control, model-routing control or equivalent assurance, human approval, kill switches, and standardized evidence.

The strategic question is not which product to buy first. It is what architecture and governance model will prevent agentic AI from becoming a collection of vendor-specific runtimes, policies, and audit trails.

Terminology: this paper uses **application-embedded agents** to mean vendor-provided agentic capabilities built directly into enterprise software, cloud services, productivity platforms, developer tools, data platforms, and workflow systems. "SaaS-native" is a narrower term and is avoided because the pattern is broader than SaaS alone.

1. Problem statement

Agentic AI changes the business architecture of enterprise software. Traditional application features help users perform tasks inside one system. Agents are valuable because they can reason across context, retrieve data, call tools, take actions, generate content, and chain steps across systems. The value is in orchestration, not just assistance.

Consider the difference between an agent that summarizes a Salesforce opportunity and an agent that reviews account history, inspects open ServiceNow incidents, checks contract terms in a document repository, queries consumption and billing data in Snowflake, drafts a renewal strategy, opens an approval, updates the CRM, and prepares a customer email.

The first pattern is local productivity. The second pattern changes business outcomes. It is also exactly where application-specific agents struggle, because the work crosses product boundaries and authority boundaries.

Domain	Fragmented application-embedded agent model	Governed enterprise-agent model
Inventory	Agents discovered per platform, often after enablement	Central inventory with owner, purpose, risk tier, tools, models, and data classes
Policy	Different controls per application	Common policy vocabulary with platform-specific adapters
Identity	Vendor-specific interpretation of user permissions	Enterprise identity, delegated authorization, and explicit acting authority
Tool execution	Broad app-level capabilities or opaque vendor actions	Narrow tools with authorization at invocation and execution time
Data flow	Hard to prove what data reached which model	Gateway, retrieval, redaction, classification, residency, and egress controls
Model risk	Model choice hidden, vendor-bound, or tenant-configured inconsistently	Approved model catalog, routing policy, fallback rules, and evaluation
Audit	Inconsistent logs, retention, and event semantics	Standardized evidence records across agent, model, tool, policy, and approval events
Operations	Many admin consoles and exception paths	One governance plane with federated enforcement and incident response

THE AUDIT TEST

A regulated enterprise must be able to answer who accessed sensitive data, which identity was used, which model processed the data, which policy allowed or blocked the flow, whether data was minimized, whether state was mutated, and whether the flow can be disabled, replayed, or investigated.

If the answer is "it depends which platform ran the agent", the organization does not have a defensible historized view when auditability, retention, model-risk oversight, EU AI Act obligations, GDPR, DORA, sector guidance, or contractual evidence requirements apply.

The issue is not that every AI feature is automatically a legally defined high-risk AI system. The issue is that regulated organizations, and any company that needs auditability, must be able to produce a consistent history of access, data movement, decisions, approvals, and actions across systems.

2. Architecture principle

Keep systems of record distributed; keep governance intent centralized; enforce controls close to execution.

The enterprise should not try to move all data, workflows, or runtime execution into one AI product. Existing systems of record and systems of engagement will remain distributed across enterprise applications, SaaS suites, data platforms, internal APIs, file repositories, collaboration tools, and custom applications.

But the governance of agentic execution must be coherent. That requires a common architecture model, shared governance standards, and physically distributed enforcement points. A central control plane is one strong implementation pattern, but the principle is broader than any product or deployment model.

DESIGN PRINCIPLE

Agents may run in multiple places, but they must be governed through common inventory, policy, identity, approval, routing, and evidence standards. Embedded agents that cannot meet those standards are separate AI control surfaces, not harmless product features.

No product or internal platform can realistically build and maintain deep native integrations for every SaaS platform, application, data product, developer tool, and workflow system that adds agentic features. The practical governance model is twofold: integrate deeply where enforcement is possible, and restrict opaque embedded agents by data classification, workload criticality, action risk, user population, geography, and evidence requirements where enforcement is not possible.

3. Architecture concept

The core architecture is simple: centralize governance intent, enforce close to execution, and preserve evidence without copying every sensitive payload.

Operational consideration: The architecture should also define who operates and maintains these planes in production. This includes platform monitoring, lifecycle management, integration maintenance, controlled updates, and support processes.

Control plane	Data plane	Evidence plane
Manages governance intent and configuration: registries, policy, risk tiers, approvals, model catalog, identity mapping, kill switches, rollout controls, and evidence schemas.	Executes the work: invocation, planning, retrieval, model calls, tool calls, streaming, redaction, validation, rate limits, cost budgets, timeouts, and step limits.	Records what happened for security, compliance, audit, legal, and operations. It supports metadata-only events, redacted payloads, hashes, retention classes, access control, and tamper evidence.

Governance should be logically consistent even when enforcement is distributed. That does not mean every enforcement decision must call one central service synchronously. Policy can be compiled, cached, and distributed to enforcement points, provided versioning, expiry, revocation, and audit semantics are preserved.

4. Governance boundary

The goal is not to connect every embedded agent on day one. The goal is to define which agentic activity is inside the enterprise governance boundary, which activity is outside it, and what limits apply to anything outside.

In practice, establishing this boundary starts with visibility. An AI flow assessment can identify existing AI traffic, model integrations, MCP servers, and agentic workloads and distinguish governed paths from shadow or insufficiently controlled AI.

This is the architecture conversation behind the Salesforce, Snowflake, and other vendor-agent question. Some platforms can participate in governed execution. Some can partially comply. Some remain opaque. Each category needs a different operating model.

Inside the governed boundary <i>Full enforcement and evidence</i>	Enterprise agent runtime: Registered agents, approved tools, governed prompts, runtime limits LLM gateway: Model routing, redaction, inspection, residency, quota, and audit MCP and tool gateway: Tool allow-lists, identity binding, schema validation, action receipts
Partially inside the boundary <i>Some controls, restricted use</i>	Vendor agent using enterprise LLM gateway: Model flow is governed, but tool behavior or runtime state may be vendor-controlled Platform with exportable audit logs: Visibility exists, but enforcement may be after-the-fact SaaS agent with approval hooks: High-impact actions can pause, but retrieval and prompts may remain opaque

Outside the governed boundary*Opaque or vendor-controlled***Standalone embedded agent:** Uses its own model, tools, memory, and policy interpretation**Opaque SaaS automation:** No standard evidence, no policy hook, no enterprise model routing**Unapproved agent platform:** Limited to low-risk data and draft/read-only use, or prohibited**GOVERNANCE BOUNDARY RULE**

Inside the boundary, agents may handle sensitive data and high-impact workflows according to policy. Partially inside, they need compensating controls and workload limits. Outside the boundary, they should be restricted to low-risk data, low-criticality work, read-only or draft-only assistance, and time-bound exceptions.

5. Reference flow

The reference flow shows how governed agentic execution should move from request to response or action, with control-plane rules applied at the points where risk enters: invocation, retrieval, model use, tool execution, approval, and evidence capture.

Figure 1. Reference flow for governed agentic execution

CONTROL PLANE	Inventory and risk tier · Policy and approval · Model, tool, and data rules
DATA PLANE	User or workflow → Approved channel → Agent runtime → Retrieval gateway LLM gateway Tool gateway
EXTERNAL SYSTEMS	Documents and data platforms · Approved model providers · Enterprise apps, APIs, and workflows
EVIDENCE PLANE	Invocation and identity · Policy, model, retrieval, and tool records · Approval and action receipts

Each step carries identity, purpose, policy version, data classification, correlation ID, and outcome metadata into the evidence store.

The LLM gateway governs model interaction. It does not, by itself, govern agent behavior. Tool execution, retrieval, memory, approval, and runtime constraints require separate enforcement.

MCP and tool access must be treated as privileged execution surfaces. A governed tool layer should enforce tool allow-lists, schema validation, identity binding, row-level and field-level authorization, rate limits, tool-result scanning, server authentication, network egress limits, action receipts, and idempotency keys.

Retrieval is not a passive search problem. It is a policy enforcement point covering classification-aware search, purpose-bound retrieval, residency constraints, row-level and field-level filtering, context minimization, sensitive span redaction, embedding governance, and retrieval event logging.

6. Identity and policy

Agent activity must be attributable. "The agent did it" is not an acceptable audit answer.

Authority mode	Meaning	Required controls
User-delegated	Agent acts with the invoking user's permissions	User-bound token, tool-time authorization, least privilege, audit to user and agent
Service-principal	Agent acts as an application or workload identity	Narrow scopes, owner approval, credential rotation, purpose binding, stronger monitoring
Approved elevation	Agent performs an action above normal user privilege after approval	Approval record, expiry, policy recheck, action receipt, elevated audit
Break-glass	Exceptional privileged action during incident or emergency	Time-bound access, dual control, alerting, post-action review
Impersonation	System pretends to be a user	Avoid by default; if unavoidable, require explicit risk acceptance and strong evidence

Authorization should be evaluated at invocation time, retrieval time, model time, tool-call time, approval time, execution time, and post-action time. Long-lived credentials and broad service principals are common failure modes. Agents should use short-lived, scoped credentials wherever possible.

Policy decision semantics

Governance fails when every platform invents its own words for the same decision. The enterprise should standardize policy outcomes: allow, flag, redact, require approval, escalate, and block.

DECISION PRECEDENCE

block > require approval > escalate > redact > flag > allow

Each decision should include policy ID and version, enforcement point, subject identity and authority mode, agent ID and version, evaluated resource/tool/model/data class, requested action, decision, reason code, timestamp, and correlation ID.

7. Threat model

Agentic AI adds a privileged reasoning and execution layer across enterprise systems. The threat model must cover more than prompt content.

Threat	Example	Required controls
Prompt injection	Malicious ticket text instructs the agent to ignore policy and exfiltrate records	Tool-result scanning, instruction hierarchy, content isolation, retrieval provenance
Confused deputy	Agent uses a service principal to access data the user cannot see	User-bound authorization at retrieval and tool-call time

Threat	Example	Required controls
Tool overreach	Generic database or browser tool can reach unrestricted systems	Narrow tools, allow-lists, egress controls, schema validation
Tool poisoning	Untrusted MCP server advertises misleading or dangerous tool behavior	Trusted registry, server authentication, signed metadata, approval before enablement
Data leakage	PII or regulated data sent to an unapproved model or region	Classification, redaction, model-routing policy, residency constraints
Action abuse	Agent sends customer communications, grants access, or initiates payment incorrectly	High-impact action approval, dry-run/propose/commit modes, action receipts
Runaway autonomy	Agent loops through costly model and tool calls	Step limits, cost budgets, timeouts, concurrency limits, kill switches
Audit tampering	Operator or compromised runtime deletes traces of agent behavior	Append-only logs, tamper evidence, separation of duties
Vendor opacity	Embedded agent hides prompts, model route, tools, or retention behavior	Minimum-control criteria, exception process, restricted risk tier
Memory contamination	Sensitive or malicious content persists in agent memory	Memory classification, retention controls, sanitization, user separation
Partial execution	Agent updates one system but fails before updating another	Idempotency keys, correlation IDs, saga patterns, compensation workflows

Design requirements

1. Central inventory

Know which agents exist, who owns them, what they access, which tools and models they use, where they run, and which business process they support.

2. Explicit runtime boundaries

Define request scope, maximum steps, time, cost, concurrency, tool allow-lists, memory retention, network egress, code execution, retries, and kill switches.

3. Identity propagation

Attribute each flow to a user, system principal, approved elevation, or break-glass authority with authorization checks at every critical stage.

4. Common policy decisions

Use a common vocabulary and precedence model so compliance teams can compare risk and enforce standards.

5. Enterprise-controlled model routing

Use enterprise LLM gateway routing or demonstrably equivalent controls; otherwise treat the capability as a higher-risk exception.

6. Tool-level authorization

Use narrowly scoped tools with schema validation, policy enforcement, identity binding, and action receipts.

7. Data minimization

Govern classified data before retrieval, model invocation, tool execution, and logging.

8. Human approval

Require review for payments, account changes, employee decisions, customer communications, access grants, legal submissions, regulatory filings, and production changes.

9. Evidence by design

Produce evidence without reconstructing logs from multiple vendor consoles.

10. Safe failure modes

High-risk flows fail closed when policy, identity, approval, or audit services are unavailable.

Failure modes and resilience

Agentic systems are distributed systems with probabilistic components. The architecture must define failure behavior before production rollout.

Failure	Required behavior
Policy service unavailable	High-risk actions fail closed; cached policy may be used only within expiry for approved low-risk flows
Identity service unavailable	No privileged action; user-delegated flows fail unless valid short-lived credentials already exist
Audit sink unavailable	High-risk actions fail closed or write to durable local queue; no silent success
LLM gateway unavailable	Route only through approved fallback; otherwise fail with explicit reason
Model provider failover	Preserve residency, retention, and model-risk constraints; no automatic unsafe region fallback
Tool connector timeout	Do not infer success; record unknown state and require reconciliation
Partial workflow execution	Use correlation IDs, idempotency keys, action receipts, and compensation where possible
Streaming violation detected	Interrupt stream, record partial delivery, and notify according to policy
Approval expires	Re-request approval and re-evaluate policy before execution
Stale retrieved data	Mark provenance and timestamp; revalidate before high-impact action

Do not convert control-plane failure into data-plane permission. "The policy service was down, so we allowed it" is not a governance strategy.

8. Decision framework for application-embedded agents

Application-embedded agents are not automatically wrong. Many will be valuable. But they should be assessed using an enterprise architecture decision framework.

Question	Required answer before broad enablement
Can we centrally inventory the agent, owner, purpose, risk tier, tools, and model route?	Yes
Can the agent preserve enterprise IAM, RBAC, and delegated authorization?	Yes
Can tool calls and actions be authorized at execution time?	Yes
Can model calls route through an enterprise LLM gateway or equivalent control?	Yes, or explicit risk exception
Can we control what data reaches the model?	Yes
Can logs flow into the standard evidence model?	Yes
Can high-risk actions require approval?	Yes
Can the agent be disabled centrally during an incident?	Yes
Can compliance teams understand the evidence without vendor-specific interpretation?	Yes
Can the vendor commit to retention, training, residency, and incident-response terms?	Yes

Embedded-agent maturity model

Level	Capability	Suitable use
0	No visibility, no control, opaque model route	Prohibit or explicit risk acceptance only
1	Admin enablement and coarse audit	Low-risk productivity pilots
2	Enterprise identity and permission preservation	Low-risk workflows with limited data access
3	Exportable logs into central evidence store	Governed use with monitoring
4	Policy hooks for tool/action approval	Controlled business workflows
5	Customer-controlled model routing or gateway integration	Regulated data and higher-risk use cases
6	Full integration with enterprise control plane and enforcement standards	Strategic enterprise agent runtime

Risk-based limits for opaque embedded agents

When an embedded agent cannot provide central visibility, policy hooks, model-routing control, or standard evidence, the enterprise should define where it may and may not be used.

Constraint	Safer allowance	Restricted or prohibited use
Data classification	Public, internal, low-sensitivity knowledge	Confidential, regulated, personal, customer, patient, employee, trading, source-code, or crown-jewel data
Workload criticality	Drafting, summarization, search assistance, low-impact productivity	Revenue-impacting, safety-impacting, operationally critical, legal, regulatory, financial, or production workflows
Action type	Read-only or draft-only assistance	State mutation, external communication, access grants, payments, submissions, or irreversible actions
Identity model	User-visible activity with preserved permissions	Broad service-principal access, unclear delegation, impersonation, or hidden elevation
Evidence quality	Exportable logs with timestamp, actor, action, and object	Opaque prompts, hidden model route, incomplete action logs, or vendor-only evidence
Geographic and legal scope	Regions and business units with compatible controls	Jurisdictions with strict residency, sectoral, labor, financial, health, or public-sector obligations

KEY DESIGN CONSIDERATION

If an embedded agent cannot be governed centrally, do not pretend it is governed. Limit what data it can touch, what actions it can take, which workloads it can support, and how long the exception lasts.

Anti-patterns to avoid

An anti-pattern is a tempting design choice that looks efficient at first but creates governance, security, or operational failure later. These are the patterns most likely to fragment the enterprise AI estate.

"The data is in the platform, so the agent should run there"

Data locality is not the same as business value or governance maturity.

"Application-specific agents are our AI strategy"

Local assistants should not define enterprise AI architecture.

"The LLM gateway solves agent governance"

The LLM gateway is necessary but insufficient. Agents retrieve data, use memory, call tools, mutate state, schedule work, communicate externally, and retry failed operations.

"Each vendor's guardrails are good enough"

Vendor guardrails are not a substitute for enterprise-wide policy, evidence, and accountability.

"Audit can be solved later"

Late audit is incomplete. Overbroad audit becomes a sensitive-data landfill.

"Pilots do not need architecture"

Pilots become production by accident. The first ten agents set patterns the next hundred will copy.

"A central platform means one giant box"

The goal is common control, not architectural fantasy. Expect federated runtimes and distributed enforcement.

9. Recommended operating model

The operating model should not be a committee that blocks delivery. It should be a paved road: approved runtimes, reusable connectors, standard policies, evidence templates, risk tiers, clear exceptions, and fast review paths.

For organizations that do not want to build the complete operational capability internally, a specialized service partner can provide that paved road: onboarding and integration, platform monitoring and support, coordinated updates and maintenance, and secure enablement of new agents and use cases.

Capability	Typical owner
Agent runtime standards	CIO / Enterprise Architecture / Platform Engineering
Control-plane architecture	Enterprise Architecture / Platform Engineering / Security Architecture
Model risk, acceptable use, and AI policy	Risk, compliance, legal, responsible AI
LLM gateway and model routing controls	Security Architecture / Platform Engineering
MCP, tool, and connector governance	Platform Engineering / Application Owners / Security Architecture
Agent design and business value	Business product owners
Data access and classification	Data Owners / Application Owners / Privacy
Approval policy for high-impact actions	Business Owners / Risk / Compliance
Monitoring and incident response	Security Operations / AI Operations
Evidence and audit readiness	Compliance / Internal Audit

For embedded-agent exceptions, document the business justification, data classes exposed, model and provider posture, identity and permission behavior, audit limitations, retention and training terms, compensating controls, expiry date, and accountable owner.

10. Where AI Warden fits

The architecture and governance model in this paper stands on its own. Enterprises may implement it with internal platforms, existing security infrastructure, specialist products, or a combination of all three. AI Warden is designed to support that model by providing enforcement and evidence capabilities where agentic AI moves from experimentation to governed operation.

- LLM gateway controls for policy enforcement, routing, scanning, redaction, quotas, and audit
- MCP gateway controls for governing agent access to tools and enterprise systems
- Agent and tool registries for inventory, ownership, approval, and operational visibility
- Policy decisions across allow, flag, redact, block, and approval-oriented workflows
- Identity-aware request logging for users, system principals, and delegated flows
- Approval and evidence workflows for governed AI operations
- Audit surfaces for security, compliance, platform, and operations teams

BOUNDARY

AI Warden should not be framed as a universal adapter for every embedded agent in every SaaS product. That would be bad architecture and an impossible delivery promise.

Its role is to provide strong inline control where traffic can be routed through governed LLM, MCP, tool, and connector paths, and to help enterprises define risk-based acceptance criteria for embedded agents that remain outside those paths. It complements IAM, SIEM, GRC platforms, API gateways, data catalogs, DLP tools, workflow systems, and systems of record. The point is not that every enterprise must buy a single product. The point is that every enterprise needs a coherent architecture for agent governance.

Where an operations partner fits: The control platform still needs to be securely integrated, operated, and evolved within the enterprise environment. Cyway complements AI Warden with Swiss-based expert operations covering onboarding and integration, continuous monitoring and support, coordinated platform lifecycle management, and secure enablement of new agentic use cases.

11. First 30 days: practical next steps

1. **Inventory agentic exposure.** Identify embedded agents, copilots, assistants, plugins, automations, MCP servers, model integrations, and planned agent features.
2. **Define the enterprise governance model.** Decide where agents, tools, models, policies, approvals, evidence, and exceptions are registered and governed, whether through an internal platform, AI Warden, existing security infrastructure, or another implementation.

3. **Map enforcement points.** Identify where controls run inline today: LLM gateways, API gateways, MCP gateways, retrieval services, SaaS hooks, workflow brokers, and logging pipelines.
4. **Classify agent actions by risk.** Separate low-risk assistance from high-impact business execution, external communication, access changes, financial movement, legal action, regulatory work, and production changes.
5. **Define policy semantics.** Standardize allow, flag, redact, require approval, escalate, and block decisions, including precedence and required evidence fields.
6. **Create embedded-agent acceptance criteria.** Do not enable agentic features unless they meet minimum controls or receive explicit risk acceptance.
7. **Define limits for opaque embedded agents.** Specify allowed data classifications, workload criticality, action types, user groups, geographies, and expiry dates for agents that cannot be centrally governed.
8. **Pilot one governed cross-system workflow.** Choose a valuable process that uses retrieval, model routing, tool execution, approval, and evidence.
9. **Validate in the customer's environment.** A focused discovery workshop can bring the platform to the enterprise environment, connect an initial provider, MCP server, and policy, and use the resulting evidence to define the prioritized production rollout.
10. **Test failure modes.** Simulate policy outage, audit outage, connector timeout, model failover, approval expiry, and partial execution.
11. **Produce evidence.** Confirm security, compliance, and audit teams can answer who did what, with which data, using which model, under which policy, and with what outcome.

Closing position

Agentic AI will not arrive as one enterprise platform. It will arrive embedded in every enterprise application, SaaS product, workflow tool, development environment, and data platform.

That is exactly why enterprises need an architecture.

The winning strategy is not to reject vendor innovation. It is to prevent every enterprise software platform from becoming a separate AI strategy and governance domain.

DECISION QUESTION

Are we adopting an AI capability, or are we accidentally creating another ungoverned AI control plane?

For organizations that want to move from architecture to controlled production, Cyway and AI Warden can jointly assess the current AI footprint, identify governance gaps, define the target architecture, validate it in the customer's environment, and establish a governed path into production and ongoing operations.

This whitepaper has been jointly created by Cyway and AI Warden on 28.09.2026