White Paper

Broadcom® Agentic Identity Fabric

Securing the Agentic AI Enterprise

TABLE OF CONTENTS

Executive Summary

There is a pattern in enterprise technology that has repeated itself with remarkable consistency over the past twenty-five years. Every fundamental shift in how applications are built and deployed has moved the security perimeter, and the company that secures the new perimeter first does not just win a deal. It defines a category.

In 1997, as the web pushed enterprise applications beyond the data center firewall, the security perimeter shifted from the network to the web perimeter defined by the user's session cookie. Enterprises suddenly faced a new question, "Who is accessing my web applications, and are they who they claim to be?" There was no product category or analyst quadrant that tracked this information. There was just a problem every enterprise recognized but nobody had solved.

The SiteMinder™ platform provided the answer. The product did not just enter a market, it created one. The product defined identity management for the web era and became the enterprise standard for over a decade, and powered much of the Fortune 500.

The pattern continued with federation and mobile. These technologies pushed users beyond organizational boundaries, shifting the perimeter from session to assertion. The industry answered with federated SSO and SAML. When APIs and microservices disaggregated the monolith, the perimeter moved from the user to the token. The industry answered with API gateways and OAuth.

Each time a shift in application architecture moved the security perimeter. Each shift changed the security model in ways the previous generation of tools could not address. As AI-driven and increasingly agentic architectures emerge, the perimeter is shifting again from static applications and human users to autonomous software entities that reason and act at runtime. AI workloads now demand entirely new approaches to govern identity, access, and behavior at machine speed.

The Evolution of Application Architecture

To understand where the security perimeter is moving, it helps to understand how applications themselves are evolving. The shift is not incremental. It is a fundamental change in how software is built, deployed, and operated.

In the first phase, the world we have known for decades, human developers write every line of code. They call APIs explicitly, configure credentials at deploy time, and ship applications that run deterministically. Every action the application takes was precoded by a developer. Every path through the code can be reviewed, tested, and audited. The security model for this world is well understood and well solved: authenticate the user, authorize the request, and audit the action. Firewalls, web application firewalls, identity and access management (IAM) systems, and role-based access control were built for exactly this architecture. The security model works because the application behaves predictably.

The second phase is emerging now. Developers use AI-powered tools (Cursor, GitHub Copilot, ChatGPT) to write code faster. The AI assists at development time by suggesting code completions, generating boilerplate, and even writing entire functions, but the deployed application is still traditional. The application still runs deterministically with the same APIs, the same credentials, and the same control flow. From a security perspective, nothing has changed at runtime. Whether a human typed the code or an AI suggested it, the application behaves the same way.

The third phase is the real shift, and it is arriving now. In this phase, applications embed LLMs as runtime decision engines. The developer no longer writes step-by-step logic for every action. Instead, the developer defines intent, constraints, and access. The LLM reasons at execution time about what to do, which APIs to call, what data to access, how to sequence actions, and what information to retrieve and synthesize. The application's behavior becomes nondeterministic; the LLM chooses different paths based on context, prompts, user input, and its own reasoning process. Vertical LLMs are trained on domain-specific knowledge (HR systems, enterprise resource planning workflows, and financial regulations). General-purpose LLMs are given access to enterprise systems through Model Context Protocol (MCP) servers, and function calls make the runtime decisions that developers would previously hard-code.

At this point the security model breaks. The decisions about what APIs to call, what data to access, and what actions to take are no longer made by a developer at deployment time and captured in source code. The decisions are now made by an LLM at execution time based on reasoning. There is no static code path to review. There is no pre-configured credential scope that covers every possible runtime decision. The LLM is improvising and traditional security tools have no mechanism to govern improvisation.

The fourth phase is the destination, and enterprises are already beginning to deploy it as multi-AI agent ecosystems. AI agents orchestrate other AI agents. AI agents spawn subagents and delegate credentials. AI agents communicate across organizational boundaries through protocols like Agent2Agent (A2A). MCP servers provide structured API front ends that AI agents use to access resource servers. Hundreds or thousands of autonomous actors make real decisions against real enterprise systems. They transfer money, access HR records, query mergers and acquisitions data, and modify enterprise resource planning entries, often with no humans involved in many transactions. In this environment, the security model is not just broken; it is completely absent.

Each phase multiplies the number of autonomous actors making decisions against enterprise resources. Phase one has humans. Phase two still has humans, just coding faster. Phase three introduces LLMs as runtime decision-makers. Phase four introduces entire ecosystems of AI agents reasoning, delegating, and acting autonomously. Identity middleware is the only layer that can handle this complexity at scale. It governs who is acting, what they are authorized to do, why they are doing it, and provides evidence that they did it correctly, regardless of whether the actor is a human, tool, LLM, or AI agent.

What Changes at Runtime

The distinction between today's applications and tomorrow's AI-orchestrated applications is most evident during execution. In a human-coded application, the developer decides which APIs to call, what data to access, what actions to take, and what credentials to use. These decisions are made at deployment time, captured in source code, and are deterministic and auditable. The security approach is straightforward: review the code, configure IAM policies, set up role-based access control, and deploy. The application will do exactly what the code says, every time.

In an AI-orchestrated application, the LLM makes these same decisions at execution time, based on reasoning. The LLM decides which APIs to call, what data to access, what actions to take, and what credentials to use. These decisions are made at runtime, based on the LLM's reasoning process. They are non-deterministic and opaque. The security question becomes, "Review what, exactly?" Should this question be the LLM's reasoning chain? Should the review be done per request, per AI agent, or per session? Traditional security was built for a world where application behavior was predictable. In the AI-orchestrated world, application behavior is emergent.

Identity middleware must serve as the runtime control plane, governing every decision, at every boundary, for every actor. The old security tools are not bad, but they were designed for a world that no longer exists. The LLMs were not runtime decision engines.

The LLM: A New Architectural Tier

There is a useful way to understand this shift through the lens of application architecture history. Every generation of enterprise computing has added a new tier to the application stack, and every new tier has eventually required its own security layer.

In the 1990s, the client-server architecture had three tiers: the client (browser), the web server, and the database. In the 2000s, the J2EE era added a fourth tier, the application server sitting between the web server and the database. The application server introduced its own security requirements that includes container-managed security, JAAS, and role-based access control. No one claimed the web server's security was adequate for the application server. The new tier received its own security layer.

The LLM is the new tier. In the agentic architecture, users and applications are at the top of the stack, followed by AI agents, LLM services (the reasoning tier), MCP servers and tool interfaces, and resource servers at the bottom. The LLM service is shared infrastructure with multiple applications, AI agents, and tools calling into the same LLM instances. This architecture similar to how multiple applications once shared application servers or database connections.

Unlike every previous tier in the history of application architecture, the LLM tier has no built-in security layer:

- **No authentication:** Anyone with an API key can call the model.
- **No authorization:** There is no native mechanism to restrict which users or AI agents can use which models for which purposes.
- **No cost attribution:** Token spend is not attributed to identities or business units.
- **No data classification enforcement:** The LLM processes whatever data is sent to it without regard to sensitivity.
- **No audit trail:** There is no built-in record of who called the model, what they sent, or why.

Every new tier in application architecture has historically required its own security layer. The LLM tier has none. This is where the model gateway must operate. The model gateway serves as a policy enforcement point (PEP) and identity middleware, and acts as the policy decision point. The model gateway provides the security layer that the LLM tier is missing, just as container security provided the security layer that application servers needed.

The same forces that built these categories are at work again. This time the perimeter is not moving from network to session, or from session to token. The transition is happening from human users to autonomous entities. The entities are software programs that operate independently at machine speed, and these entities can interact with every system they can access.

The Agentic AI Ecosystem

The agentic AI ecosystem is not a single application or a single protocol. It is a network made up of interconnected components, that includes people, tools, AI agents, models, servers, and data stores. There are flows between the components that cross multiple trust boundaries. Understanding this ecosystem is essential to understanding where identity governance must be applied.

At the top of the ecosystem are people: developers using AI-powered tools like Cursor, GitHub Copilot, and IDE extensions, and business users launching enterprise applications. These tools and applications are increasingly embedding or spawning AI agents. These AI agents are autonomous actors that reason, make decisions, and take actions on behalf of users.

The AI agents use LLMs as their reasoning engines. Some AI agents use hosted LLM services (GPT, Claude, Llama through cloud APIs like AWS Bedrock or Azure AI). Other AI agents use self-hosted models running on private infrastructure. The LLM is the reasoning tier. The AI agent calls the LLM to decide what to do next, which APIs to invoke, what data to retrieve, and how to synthesize results.

AI agents access enterprise resources through multiple paths. MCP servers provide structured tool interfaces. They act as API front ends that convert AI agent requests into specific resource server API calls. Resource servers (SAP, databases, SaaS platforms, and internal APIs) expose the actual business data and operations. Knowledge bases (vector databases, document stores, and data lakes) provide content for retrieval-augmented generation (RAG), where AI agents retrieve relevant information to ground their responses in factual, enterprise-specific data.

AI agents also interact with other AI agents. Through protocols like A2A, AI agents delegate tasks, spawn AI subagents, and collaborate across organizational boundaries. A procurement AI agent might delegate price verification to a market analysis AI agent, which in turn queries multiple vendor APIs through MCP servers. The delegation chain can extend several levels deep, with each link inheriting or narrowing the authorization of the link above it.

At every junction in this ecosystem, identity governance decisions must be made:

- **When a person uses a tool:** Who is this person, and what are they authorized to use?
- **When a tool or AI agent calls an LLM:** Which model is authorized, what data classification is permitted, and what cost tier applies?
- **When an AI agent accesses a resource through an MCP server:** Is this AI agent authorized for this tool, what is its declared intent, and what scope of access should it receive?
- **When an AI agent delegates to another AI agent:** Who originated the delegation, is there consent, and can the chain be traced?
- **When an AI agent retrieves content from a knowledge base:** What content classification applies, does the AI agent have the required clearance, and is this research or potential exfiltration?

These decisions require PEPs. The PEPs are gateways positioned at every access boundary to enforce governance decisions. A model gateway PEP controls access to LLMs. An API gateway PEP controls access to resource servers through MCP interfaces. A Content Access Gateway PEP controls access to knowledge bases for RAG retrieval. Each PEP enforces decisions but does not make them alone. The decisions are made by a central policy decision point (PDP), also known as the identity middleware. The PDP evaluates authentication, authorization, intent, delegation chains, and content classification policies. The PDP is the brain; the PEPs are the enforcement arms.

Without this PEP/PDP architecture, every component in the ecosystem connects to every other component without governance. AI agents access any model, any API, any content store, with whatever credentials they happen to have. The result is not a governed enterprise system. It is a network without access control, and no enterprise would deploy a network without access control.

A Day in the Life of an AI Agent

To demonstrate these risks, consider what a typical enterprise AI agent does during a single task and what occurs when identity governance is not present.

A procurement AI agent receives a task from an enterprise application: evaluate vendor proposals for a new cloud infrastructure contract, analyze pricing against budget authority, and prepare a recommendation for the procurement committee. This task is a legitimate business workflow that organizations are already automating with AI agents.

Here is what the AI agent does:

1. Authenticates to the enterprise's vendor management system using a service account.
2. Queries the vendor database for all active proposals, retrieving pricing, terms, and performance history.
3. Accesses the company's financial system to check current budget allocations and spending authority limits.
4. Queries the HR system to verify the requesting manager's authorization level.
5. Connects to external market data APIs to benchmark pricing.
6. Generates a comparative analysis, writes the recommendation to the procurement workflow system, and spawns an AI subagent to verify vendor compliance certifications through a third-party service.

The following events occurred while the AI agent was performing the task. The AI agent authenticated with a shared service account, the same credential used by dozens of other AI agents across the organization. There is no way to distinguish this AI agent's actions from any other AI agent using the same account. The AI agent queried the vendor database and retrieved all proposals, not just the ones relevant to cloud infrastructure, but every active proposal including those for classified defense contracts and pending mergers and acquisition evaluations. Nothing restricted its scope. The AI agent accessed the financial system and read not just the relevant budget line, but all budget data it could reach, because its service account had broad read access. The AI agent queried the HR system, a system entirely outside its intended scope, because its reasoning chain determined that verifying the manager's authorization level required HR data. No policy prevented this lateral movement from procurement into HR. The AI subagent the procurement AI agent spawned inherited the parent's credentials and access scope, with no reduction in privilege and no record of the delegation chain.

None of these events required malicious intent. The AI agent was performing exactly as intended, comprehensively evaluating the procurement decision. The problem is that *comprehensively* for an autonomous reasoning engine means accessing everything it can reach that might be relevant, and the boundaries of *relevant* are determined by the LLM's reasoning, not by pre-coded business rules. There was no authentication of the AI agent's own identity, no authorization scoped to its specific purpose, no intent validation, no delegation governance, and no audit trail that a regulator could reconstruct. The AI agent did its job. Nobody can prove that it stayed within its boundaries, because no boundaries existed.

The Agentic Shift: Three Risks No Existing Architecture Was Built For

The enterprise shift to agentic AI represents the most fundamental change to the security perimeter since the invention of the web application. AI agents are something genuinely new in enterprise computing: autonomous entities that reason, decide, and act across enterprise systems on behalf of users without human intervention at each step, and communicating entirely through APIs.

This transition creates three primary risks that no existing security architecture was designed to handle.

The first risk is unmanaged machine identities. Every AI agent that operates within an enterprise is, from a security perspective, a new identity that authenticates, holds credentials, is granted permissions, and takes actions. Unlike human identities which enterprises have spent decades learning to manage through IAM and PAM systems, machine identities for AI agents are proliferating without governance. Enterprises are deploying AI agents with service accounts, shared credentials, or overly broad API keys, because there is no established framework for AI agent identity lifecycle management. The result is an expanding attack surface of unmanaged, non-auditable, over-privileged machine identities. This environment provides the exact conditions that existed before every major breach.

Enterprises have spent decades building identity governance for humans (offboarding, role changes, periodic access reviews). None of that work extends to AI agents today. When an AI agent is deprecated or its purpose changes, its credentials and permissions typically persist and accumulate unchecked. The governance gap is real and it is growing with every AI agent deployed.

The second risk is data exfiltration through nondeterministic reasoning. This is the risk that distinguishes AI agents from every previous class of software. Traditional applications are deterministic. If they are given the same input, they produce the same output and access the same resources. AI agents are fundamentally nondeterministic. They reason over data, decide which systems to query, determine what information to retrieve, and synthesize responses in ways that cannot be fully predicted from their inputs.

This nondeterminism means that an AI agent granted legitimate access to a system may use that access in ways its operators never intended. An AI agent authorized to query a customer database for billing purposes may, through its reasoning process, access demographic data, cross-reference it with another system, and surface sensitive information in its output. The issue occurred not because the AI agent was programmed to access sensitive information, but because its reasoning chain led it there. The traditional security model of authorize access to a resource is insufficient when the entity accessing the resource makes autonomous decisions about how to use what it finds. Data exfiltration in the AI era does not require a malicious actor. It requires only an AI agent with broad permissions and an objective that intersects with sensitive data.

The third risk is the lack of auditable intent. In human-driven workflows, intent is implicit. When an employee queries a database, there is an understood purpose tied to their role, their current task, and organizational context. When an AI agent queries the same database, its intent is embedded in a reasoning chain that is opaque to administrators, auditors, and regulators. The AI agent has an objective, but the path it takes to fulfill that objective through the systems it accesses, the data it reads, and the conclusions it draws is not predetermined and often not visible after the fact.

Regulators are already focused on this behavior. The EU AI Act, SEC cybersecurity disclosure rules, and banking regulators across the globe are converging on a single principle. Enterprises must be able to explain and justify what their AI systems did, why they did it, and whether they stayed within authorized boundaries. Without auditable intent, enterprises cannot satisfy these requirements. For the regulated industries (financial services, healthcare, government, and critical infrastructure) that form our strongest customer base, the inability to demonstrate auditable intent is a deployment blocker.

Identity Is Not a New Problem

There is an important historical perspective that puts the current moment in context; every fundamental shift in computing architecture has required a new identity control plane. Identity is not a new problem, but a recurring one where the companies that solve the issue first define the category.

When mainframes dominated enterprise computing, identity was provided by the Resource Access Control Facility and Top Secret™ products. These systems controlled who could access which datasets and programs on the mainframe. When client-server architectures emerged, LDAP and Kerberos provided identity for networked environments where users accessed resources across multiple servers. When web applications moved enterprise systems outside the data center firewall, SSO and web access management, through SiteMinder, provided identity for the web era. When APIs and microservices disaggregated the monolith, OAuth2 and API gateways provided identity for machine-to-machine communication.

Each transition followed the same pattern. A new computing paradigm introduced new types of actors accessing new types of resources through new types of interactions. The existing identity tools, designed for the previous paradigm, could not govern the new one. A new identity control plane was required. The company that built it defined the category.

AI agents are the next paradigm. They are a new type of actor that is autonomous, reasoning, and non-deterministic. AI agents access enterprise resources with new types of interactions through MCP servers, function calling, A2A delegation, and RAG retrieval. The existing identity tools, designed for human users and deterministic API calls, cannot govern AI agents that reason about what to access, delegate credentials to AI subagents, and make runtime decisions that no developer precoded.

What identity must become for AI agents goes beyond what it provided for humans. AI agents need their own identities, not borrowed human credentials or shared service accounts. AI agents need first-class identities with their own lifecycle: provisioning, credential issuance, scope binding, rotation, and revocation. AI agents need intent-based authorization. The authorization cannot be just the ability for the entity to access a resource. The authorization must allow an AI agent to accomplish a specific objective, through specific systems, on behalf of a specific user, within specific constraints. AI Agents need delegation governance to track who authorized which AI agent to act on whose behalf, with what scope, and to what depth. AI agents need identity-attributed audit that is reconstructible after the fact, where every action is tied to a specific AI agent identity, a specific delegation chain, and a specific policy decision.

The Governed AI Agent: Same Day with Identity

Return to the procurement AI agent from the previous scenario. Same task, same systems, same business objective, but with identity governance in place.

The procurement AI agent authenticates with its own unique identity, not a shared service account. The credential is issued specifically to this AI agent, scoped to procurement workflows, with a defined delegation chain linking it to the requesting user and the enterprise application that launched it. Before accessing any resource, the AI agent's request passes through a PEP. The PEP is an API gateway that evaluates the request against the PDP (identity middleware).

When the AI agent queries the vendor database, the PEP enforces scope. The AI agent can access proposals related to cloud infrastructure contracts, not defense contracts or mergers, or acquisition evaluations. Its declared intent to evaluate cloud-infrastructure vendor proposals is captured in its authorization context. When the AI agent accesses the financial system, the PDP evaluates whether a procurement AI agent is authorized to read budget data, and if so, limits the scope to the relevant cost center. When the AI agent's reasoning chain leads it toward the HR system, the PEP denies the request. The AI agent's intent context does not include HR data access, and its delegation scope does not authorize cross-domain queries from procurement into human resources. The lateral movement that occurred in the ungoverned scenario is prevented by policy, not by luck.

When the AI agent spawns an AI subagent for vendor compliance verification, the delegation is recorded. The AI subagent receives a narrower scope than its parent (compliance data only, read-only, time-limited), and the full delegation chain is captured in the audit trail. The AI subagent cannot escalate its privileges beyond what the parent explicitly delegated.

At the end of the workflow, a complete, identity-attributed audit trail exists. Every API call, every resource access, every delegation, every policy decision is tied to the AI agent's identity, the user on whose behalf it acted, the intent it declared, and the policy that governed it. A complete, reconstructible evidence trail can be provided when a regulator asks, "What did this AI agent do, and was it authorized?" This is the difference between ungoverned AI and governed AI. The AI agent did the same work, but now there is evidence that it stayed within its boundaries.

The Governance Imperative: Security, Observability, and Identity

These three risks all point to the same missing piece, a hardened governance plane purpose-built for autonomous AI entities. You cannot solve each risk independently. An identity solution that does not cover the API layer leaves AI agent actions ungoverned. An API gateway that does not understand identity context cannot enforce meaningful policy. Without observability, you have no way to know whether your security is working, or your AI agents are staying within their authorized boundaries.

This last point, observability, deserves more than a passing mention. Observability is the pillar that separates governed AI from ungoverned AI in the eyes of regulators, auditors, boards, and customers.

Security prevents bad outcomes, and observability proves that you prevented them. In the agentic world, this distinction becomes existential. An enterprise can deploy the most rigorous identity and access policies imaginable, but those policies might as well not exist from a compliance perspective if it cannot demonstrate to a regulator what its AI agents actually did. The audit must show what systems the AI agent accessed, what data the AI agent consumed, what reasoning path led to each action, and was every step performed within authorized boundaries. Observability is not a reporting feature layered on top of security. It is the mechanism that makes security provable, auditable, and defensible.

The challenge with observability for AI agents is that traditional monitoring is inadequate. Application performance monitoring tells you whether a service is up. Log aggregation tells you what events occurred. Agentic observability requires something deeper: the ability to trace an autonomous AI agent's full execution chain across multiple systems, correlate every action to the identity that performed it, capture the authorization context that governed each step, and reconstruct the AI agent's intent trail after the fact. Observability must be identity-aware: every trace, every log entry, every metric must be bound to a specific AI agent identity, its delegation chain, and the policy that authorized the action. Without identity attribution, observability produces noise. With identity attribution, observability produces evidence.

Security, observability, and identity are not three separate capabilities to be sourced from three separate vendors. They are three dimensions of a single governance plane, and they only work when they are integrated. Security defines what AI agents can do. Observability captures what AI agents did do. Identity is what binds the two together; it gives security a subject to enforce against and gives observability a subject to attribute to.

What is needed is a governance layer that manages the full identity lifecycle of every AI agent, including creation, credential issuance, scope definition, delegation, rotation, and revocation. The governance must be applied with the same rigor that enterprises apply to human identities and privileged accounts. It must enforce fine-grained, context-aware authorization at every API interaction. AI agents can only access what they are explicitly permitted to access, for the purposes they are authorized to pursue. The governance layer must provide end-to-end, identity-attributed observability across the full AI agent execution path. Observability is more than just logging, it provides a complete, auditable evidence trail that can answer any question a regulator, auditor, or board member asks about what an AI agent did and why.

The Broadcom® Governance Plane: IDSP, PAM, and Layer7 API Gateway

Broadcom provides this governance plane today through three platforms that are already in production at enterprise scale: Symantec® Identity Security Platform (IDSP), Symantec Privileged Access Management (PAM), and Layer7 API Gateway.

IDSP

IDSP is the modern, cloud-native identity engine at the center of the governance plane. Built on a full OAuth2/OIDC implementation with fine-grained scope-based policy enforcement, delegated identity chains, and multi-tenant architecture; IDSP provides the identity control plane that AI agents require.

IDSP manages the complete AI agent identity lifecycle: issuance, delegation, scope binding, and revocation. The platform enforces authorization policies that go beyond simple access control to include purpose limitation, temporal constraints, and delegation depth. The platform provides the observability substrate that regulators and auditors demand by generating a comprehensive audit trail of every token issued, every scope granted, and every authentication event.

IDSP is not a legacy IAM system with extended AI capabilities. It is a platform built from inception for the architectural realities of modern enterprise computing, and the architectural decisions made turn out to be exactly what the agentic world now demands.

Architecturally, IDSP serves as the PDP, and the platform evaluates every authentication, authorization, and intent policy decision. It is the system that PEPs query to determine whether an AI agent's request should be permitted, denied, or constrained. When an API Gateway, acting as a PEP, intercepts an AI agent's request to access a resource server through an MCP interface, it queries IDSP. The platform determines if the AI agent identity is valid, authorized for this intent, supported within the delegation chain for this scope, and permitted to access the data classification of the target resource. IDSP evaluates this information and returns a policy decision. The PEP enforces the policy.

This PDP and PEP architecture extends naturally to the new access patterns that agentic AI introduces. For RAG use cases, where AI agents retrieve content from knowledge bases, a content access gateway PEP enforces content classification and access authorization based on the AI agent's identity and declared purpose. For model access, a model gateway PEP controls which AI agents can use which LLMs, at what cost tier, and with what data classification constraints. For A2A delegation, IDSP tracks the full delegation chain and ensures that each link in the chain narrows rather than expands the authorization scope.

Agentic security is not primarily about protocols or tokens. OAuth2 and OIDC are the transport layer. They handle how AI agents authenticate and carry credentials. The real security challenge with AI agents is not authentication, it is authorization at the level of intent. Traditional access control asks, "Can this entity access this resource?" That question is insufficient for an autonomous AI agent, because the AI agent does not just access a resource. The AI agent reasons over what it finds and decides what to do next. The correct question for agentic security is, "What is this AI agent trying to accomplish, and is that intent within its authorized boundaries?"

The IDSP architecture provides a decisive advantage. The platform was not built around coarse-grained role assignment. IDSP provides fine-grained, scope-based policy enforcement with precise, purpose-limited, time-bound, and context-aware grants. When this platform was designed, the goal was to solve the complexity of modern enterprise authorization. What was built is an intent-based policy engine. The system can enforce not just the access the AI agent has to an API, but enforce if the AI agent can access the API for billing purposes, on behalf of a specific user, within a specific amount of time, and must not access demographic data. This precision is exactly what agentic security requires, policy that governs intent and purpose, not just access.

IDSP was built to run anywhere (on-premises, cloud, hybrid, multi-cloud), because enterprises exist across environments. The platform was built multi-tenant from the start, because enterprise identity governance requires isolation between organizational boundaries. Delegation chains were built because the future of identity is not just about who is this user, but about *who authorized this AI agent to act on this user's behalf, with what constraints, and to what depth*. Each of these decisions, made before the current AI moment, turns out to be the correct architecture for governing autonomous AI agents. IDSP provides the correct technology stack at the right time.

The most compelling proof that IDSP works at enterprise scale is not its feature set. It is what was done with it inside Broadcom. IDSP was used to completely replace Broadcom's own Okta deployment. This work includes customer identities that span more than 9000 identity providers and workforce identities spanning more than 2500 applications.

PAM

PAM extends the governance plane into privileged and sensitive access. The same rigor must be applied to AI agents as privileged human users that interact with critical systems such as, databases, infrastructure controls, and financial systems. The AI agents require session recording, just-in-time access, credential vaulting, and behavioral monitoring. PAM ensures that when an AI agent accesses a privileged resource, the access is scoped, time-bound, recorded, and revocable. This access is essential for addressing the nondeterministic reasoning risk. When an AI agent's reasoning chain leads it toward sensitive data, PAM enforces the boundary that prevents unauthorized access and captures the audit trail that proves compliance.

Layer7 API Gateway

Layer7 API Gateway is the enforcement point where identity-driven policy meets the reality of API traffic. Every autonomous action an AI agent takes is, at its core, an API call. Layer7 API Gateway is already deployed in the data path of enterprise API traffic across thousands of customers. The platform enforces rate limiting, payload inspection, scope validation, and threat detection at the API layer. This enforcement ensures that every interaction between an AI agent and an enterprise system passes through a governed control point. Broadcom does not need to convince enterprises to insert a new component into their architecture; Layer7 API Gateway is already there.

Observability

Observability across the governance plane is not a separate product added on at the end. It is woven into every layer of the stack. IDSP generates a complete identity event stream: every token issued, every scope granted, every authentication and authorization decision, and every delegation chain exercised. PAM captures full session recordings and behavioral logs for every privileged interaction an AI agent has with a sensitive system. Layer7 API Gateway logs every API call that passes through the gateway.

Unlike generic monitoring, identity is built into the entire observability stack. Every event generated by IDSP, every session recorded by PAM, and every API log captured by Layer7 API Gateway can be correlated through a single AI agent identity. An enterprise can answer the question every regulator will ask,"What did this AI agent do, on whose behalf, and was it authorized?" The answer is not provided by piecing together logs from five different vendors, but from a single, identity-attributed observability trail that spans the full execution path.

From Modern Identity to Agentic Governance

The shift from securing human users to governing AI agents does not require a complete re-architecture of enterprise security. It requires extending identity, access control, and observability systems into the places where AI agents operate: at the API boundary, at the model interface, at the data retrieval layer, and across A2A delegation chains.

IDSP, PAM, and Layer7 API Gateway were not built specifically for agentic AI. The platforms were built to solve the foundational problems of modern enterprise identity: scope-based authorization, delegation chains, credential lifecycle management, privileged access governance, API-level policy enforcement, and identity-attributed observability. However, these features are precisely the capabilities that agentic AI now requires. AI agents are not a discontinuity from the past, they are the logical next step in a progression from human-driven workflows, to API-driven integrations, and to autonomous, reasoning entities.

The key point is that each of these capabilities builds on the architectural foundation that already exists. Broadcom is not redesigning the platforms for agentic AI. Instead, platforms that were designed for modern enterprise computing with fine-grained authorization, delegation, API governance, and observability are extended into the agentic domain. The architectural decisions that were made for modern identity turn out to be the right decisions for agentic identity. The problems are fundamentally the same: managing lifecycles, enforcing intent, attributing actions, and providing evidence.

The Imperative

The shift to agentic AI is not theoretical. It is happening now. Early adopters are deploying AI agents into production workflows. The question is not whether enterprises will adopt agentic AI, but whether they will govern it before it becomes a liability.

The risks are real and measurable:

- Unmanaged machine identities proliferate with every AI agent deployed, creating credential sprawl and expanding the attack surface in ways traditional IAM cannot track.
- Non-deterministic reasoning makes it impossible to predict or audit what data an AI agent will access. Traditional access control is insufficient for autonomous actors that reason about what to do with the resources they can access.
- Lack of auditable intent means enterprises cannot demonstrate to regulators, boards, or customers that their AI agents stayed within authorized boundaries.
- Without delegation governance, every A2A interaction becomes a credential inheritance problem, and every delegation must narrow rather than expand the authorization scope.
- Without identity-attributed observability there is no way to reconstruct what an AI agent did, why it did it, and whether it was authorized.

These are not future risks. These risks exist today in every organization experimenting with AI agents. The governance gap is open. The question for every enterprise is whether it will close this gap before or after the consequences arrive.

Broadcom provides the governance plane which includes IDSP for human and non-human identity, human and non-human risk, authentication, and authorization; PAM for privileged access; Layer7 API Gateway for API enforcement and unified observability across all three platforms. The governance plane makes agentic AI safe to deploy at enterprise scale. The architecture exists. The platform is proven. The time to govern is now.

For more information, visit our website at: **www.broadcom.com**

BAIF-SAAE-WP100 March 13, 2026