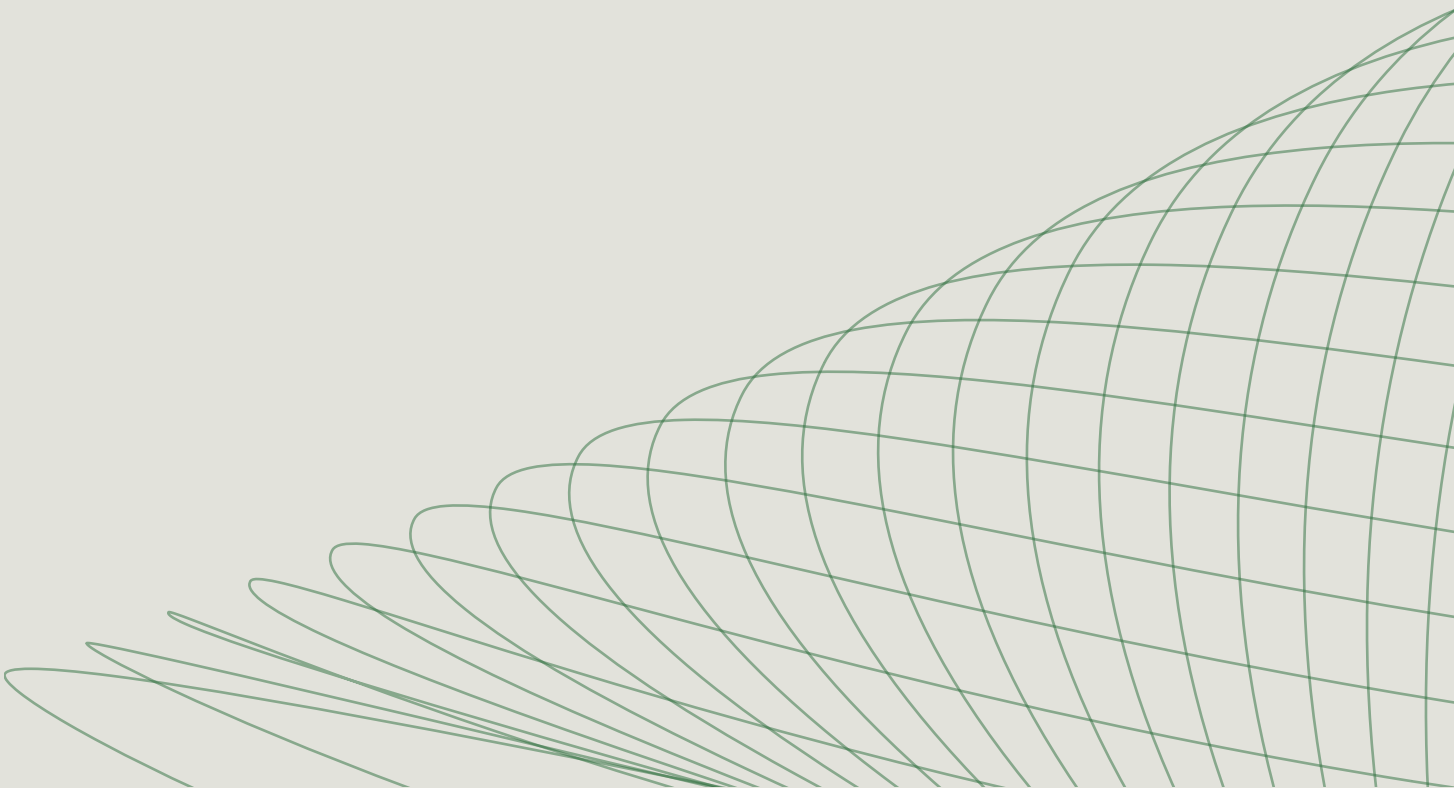
STACKLOK

AI Control Plane Buyer's Guide for Retail

Key considerations for leaders who need the governance and controls to run AI agents in production at scale



Contents

Over the past year, we have worked with retail enterprises to drive deeper adoption of AI agents and MCP servers. This guide distils what they told us matters most, and what separates governance theater from genuine control.

1. Why Retail Leaders Must Act Now

AI agents have already changed how retail enterprises operate. They are in production optimising inventory replenishment, personalizing customer experiences, automating supplier communications, and triaging contact center queues. Retailers are connecting agents to their core systems so they can take action. That means those agents are touching customer purchase history, pricing engines, loyalty data, supplier contracts, and order management platforms.

This transformation is happening with or without a governance framework in place. Engineering and data teams are standing up MCP servers that connect agents to product catalogs, OMS platforms, and customer databases, before any policy exists to govern those connections. By the time the platform team or security function becomes aware, those connections are embedded in workflows that teams depend on.

This creates a specific and urgent problem for retail technology leaders. The consequences of ungoverned AI agents in retail are not abstract. An agent with unaudited write access to a pricing engine can cause real commercial harm before anyone notices. An agent querying customer purchase history through a shared service account creates PCI DSS and GDPR exposure. An agent connecting to a supplier portal through an unvetted community MCP server is a supply chain risk inside your procurement workflow. These examples are the natural endpoint of adoption without governance.

What makes this moment particularly consequential is the compounding effect of delay. Retail moves fast, and so teams do not wait for governance frameworks before deploying tools that give them an edge. Every week without a control plane is a week in which more ungoverned deployments accumulate, more shadow integrations embed themselves into critical workflows, and more technical patterns calcify in ways that will be expensive to retrofit.

The leaders who resonate most with this challenge are the ones who own the board-level pressure to demonstrate an AI strategy. They believe in the technology and understand that to move with speed, they must have the right underlying control plane.

Retail enterprises that establish a genuine AI control plane now will define how their organization runs agentic AI through the next decade of commerce. Those who wait will spend that time in remediation: unpicking shadow deployments, responding to data incidents, and trying to retrofit controls onto an estate that was never designed to be governed.

2. The Reality on the Ground: What We See in Retail Today

The picture we encounter at retail enterprises evaluating AI infrastructure is remarkably consistent:

MCP sprawl has already started

Teams across merchandising, supply chain, e-commerce, and customer experience are independently connecting AI agents to enterprise systems. In organizations with multiple brands or business units, this fragmentation compounds quickly: different teams deploying different MCP servers against the same downstream systems, with no shared visibility and no unified policy layer.

Authentication is improvised and data exposure is uncontrolled

Most early MCP implementations rely on shared API keys, service accounts with broad permissions, or personal access tokens that expose far more of the underlying system than the agent needs. A customer service agent that looks up an order should not be connecting through a credential that also has write access to fulfillment instructions. That over-permissioning is standard today because the tooling to scope it precisely does not yet exist in most environments.

Security teams are blocking rather than enabling

The right response to security team concerns is to give them the audit logs, access controls, identity passthrough, and policy framework they need to say yes. Without that, every new MCP server requires a bespoke review, adoption slows to a crawl, and developers route around controls.

Governance is being invented ad hoc

Reference architectures for governing AI agents in retail do not yet exist at industry scale. Teams are making it up as they go. Patterns established under peak-season time pressure in one team bear no resemblance to those established by a different team six months later. The result is a patchwork of governance approaches that provides the appearance of control without the substance.

3. The Distinct Stakes for Retail

Retail enterprises face a version of the AI governance problem that is shaped by the specific characteristics of the industry. Several factors compound the challenge:

Customer data at scale creates outsized exposure. Retail enterprises hold some of the most voluminous consumer data of any sector: purchase history, browsing behavior, loyalty data, payment information, and delivery addresses. AI agents connecting to these systems through poorly governed MCP integrations create real PCI DSS, GDPR, and CCPA exposure.

Multi-banner and omnichannel complexity works against centralized control. Large retail groups operate across multiple brands, channels, geographies, and fulfillment models, each with its own technology stack and operational cadence. Without deliberate multi-tenancy, every new AI deployment creates a new governance exception.

Supplier and partner integrations expand the attack surface. Retail AI agents increasingly interact with external supplier portals, logistics providers, and marketplace platforms through MCP integrations. Community-sourced MCP servers connecting to supplier APIs may carry unvetted dependencies. Several enterprises we work with have already experienced supply chain incidents through AI developer tooling.

Peak season compresses the risk window. The most commercially critical periods of the year are also the periods when ungoverned AI deployments are most likely to proliferate and when the consequences of something going wrong are highest. The time to establish governance is before peak, not after an incident.

Speed-to-market pressure defeats slow procurement. By the time formal procurement processes complete, developers have already built workarounds. The enterprises handling this best are using open source evaluation paths to build internal confidence before a procurement conversation, and governance frameworks that are lightweight enough that the official channel is also the fast channel.

4. The Core Problems an AI Control Plane Must Solve

Before evaluating any platform, it is worth being explicit about the problem surface. Five categories of challenge consistently emerge as retailers try to scale AI agents from experimentation to production:

Identity and access control is the hardest technical problem. When an AI agent calls a tool (for example querying a customer's order history, updating a product description, or checking stock levels) which identity does the downstream system see? In most current implementations, it sees a shared service account. That breaks accountability, creates over-permissioned access to sensitive customer and commercial data, and makes it impossible to trace which action was triggered by which agent or user.

Governance and policy enforcement at scale is a fundamentally different problem from governance at five servers. Governing fifty MCP servers spanning merchandising, supply chain, customer experience, and finance requires a policy model that can express fine-grained controls. Not just who can access a server, but what tools they can invoke, under what conditions, and against which data. Those policies need to be declarative, version-controlled, and enforceable at runtime without redeployment.

Discoverability without shadow IT is a genuine tension. The harder you make it to consume MCP servers through official channels, the more retail teams will route around those channels. Without a curated registry and an approval workflow, you have no visibility into what is running against your production systems. The right answer is a registry that is frictionless enough to be the default path, not a compliance checkbox developers treat as an obstacle.

Operational observability is non-negotiable for production workloads. You need to know what MCP servers are running, who is calling them, what tools are being invoked, and whether those invocations are succeeding or failing. That telemetry needs to flow into your existing observability stack. In retail, you also need usage attribution granular enough to support cost chargeback across banners and business units, and anomaly detection capable of flagging unexpected agent behavior before it becomes a commercial incident.

Deployment flexibility across heterogeneous environments is a hard requirement. Retail technology estates are rarely homogeneous. You likely have Kubernetes clusters across multiple clouds, on-premises infrastructure in distribution centers, and a mix of local developer tooling across geographies. A platform that only operates in one deployment model will create governance gaps wherever it does not reach.

5. The Capability Checklist

Use this framework when evaluating vendors. These requirements reflect the real challenges that retail enterprises are working through as they scale AI agents to production. Where capabilities are table stakes, they are noted as such. Where they represent a higher bar that separates serious enterprise platforms from lighter-weight alternatives, that distinction is called out.

Section 1: Identity, Authentication & Audit

This is the category where the gap between credible enterprise platforms and lightweight alternatives is widest, and where the data protection and commercial risk stakes for retail are highest. Treat any vendor that cannot give detailed, concrete answers here with significant skepticism.

Capability	Why it Matters	What to Require
Enterprise IdP integration (Okta, Entra ID, Ping, Google)	Developers must authenticate through your existing identity provider. Separate credentials for MCP access fragment your IAM posture, which is particularly problematic when agents are accessing systems that hold customer PII or payment data.	Require native OIDC/OAuth SSO with your IdP, with no stored API keys or personal access tokens in client configurations. If you are on Entra ID, verify the vendor understands Entra's dynamic client registration limitations.

Per-user identity passthrough to downstream systems	When an AI agent calls a tool, the downstream system should log the actual user's or agent's identity, not a shared service account. Without this, you cannot trace a pricing error or data access event to its source.	Require token exchange flows (OAuth on-behalf-of / RFC 8693) that preserve and propagate end-user identity to downstream systems. Ask for an end-to-end architecture walkthrough of exactly how this works.
Short-lived, scoped tokens with automatic rotation	In retail, where a single service account may have access to pricing, customer data, and order management simultaneously, the blast radius of a compromised static credential is severe.	Require that the platform issues short-lived tokens per session or per request, with no requirement for developers to manage credential rotation manually.
Non-human identity support for agents	Inventory replenishment agents, pricing optimization agents, and supplier communication agents operate without a human in the loop. Your governance model needs a story for workload identity. Unmanaged service accounts for automated agents are a significant exposure.	Require support for workload identity patterns including SPIFFE/SPIRE-based JWT authentication for non-human agents running in automated pipelines and scheduled workflows.
Read-only and granular operation-level scoping	An agent that can read product data should not be able to update it. An agent that can query order history should not be able to modify fulfillment instructions. These are real operational requirements in retail, not theoretical constraints.	Require the ability to scope tool permissions by operation type (read, write, delete) at the policy level, not just at the server access level. Verify enforcement happens at the gateway, not just in client configuration.
Audit log export to your SIEM	In the event of a data incident, a pricing anomaly, or a supplier integration failure, an isolated audit log that cannot be correlated with your wider security telemetry is of limited value.	Require log export in formats compatible with your SIEM platform (Splunk, Elastic, Microsoft Sentinel, etc.). Confirm logs are tamper-evident, include full identity and tool call context, and can be retained for the periods your compliance framework requires.

Section 2: Governance, Policy & Compliance

Capability	Why it Matters	What to Require
Curated server registry with approval workflows	Without a registry, you have no visibility into what MCP servers are connecting AI agents to your production systems — customer databases, pricing engines, and OMS platforms. A shadow MCP server connecting to any of them is an uncontrolled risk.	Require a registry that supports metadata (owner, security review date, data classification, approved client list) and a vetting workflow that integrates with your existing approval processes. The registry must be easy enough to use that it is the default, not an obstacle teams route around.

Supply chain security for MCP servers	In retail, where supplier and logistics integrations are frequent targets for credential theft, the supply chain risk from unvetted MCP servers is amplified. Several enterprises we work with have already experienced supply chain incidents through AI developer tooling.	Require CVE scanning and SLSA attestation for all servers in the registry. Ask whether the vendor maintains hardened versions of high-usage community MCP servers (Salesforce, SAP, Shopify, Atlassian, etc.) with ongoing security patch commitments.
Granular, declarative policy engine	A store associate agent, a merchandising analyst agent, and a supplier management agent should each see a different, appropriately scoped set of tools, not the same server with the same permissions.	Require a policy engine that supports RBAC, ABAC, and claim-based authorization at both the server and tool invocation level. Policies should be declarative code. Verify compatibility with your existing policy framework (Cedar, OPA/Rego, or equivalent).
Policy changes without redeployment	Retail governance needs to respond fast to a live pricing incident, a data access anomaly, or an agent behaving unexpectedly during a peak trading period. If changing a policy requires a redeployment cycle, your incident response is constrained at exactly the wrong moment.	Require that policy changes take effect at runtime without server or gateway redeployment. This is a meaningful differentiator that many platforms cannot deliver.
Multi-banner and multi-business-unit controls	Large retail groups operate across banners, channels, and geographies that each have their own technology stacks and operational requirements. A governance model that cannot scope controls to banner or business unit level is inadequate for how retail organizations actually operate.	Require namespace-level isolation with RBAC controls that enable different banners, business units, or regional teams to have distinct server catalogs and access policies without central admin intervention for routine changes.
Client-side enforcement hooks	In retail engineering teams under delivery pressure, a developer bypassing the server layer and connecting a local MCP server directly to a production system is a common behavior when the official path is too slow.	Require hooks-based enforcement for your approved IDE clients (VS Code, Cursor, Windsurf, Claude Code). Ask specifically which clients are supported today, and get explicit answers on any gaps.
Tool filtering and virtual MCP server composition	An agent that only needs to check order status should not be presented with every capability in your OMS. Over-exposure is both a governance risk and a performance problem.	Require the ability to define virtual MCP servers tuned per role, team, or use case, exposing only the tools appropriate to that context. This is a meaningful differentiator that most lightweight platforms do not support.

Section 3: Runtime Security & Deployment

Capability	Why it Matters	What to Require
Kubernetes-native operator	MCP servers need to be managed like any other production workload, with lifecycle management, native scheduling, and integration into your existing operational tooling. Adding an isolated layer for AI agents creates unnecessary overhead and governance gaps.	Require a native Kubernetes operator with CRD-based server lifecycle management. Verify support for your specific distribution (EKS, AKS, GKE, OpenShift, or on-premises).
Container isolation per server	In a retail environment where a single MCP estate may contain servers connecting to customer data, pricing systems, and supplier portals simultaneously, a compromised server in a shared runtime can affect all others.	Require per-server container isolation with configurable network egress and filesystem access controls. For environments handling payment data or customer PII, ask specifically about hardware-isolated microVM runtimes (Kata Containers, Firecracker).
Fully self-hosted deployment	Retail enterprises with data residency requirements, on-premises infrastructure in distribution centers, or cloud environments subject to internal data handling policies cannot always route AI agent traffic through external SaaS platforms.	Confirm the platform is fully self-hostable with no required SaaS components or external callbacks. Verify this is available now, not on the roadmap.

Section 4: Observability & ROI Measurement

You cannot govern what you cannot see — and in retail, where AI agent spend must be justified against measurable trading outcomes, you cannot build a business case for something you cannot measure.

Capability	Why it Matters	What to Require
OpenTelemetry-native telemetry	You already have an observability stack. Your AI agent telemetry should flow into it, not create a parallel system to maintain. In retail, where engineering teams manage complex observability across distributed commerce platforms, a separate monitoring layer adds overhead without adding insight.	Require OTLP-compatible traces, metrics, and logs following official OpenTelemetry MCP semantic conventions. Verify export to your existing backend (Datadog, Splunk, Grafana, Dynatrace, Prometheus, etc.).
Baseline and anomaly visibility	In retail, unexpected agent behavior is often a commercial risk before it becomes a security risk. An agent making unusually high-frequency calls to a pricing API, or accessing customer records outside its normal pattern, may indicate a misconfiguration or a compromise.	Require sufficient telemetry granularity to establish per-user, per-team, and per-server usage baselines over time. This is the foundation for detecting compromise or misconfiguration before it affects trading operations.

Usage attribution for cost chargeback	In retail groups operating across multiple business units, AI infrastructure costs need to be attributable to the part of the business generating them.	Require telemetry granular enough to support chargeback models: tool call counts, token consumption estimates, and user-level attribution available in the telemetry stream.

Section 5: Developer Experience & Adoption

The best AI control plane platforms make the governed path the easy path. In retail, where engineering teams face constant delivery pressure, a cumbersome governance layer is not just a friction problem — it is an active driver of shadow IT.

Capability	Why it Matters	What to Require
Self-service discovery portal	Retail data and engineering teams need to find approved MCP servers without filing a ticket or waiting for an administrator. Adding approval latency to every new AI integration is a reliable way to guarantee that teams find workarounds.	Require a self-service portal that developers can access without admin assistance, supporting both a human-readable UI and a machine-readable API for programmatic integration.
Automatic IDE client configuration	Requiring developers to manually configure each IDE client for every MCP server is friction that slows adoption and creates inconsistency across the estate. In retail teams that span multiple tooling environments and geographies, manual configuration does not scale.	Require automatic client configuration: when a developer gains access to a server, their IDE should be configured automatically, not through a manual YAML edit.
Context optimization	As MCP server counts grow and span data, OMS, CRM, supplier portals, and analytics platforms, context window bloat degrades agent performance. An agent handling a customer query should not be presented with every capability across your commerce stack.	Require a gateway-level context optimizer that dynamically surfaces relevant tools based on the task, rather than flooding the model with every available tool on every request.
Composite tool and workflow support	High-value retail workflows require sequencing multiple tool calls across multiple systems in a defined order. Leaving this to the LLM produces inconsistent results that are difficult to audit.	Require the ability to define declarative composite tools that chain multiple underlying tool calls into a single, deterministic workflow. Ask for a demo with a realistic multi-system retail workflow.
Support for both local and remote server models	Retail engineering teams prototype locally and deploy centrally. Both models are valid and both will be in use simultaneously. The platform must govern both with the same policies, telemetry, and identity controls.	Require that local and remote server deployments are subject to the same registry policies, the same telemetry requirements, and the same identity controls.

Section 6: Commercial & Vendor Considerations

Capability	Why it Matters	What to Require
Open source core with commercial enterprise layer	In retail, where formal procurement processes can be slow relative to engineering delivery cycles, the ability to evaluate a platform without a commercial conversation is genuinely valuable. Open source also signals a vendor philosophy compatible with the build culture common among retail engineering teams.	Require an Apache 2.0 (or equivalent permissive) licensed core that includes the runtime, registry, and gateway. Enterprise tiers should add hardening, support SLAs, and convenience features.
Production support SLAs	When AI control plane infrastructure becomes operationally critical, you need a vendor with defined escalation paths. Community support is not adequate for infrastructure that affects trading operations.	Require defined SLA tiers with escalation paths. Confirm support coverage matches your operational hours, and that critical incident response times are contractually defined.
Outcomes-oriented engagement model	Retail technology teams are results-oriented and operate under delivery pressure. The most successful deployments involve vendor engineering resources working alongside your team to implement, validate, and iterate.	Ask specifically about forward-deployed engineering availability. Is there a dedicated deployment engineer for the initial rollout? Will engineers write code alongside your team? A vendor who can embed is worth significantly more than one who sends documentation.
Modular architecture ('use what you need, bring what you have')	Your organization has already invested in identity systems, policy engines, observability platforms, and integration middleware. A platform that requires you to replace existing investments is a harder internal sell, and introduces migration risk that retail technology organizations typically cannot absorb.	Require explicit confirmation of which components are swappable. Can you use your existing OPA/Rego policies instead of Cedar? Can the registry coexist with your existing artifact management? Every confirmed modularity point de-risks the procurement conversation.

6. Prioritizing Your Evaluation

Not every capability carries equal weight for every organization. Use the following framework to sequence your evaluation based on your current situation.

If your primary concern is customer data protection and compliance

Lead with per-user identity passthrough, SIEM-ready audit log export, configurable retention, and granular operation-level scoping. These capabilities allow you to demonstrate to your data protection officer, legal team, and external auditors that AI agents accessing customer data are doing so under appropriate controls with a complete, attributable audit trail.

If your primary concern is commercial risk, including pricing, inventory, and trading operations

Lead with granular policy enforcement, tool filtering and virtual MCP server composition, and policy changes without redeployment. An agent that touches a pricing engine or inventory allocation system needs precisely scoped access, and your governance layer needs to be able to respond to anomalies in real time.

If your primary concern is security team buy-in

Lead with authentication and identity passthrough, client-side enforcement hooks, supply chain security, and container isolation. These capabilities allow security teams to say yes to broader adoption rather than defaulting to binary block decisions. A platform team's job is often to get security comfortable enough to unlock rollout.

If your primary concern is developer adoption without shadow IT

Lead with the self-service discovery portal, automatic client configuration, and the open source evaluation path. In retail engineering teams under delivery pressure, the governed path must also be the fast path. Developers who hit friction do not escalate, they build the workarounds you are trying to prevent.

If your primary concern is scaling from your current footprint

Lead with the Kubernetes operator, multi-banner controls, virtual MCP server composition, and context optimization. The patterns you establish for five servers and one banner need to hold for fifty servers across ten banners. Platforms that seem adequate at small scale consistently fail to make that transition gracefully.

If your primary concern is justifying AI investment to the board

Lead with observability, usage telemetry, and tool call performance instrumentation. Board-level AI investment in retail gets justified by measurable outcomes: contact center deflection rates, inventory accuracy improvements, productivity gains in merchandising. You cannot construct that ROI narrative without granular telemetry attributable to specific use cases and business units.

7. What the Market Actually Looks Like

The AI control plane and MCP platform market is early. Several vendors have entered the space, and it is worth being clear-eyed about where they focus and where they fall short when evaluated against retail enterprise requirements.

Lightweight gateway tools can proxy MCP traffic and provide basic routing, but they typically lack the runtime management, supply chain security, multi-tenant governance, and identity depth that production retail deployments require. They are useful for prototyping. They are not adequate for governing a multi-team MCP estate that connects to customer data, trading systems, and supplier integrations.

Developer-tooling platforms that have added MCP support often treat governance as a secondary concern. They are optimised for individual developer productivity, not organizational deployment at scale. Their audit trails are thin, their identity integrations are limited, and they were not built with the data protection, commercial risk, or multi-banner complexity of large retail enterprises in mind.

Cloud provider offerings address deployment but typically lock you into a single-cloud architecture, lack the vendor-neutral flexibility that heterogeneous retail environments require, and do not address the developer adoption and agent performance challenges that retail technology teams actually face.

The right platform will be purpose-built to solve the enterprise AI governance problem: a Kubernetes-native operator, a full identity and policy stack, supply chain security, multi-tenant controls, and an open source core that lets you evaluate without asking for permission. It treats governance not as a constraint on developer productivity, but as the foundation that makes broader, safer adoption possible.

The questions in this guide will help you tell the difference.