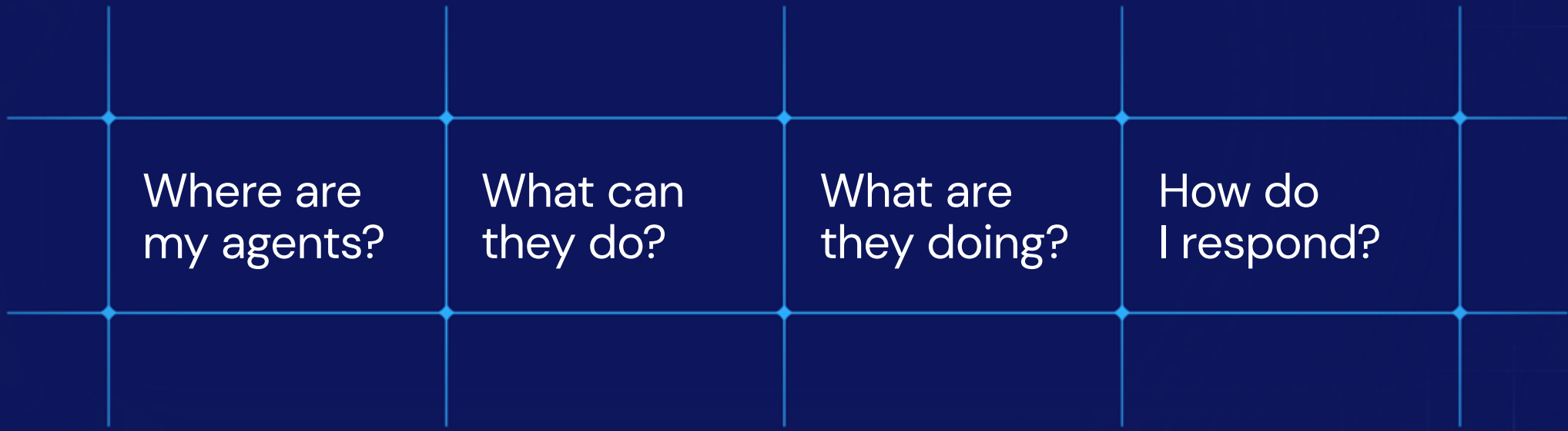
Whitepaper

Governing Agentic Execution

An architectural blueprint for the secure agentic enterprise



Authored by the founding members of the Blueprint Alliance
A cross-industry reference architecture for CISOs, CIOs, CTOs, and Enterprise Security Architects

Blueprint Alliance

Table of Contents

Executive Summary

The traditional enterprise security perimeter is being challenged in the face of AI Agents. Experimental LLM chatbots have given way to autonomous, multi-step agentic workflows in barely two years. Software used to do what it was told; now it decides, acts, and connects on its own. That autonomy offers real business value, but introduces a governance challenge most security teams are not yet equipped to manage.

Gartner predicts that by 2028, an average global Fortune 500 enterprise will have over 150,000 agents in use, up from less than 15 in 2025, generating significant agent sprawl, IT complexity and management challenges. These agents don't just display data. They reason, route, call downstream APIs, spawn sub-agents, and execute multi-step operations across numerous enterprise systems.

Without a unified architectural standard, organizations face a widening **identity, security, and governance gap:** shadow agents multiply, credentials get inherited across trust boundaries, and execution goes uncontained. The Blueprint Alliance Reference Architecture is an open, multi-vendor framework, published for open community use, that shifts security from the network perimeter to a unified, zero-trust, agentic control plane, giving technology leaders a concrete way to scale agentic deployments without losing control.

Introduction

For decades, enterprise security relied on a clear separation between human identities (managed via IAM, MFA, SSO) and machine identities (managed via static service accounts, API keys, or cloud-based compute services). Autonomous AI agents collapse this dichotomy. When an AI agent acts on behalf of a human employee, inherits access privileges, queries vector databases, and triggers financial transactions, it becomes an active security principal and operational actor requiring explicit identity, access, and governance controls.

Security cannot be retrofitted at the model or network layer alone. It has to be anchored in an agentic control plane that continuously governs visibility, connectivity, and execution.

Traditional Software (Deterministic)	Agentic AI (Non-Deterministic)
Executes explicit, step-by-step code paths.	Receives high-level human intent and formulates multi-step execution plans.
Relies on static, predefined permissions and service accounts.	Dynamically selects APIs and applications, leveraging token vaults, credential brokers, and delegated sessions.
Limited strictly to hardcoded integration paths.	Acts autonomously, utilizes tools via Model Context Protocol (MCP), and spawns sub-agents.
Operates bound within authenticated user sessions.	Operates asynchronously, often long after the human delegator has logged off.

The Governance Gap:

Five Questions Every Board Is Asking

AI agents are multiplying faster than most security teams can keep track of. Gartner predicts that by 2028, an average global Fortune 500 enterprise will have over 150,000 agents in use, up from less than 15 in 2025, generating significant agent sprawl, IT complexity and management challenges. Only 13% of organizations think they have the right AI agent governance in place. That gap is why cybersecurity and compliance committees keep coming back to the same five questions.

1. Where are our AI agents, and who owns them?
2. How do we restrict what agents can execute and stay on task?
3. Can we audit and explain autonomous decision-making after the fact?
4. What is our data exposure and third-party risk across the agent supply chain?
5. Do we have a verified kill switch and rollback plan if an agent goes rogue?

These five questions map onto the four pillars of the Blueprint reference architecture in Section 3. Discovery and identity registration answer question one, Access Management answers question two, Runtime Authorization and Monitoring answers questions three and four, and Active Containment answers question five. Answering these five questions with evidence, not just assurances, lets a CISO safely accelerate agent adoption while matching control depth to regulatory and operational exposure. Those who cannot remain vulnerable to significant and increasing security incidents.

These questions serve as a diagnostic tool to evaluate risk rather than a uniform checklist. Security requirements naturally vary by operational scope, as a read-only productivity assistant does not need the same containment controls as an agent executing financial transactions. Diagnosing which risks actually apply allows security teams to match controls to real-world exposure. This keeps organizations from over-engineering low-risk deployments and slipping into multi-month approval delays, while ensuring high-risk agents receive appropriate operational boundaries.

Blueprint Principles

The Blueprint Alliance framework extends zero trust to agents: treat them as first-class identities, not as code, credentials, or an afterthought bolted onto the human identity stack. These six principles hold up everything that follows.

Every hosted agent is a distinct security identity, not an afterthought. Agents get provisioned, authenticated, and de-provisioned with the same rigor as employees and service accounts. All enterprise-hosted agents, including pilots and internal tools, should operate within an appropriate governance framework, with control depth proportionate to their risk and operational scope. Local runtimes operate under specialized endpoint containment rather than formal identity-directory registration.

Access is scoped to each task, not standing. While task-scoped dynamic grants represent the aspirational goal, practical implementations pair short-lived task grants with resilient baseline permissions to balance overhead.

Delegation is traceable end-to-end. When a human delegates to an agent and that agent spawns a sub-agent, the chain of accountability must survive each hop. Every action should trace back to the human or process that authorized it. While end-to-end multi-hop delegation remains an aspirational target given today's tooling, this blueprint defines the reference trajectory as standards mature.

Agent Runtime is isolated and monitored, not just provisioned. Access decisions made at setup time are necessary, not sufficient. The blueprint requires continuous observation across execution runs to establish behavioral baselines, not just what it was granted.

Containment is instant and reversible. Every agent, regardless of vendor or platform, needs a targeted kill switch that can suspend, quarantine, or terminate it in real time, with a clear way back. Pairing continuous behavioral monitoring with a structural execution boundary closes the gap between 'authorized' and 'safe' by containing the blast radius the moment an agent deviates from its baseline.

Governance adapts at the velocity of AI. Static security policies cannot keep pace with rapidly evolving agentic capabilities and governance has to catch what policies alone can't anticipate. This blueprint operates as an adaptable architecture that scales alongside emerging agent behaviors, pairing static permission ceilings with dynamic real-time scoping, allowing organizations to maintain hard guardrails while adapting access to context.

Together, these six principles extend zero trust to a new class of identities: agents that operate constantly, persistently, and autonomously across the enterprise. Get them right, and the identity, security, and governance gap closes.

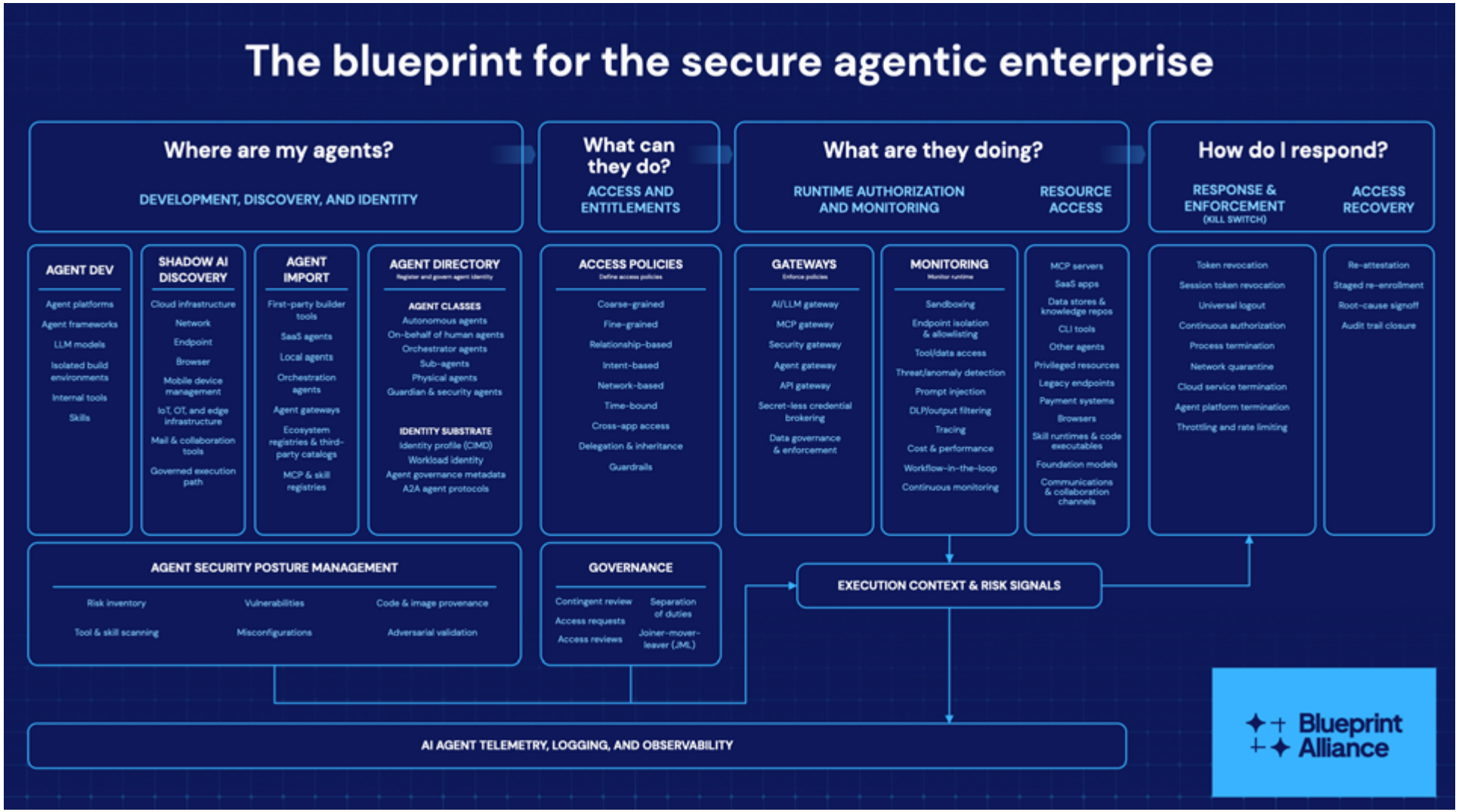
Blueprint Reference Architecture

The Blueprint Reference Architecture grounds agentic enterprise security in four operational questions required to govern autonomous execution, each mapping to a pillar of the reference architecture:

1. Where are my agents?
2. What can they do?
3. What are they doing?
4. How do I respond?

Spanning all of these questions is the need for a strong foundation of execution context and risk signals, fed continuously by runtime telemetry, logging, and observability. Together, these act as the connective tissue that enables the system to see risk patterns, and respond in concert.

Blueprint Reference Architecture



3.1 Pillar 1:

Where are my agents? (Development, Discovery & Identity)

The foundational challenge of securing the agentic enterprise is visibility. Organizations cannot govern or protect what they do not know exists. This pillar establishes a comprehensive tracking capability across all corporate environments to identify, catalog, and evaluate every AI agent traversing the corporate infrastructure, whether sanctioned or unmanaged. This pillar concludes by registering validated agents as governed identities with accountable owners, producing the authoritative inventory that Pillar 2 binds access policies to.

3.1.1 AI Agent Discovery

AI Agent Discovery serves as the continuous detection mechanism for the enterprise agent footprint. It maps the complete lifecycle of agents from internal engineering pipelines to external third-party integrations, ensuring that no active model or autonomous process operates in a blind spot.

3.1.1.1 Agent Development

This component focuses on internal engineering environments where proprietary agents are actively built, trained, and tuned. Securing the development pipeline helps ensure security-by-design before an agent ever interacts with production data.

- **Agent platforms:** Enterprise-managed orchestration environments where agents are hosted, trained, and systematically deployed.
- **Agent frameworks:** The underlying software libraries and architectures (such as LangChain, LlamaIndex, or AutoGen) utilized by developers to build agentic logic.
- **LLM models:** The foundational and fine-tuned large language models that serve as the cognitive engines driving the agents.
- **Isolated build and execution environments:** Secure, compute-isolated CI/CD boundaries that run agent code during testing and evaluation, ensuring security teams discover ephemeral test executions rather than relying solely on static registry declarations.
- **Additional internal tooling:** Custom code repositories, code assistants, and internal model registries integrated into the development pipeline.
- **Skills:** Defined procedural modules, code snippets, or tools built out to extend the capabilities of an agent.

3.1.1.2 Agent Import

Modern enterprises rarely rely solely on a single source of intelligence; instead, they ingest diverse agent architectures that vary by origin, framework, and compute boundary as a hedge against rapidly shifting agentic capabilities. Agent Import monitors and governs the introduction of these varying agent classes into the corporate ecosystem, focusing heavily on how data flows across the organization based on their underlying deployment model.

Agent Type, Category, and Channel are three separate questions you can ask about any agent you import, and a single agent will usually have an answer to all three at once, for example, a SaaS agent (built by a vendor) used for internal HR workflows (B2E), pulled in through an agent gateway.

Agent Type: Who built it?

- **First-party agents:** Internally developed AI capabilities ranging from fully custom implementations built with raw code (such as Python and LangChain) running on general-purpose compute (e.g., Kubernetes or Serverless Functions), to agents constructed using provider-specific agentic frameworks that natively handle the runtime, memory, and workflow orchestration.
- **SaaS agents:** Purpose-built, third-party AI products that operate as entirely independent entities within the enterprise software ecosystem.
- **Local agent:** A client architecture installed directly on the employee's local device, where every single reasoning step is sent to a remote cloud LLM provider. It behaves as a non-deterministic layer between human intent and tool action, with data crossing the organizational boundary on every inference call.
- **Orchestration agents:** Agents built on visual orchestration platforms that serve to "glue" disparate enterprise APIs together.

Agent Category: Who is it for?

- **B2E (Business-to-Employee) agents:** Internal, employee-facing AI agents deployed to automate corporate workflows, IT/HR operations, and daily employee productivity within internal enterprise identity and role-based access controls.
- **B2B agents:** Business-facing AI agents that operate across corporate boundaries to automate enterprise-to-enterprise workflows, supply chain transactions, and cross-tenant API integrations.
- **B2C agents:** Consumer-facing AI agents deployed to interact directly with external end-users for service, commerce, or support, requiring strict defenses against public prompt injections, brand exposure, and consumer PII leaks.
- **B2B2C agents:** White-labeled or vendor-provided AI agents delivered through a corporate client to serve that partner's end-consumers, operating under multi-tier isolation and transitive trust boundaries.
- **Personal Assistant agents:** Consumer-grade, user-directed AI assistants operating on an individual's behalf across both personal and enterprise contexts, requiring boundary controls between personal and corporate data access.

Ingestion Channel: How is it found?

- **Agent gateways:** Ingestion points and proxies through which these external or imported agent types interact with internal enterprise APIs.
- **Ecosystem registries & third-party catalogs:** Automated connector pipelines and API adapters that continuously monitor, pull, and map agent definitions from external vendor-native registries, ensuring agents are automatically cataloged without intervention.
- **MCP & Skill registries:** Ingestion points through which agents pull tool schemas, skills, and context definitions from public or vendor-hosted MCP and Skill registries. Registries are themselves an injection vector, a malicious or disguised entry can be pulled in and consumed automatically, so every registry-sourced tool or skill is treated as untrusted input subject to the supply-chain scanning in Section 3.1.2 before an agent can invoke it.

3.1.1.3 Shadow AI Discovery

Employees naturally turn to unapproved AI tools to optimize their workflows, introducing compliance, data exfiltration, and operational security risks. Shadow AI Discovery actively scans corporate perimeters to detect unauthorized agent deployments and usage. Discovery and monitoring capabilities should be implemented in accordance with applicable privacy, employment, labor, works-council and data-protection requirements, with appropriate transparency, proportionality, data minimization, retention and access controls.

- **Cloud infrastructure:** Cloud compute instances, serverless environments, and unmanaged API endpoints discovered via active network/API inspection.
- **Network:** Real-time traffic analysis and egress monitoring to identify unrecognized API calls to external LLM providers and AI endpoints.
- **Endpoint:** Device-level monitoring to track unauthorized execution of AI binaries, scripts, or local model weights.
- **Browser:** Extension tracking and browser session monitoring to detect when users log into unapproved web-based AI platforms or chat systems.
- **Mobile Device Management (MDM) & Endpoint management:** Continuous device-level auditing to detect unapproved local AI executables, local model weights, CLI agent tools, and IDE extensions, while enforcing baseline security posture and blocking unmanaged agent runtimes at the operating system layer.
- **IoT, OT & edge infrastructure:** Industrial security monitoring and network protocol inspection to detect Small Language Models (SLMs) and autonomous agents on manufacturing equipment, PLCs, and embedded hardware where traditional endpoint software cannot be deployed.
- **Mail and collaboration flow:** Analysis of message, chat, and calendar traffic to detect unsanctioned agents sending messages, auto-replying, or managing calendars on a user's behalf before activity becomes visible in endpoint or network egress logs.
- **Governed execution path as a detection surface:** An architectural boundary where all sanctioned agents execute through a centrally managed control plane; any agentic process operating outside this boundary is automatically flagged as a shadow AI signal without requiring separate environment-specific detection rules.

3.1.2 AI Agent Security Posture Management (AIASPM)

Once discovered, agents must be continuously evaluated. AIASPM provides the diagnostic layer that assesses risk profiles, configuration hygiene, and known software weaknesses across the entire agent inventory.

- **Risk Inventory:** A centralized, dynamic ledger tracking security posture, vulnerabilities, software dependencies, and misconfigurations across discovered agents regardless of lifecycle state, provisioning the security-risk lens to complement the IAM-focused Agent Directory.
- **Vulnerabilities:** Standard software vulnerability management extended to agentic dependencies and model endpoints, covering CVEs across the agent's codebase, supply chain, and foundational models, mapped to standardized frameworks for consistent risk scoring.
- **Code & image provenance:** Cryptographic verification of the origin, build integrity, and signatures of agent code, container images, dependencies, and model weights (e.g., SLSA framework attestations) prior to production execution. Utilizing minimal, pre-attested base images stripped of unneeded packages and rebuilt against current CVE data reduces supply chain attack surface and simplifies provenance verification at build time.
- **Tool & skill supply-chain scanning:** Automated static and dynamic inspection of local agent instruction sets, execution skills, and remote Model Context Protocol (MCP) tool schemas, including entries pulled from MCP and Skill registries, to detect embedded malicious code, concealed operations, and unauthorized egress paths prior to registration.
- **Misconfigurations:** Continuous scanning to catch weak parameter settings, insecure system prompts, hardcoded long-lived credentials, impersonation vectors and configuration drift introduced after an agent's initial review passed.
- **Adversarial validation:** Continuous, automated adversarial testing of agents and their guardrails against known prompt injection, jailbreak, and data-exfiltration paths inside disposable, isolated execution boundaries to safely evaluate real-world exploitability without risking production impact.

3.1.3 Agent Directory (Register & Govern Agent Identity)

Where Agent Import (Section 3.1.1.2) captures provenance and metadata during discovery, the Agent Directory is the authoritative identity repository where cryptographic credentials, ownership, and policy entitlements are formally registered and maintained for all validated, enterprise-hosted agents. Registration is the boundary between a discovered agent and a governed one, and it is the prerequisite for every access decision in Pillar 2.

Every hosted agent and every sub-agent with distinct entitlements must be registered here with a verified profile, cryptographic identity, accountable owner, and clear structural relationships before it can request access to enterprise systems. Keep registration lightweight enough that it doesn't push teams to bypass it by using shared human credentials. Local agents and ephemeral sub-agents that inherit a parent's scope are governed by the containment and delegation controls described below rather than by individual directory entries. Once an agent is imported, the Directory answers two more questions about how it operates in your enterprise: what role does it play and how does it prove its identity.

Agent Classes: What role does it play?

- **Autonomous agents:** Fully independent agents that operate asynchronously to achieve long-running business goals without immediate human oversight.
- **OBO (On-Behalf-Of) human agents:** Agents executing tasks directly tied to a specific human user's session, acting as a proxy and constrained by that user's underlying permissions.
- **Orchestrator agents:** Parent-level agents that decompose an objective into sub-tasks and spawn, coordinate, and supervise downstream sub-agents; the directory must track delegation chains and spawn relationships as first-class attributes.
- **Sub-agents:** Specialized child agents dynamically spawned or invoked by a parent orchestrator to execute discrete sub-tasks, governed by inherited session scopes rather than individual directory registration. Because in-model task boundaries enforced within an LLM can fail, sub-agents must be wrapped in isolated execution boundaries that structurally enforce inherited permission scope, preventing reasoning errors from escalating privileges.

- **Physical/Embodied agents:** Agents controlling robotics, actuators, or other physical systems. This architecture governs identity and access, but complements, rather than replaces, existing plant safety, OT security, and change-management controls.
- **Guardian & Security agents:** Specialized administrative AI agents operating within the security control plane (e.g., SOC triage agents, dynamic policy engines, continuous AIASPM scanners). Given their elevated privileges to inspect telemetry and execute containment actions, guardian agent identities require mandatory cryptographic workload attestation, strict separation of duties, and continuous owner sign-off.

Identity Substrate: What proves it's really that agent?

- **Agent Identity metadata profile:** Rich metadata profiles (CIMD) that define the agent's intent, origin, lineage, and operational context to inform risk-based access decisions.
- **Workload identity:** Cryptographic, platform-agnostic, standards-based identity production frameworks used to issue secure, verifiable IDs to agents across dynamic cloud environments.
- **Agent governance metadata:** Structured metadata bound to the agent's directory record, including its accountable human owner, designated operational team, or on-call queue, lifecycle status (e.g., active, suspended, retired), risk tier, and operational scope, to be used continuously by policy engines for access control authoring and runtime evaluation.
- **A2A agent protocols:** AI agents that leverage standardized interoperability frameworks to dynamically discover capabilities, delegate tasks, and collaborate peer-to-peer across enterprise boundaries. When interacting across organizations without a shared directory or trust root, agents utilize cryptographic attestation and token-exchange standards (e.g., WIMSE/AIMS, OAuth Token Exchange, HTTP Message Signatures) to preserve delegation chains and verify principal identity across intermediaries.

3.2 Pillar 2:

What can they do? (Access & Entitlements)

Autonomous agents are fundamentally designed to act, which requires them to connect to data, APIs, and systems. Building on the verified identities established in Pillar 1, this pillar defines the boundaries and permissions that govern how agents interact with corporate resources across both direct human-delegated sessions and complex, multi-agent workflows, augmenting legacy human-centric models with machine-to-machine AI governance.

3.2.1 Access Policies (Define access policies)

Access Policies specify the operational boundaries for what an identity-verified agent is allowed to read, write, or execute. Given the unpredictable nature of generative AI outputs, these policies must be deeply contextual and adaptive.

- **Coarse-grained:** Broad, role-based controls defining which high-level environments or macro-services an agent can connect to.
- **Fine-grained:** Attribute-based and object-level permissions restricting an agent to specific database rows, files, or specific API methods, explicitly taking into account corporate data classification labels (e.g., Confidential, PII) to enforce compliant data boundaries. Policy constructs may span data classification, agent identity, risk scope, environmental context, and action type.
- **Relationship-based:** Access derived from an agent's graph relationships: who spawned it, who it acts on behalf of, and what it's nested under.
- **Intent-based:** Advanced policy engines that analyze the semantic intent of the agent's prompt or plan before granting access to resources. Inspecting a plan is achievable with current tooling; treating a literal user prompt as the sole trigger is aspirational, since prompts aren't always what an agent acts on. Ephemeral, time-bound privileges are created strictly for the duration of a specific task and immediately revoked upon completion.
- **Network-based:** Macro-segmentation restricting agent traffic across enterprise VPCs, combined with L3/L4 micro-segmentation controls restricting agent-to-target communication to authorized IP ranges or service meshes.

- **Time-bound:** Time-bound grants are scoped to a single task or session and are automatically revoked upon completion or timeout. Because auto-approved, auto-expiring grants add limited control value on their own, pair time-bound scoping with the anomaly detection in Section 3.3.1.2 rather than treating expiry alone as the safeguard.
- **Cross-app access:** Policy frameworks and boundary controls that govern an agent's ability to traverse distinct application ecosystems, enabling secure data translation, token exchange, and function execution as it orchestrates workflows across different enterprise platforms.
- **Delegation & inheritance:** The effective permission boundary for a sub-agent is computed as the intersection of its own entitlements and the delegating agent or human's entitlements to prevent confused-deputy escalation. Policy should also cap the breadth and depth of delegation to prevent unbounded fan-out.
- **Guardrails:** Hard, deterministic safety constraints and behavioral boundaries that override autonomous reasoning loops to globally prevent non-compliant actions, toxic outputs, or high risk operational steps regardless of the agent's intent or fine-grained privileges.

3.2.2 Governance

Governance provides the compliance baseline and continuous oversight loop, using automated tooling rather than manual rubber-stamp reviews to keep agent privileges aligned with the principle of least privilege at agent velocity.

- **Contingent Review:** Trigger automated or human-in-the-loop workflow approvals whenever an agent's action crosses defined financial or operational thresholds (e.g., a purchase or booking exceeding $1,000), regardless of whether the action falls within its granted technical permissions.
- **Access reviews:** Automated, continuous entitlement scanning helping system owners to validate non-conforming permissions.
- **Access requests:** Formal, audited workflows for provisioning new capabilities, models, or system access to an existing agent. While requests may originate from human owners or dynamically from agents at runtime, agent-initiated requests must be strictly evaluated against static permission ceilings and require mandatory owner verification for scope expansion to prevent agents from manipulating automated workflows into granting excessive access.
- **Separation of Duties (SoD):** Algorithmic rules preventing a single agent from possessing conflicting privileges, such as both generating and approving financial transactions.
- **Joiner-mover-leaver (JML):** Apply the same onboarding, change management, and deprovisioning discipline used for human identity lifecycles to an agent's scope, ownership, or model version evolution.

3.3 Pillar 3:

What are they doing? (Runtime Authorization & Control)

Visibility and identity are ineffective without inline runtime monitoring and enforcement. This pillar focuses on intercepting, inspecting, and authorizing agent actions in real time as they execute tasks, ensuring they do not engage in malicious behavior, mishandle sensitive data, or compromise critical enterprise targets.

3.3.1 Runtime Authorization and Monitoring

Runtime Authorization and Monitoring serves as the active inspection layer for live agent sessions. It sits directly in the execution path, analyzing both the inputs flowing into the agent and the actions or outputs radiating back out.

3.3.1.1 Gateways (Enforce access policies)

Gateways act as policy enforcement points (PEPs) in a defense-in-depth chain. They intercept traffic between the agentic layer and the rest of the enterprise stack, parse protocols, and block unauthorized calls before they reach their destination. Downstream API Gateways maintain ultimate authorization authority, performing their own validation of every call regardless of what upstream AI/Agent Gateways have already inspected. AI/LLM, MCP, Security, Agent, and API Gateways are often deployed together, but each owns a distinct control point:

AI/LLM Gateways govern model-API traffic; MCP Gateways govern tool/context-sharing protocol traffic; Security Gateways provide protocol-agnostic threat inspection; Agent Gateways govern agent-specific identity and delegation concerns; and API Gateways enforce classic API-layer controls for the tools and services agents call.

- **AI/LLM gateway:** Intercepts low-level API calls to foundational models, managing rate-limiting, token counts, and system-prompt overrides.
- **MCP (Model Context Protocol) gateway:** A specialized proxy optimized for managing context-sharing protocols and open-standard integrations between agents and applications with a focus on MCP servers.
- **Security gateway:** An inline threat-mitigation layer that provides dedicated content security and protocol verification for agent interactions. It sits directly in the traffic flow to perform real-time, TLS-terminating deep payload inspection, actively intercepting and filtering out complex prompt injections, jailbreaks, hidden adversarial text strings, and malicious code payloads before they hit the agent or downstream resources. It also handles data classification scanning to enforce outbound data redaction, structural integrity checks on model outputs, and cryptographic inspection of encrypted agent payloads.

- **Agent gateway:** Serves as the agent-specific runtime control plane: binding agent identity to every request, maintaining the tool/MCP registry agents draw from, and governing agent actions, not just access, based on identity, context, and the specific action being requested. It stops unsafe interactions in real time, blocking or sanitizing risky model traffic and denying or requiring approval for sensitive tool actions via built-in and custom policies, and it injects secret-less, short-lived downstream credentials into API calls at execution time so agents never handle, store, or expose standing enterprise API keys or service account secrets. It can provide governed telemetry across supported multi-agent interactions and can detect or redact sensitive data, including PII, according to configured policies and capabilities while defending against unsafe behaviors and malicious prompt-based attacks.
- **API gateway:** Enforces classic API-layer controls, rate limiting, quotas, schema validation, mTLS, WAF, traffic shaping, and revoke-at-gateway for the tools and services agents and MCP servers use, performing its own final authorization check on every call rather than assuming upstream AI/Agent Gateways already caught every anomaly.

3.3.1.2 Monitoring (Monitor runtime)

Monitoring provides deep observability, threat detection, and active enforcement, scanning runtime telemetry for anomalous behavior, data leakage, and external exploitation attempts, and applying controls such as sandboxing and endpoint isolation in real time.

- **Sandboxing:** Isolating agent execution (especially untrusted, third-party, or newly updated agents) within secure, constrained runtimes to prevent lateral movement and contain execution risks through structural boundaries.
- **Endpoint isolation & allowlisting:** Local runtime controls for endpoint-hosted agents, including OS-level sandboxing, endpoint application allowlisting, and local device-posture checks to constrain local agent command execution and file access.
- **Tool/Data access:** Real-time structural inspection and auditing of arguments, payloads, schemas, and queries passed to downstream tools and repositories. Performing inspection inside or directly adjacent to the execution boundary helps ensure dynamic execution plans do not deviate from authorized parameters or introduce unintended mutations mid-session.
- **Threat/anomaly detection:** Machine learning engines tracking agent behavioral baselines to flag unusual spikes in data access, rapid API loops, or bizarre execution patterns.

- **Prompt injection:** Inline scanners designed to catch adversarial text strings, both direct and indirect, meant to hijack the agent's core logic or override system instructions.
- **DLP/output filtering:** Data Loss Prevention tools scanning agent outputs to mitigate the risk of leaking proprietary source code, PII, or trade secrets to external providers or unauthorized users.
- **Tracing:** Comprehensive transaction mapping that charts the exact execution path, call stacks, and sub-agent invocations triggered by a primary user prompt.
- **Cost & Performance:** Tracking token usage, operational costs, latency, and resource consumption to prevent runaway agent loops or resource exhaustion attacks.
- **Workflow-in-the-loop:** Programmatic triggers that pause agent execution for high-risk actions, routing to automated policy cross-checks or explicit human sign-off before they proceed.
- **Continuous Monitoring:** Continuous monitoring and exportability of the agent's activity, including granular tracing, audit trail logging, and non-repudiation of the agent's actions.

3.3.1.3 Continuous Intent and Alignment Enforcement

Continuous intent and alignment enforcement determines whether an agent remains within the boundaries of its delegated task throughout execution or has drifted into unauthorized purpose expansion. It complements initial access policies: intent-based policy determines what an agent may initially access, whereas continuous alignment enforcement evaluates whether evolving, multi-step behaviors remain appropriate over time.

Key capabilities include:

- **Purpose binding:** Enforcing strict linkage between the delegating user, agent identity, approved workflow, and requested task.
- **Plan vs. action evaluation:** Runtime comparison of planned tool calls against observed actions to catch task drift, injection-driven deviations, and confused-deputy behavior.
- **Context-aware enforcement:** Executing real-time actions, including allow, deny, redact, step-up approval, scope reduction, suspension, or containment.
- **Traceable decision evidence:** Maintaining non-repudiable audit logs explaining why each material action was allowed, modified, or blocked.

3.3.2 Resource Access

Resource Access delineates the actual blast radius of an agent. It categorizes the diverse downstream targets that an agent can interact with, helping ensure that appropriate security wrappers surround every interface.

3.3.2.1 Target Resources (Components being accessed)

Target Resources represent the destinations of an agent's actions. Whether pulling context or pushing changes, these endpoints must be continuously cataloged and defended.

- **MCP servers:** Data repositories and utility servers explicitly configured to share contextual information (structured data, documents, embeddings, tools, and state) with agents via the Model Context Protocol.
- **SaaS apps:** Enterprise productivity and core business applications containing vast stores of sensitive corporate data.
- **Data stores & knowledge repositories:** Structured and unstructured repositories, including vector databases, data lakes, warehouses, and transactional storage systems, leveraged by agents to extract business context or commit operational outputs. These repositories are governed by data security posture management (DSPM) practices, tracking lineage and classification as agents access each store.
- **CLI tools:** Command-line interfaces allowing agents to execute system-level scripts, configurations, or operational commands.
- **Other agents:** Downstream sub-agents or specialized agents tasked with executing fractional components of a broader workload.
- **Privileged resources:** Highly sensitive target environments, including root directories, identity stores, and master configuration consoles.
- **Standard & Legacy endpoints:** Enterprise REST/GraphQL APIs, databases, and legacy applications that do not natively support agentic context protocols like MCP.
- **Payment systems:** Financial networks, transaction gateways, and accounting ledgers require the highest degree of cryptographic validation and human oversight.
- **Browsers:** Web automated sessions utilized by agents to scrape public intelligence or interact with web-based external consoles.
- **Skill runtimes & code executables:** Code interpreters, script execution sandboxes, and procedural modules invoked dynamically by agents to run generated code, requiring isolated compute boundaries to mitigate runtime-escape risks.
- **Foundation models:** The raw intelligence providers where weights are hosted, necessitating secure, validated model API keys.
- **Communication and collaboration channels:** Email, chat, and ticketing systems through which agents interact, representing untrusted prompt ingress paths subject to deep payload inspection and scanning.

3.3.2.2 Purpose Aware Data Security

Purpose aware data security evaluates whether a given agent should access, process, transform, summarize, disclose, or transmit specific data based on the delegated purpose, user authority, agent identity, workflow state, data classification, recipient, destination, and downstream action. It applies data controls at the moment of use, not only at the moment of access. This matters because agents combine data across systems, infer sensitive information from non sensitive inputs, and disclose results through outputs, APIs, messages, files, and sub agent calls. Classification and DLP determine what the data is and whether it is sensitive; purpose aware data security determines whether this use of it is appropriate in context. Key capabilities include purpose binding between the user request, agent task, data source, and permitted use; runtime evaluation of whether access and disclosure suit the current workflow step; controls for summarization, transformation, export, cross app movement, and third party agent disclosure; detection of data overreach, excessive retrieval, unauthorized inference, and inappropriate sharing; and enforcement spanning redaction, masking, partial response, approval workflow, logging, blocking, or containment.

3.4 Pillar 4:

How do I respond? (Active Containment)

The final pillar of the blueprint transforms telemetry into immediate containment. When monitoring systems flag anomalous behavior, compromised credentials, or a prompt-injection attack, the enterprise must have highly automated, surgical mitigation mechanisms that favor proportional, least-disruptive responses (rate-limiting, quarantine) over destructive actions that could cause a self-inflicted outage.

3.4.1 Response/Enforcement (Throttling & Containment)

The automated operational control loop is designed to respond to risk signals in real time. It acts as a coordinated control layer, not a single centralized chokepoint, executing targeted, proportional security responses across the entire technical stack.

Enforcement is most effective when paired with a standardized notification protocol: the system that detects risk publishes a Kill Switch event, the subscribing systems act, and the outcome is published back as a confirmation signal.

3.4.1.1 Enforcement Actions (Mitigation mechanisms)

Enforcement Actions represent the concrete, multi-layered levers security teams and automated orchestrators can pull to isolate, contain, or completely neutralize a problematic or compromised agent. These actions form a SOC-owned library of response options, not isolated one-off switches: responders select the least-disruptive action proportional to the aggregated Risk signal.

Containment actions triggered by guardian agents (suspend, isolate, terminate) require the target agent to operate within an execution boundary that supports granular session-level control, whether running in a managed cloud service, SaaS application, or containerized runtime. Without underlying platform or runtime support to pause an agent without collateral impact to adjacent workflows, API-driven containment provides a false sense of security. When containing agents tied to physical equipment or OT, automated triggers must be calibrated to ensure abrupt isolation doesn't create physical safety or operational hazards on the plant floor.

- **Token revocation:** Instantly invalidating active OAuth or API access tokens, immediately freezing the agent's ability to call connected applications.
- **Session Token Revocation:** Invalidating active OAuth tokens and revoking downstream session contexts across connected enterprise services and/or severing the active runtime connection of a single, specific interaction string without necessarily deleting the agent's broad identity.
- **Continuous authorization:** Dynamic, real-time re-evaluation of security contexts that instantly downgrades permissions the exact moment a risk posture changes.

- **Process termination:** Force-closing compute runtimes as a last resort when quarantine fails; note that termination sacrifices in-memory forensic telemetry, so prefer network quarantine, session suspension, or execution environment suspension where possible.
- **Network quarantine:** Utilizing micro-segmentation rules to isolate the agent's hosting environment, cutting off its communication with both the internal network and the open internet.
- **Cloud service termination:** Instantly shutting down the underlying cloud instances, serverless functions, or virtual machine clusters hosting the rogue agent.
- **Agent platform termination:** A top-level administrative command that instructs the centralized agent orchestration platform to suspend the agent's operational account and halt all scheduled tasks.
- **Throttling and rate limiting:** Progressively constraining an agent's request rate, token budget, or concurrency as a proportionate first response, degrading capability without fully severing the session.

3.4.2 Access Recovery

A kill switch is only half the control loop: every containment action needs a matching path back to a verified, last known good state. Access Recovery governs how a contained, quarantined, or terminated agent is restored to service, confirming that reinstatement is deliberate, documented, and traceable, rather than an automatic reversal of the enforcement action. Fully automated, standardized recovery workflows remain a target state; most organizations today still require manual sign-off at one or more of the steps below.

- **Re-attestation:** Re-verifying the agent's identity, code, and configuration against a known-good baseline before any access is restored in a fresh, isolated execution instance.
- **Staged re-enrollment:** Restoring access incrementally (e.g., read-only before write, sandboxed before production) rather than reinstating full privileges at once.
- **Root-cause sign off:** Requiring the human owner or security team to document root cause and remediation before re-enabling the agent.
- **Audit trail closure:** Linking the original Kill Switch event, remediation actions, and re-enrollment decision into one closed, auditable incident record.

3.5

Cross-Cutting Foundation Components

These architectural layers do not live in silos; they sit underneath and across all four pillars, serving as the tissue that connects visibility, identity, monitoring, and response into a unified security fabric.

3.5.1 Execution Context & Risk Signals

Execution Context & Risk Signals serve as the connective analytical tissue of the blueprint. Fed continuously by real-time runtime telemetry, security posture assessments, and delegating user identity signals (such as account compromise indicators or elevated principal risk), this orchestration engine dynamically evaluates threats by correlating individually low-risk conditions into toxic combinations. It also publishes positive execution context, structural confirmation of what's already known to be valid, not just alerts about what's gone wrong. When a threshold is crossed (e.g., a sudden prompt injection combined with an attempt to exfiltrate financial data), Risk Signals trigger the corresponding Response/Enforcement mechanisms directly and programmatically. To help ensure real-time interoperability and coordinated responses across heterogeneous vendor environments, this system can leverage open security engineering standards such as the **Shared Signals Framework (SSF)** and the **Continuous Access Evaluation Profile (CAEP)** to asynchronously publish and consume zero-trust state modifications.

3.5.2 AI Agent Telemetry, Logging, and Observability

This is the foundational baseline layer supporting all pillars of the reference architecture. It serves as the immutable data plane, capturing execution metadata throughout the enterprise stack, backed by mandatory inline payload redaction to prevent PII, secrets, and credentials from entering the log lake. Without this structured, high-fidelity logging lake, automated threat detection is impossible, posture assessments lack data, and forensic incident responders are left completely in the dark. To help ensure consistency and reduce vendor silos in complex multi-agent interactions, this layer can leverage open standards such as the **Open Cybersecurity Schema Framework (OCSF)** to normalize diverse log streams into a single, unified security taxonomy.

- **Tamper-resistance telemetry origin:** Capturing telemetry at the execution boundary rather than relying on self-reported agent logs helps ensure non-repudiation, preventing a compromised agent from suppressing, modifying, or falsifying its execution record.

4

Operationalizing with Global System Integrators (GSIs)

An architectural blueprint requires an operational deployment model to achieve lasting enterprise value. The sections below define how security teams and advisory partners operationalize these four pillars within existing enterprise workflows. Organizations may operationalize the architecture through internal enterprise teams, GSIs, managed-service providers or other qualified partners, depending on organizational capabilities and requirements. A reference architecture published without an operating model becomes shelfware within a year; the sections below define how that architecture is actually stood up, staffed, and sustained within an enterprise.

4.1 Deployment Advisory & Architecture Enablement

GSIs translate the four pillars into a sequenced, organization-specific rollout rather than a simultaneous, all-at-once deployment. This includes:

- **Current-state assessment:** Mapping an enterprise's existing IAM, network, and monitoring stack against the Blueprint Reference Architecture to identify true gaps versus capabilities that already exist under a different name.
- **Target operating model design:** Defining clear operational ownership across pillars to prevent accountability gaps across teams. This model typically assigns Gateways and Monitoring to Security Engineering, Agent Directory, Access Policies and Governance to IAM, Response/Enforcement to the SOC, and execution boundaries (sandboxing, isolation runtimes, lifecycle management) to Compute and Platform Engineering.
- **Phased rollout planning:** Sequencing deployment adapted to the client's existing tooling investments and regulatory timeline.
- **Vendor interoperability integration:** Configuring the specific AI/LLM, MCP, Security, Agent, and API Gateways an enterprise has already licensed to interoperate through the shared telemetry and Risk Signals layer, rather than requiring a rip-and-replace of existing infrastructure.

4.2 Standardized Risk Scoring Matrices

Not every agent warrants the same level of control depth, and GSIs provide calibrated scoring frameworks that prevent organizations from over-engineering low-risk deployments or under-protecting high-risk ones. This includes:

- **Risk tiering models** that score agents against operational scope (read-only vs. transactional), data classifications, autonomy level (OBO vs. fully autonomous), and blast radius of target resources.
- **Industry-calibrated thresholds:** A financial services agent executing payment workflows and a manufacturing agent reading OT sensor data warrant materially different risk baselines; GSIs bring the cross-industry pattern library to calibrate these thresholds rather than starting from zero.
- **Business-impact scoring for semantic governance:** Defining the dollar, regulatory, or reputational thresholds that trigger mandatory human-in-the-loop approval, tuned to the client's actual risk appetite rather than a generic default.
- **Isolation depth scaled to risk tier:** Risk tiering must scale the strength of the structural execution boundary (e.g., kernel-level sandboxing, ephemeral runtime lifecycles) alongside review gates, ensuring that a high-risk agent does not share the same physical blast radius as a low-risk tool if compromised.

4.3 Continuous Lifecycle Management

Agent governance is not a one-time certification; GSIs operate the ongoing lifecycle processes that keep the Agent Directory and Access Policies accurate as the environment changes:

- **Automated access certification cycles:** Managed-service pulls of the Agent Directory and Access Policies state, flagging agents whose human owners have left the organization or whose entitlements have drifted from least privilege.
- **Joiner-mover-leaver (JML) parity for agents:** Applying the same deprovisioning discipline used for human offboarding to agent ownership changes, model version upgrades, and sub-agent retirement, so orphaned credentials don't accumulate as the primary source of the identity gap.
- **Separation of Duties (SoD) exception routing:** Operating the workflows that route conflicting-privilege exceptions (e.g., an agent that can both generate and approve a financial transaction) to the correct system owner for resolution, on the same cadence as human identity governance.
- **Execution environment drift management:** Scheduled patching, dependency updating, and re-attestation of agent runtimes and underlying platform configurations to help eliminate vulnerability drift across hosted, cloud, and containerized deployments.

4.4 Regulatory & Compliance Alignment

GSIs bridge the architecture to the specific regulatory regimes an enterprise operates under, translating technical controls into evidence designed to support regulatory, compliance, and audit requirements:

- **Control mapping to regulatory frameworks:** Aligning telemetry, audit trail, and access review outputs to sector-specific requirements (e.g., financial services model risk management, healthcare data handling, critical infrastructure protection) as they evolve alongside agentic AI-specific guidance.
- **Audit-ready evidence packages:** Structuring the telemetry and logging layer's output so that Access Reviews, Containment & Throttling events, and Access Recovery records (re-attestation, root-cause sign-off, audit trail closure) satisfy examiner and auditor documentation standards, not just internal SOC needs.
- **Cross-border and data residency guidance:** Advising on how Agent Import channels (SaaS agents, first-party agents crossing organizational boundaries) intersect with data residency and cross-border transfer requirements specific to each jurisdiction the enterprise operates in.
- **Tamper-evident audit provenance:** Structuring telemetry collection so regulators and examiners receive non-repudiable audit logs generated directly at the execution boundary, reducing reliance on secondary log reconstruction from surrounding network infrastructure.

4.5 Sector-Specific Playbooks

Control requirements vary by sector, requiring GSIs to maintain playbooks tailored to distinct operational environments:

- **Financial Services:** Prioritizes strict execution isolation, high-assurance transaction thresholds, and regulatory data controls.
- **Manufacturing & Industrial:** Emphasizes OT, IoT, and edge agent discovery across legacy hardware.
- **Retail & Commerce:** Focuses on B2C prompt-injection defense, brand protection, and customer PII leakage.

Founding members continuously update these playbooks as joint interoperability results are published, verifying the operating model adapts alongside emerging agentic capabilities.

Conclusion: Shaping the Future of AI Security

The Blueprint Alliance governs agent behavior at production runtime. Built directly into the enterprise stack rather than retrofitted as an external layer, this architecture provides the structural safeguards required to deploy agentic automation safely at scale.

To deliver true multi-vendor interoperability, founding members actively validate their platforms against core open standards, including Model Context Protocol (MCP) for tool interaction, Open Cybersecurity Schema Framework (OCSF) for unified logging, Shared Signals and Events (SSF/CAEP) for real-time risk exchange, and HTTP Message Signatures (RFC 9421) for request authentication. This shared telemetry baseline helps ensure that a threat signal or boundary violation detected in one runtime environment can be natively ingested and acted upon across the entire control plane. Capabilities will vary by vendor and implementation; reference integrations and validated interoperability results will be publisjed as testing is completed.

By regularly publishing joint interoperability results and reference integrations, the Alliance delivers an evolving, multi-vendor ecosystem that moves at the speed of agentic innovation.

Blueprint in Action

The five scenarios below show how the pillars in Section 3 work together in practice. Each is a composite, built from patterns seen across early Alliance engagements rather than any single named customer. These composites illustrate the architecture's intended behavior; they are not evidence of deployed outcomes at Alliance-member scale today. Read them as the architecture in motion, not a case study of one deployment.

From shadow agent to registered identity. A finance analyst connects a browser-based AI assistant to their corporate SSO session to summarize vendor contracts. Shadow AI Discovery flags the unrecognized egress traffic on its very first connection. AI Agent Security Posture Management checks the tool against known vulnerabilities and routes it to the Agent Directory. Before sensitive data is ever accessed, the assistant is either registered as a governed identity with a verified metadata profile and scoped access or blocked at the network layer. What used to stay invisible for months is now visible, checked, and decided on before it ever touches sensitive data.

Preserving delegation across a multi-agent workflow. A sales operations lead asks an orchestrator agent to update a set of CRM records and notify a distribution list. The orchestrator spawns a sub-agent to handle the write and another to draft the notification. Each hop carries a nested actor claim back to the original human session, so the Agent Directory and Access Policies engine can enforce OBO constraints at every step. If a sub-agent tries to reach a system that the original human could not, core IAM and policy boundaries, including Cross App Access, block it, and because each sub-agent runs in its own isolated execution context, a blocked attack cannot impact the orchestrator session or sibling tasks, while the entire workflow remains unified in a single execution trace.

Right-sizing access to what the agent is actually trying to do. A support agent is provisioned with coarse-grained read access to a customer data platform. When it attempts to issue a refund, the intent-based policy engine detects the shift from lookup to transaction and requires human-in-the-loop approval before granting a fine-grained, ephemeral privilege scoped to that single order. Executing the refund within an ephemeral runtime scoped to that single approved task helps ensure the elevated privileges and runtime environment are decommissioned upon task completion, reducing standing permissions and persistent attack surfaces. The elevated privilege expires the instant the task is done.

From anomaly to automated containment. Runtime monitoring, typically a traditional ML anomaly-detection model rather than an LLM, detects an agent making an unusually high volume of calls to an internal API it rarely touches; an inline Guardian AI Agent separately flags a prompt-injection signature in its most recent input. Risk Signals correlates both events through the Shared Signals Framework, crosses a defined risk threshold, and automatically triggers token revocation and network quarantine, and suspension of the agent's active execution environment, preserving its in-memory state for forensic review rather than terminating it outright, all without waiting for a human to notice the alert. The security team receives the full trace as a closed-loop confirmation record, compressing containment from a multi-hour manual investigation into an automated, near-instantaneous response.

Turning access reviews into a continuous process. During each continuous access-certification cycle, a Global System Integrator's managed service pulls the full Agent Directory and Access Policies state through the blueprint's open telemetry layer, flags agents whose owners have left the company, and routes Separation of Duties exceptions to the right system owner. A manual, spreadsheet-driven review becomes a continuous, audit-ready workflow.

Best Practices for Getting Started

You don't need every capability in this blueprint on day one. Here's the order Alliance member deployments have actually followed.

Stand up unified logging and telemetry first. You cannot debug policy enforcement or baseline behavior without a single, normalized log lake; treat this as the prerequisite beneath every step below.

Then start with discovery, not policy. You can't govern what you can't see. Stand up Shadow AI Discovery and an initial Agent Inventory before you write a single access policy. Most teams are surprised by how many agents are already running.

Register before you restrict. Bring all discovered agents into a single Agent Directory with verified identities and owners before you layer on fine-grained or intent-based policies. A policy on an unregistered agent is a policy on nothing.

Layer authorization. Fine-grained (FGA) and relationship-based (ReBAC) controls have to be in place before intent-based authorization can correctly scope down agent permissions.

Instrument runtime before you automate the response. Deploy monitoring, tracing, and DLP/output filtering widely enough to establish a real behavioral baseline before wiring automated enforcement to Risk Signals. Automate containment against noisy or incomplete telemetry, and false positives will erode trust in the whole system.

Pressure-test the kill switch before you need it. Validate token revocation, session termination, and network quarantine against a non-production agent on a regular cadence. A kill switch you've never tested is a hypothesis, not a control.

Treat governance as continuous, not a project. Access Reviews, Access Requests, and Separation of Duties checks should run on the same cadence as human identity governance, not as a one-time agentic AI initiative. A Global System Integrator can help make that shift from project to operating model.