Enterprise Context Layer for AI Agents
Reference Architecture

Executive summary

The limiting factor for enterprise agents is increasingly not access to a capable model. It is access to the right enterprise context under the right identity and policy, with enough evidence to support a trustworthy response or action.

Enterprise context includes documents and policies, but also current operational state, workflow status, user authority, asset relationships, business definitions, effective dates, and permitted actions. The Enterprise Context Layer is a policy-aware mediation layer that provides this context without requiring every agent to become its own integration platform. It provides stable interfaces for context requests, identifies relevant approved sources, retrieves or composes information, applies policy and minimization, attaches source and freshness evidence, and returns a structured context package to the agent.

The model performs reasoning. The agent runtime manages the interaction loop and approved tool calls. The governed workspace constrains execution. Source systems remain authoritative for their data and permissions. Purpose-built action services independently control state-changing operations.

Executive definition: The Enterprise Context Layer gives agents the right enterprise context, for the right user and task, from the right sources, with evidence and control.

Introduction

A model can generate, summarize, and reason over information placed in its context. An enterprise agent adds a runtime that can maintain a session, call tools, and propose or perform actions. Neither component automatically knows which enterprise sources are authoritative, what a field means, whether a user may access a record, how current the information is, or which next action is permitted.

The Enterprise Context Layer provides AI agents with a consistent, governed interface to authorized enterprise information while preserving source-system ownership and access controls. It retrieves and composes task-relevant context, applies enterprise policy, and returns the result with provenance, freshness, limitations, and permitted next actions.

This paper treats context as an enterprise architecture concern. Context is not simply a prompt or retrieved document. It is an authorized, task-relevant representation of enterprise knowledge and current state supplied to an agent for a defined purpose.

The **Intended Audience** for this paper is tailored for individuals or teams with a strong background in:

- Enterprise and solution architects
- AI platform and agent-engineering teams
- Data, integration, and application architects
- Security, identity, privacy, and governance teams
- Infrastructure architects responsible for workstation, server, and cluster placement
- Service owners evaluating operational agents and workflow automation

The **Scope and Non-Goals** considered in this paper are:

- The paper covers identity-aware context access, source registration, structured and unstructured retrieval, multi-source composition, business meaning, policy filtering, provenance, freshness, security boundaries, deployment, operations, and validation.
- It does not define a new agent framework or model, replace source-system authorization, require centralized replication of all enterprise data, replace sandboxing or inference services, or give agents authority to approve high-impact actions.

Business Problem and Use Case

This section explains why direct source connectivity does not by itself provide trustworthy enterprise context. It identifies the recurring quality, governance, and integration problems that the architecture is designed to address.

Enterprise agents do not fail only because they lack access to data. They also fail when information supplied to the model is incomplete, inconsistent, stale, over-scoped, or disconnected from its business meaning. Connecting an agent to more systems increases available information but does not establish which sources are authoritative, how results should be combined, what the requester may see, or whether the result is current enough for a decision.

In this section we cover the following topics:

- Enterprise Agent Context Problem
- Direct-Connection Anti-Pattern
- Why RAG and MCP Are Necessary but Not Sufficient

Enterprise Agent Context Problem

The following recurring problems show why enterprise context must be governed as an architectural capability rather than assembled independently by each agent.

- **Context-quality fragmentation**: agents receive raw responses from different tools and must reconcile schemas, relationships, definitions, and timestamps.
- **Trust and governance exposure**: credentials, permissions, sensitive data, and state-changing tools may be exposed without consistent policy, minimization, or attribution.
- **Integration duplication**: each agent independently implements connectors, source selection, error handling, and audit behavior.

Direct-Connection Anti-Pattern

A direct-connection design multiplies integrations, credentials, and policy decisions across agents. The followiung figure contrasts that anti-pattern with a governed mediation layer that centralizes reusable context controls.

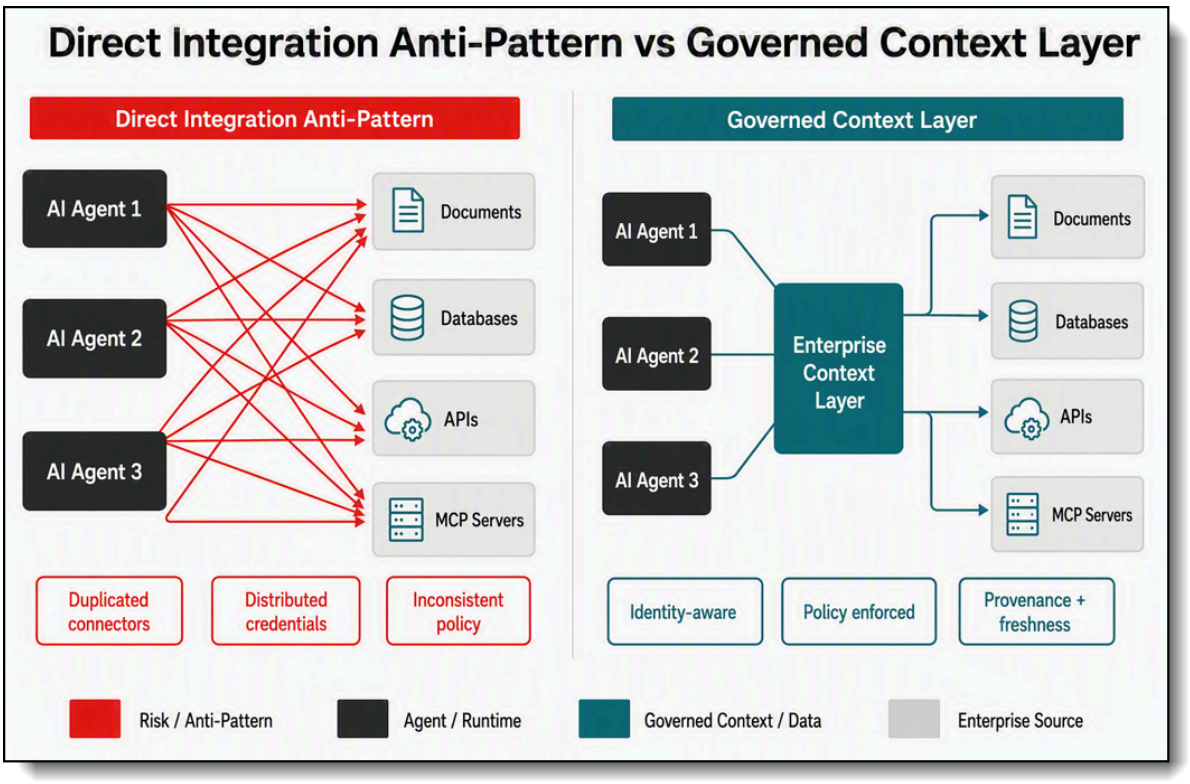


Figure 1. Direct-Connection Anti-Pattern vs Governed Context Layer

Why RAG and MCP Are Necessary but Not Sufficient

RAG and MCP each solve important parts of the context problem, but neither by itself supplies the complete set of enterprise governance, identity, freshness, and operational controls described here.

Retrieval-augmented generation is useful for discovering unstructured knowledge such as policies and runbooks. Operational agents also require structured, current state from systems such as asset repositories, lease databases, workflow platforms, and monitoring systems. A complete answer may require both document retrieval and live source queries.

MCP provides a standardized protocol for context exchange among hosts, clients, and servers. Official MCP documentation states that the protocol focuses on context exchange and does not dictate how applications use models or manage the context they receive. The Context Layer uses MCP where appropriate, while adding source governance, identity continuity, retrieval strategy, policy, provenance, freshness, tenancy, and observability.

- For details on Model Context Protocol, “Architecture overview,” documentation, see: https://modelcontextprotocol.io/docs/2026-07-28/learn/architecture

Solution Overview

The solution separates agent reasoning, governed execution, context mediation, authoritative sources, inference, and controlled actions. The following principles, architecture view, and request flows show how

those responsibilities work together.

Note, that the following is a working definition used throughout this publication.

> **Definition**: The Enterprise Context Layer provides AI agents with a consistent, governed interface to authorized enterprise information while preserving source-system ownership and access controls. It retrieves and composes task-relevant context, applies enterprise policy, and returns the result with provenance, freshness, limitations, and permitted next actions.

In this section we cover the following topics:

- Architectural Principles
- High-Level Reference Architecture
- Request and Data Flows

Architectural Principles

The design principles in the following table translate the governing intent of the Enterprise Context Layer into practical architectural implications. Lists in this table link each architectural principle to its design implication.

Table 1. Architectural principles and design implications

Principle	Design implication
Governed access path	Agents use approved context interfaces instead of unrestricted direct backend access.
Identity follows the request	User, workspace, agent, tenant, and delegated purpose remain attributable.
Capabilities, not raw secrets	Credentials remain outside prompts, model context, and agent memory.
Least context necessary	Return the smallest authorized result that satisfies the task.
Read and action separation	Retrieval authority does not imply authority to change system state.
Meaning and evidence travel together	Return business meaning, source, freshness, transformations, and limitations.
Stable contracts	Agents depend on context contracts rather than source-specific schemas.
Observable and revocable	The complete context and action path are traceable and access can be revoked.

High-Level Reference Architecture

The following figure presents the major logical layers and shows the Enterprise Context Layer positioned between the governed agent workspace and authoritative enterprise sources.

AI agents use a governed workspace and a shared Enterprise Context Layer to reach authorized sources and controlled action services.

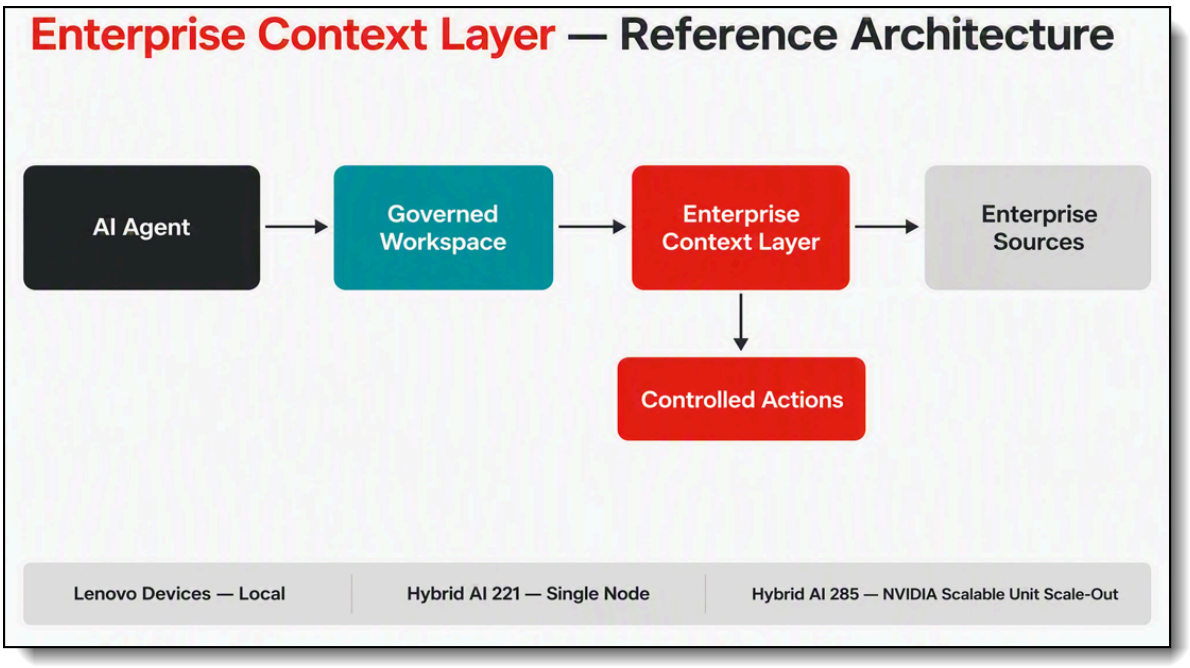


Figure 2. Enterprise Context Layer flow

Request and Data Flows

The following figure separates the read path from the action path. This distinction allows context retrieval and state-changing operations to use different authority, approval, and evidence controls.

The read path returns minimized, source-backed context; the action path applies separate control checks.

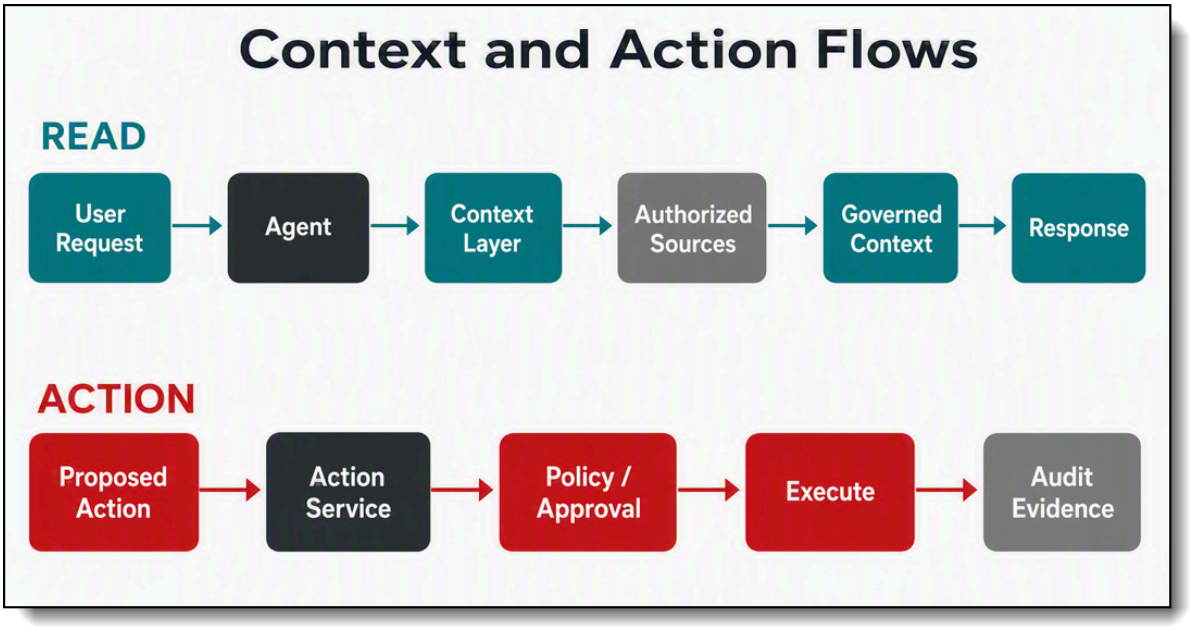


Figure 3. Context and Action Flows

Architecture Components

The architecture is organized as the following responsibilities. Each bullet describes a distinct boundary that can evolve independently while participating in the same governed context flow.

- Agent Runtime: Receives the user request, maintains the interaction loop, calls the configured model, exposes approved tool descriptions, executes selected tool calls, and returns the response.
- Governed Agent Workspace: Provides the execution and policy boundary for the agent, including sandboxing, network restrictions, file and process controls, and lifecycle management.
- Enterprise Context Layer Services: Provide the gateway, source registry, identity and scope mediation, retrieval and federation, semantic interpretation, policy filtering, provenance, freshness, scoped caching, and observability.
- Enterprise Sources: Remain authoritative for their own information, permissions, schema, quality, and transaction semantics.
- Shared Inference: Provides approved model endpoints independently from enterprise context services. The placement may be local, centralized, or hybrid.
- Controlled Action Services: Expose narrow, typed operations for state-changing work. They independently verify authorization, preconditions, approval, idempotency, and transaction evidence.

Lenovo Infrastructure: Lenovo infrastructure provides deployment choices for the Enterprise Context Layer, agent runtime, inference, storage, networking, and operational services.

- **Lenovo Devices** support local or workstation-based agent deployments.
- **Lenovo Hybrid AI 221** is the primary single-node hardware deployment pattern for compact, inference-focused implementations.
- **Lenovo Hybrid AI 285** is the scale-out hardware deployment pattern built around NVIDIA Scalable Units, providing repeatable multi-node compute, networking, storage, and platform expansion.

The logical Enterprise Context Layer remains consistent across these patterns; only the infrastructure scale, resiliency, capacity, and operating model change.

- Context Request and Response Contract

The following table identifies the information that should accompany a context request.

Table 2. Context request elements

Request element	Purpose
Identity context	User or sponsor, workspace, logical agent, service, and tenant.
Task purpose	The decision or workflow that requires context.
Requested context	Domain, entities, fields, time range, and freshness expectation.
Delegated scope	Permitted sources, tools, resources, duration, and approval mode.

The following table identifies the evidence and controls that should accompany a context response.

Table 3. Context response elements

Response element	Purpose
Result	Structured context or domain answer.
Sources and freshness	Origins, retrieval times, and currency.
Business meaning	Definitions, relationships, ownership, units, or effective dates.

Response element	Purpose
Policy and limitations	Allowed, filtered, redacted, partial, stale, unavailable, or denied.
Permitted next actions	Actions that may be proposed, not automatic authority.
Trace reference	Evidence for audit and operations.

Security and Governance Design

This section applies deterministic controls around identity, credentials, tools, data, tenants, and actions.

The Context Layer sits between probabilistic reasoning and high-value enterprise systems. Controls must therefore remain deterministic and external to the model wherever authority, credentials, network access, sensitive information, tenant isolation, or state-changing actions are involved.

- Threat Model and Trust Boundaries
- Identity and Delegated Authority
- Credential Mediation
- Network, Tool, and Data Controls
- Tenant Isolation
- Prompt-Injection Controls
- Action Governance
- Audit, Revocation, and Risk Management

Threat Model and Trust Boundaries

The following figure shows the trust boundaries and layered controls that must remain external to probabilistic model reasoning.

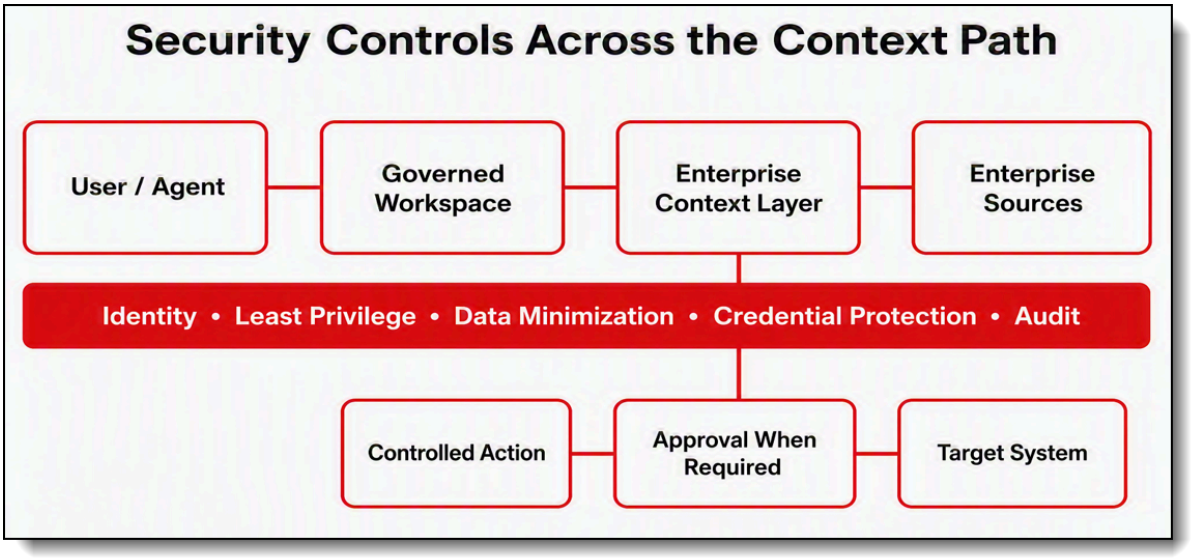


Figure 4. Layered controls around the enterprise context path

The following supporting table shows the representative threats mapping to their exposure and required control.

Table 4. Threats, exposures, and required controls

Request element	Purpose	Required Control
Prompt injection	Retrieved content attempts to redirect the agent or invoke tools.	Treat content as untrusted; isolate instructions; validate tool arguments; enforce authority outside the model.
Tool abuse	An overbroad tool or unsafe parameter causes unintended behavior.	Expose the minimum tool set, use typed parameters, and independently authorize actions.
Sensitive-data disclosure	The context response contains excessive records, fields, secrets, or cross-tenant information.	Apply source authorization, purpose limits, minimization, redaction, and tenant isolation.
Credential exposure	Secrets reach prompts, memory, files, environment, or logs.	Use credential mediation, short-lived credentials where available, and log scrubbing.
Data exfiltration	Context or output is sent to an unapproved destination.	Apply default-deny egress and monitor denied or anomalous paths.
Memory or cache poisoning	Malicious data persists and affects later sessions.	Use explicit ownership, scoped keys, TTL, invalidation, integrity checks, and isolation tests.
Unauthorized action	Read authority is converted into a write.	Use a separate action service with authorization, preconditions, approval, and evidence.
Supply-chain compromise	A connector, MCP server, skill, or package is compromised.	Maintain inventory, trusted sources, version control, validation, scanning, and rollback.

Identity and Delegated Authority

Identity and delegated authority should remain explicit throughout the request path so that every decision can be attributed to the correct user, agent, service, tenant, and purpose.

- Keep the user or sponsor, workspace, logical agent, service, connector, and source identities distinct.
- Carry task purpose and tenant explicitly.
- Bind delegated resources, tools, duration, approval mode, and revocation to the request or engagement.
- Revalidate authorization at the Context Layer and at each authoritative source or action service.
- Do not allow the agent to expand its own scope, tool set, or policy.

Credential Mediation

Credential mediation keeps secrets outside the agent and model context while providing only the scoped, revocable access required for an approved task.

- Keep backend and model credentials out of prompts, model context, agent memory, and user-visible errors.
- Use a secret manager or credential proxy rather than persistent secrets inside the agent environment.
- Do not turn the Context Layer into an unrestricted shared service credential.
- Rotate credentials and validate revocation during long-running workflows.

Network, Tool, and Data Controls

Network, tool, and data controls limit the agent to approved endpoints, capabilities, and information, preventing bypass of governed mediation and minimizing exposure.

- Apply default-deny egress at workspace and runtime boundaries.
- Allow only approved context, inference, telemetry, and lifecycle endpoints.
- Prevent direct bypass to enterprise systems when governed mediation is required.
- Filter tool discovery and expose only the smallest useful surface.
- Apply record and field minimization before returning context.

Tenant Isolation

Tenant isolation prevents context, credentials, cache entries, traces, and recovery data from crossing identity or organizational boundaries.

- Scope query results, caches, embeddings, indexes, credentials, logs, and memory to the correct identity boundary.
- Include tenant, identity, source, policy, and purpose in scoped cache decisions.
- Test cross-tenant retrieval, cache confusion, trace visibility, credential separation, backup, and restoration.

Prompt-Injection Controls

Prompt-injection controls treat retrieved content and tool output as untrusted data, ensuring that model-visible instructions cannot override deterministic authorization, policy, or approval boundaries.

- Treat all retrieved content and tool output as untrusted data.
- Prevent retrieved content from redefining system instructions, tool permissions, or approval rules.
- Validate model-generated tool arguments before connector or action execution.
- Use deterministic policy, egress, and approval controls outside the model.

Action Governance

The following table distinguishes action classes and the controls expected for each class.

Table 5. Action classes and required controls

Action class	Example	Control
Read-only	Query assets visible to the requester.	Context policy and source authorization.
Reversible write	Create a draft request.	Confirmation or policy approval plus idempotency.
Operational write	Reserve capacity or provision infrastructure.	Independent authorization, preconditions, approval, and evidence.
Destructive / external	Reclaim, delete, revoke, publish, or send externally.	Mandatory human review, narrow privilege, and recovery plan.

Audit, Revocation, and Risk Management

Audit should correlate the user request, agent identity, policy evaluation, source access, transformations, context returned, model response, approval, action, and final outcome. Revocation must apply to users, agents, workspaces, connectors, policies, and credentials.

NIST describes its AI Risk Management Framework as a voluntary resource to help organizations incorporate trustworthiness considerations into the design, development, use, and evaluation of AI systems; the Generative AI Profile is a companion resource for generative AI.

- For details on NIST, “Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile,” NIST AI 600-1, see: https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence

Deployment Architecture

The Enterprise Context Layer can run locally on Lenovo Devices, as a compact single-node deployment on Lenovo Hybrid AI 221, or as a scale-out deployment on Lenovo Hybrid AI 285 using NVIDIA Scalable Units.

The following figure introduces three Lenovo Deployment Patterns. The subsections that follow explain when local, shared, enterprise, and hybrid placement models are appropriate.

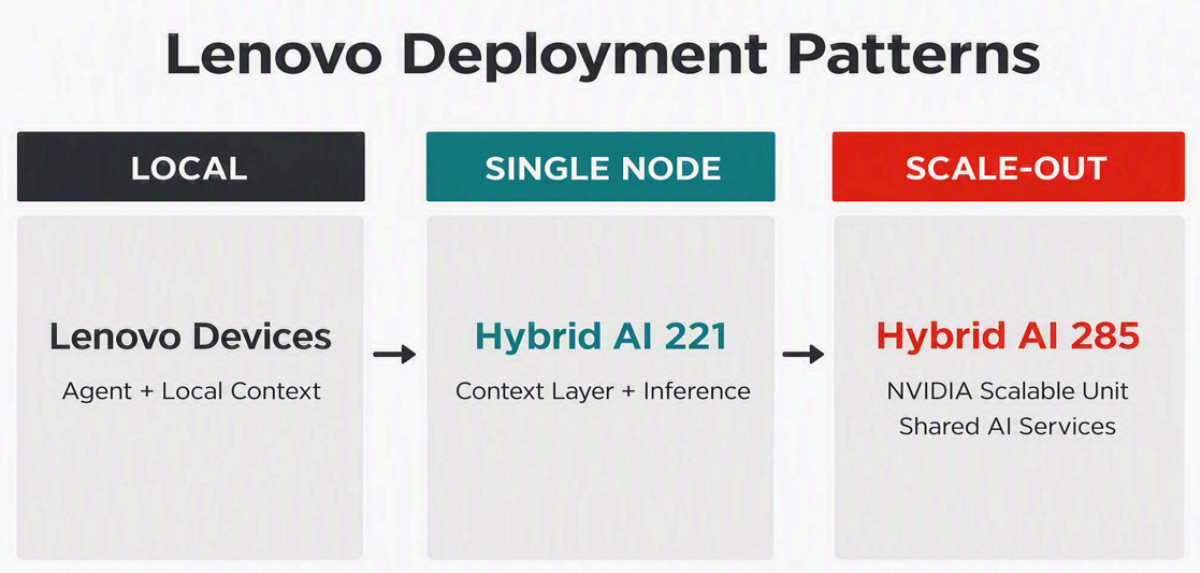


Figure 5. Lenovo Deployment Patterns

Topics in this section:

- Local or Workstation Deployment
- Domain-Specific Shared Service
- Departmental and Enterprise Deployment
- Hybrid Deployment
- Placement and Scaling Considerations

Local or Workstation Deployment

A bounded proof point can co-locate one governed workspace, a local context service, and local or approved remote inference on workstation-class infrastructure. This pattern is appropriate for controlled evaluation, domain validation, and privacy-sensitive local processing, but it requires disciplined local lifecycle and policy management.

Domain-Specific Shared Service

A domain facade such as LabOps provides curated source connectivity, business meaning, and typed context tools for a bounded process. It is the recommended first shared pattern because it creates reuse without claiming a universal enterprise data plane.

Departmental and Enterprise Deployment

Shared deployments centralize source registration, policy, observability, and operating ownership while preserving identity, tenant, credential, cache, and result isolation. They require resilient services, capacity management, and clearly defined administrative boundaries.

Hybrid Deployment

Agent execution can remain local while inference, context, data, or lifecycle services run centrally. Placement decisions should consider data locality, latency, source connectivity, security boundaries, resilience, and operating responsibility.

Placement and Scaling Considerations

Placement and scaling decisions should balance workload demand, source characteristics, context freshness, isolation, resilience, and operational constraints.

- Number of workspaces and concurrent context requests
- Source latency and consistency requirements
- Context size, query cost, and freshness expectations
- Cache scope and isolation •Inference placement and capacity
- High availability, backup, recovery, and change windows
- Tenant trust model and compliance requirements

Implementation Example and Validation

This section maps the reference pattern to a Lab Operations Agent example and distinguishes architectural illustration from approved validation evidence. The implementation example uses the Lab Operations Agent because it requires current state from multiple operational systems and has a natural distinction between read context and state-changing actions. Internal architecture artifacts describe Hermes skill modules using Spice.ai for the read surface and infrastructure-specific MCP servers for write operations.

Topics in this section:

- Lab Operations Agent
- Stateless Event-Driven Architecture
- Source Systems and Context Flows
- Read Context Path
- Controlled Action Path

Lab Operations Agent

The Lab Operations Agent demonstrates how governed read access and separately controlled actions can be combined in an operational workflow. Figure 6 presents the logical paths and the accompanying detailed architecture.

The Lab Operations Agent is a governed AI operations assistant designed to manage the end-to-end lifecycle of shared lab infrastructure. It provides a conversational interface through Microsoft Teams while coordinating deterministic workflows for asset discovery, request intake, approval, reservation, provisioning, lease monitoring, health monitoring, extension review, reclamation, and audit reporting.

The agent is intended to reduce the manual effort and elapsed time required to allocate and manage lab resources without removing operational control from administrators. Users can describe workload and capacity requirements in natural language. The agent then retrieves current asset, request, lease, workflow, and health information; evaluates the request against defined rules; and guides it through the appropriate approval and fulfilment process.

The implementation separates reasoning from authority:

- **Hermes** manages agent reasoning, planning, skills, and tool selection.
- **NemoClaw and OpenShell** provide the governed execution boundary.
- **Spice.ai** provides the controlled context and data-access path.
- **NetBox** supplies infrastructure inventory and allocation state.
- **Monday.com** supports request tracking and human approval.
- **PostgreSQL and RabbitMQ** maintain operational state and event-driven processing.
- **Infrastructure-specific MCP services** execute approved reservation, provisioning, monitoring, and reclamation operations.
- **Human reviewers** retain authority for actions that require administrative or risk-based approval.

LabOps is used in this paper as a practical validation of the Enterprise Context Layer pattern because it requires both current multi-source context and tightly controlled state-changing actions. It demonstrates how an agent can understand operational state through a governed read path while using separate action services for execution.

The following image is a high level flow of LabOps.

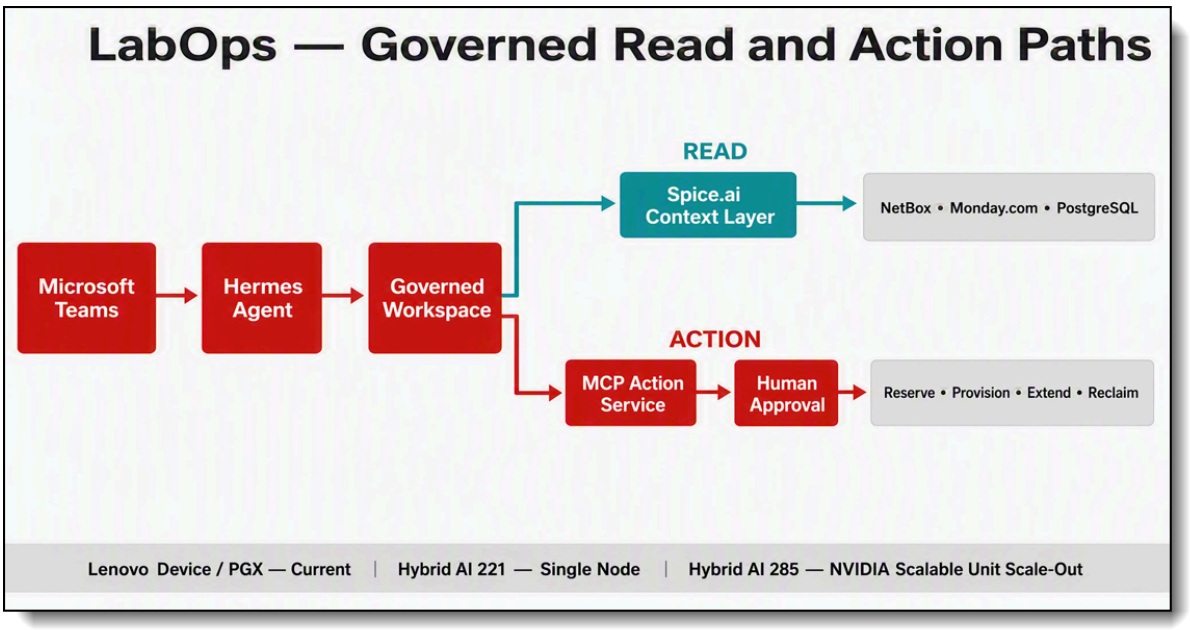


Figure 6. High level flow of LabOps

The following image is a representation on how LabOps applies the Enterprise Context Layer pattern by separating governed information access from approved infrastructure actions and by supporting progression from local hardware to single-node and scale-out deployment patterns.

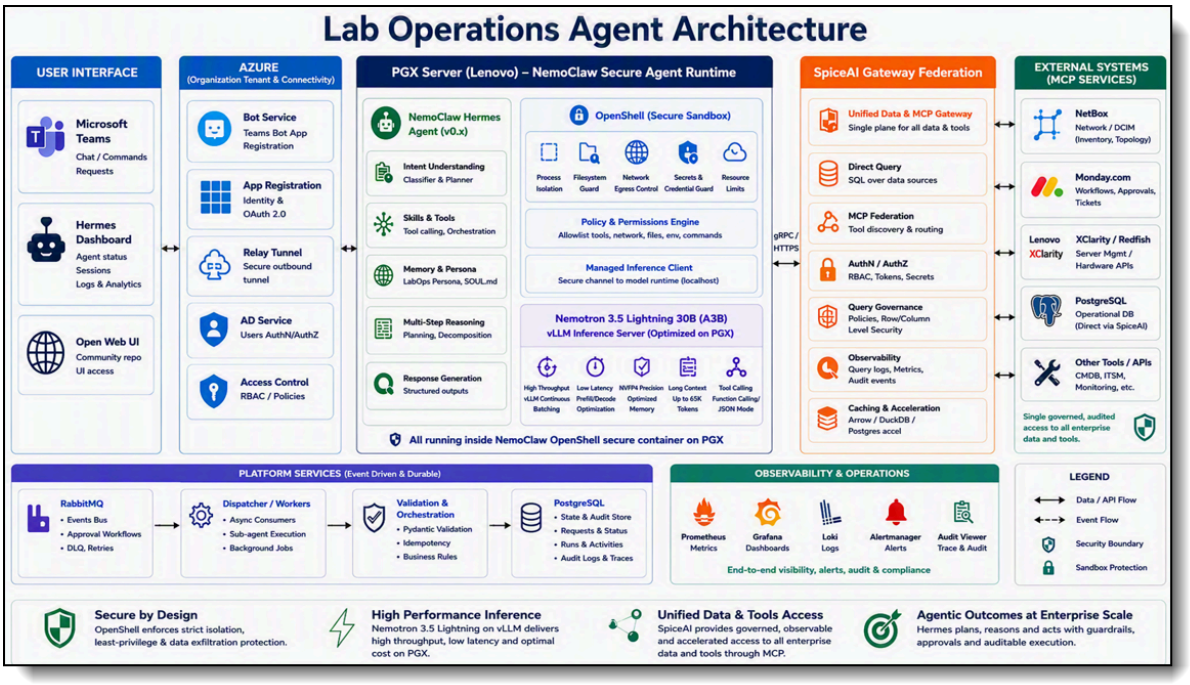


Figure 7. LabOps applying the Enterprise Context Layer pattern

Stateless Event-Driven Architecture

The following diagram represents a multi-agent, event-driven architecture for IT asset and service operations, where Microsoft Teams acts as the user interaction channel and communicates through a Bot Adapter with the central Hermes Agent, which orchestrates user requests by separating them into synchronous and asynchronous execution paths.

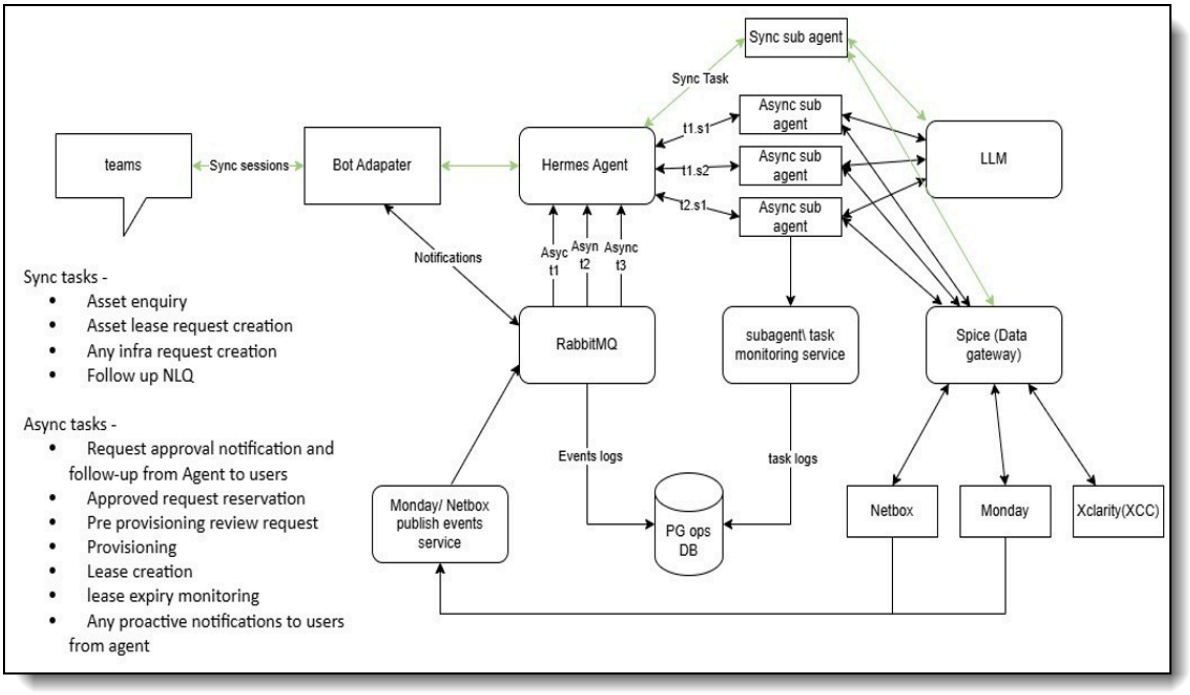


Figure 8. LabOps applying the Enterprise Context Layer pattern

Synchronous requests, such as asset inquiries, asset lease creation, infrastructure request creation, and follow-up natural-language queries, are routed directly to a Sync Sub-Agent for immediate processing, while longer-running workflows, including approval notifications, approved-request reservation, pre-provisioning reviews, provisioning, lease creation, lease-expiry monitoring, and proactive notifications, are delegated through RabbitMQ to multiple Async Sub-Agents (t1.s1, t1.s2, t2.s1, etc.), enabling decoupled and scalable background processing. The sub-agents interact with an LLM for language/reasoning capabilities and access enterprise operational systems through Spice (Data Gateway), which provides connectivity to NetBox, Monday, and XClarity (XCC).

A dedicated sub-agent task monitoring service captures execution/task logs, while RabbitMQ event logs and task-monitoring data has persisted in the PG Ops DB for operational tracking. Monday/NetBox publisher event services publish infrastructure and workflow events into RabbitMQ, allowing asynchronous processing and proactive agent-driven notifications, which are sent back through the Bot Adapter to Teams. Overall, the architecture combines agent orchestration, LLM-driven processing, message-queue-based asynchronous execution, centralized data integration, task observability, and event-driven automation to support scalable infrastructure and asset lifecycle operations.

Source Systems and Context Flows

The following table maps common LabOps needs to required context and illustrative sources.

Table 6. Common LabOps needs, required context and sources

Need	Context	Illustrative source
Find capacity	Asset configuration, health, allocation, lease, pending request	CMDB/NetBox, lease records, workflow
Create or track request	Requester, required fields, approver, state	Identity and workflow system
Recommend provisioning	Capacity, availability, topology, policy, risk	Composed operational context
Monitor lifecycle	Lease, health, ownership, exceptions	Lease, monitoring, and asset systems
Reclaim asset	Expiry, notification, exception, approval, recovery posture	Lease engine and action service

Read Context Path

For the Read Context Path, the following steps need to be followed:

1. The user asks a LabOps question through an approved channel.
2. Hermes calls the configured model and exposes approved context tools.
3. The model selects a tool; Hermes executes the MCP call.
4. OpenShell permits the approved context endpoint.
5. The context service retrieves authorized source data and composes the domain result.
6. The model receives minimized context with source, freshness, limitations, and permitted next action.

Controlled Action Path

For the Controlled Action Path, the following steps need to be followed:

1. The agent retrieves current state and action prerequisites.
2. The agent proposes a typed action rather than using broad backend access.
3. The action service independently checks identity, parameters, risk, policy, preconditions, and approval.
4. A human review gate is applied where required.
5. The service executes with narrow authority and returns transaction evidence.

Design Considerations and Recommendations

The recommendations in this section help architecture teams decide when the pattern is warranted and how it differs from adjacent integration and retrieval approaches.

Topics in this section:

- When to Use the Pattern
- Context Layer versus Direct Connector
- Context Layer versus RAG Platform
- Context Layer versus MCP Gateway
- Build versus Integrate

When to Use the Pattern

Use the Enterprise Context Layer when context access, policy, evidence, and operational controls must be shared or consistently governed across sources, agents, or actions.

- An agent needs context from more than one enterprise source.
- Multiple agents will reuse the same sources, semantics, or policy.
- The context contains sensitive or tenant-scoped information.
- The answer requires current structured state as well as documents.
- Actions require approval, preconditions, or transaction evidence.
- The organization needs consistent provenance and operating ownership.

Context Layer versus Direct Connector

A direct connector can be sufficient for a controlled, single-source, low-risk, read-only use case. The Context Layer becomes valuable when integration, policy, composition, evidence, and operations must be reused or governed consistently.

Context Layer versus RAG Platform

RAG platforms primarily retrieve knowledge for model grounding. The Context Layer also addresses current structured state, workflow context, identity and purpose, multi-source composition, action prerequisites, provenance, and policy.

Context Layer versus MCP Gateway

An MCP gateway governs connectivity, discovery, routing, and often authentication or authorization for MCP servers. The Context Layer includes the broader enterprise responsibility for source governance, identity continuity, retrieval and composition, semantics, policy, freshness, provenance, tenancy, and operations.

Build versus Integrate

The logical pattern should remain vendor-neutral. Products can implement several responsibilities, but no component should be assumed to provide the complete architecture. Spice.ai documents MCP federation and the combination of MCP tool output with federated SQL and vector search; this makes it relevant to the implementation example, while a dedicated MCP gateway or enterprise identity service may still own capabilities outside its validated scope.

- For details on Spice.ai, “Federated MCP Client for Distributed Tool Ecosystems” and “MCP Server and Gateway for Enterprise AI Agents.”, see: https://spiceai.org/docs/use-cases/ai/federated-mcp-server

Operations and Lifecycle

Operational ownership is required to keep policies, connectors, credentials, caches, and recovery processes current throughout the service lifecycle.

Topics in this section:

- Monitoring and Observability
- Policy and Connector Lifecycle
- Credential Rotation and Revocation
- Backup, Recovery, and Retention
- Ownership and Operating Model

Monitoring and Observability

Monitoring and observability should provide end-to-end visibility into context requests, source access, policy decisions, performance, freshness, costs, and action outcomes.

- Request and trace identifiers
- Identity and tenant attribution
- Source and connector health
- Policy permits and denials
- Retrieval latency, context volume, and freshness
- Cache activity and invalidation
- Tool and action results
- Model and context-service cost attribution

Policy and Connector Lifecycle

Source and tool registration should include ownership, schema, classification, version, endpoints, intended use, health, and retirement status. Connector and policy changes should be reviewed, tested, versioned, and rolled back independently from agent prompts and skills.

Credential Rotation and Revocation

Credential rotation, user revocation, agent revocation, workspace suspension, connector disablement, and policy updates must be testable during active workflows. Long-running agents must not retain stale authority after revocation.

Backup, Recovery, and Retention

The operating model should define which context-service state is durable, what is cached, how retention and deletion are enforced, how tenant-scoped state is backed up and restored, and how source-of-truth systems are reconciled following recovery.

Ownership and Operating Model

The following table assigns accountability across the operating model.

Table 7. Accountability across the operating model

Role	Accountability
Agent product owner	Use case, user experience, instructions, skills, and business outcome.
Context platform owner	Gateway, registry, retrieval, cache, observability, and service reliability.
Source owner	Authoritative data, schema, classification, quality, and access policy.
Security and identity owner	Identity, delegation, secrets, threat testing, incident response, and risk acceptance.
Action-service owner	Typed operations, authorization, approval, preconditions, evidence, and recovery.
Infrastructure owner	Workspace, context service, inference, storage, network, backup, and capacity operations.

Summary

This section consolidates the architectural rationale, separation of responsibilities, and practical value of the Enterprise Context Layer.

Enterprise agents require more than access to models and tools. They need context that is authorized, task-relevant, semantically meaningful, current, and traceable. Connecting every agent directly to every enterprise source transfers integration and security complexity into the least deterministic part of the system.

The Enterprise Context Layer creates a stable mediation boundary. It allows agent frameworks and enterprise systems to evolve independently while preserving identity, source authority, policy, provenance, and operational control. It also creates a deliberate distinction between understanding enterprise state and changing that state.

The Lab Operations Agent illustrates the pattern. Hermes and the model provide agent-side reasoning and tool selection. NemoClaw and OpenShell provide the governed execution boundary. Spice.ai implements selected context and federation capabilities. Source systems remain authoritative. Narrow action services execute approved changes. Together, these responsibilities form a reusable architecture without claiming that one product provides the complete solution.

Final takeaway: A context-aware agent is not simply an agent with more data. It is an agent that receives the minimum authorized enterprise context required for the task, with meaning, freshness, provenance, limitations, and controlled next actions.

Appendix A. Security Control Matrix

The following table maps security controls to enforcement locations and evidence.

Table 8. Security controls to enforcement locations and evidence

Control	Enforcement Location	Evidence
Action approval	Action service and review gate	Approval and transaction reference
Authentication	Approved user channel and enterprise identity	Successful and failed authentication evidence
Audit correlation	Workspace, Context Layer, source, action service	Common trace identifier
Credential mediation	Credential proxy or connector	No raw secret in agent inspection
Delegation	Context request, policy service, action service	Scope, expiry, and revocation evidence
Network egress	Workspace and runtime policy	Denied unlisted connection test
Source authorization	Context service and source	Allowed and denied source tests
Tenant isolation	Workspace, context service, cache, index, logs	Cross-tenant negative tests
Tool visibility	Agent host or MCP gateway	Authorized tool list by identity and role

Appendix B. Validation Checklist

The following Validation Checklist applies as follows:

- Attempt cross-tenant query, cache, index, log, and credential access.
- Attempt direct egress to an unapproved source.
- Attempt a write through the read surface.
- Change source state, expire cache, and disable a required source.
- Embed malicious instructions in a retrieved document or tool output.
- Inspect prompts, memory, files, environment, logs, and outbound calls for secrets.
- Retrieve and compose authorized context from at least two sources.
- Rotate credentials, update policy, remove a connector, and revoke an active agent.
- Run the same request under allowed and denied identities.
- Trace a response and an action to source, policy, approval, and transaction evidence.

Appendix C. Terminology and Acronyms

The following table defines the principal terms used in this publication.

Table 9. Principal terms used

Term	Definition
Action service	A narrow interface that performs a typed state-changing operation after independent controls.
Agent runtime	Software that maintains the interaction loop, calls the model, exposes tools, executes selected tool calls, and returns responses.
Data federation	Querying or composing information across connected sources without requiring all source data to be permanently copied into one store.
Enterprise context	Authorized, task-relevant knowledge and current state supplied to an agent for a defined purpose.

Term	Definition
MCP	A protocol for context exchange among hosts, clients, and servers.
Provenance	Evidence describing origin, retrieval time, transformation, policy, and trace history.
Tenant	An explicitly defined user, team, department, customer, or organizational isolation boundary.
Workspace	The managed execution and policy boundary in which an agent runs.

References

For more information, see these resources:

- Model Context Protocol, “Architecture overview,” documentation: https://modelcontextprotocol.io/docs/2026-07-28/learn/architecture
- NVIDIA, “Secure Agent Workspace Reference Design: Reference Architecture” https://docs.nvidia.com/enterprise-reference-architectures/secure-agent-workspace-reference-design/latest/reference-architecture.html
- Spice.ai, “Federated MCP Client for Distributed Tool Ecosystems” and “MCP Server and Gateway for Enterprise AI Agents https://spiceai.org/docs/use-cases/ai/federated-mcp-server
- NIST, “Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile” https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence
- OWASP, “AI Agent Security Cheat Sheet.” https://cheatsheetseries.owasp.org/cheatsheets/AI_Agent_Security_Cheat_Sheet.html
- Lenovo Press, “Lenovo Hybrid AI 285 Platform Guide” https://lenovopress.lenovo.com/lp2181-lenovo-hybrid-ai-285
- SKU Part Numbers:
  - ISG Nemoclaw in box - 5MS7C42207
  - IDG Early Preview Nemoclaw in box - 5MS1X14123

Authors

Drew Johnson is the project manager in the SSG AI team. Focusing on the project deliverables and milestones and co-ordination of different streams.

Dinesh Tripathi is a technical lead in the SSG AI team. Focuses on the technical direction and implementation of enterprise AI architectures and agent solutions.

Firoz Rangwala is Director of Architecture in the SSG AI team. Provides architectural leadership for enterprise AI initiatives and the design of governed, scalable AI solutions.

Harry Svendsgaard is a hybrid cloud lab engineer in the SSG AI team. Core developer and AI data engineer.

Harshad Vidyadhar Tatwadi is an Enterprise Architect in the SSG AI team. Develops enterprise architecture patterns that connect AI agents, governed context, infrastructure, and operational workflows.

Rishabh Sachan is a Cloud Architect in the SSG AI team. Contributes cloud architecture guidance for enterprise AI platforms, integrations, and deployment patterns and automation.

Surbhi Taneja is a Product Manager in the SSG AI team. Helps align enterprise AI capabilities and solution requirements with product priorities and customer needs.

Tabor Scott is the Cyber Security Architect in ths SSG AI team. Driving AI security, guardrail, governance at the whole within all SSG projects.

Related product families

Product families related to this document are the following:

- AI Servers
- Artificial Intelligence

Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area. Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service. Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
8001 Development Drive
Morrisville, NC 27560
U.S.A.
Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION ”AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary. Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk. Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

© Copyright Lenovo 2026. All rights reserved.

This document, LP2551, was created or updated on October 29, 2026.

Send us your comments in one of the following ways:

- Use the online Contact us review form found at:
  https://lenovopress.lenovo.com/LP2551
- Send your comments in an e-mail to:
  comments@lenovopress.com

This document is available online at https://lenovopress.lenovo.com/LP2551.

Trademarks

Lenovo and the Lenovo logo are trademarks or registered trademarks of Lenovo in the United States, other countries, or both. A current list of Lenovo trademarks is available on the Web at https://www.lenovo.com/us/en/legal/copytrade/.

The following terms are trademarks of Lenovo in the United States, other countries, or both:
Lenovo®
XClarity®

The following terms are trademarks of other companies:

Microsoft® and Microsoft Teams® are trademarks of Microsoft Corporation in the United States, other countries, or both.

NVIDIA® is a trademark of NVIDIA Corporation.

Other company, product, or service names may be trademarks or service marks of others.