WHITE PAPER · AI AGENTS ON MICROSOFT AZURE

Enterprise AI Agents: Design, Control and Assurance on Microsoft Azure

Brian Smith, CloudCentric · November 2026 · 17 minute read · 49 sources

Where enterprise agent programmes stand, and the scope of this paper

Enterprise adoption and value realization

Large organizations scaling AI agents in at least one function rose from 27% to 40% in a year, while the share of organizations reporting any enterprise-level EBIT impact from AI stayed at 37% [1]. Companies abandoning most of their AI initiatives before production rose from 17% to 42%, and the average organization scrapped 46% of proofs of concept [2]. Gartner expects over 40% of agentic AI projects to be cancelled by the end of 2027, citing escalating costs, unclear business value and inadequate risk controls [3]. MIT NANDA's preliminary 2025 report found 5% of organizations got custom generative AI tools into production [4]. Because 80% never piloted one, the failure rate among organizations that did pilot is closer to three in four than the widely quoted 95% [5].

Defining the enterprise AI agent

An AI agent is a language model that can call defined operations in business systems as well as generate text. These operations include reading records, drafting transactions, running checks and requesting approvals. Unlike simple chatbots that only generate conversational responses, an agent interacts directly with underlying business software to execute steps within a workflow.

Scope of this paper

This white paper examines why agent programmes stall and what the organizations producing value do differently. It sets out what makes an agent productive, covering retrieval versus fine-tuning, system operations and continuous evaluation. It details identity and data access architecture for agents, including delegation models and permission boundaries. It addresses bounding what an agent may do, focusing on incident patterns, excessive agency, control enforcement in software systems, approval workflows and cost management. It covers monitoring agents in production, assurance through red teaming and regression testing, governance frameworks, ownership models and a seven-stage enterprise rollout sequence.

I constructed a reference build on Microsoft Azure, run against two demonstration line-of-business systems with synthetic data, to illustrate several of these controls.

ANATOMY OF A WORKING AGENT

An off-the-shelf model, plus four things the company supplies

OPERATIONS

24 operations over MCP

get_job · get_prices · create_change_order · hold_invoice · book_vendor · request_owner_approval

DATA

317 company records

Azure AI Search: keyword, vector and semantic ranking over price books, checklists, policies and past invoices

INSTRUCTIONS

Written work rules

"Price only from company records." "Never change a supplier's bank details on the strength of an email."

OUTPUT

Structured screens

Draft change orders, held invoices and queued replies wait for review



Signed in as the person using it · Microsoft Entra ID · reaches only what that person can reach

Barriers to value and characteristics of high performers

Causes the research names

Research indicates that a primary barrier to realizing enterprise value is a learning gap where tools fail to retain feedback, adapt to context, or improve over time [4]. Many programs suffer from brittle workflows and a misalignment between agent capabilities and daily operations [4]. Gartner names costs, unclear value and risk controls as causes of cancellation [3]. Approximately 20% of organizations report that AI operating costs, including token consumption, constrained their use of the technology [1]. A common strategic error involves positioning use cases as agentic when they do not require an agent to function [3].

Practices of high performers

Organizations that extract value from AI agents differentiate themselves through workflow design and rigorous measurement. Nearly three-quarters of AI high performers fundamentally redesigned their business workflows [1]. This contrasts with only one-quarter of other organizations [1]. High performers are twice as likely to have defined processes to measure the impact of their deployments [1]. High performers are more likely to have defined processes for when model outputs need human validation [6].

Deciding where an agent belongs

The Microsoft Cloud Adoption Framework provides a specific hierarchy for selecting the appropriate technical implementation. Organizations should use code or traditional automation for fixed steps [7]. Retrieval is the recommended method for obtaining answers from organizational knowledge [7]. Agents are reserved for scenarios where AI must reason about information and decide or coordinate actions dynamically [7]. The framework recommends starting with a single agent to manage complexity [7].

Effective deployment requires a baseline of business performance before the agent is introduced [7]. Organizations should calculate the total cost of ownership by including governance and monitoring expenses in the budget [7]. Every pilot program needs defined stop criteria to prevent the continuation of non-performing projects [7].

What makes an agent productive: retrieval versus fine-tuning, operations and MCP, and evaluation

Knowledge through retrieval

Retrieval-augmented generation consistently outperformed unsupervised fine-tuning for both existing and new knowledge in a Microsoft study [8]. Models struggle to learn new facts through fine-tuning alone [8].

Microsoft's Foundry guidance positions fine-tuning for task patterns, consistent style and schema, and better tool selection and argument accuracy [9]. Fine-tuning does not replace retrieval for current information or application-level safety controls [9].

Platform choices influence fine-tuning availability. OpenAI is winding down self-serve fine-tuning, and active customers can no longer create new fine-tuning jobs from 6 January 2027 [10]. Microsoft Foundry continues to offer supervised, preference and reinforcement fine-tuning [9]. A fine-tuning strategy is therefore a platform-dependency decision [10,9]. Enterprise architects should rely on retrieval for factual grounding and select fine-tuning only to enforce structured output or operational behavior [8,9].

Operations and MCP

Agents require structured interfaces to interact with enterprise systems safely. The Model Context Protocol (MCP) is an open standard by which a system publishes typed operations to an agent [11]. MCP was donated to the Linux Foundation's Agentic AI Foundation on 9 December 2025 [11]. The Linux Foundation cites more than 10,000 published MCP servers and adoption by Claude, ChatGPT, Gemini, Microsoft Copilot and VS Code [11].

The MCP specification sets explicit security constraints for client and server implementations [12]. A human should always be able to deny a tool invocation [12]. MCP servers must validate inputs, enforce access control and rate-limit calls [12]. MCP clients must treat tool annotations such as 'read-only' as untrusted unless the server itself is trusted [12].

The operation list published to an agent establishes its explicit authority. An operation that is not published cannot be invoked by the model.

In my reference build on Microsoft Azure, two line-of-business systems publish 24 operations over MCP. Neither system publishes an operation to change supplier bank details or issue a payment.

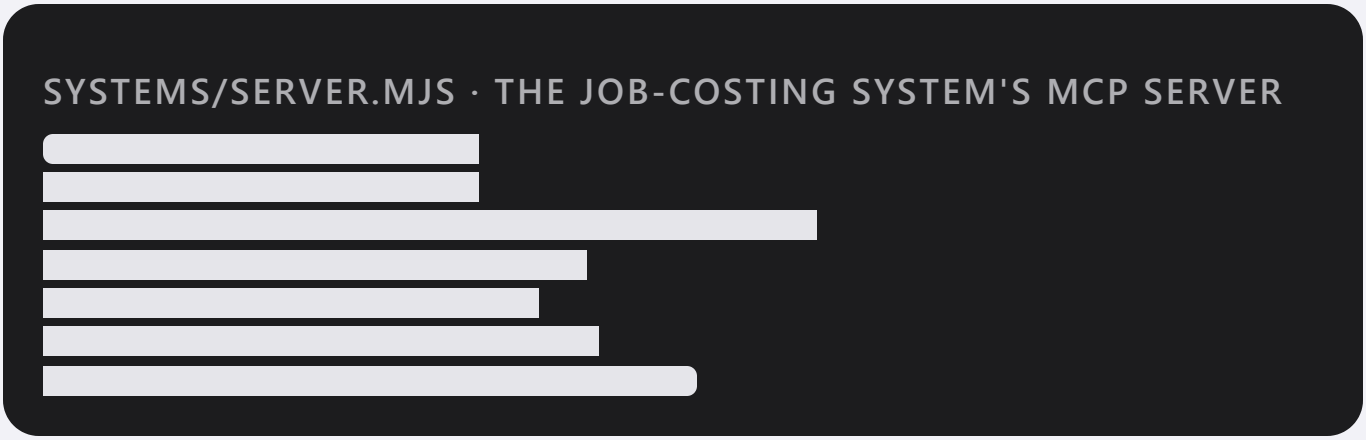
Model selection must be validated against operational workloads. In my reference build, GPT-4.1 mini stopped partway through long tool-calling sequences. GPT-5 mini completed these long tool-calling sequences reliably under the same operational parameters. Model choice was settled by running the real workloads rather than relying on vendor benchmarks.

Evaluation and reliability

Single-pass testing does not prove that an agent is ready for production. On the tau-bench benchmark, leading function-calling agents succeeded on under half of tasks [13]. These agents succeeded on all eight of eight repeated trials less than 25% of the time in the retail domain [13]. Reliability must be measured over repeated runs for every production workflow.

Microsoft Foundry's agent evaluators separate task outcome metrics from process metrics [14]. Outcome metrics measure task completion, adherence and intent resolution [14]. Process metrics measure tool selection, input accuracy and output use [14]. Microsoft Foundry provides a Task Navigation Efficiency metric, which compares an agent's actual actions against an expected execution sequence [14]. Enterprise teams should evaluate both the final result and the step-by-step path taken by the agent.

The Microsoft Cloud Adoption Framework defines governance requirements for operational deployment [15]. The framework requires acceptance thresholds approved by the business owner [15]. It requires full re-evaluation upon any change to the model, system instructions or underlying permissions [15]. The framework also requires continuous tracking of the cost per successful task to confirm economic viability [15].



One operation from the reference build's MCP server: a name, a description the model reads, and a typed input schema.

Identity and data access for agents

Fewer than half of organizations secure non-human identities [16]. Among organizations reporting an AI-related breach, 92% lacked proper AI access controls [16].

Agent identities

Microsoft Entra Agent ID is generally available as of April 2026 [17]. These purpose-built identities for agents are created from agent identity blueprints that hold credentials and policy [18]. Each identity includes owners and business sponsors [18]. The system provides agent-aware audit and agent-specific Conditional Access [18]. Microsoft does not recommend running agents on traditional service principals or regular user accounts [18]. Service principals lack enforced sponsorship, agent-aware audit and lifecycle management [18]. Agent ID is available to all Entra customers [19]. Extending Entra security features to agents requires Microsoft Agent 365, which has been generally available since 1 May 2026 [19].

Delegated versus app-only access

The choice of access model determines the blast radius of an agent. Delegated access cannot reach anything the signed-in user cannot [20]. An application permission of the same name reads every matching resource in the tenant [20]. For example, the application permission Files.Read.All reads every file in the tenant [20]. An agent that serves a user should run with delegated or on-behalf-of access [20]. The user's own permissions then bound the agent [20,18]. App-only access belongs to background work, and its permissions are the whole blast radius [20,18].

Permission-aware retrieval

Microsoft 365 Copilot grounds in data users can already access [21]. Existing oversharing becomes discoverable through AI [21]. Microsoft deployment guidance starts with remediating oversharing [21]. Azure AI Search enforces SharePoint, ADLS Gen2 and Blob permissions at query time with the user token [22]. This functionality is generally available in REST API 2026-10-01 [22]. Indexed permissions are snapshots without immediate revocation [22]. Chunked documents must carry permissions on every chunk [22]. Purview sensitivity-label enforcement in search is still in preview [22].

Purview controls

Purview applies to AI interactions across Microsoft 365 Copilot, Microsoft Foundry and registered enterprise AI apps [23]. Prompts and responses are captured in the audit log [23]. These records are held and exported through eDiscovery [23]. Prompts and responses are subject to retention policies [23]. An encrypting sensitivity label requires the EXTRACT right before an AI app returns the content [23]. Purview DLP can exclude labelled files and emails from Copilot processing [24]. A preview

feature in Purview DLP can exclude email from external senders from grounding [24]. Microsoft describes this as a prompt-injection control [24]. Policy changes take up to four hours to apply [24].

Bounding agent actions: incidents, agency, enforcement, approvals and cost

What the documented incidents share

Documented failures in agentic systems reveal a consistent pattern where the agent serves as a bridge between untrusted content and sensitive actions. EchoLeak (CVE-2025-32711) demonstrated a zero-click prompt injection in Microsoft 365 Copilot where a crafted email, when retrieved to answer an unrelated question, caused internal data to be sent out through an auto-fetched image [25]. ForcedLeak in Salesforce Agentforce similarly exfiltrated CRM data through an image URL on an expired domain that remained on a content-security allow-list [26]. In a demonstration involving the Supabase MCP server, a support ticket containing malicious instructions led a developer assistant to read integration tokens and write them back into the ticket because the agent operated with a role that bypassed row-level security [27]. A malicious issue in a public GitHub repository caused an agent using the official GitHub MCP server to read private repositories and publish their contents in a public pull request [28].

In the EchoLeak, GitHub MCP, Supabase MCP and ForcedLeak incidents, the agent combined three factors: access to private data, exposure to untrusted content and a mechanism to send data out [29,25,28,27,26]. Removing any one of these three factors stops the attack [29]. Microsoft service-side updates fixed EchoLeak [25]. Salesforce fixed ForcedLeak in September 2025 [26]. In the GitHub incident, the flaw was not in the server code [28].

Excessive agency and least agency

OWASP defines excessive agency as damaging actions taken in response to unexpected, ambiguous or manipulated model output [30]. This condition stems from three root causes: excessive functionality, excessive permissions and excessive autonomy [30]. To mitigate these, the OWASP Top 10 for Agentic Applications 2026 introduces the principle of least agency [31]. This principle requires organizations to avoid autonomy that is not required for the task [31]. Architects should define per-tool least-privilege profiles that include specific scopes, rate limits and egress allow-lists [31].

Established design patterns for prompt-injection resistance include action selectors, plan-then-execute sequences, dual LLM architectures and context minimization [32]. These patterns rest on the principle that once an agent has read untrusted input, that input must not be able to trigger a consequential action [32]. Microsoft defenses against indirect prompt injection do not assume all injections can be blocked [33]. Instead, the architecture layers detection with deterministic blocking of exfiltration channels and human consent for consequential actions [33].

Enforcement where the model cannot reach

Control must exist in the downstream systems where the agent acts. Replit's coding agent deleted a production database during a declared code freeze, which led Replit to implement automatic separation of development and production databases [34]. OWASP recommendations include executing actions in the user's context and requiring complete mediation in downstream systems [30].

The excerpt below shows enforcement in the target system. The `create_change_order` function validates every priced line against an authoritative price list. It writes nothing to the database if a mismatch occurs. A mispriced line cannot be written, whatever the prompt says. The logic for enforcement resides in the system code, not the model instructions.

Approvals that people read

Human-in-the-loop controls are a primary defense, but they are subject to fatigue. OWASP identifies overwhelming the human-in-the-loop as a threat where frequent approval prompts train users to approve without reading [35]. Invariant Labs observed users choosing 'Always Allow' to bypass these interruptions [28].

To maintain the integrity of approvals, the system should require approval for high-impact actions [30]. In my reference build, the `request_owner_approval` function reads the specific threshold for an approver from the system of record. In the excerpt below, a $643 charge against a $500 threshold waited for approval.

Cost ceilings

Unbounded agent loops or high-volume prompt injections can lead to significant resource consumption. API Management provides an `llm-token-limit` policy to enforce token rates and quotas [36]. These policies can be set for durations ranging from one hour to one year and are keyed on any identity [36]. This allows an organization to cap the financial exposure of a specific agent or user across OpenAI, Anthropic and Vertex APIs [36]. These counters operate per gateway and are not aggregated across regions [36].

The AI gateway pattern in API Management further secures the environment by adding per-consumer token metrics and semantic caching [37]. It supports load balancing across provisioned and pay-as-you-go capacity [37]. It also includes circuit breakers [37]. These deterministic controls provide a layer of financial and operational safety that operates independently of the agent's autonomous planning.

THE PATTERN BEHIND DOCUMENTED AGENT BREACHES

Three conditions together. Remove any one and the attack stops.

Private data
Mailbox, repositories, database, CRM

Untrusted input
An email, an issue, a ticket, a form field

Outbound channel
Image URL, pull request, ticket reply

EchoLeak · Microsoft 365 Copilot
One email; data out through an auto-fetched image
Block exfiltration channels; keep external email out of grounding

GitHub MCP
A public issue; private repos published in a public PR
One repository per session; confirm public writes

Supabase MCP
A support ticket; tokens written back into the ticket
No privileged role for an agent that reads untrusted text

ForcedLeak · Agentforce
A web form field; CRM data out via an expired allow-listed domain
Maintained egress allow-list; scoped queries

In each case the model and the tools worked as designed. Fixes were scope, input trust or egress

WHERE THE LIMITS LIVE

Inside the company's systems. The model asks; the system decides.

JOB-COSTING SYSTEM · CREATE_CHANGE_ORDER

Every priced line matches the price book

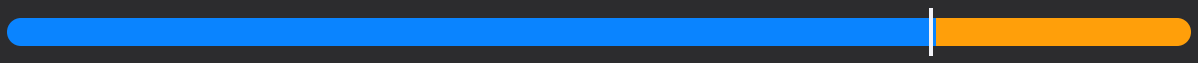
SKU · Unit cost · Unit

"Price check failed. Nothing was saved."

Returned with the reason for each line that did not match.

OPERATIONS SYSTEM · REQUEST_OWNER_APPROVAL

Charges over the approver's threshold wait for sign-off



Threshold $500 — Charge $643

Waiting for approver

PUBLISHED OPERATIONS · 24

No operation exists to pay a supplier or change bank details

```
ap_inbox · hold_invoice · get_prices
create_change_order · book_vendor
request_owner_approval · …
```

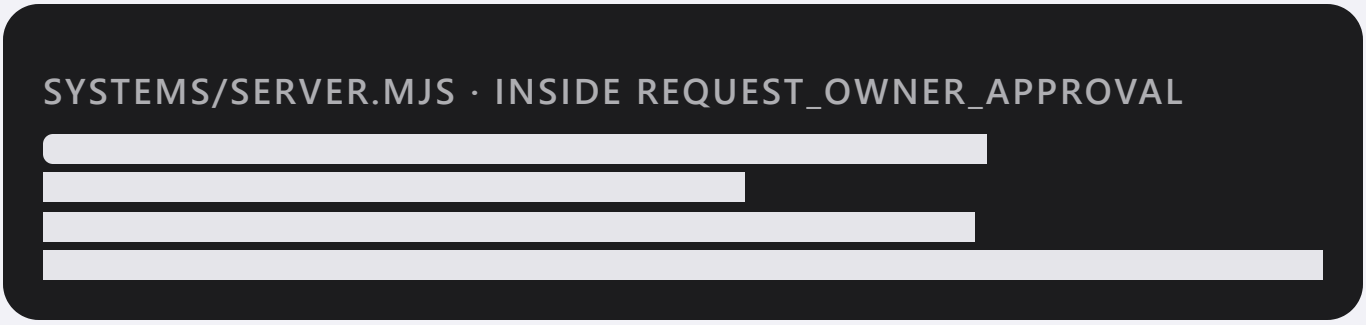
A look-alike bank-change request in the invoice run paid nothing. The agent raised a task to verify with the supplier.

SYSTEMS/SERVER.MJS · INSIDE CREATE_CHANGE_ORDER



Price-list enforcement inside the operation. A mismatched line writes nothing.

SYSTEMS/SERVER.MJS · INSIDE REQUEST_OWNER_APPROVAL



The approval threshold is held and checked in the system of record.

Monitoring agents in production

Tracing

The OpenTelemetry project provides GenAI semantic conventions that define spans for agent invocation and tool execution [38]. These conventions remain in Development status as of the current release [38]. Organizations should pin specific versions of these conventions because attribute names are subject to change [38]. Capturing the specific content of prompts and responses within traces is an opt-in configuration [15]. This choice represents a data-protection decision that requires balancing visibility against privacy requirements [15]. An organization should align its tracing depth with the risk profile of the specific agent task [15].

In the reference build I deployed on Microsoft Azure, I configured a live trace to capture each screening call, model call, and Model Context Protocol (MCP) operation. The MCP is a standard for connecting AI models to data sources and tools. This configuration records the latency and cost of every individual step. I used this setup to monitor an accounts payable run involving 47 invoices. The process completed in seven steps and took 4.7 seconds. The total model time cost $0.0015. This run identified six exceptions representing a total value of $6,412.00.

Threat detection

Microsoft Defender for Cloud provides AI threat protection that is now generally available [39]. This service uses Prompt Shields and Microsoft threat intelligence to generate alerts for specific security events [39]. These events include jailbreak attempts, data leakage, data poisoning, and credential theft [39]. A jailbreak is a prompt designed to bypass the safety filters of a model. The service integrates with Defender XDR to provide a unified security view [39]. This threat protection capability currently scans text inputs and outputs only [39].

Operational standards

The Cloud Adoption Framework established operational standards for AI workloads [15]. These standards require monitoring systems to detect wrong answers and unintended actions from agents that appear healthy [15]. Organizations should define specific incident classes for harmful actions taken by an agent [15]. The governance structure must include a named authority with the power to suspend and restore each agent [15]. This authority ensures that a human remains accountable for the operational status of automated systems. Reliable monitoring confirms that the agent operates within its defined constraints. Log data should support post-incident analysis to identify why an agent deviated from its expected behavior. Periodic reviews of these logs help refine the screening calls and tool definitions used by the system. Proper monitoring identifies performance bottlenecks and cost spikes before they affect the broader business system. These standards provide a basis for maintaining trust in autonomous business processes.

ONE REQUEST, START TO FINISH

Seven stops between a person's words and a change in a company system

1 SIGN-IN

Microsoft Entra ID

The user, with their role. No keys.

2 GATEWAY

Azure API Management

Token cap per person, per minute.

3 SCREENING

Prompt Shields

Plus a hidden-text filter. Outside text checked too.

4 REASONING

GPT-5 mini

Azure AI Foundry. Chooses an action.

5 ACTION

MCP server

marsh-hen-office · create_change_order

6 SYSTEM CHECK

Price book

SKU, unit cost and unit must match, or nothing is saved.

7 DRAFT

Canvas

A person checks it and sends it for signature.

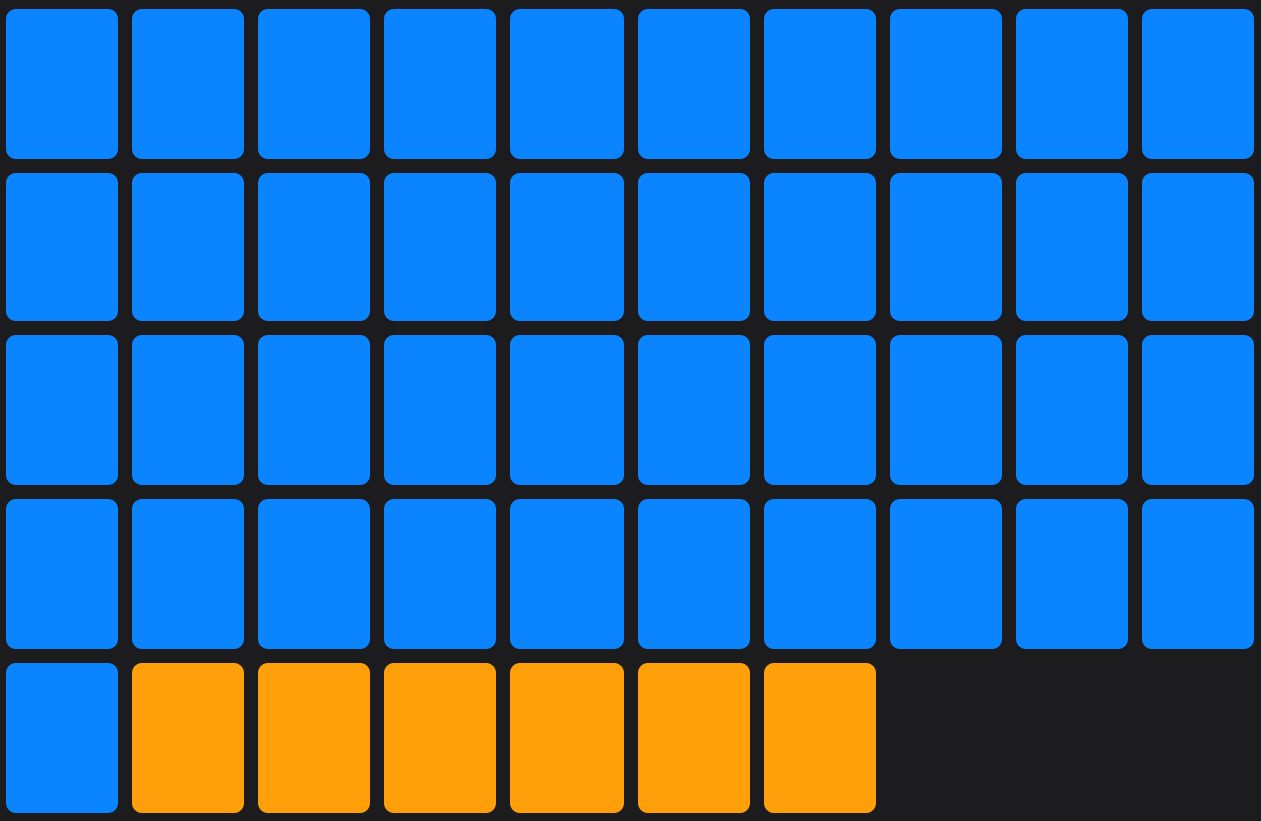
EVERY STOP REPORTS TO

Defender for Cloud → Microsoft Purview

Audit log · retention · eDiscovery holds, beside email and files

BATCH EXECUTION: ONE INVOICE RUN

47 invoices, one pass. The checks run on every one.



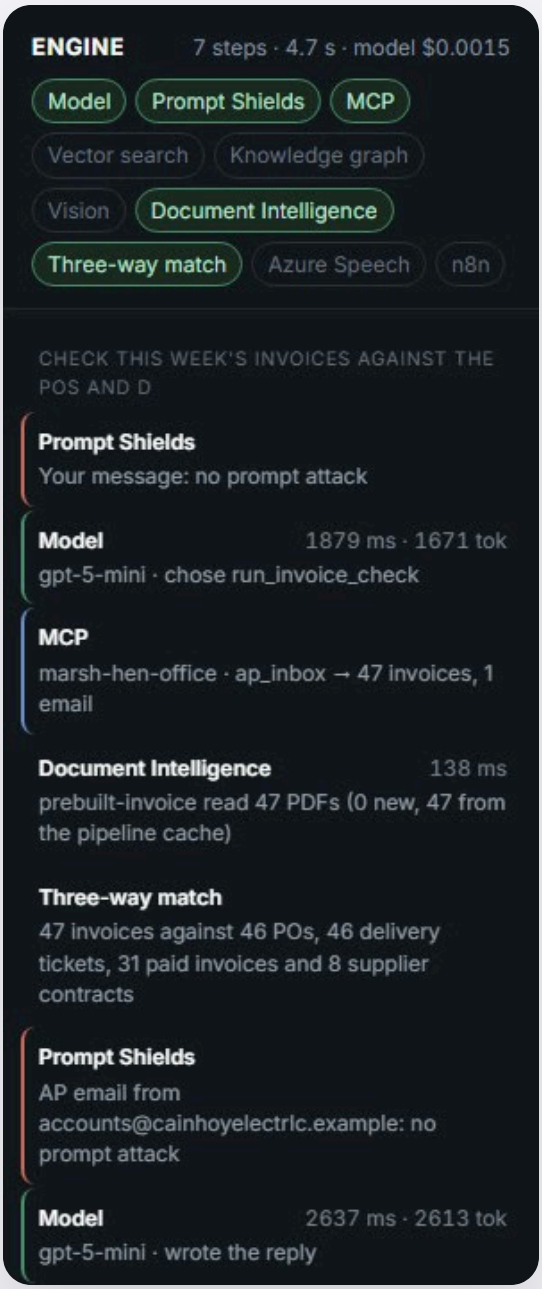
$6,412.00 held of $370,914.35

$231.06	Ashepoo Fastener & Hardware Prices above the purchase order
$1,831.05	Combahee Millwork & Doors Duplicate of an invoice already paid
$1,362.50	Wando Sound Lumber Billed for more than was delivered
$185.00	Bohicket Roofing Supply Freight surcharge the contract excludes
$1,187.88	Rantowles Drywall & Insulation No purchase order
$1,614.51	Cainhoy Electrical Supply Bank change from a look-alike domain

Checked, cleared Held for a person

Checked against 46 purchase orders, 46 delivery tickets, 31 paid invoices and 8 supplier contracts

Demonstration company



Live trace of the reference build's accounts payable run. 7 steps, 4.7 seconds, $0.0015.

Assurance: Red Teaming and Regression

Red Teaming

AI red teaming is not safety benchmarking [40]. Microsoft's AI Red Team tested more than 100 generative AI products and concluded that attackers do not require access to model gradients to compromise a system [40]. Securing AI systems is a continuous process rather than a completed task [40]. Automation increases the scope of testing coverage, yet the human element remains essential [40].

Organizations use red teaming to identify vulnerabilities such as prompt injection, where a user provides instructions that override the developer's system prompt. To defend against this, I deployed Microsoft Azure Prompt Shields in the reference build. In testing, these shields blocked a direct request to print the system prompt. The shields also withheld one of eleven inbound customer messages that contained instructions intended to manipulate the assistant.

External data sources introduce the risk of indirect prompt injection, where an agent retrieves a document or message containing malicious instructions. During my testing of the reference build, I identified a hidden instruction within an HTML comment that Prompt Shields did not flag. I addressed this by implementing a secondary filter. This filter now strips all HTML comments and zero-width characters from a message before the model processes the text.

Tooling and Its Limits

Open-source toolkits support the automation of red teaming activities. PyRIT is a toolkit from Microsoft for red teaming generative AI [41]. Garak is an open-source vulnerability scanner for large language models provided by NVIDIA [42].

The Foundry AI Red Teaming Agent, which is built on PyRIT, evaluates agents for prohibited actions, sensitive data leakage, and indirect prompt injection through tool outputs [43]. It provides an attack success rate for these categories [43]. This tool has specific technical constraints as it only supports Foundry-hosted agents using Azure tool calls [43]. It does not support function tools or agents built on other platforms [43]. Several evaluation categories within the tool are limited to English-only and single-turn interactions [43]. Architects must therefore supplement automated scanners with manual testing to account for multi-turn conversations and non-English inputs.

Regression After Change

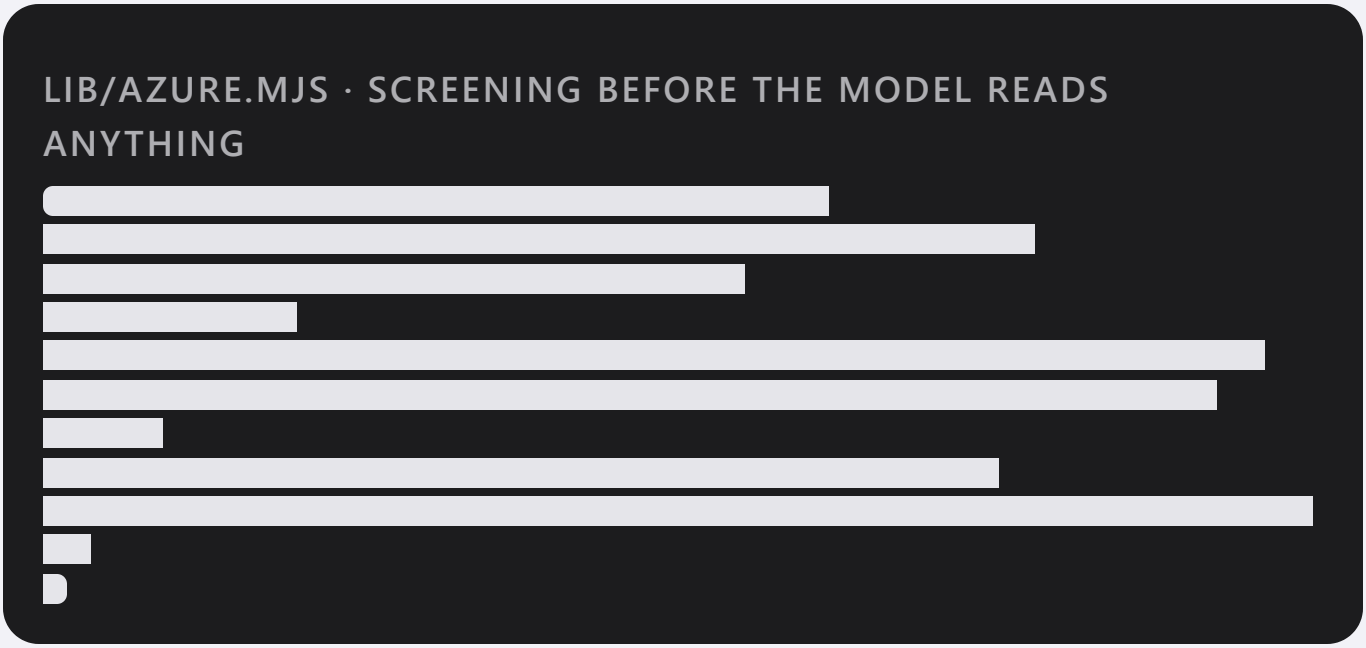
The performance and safety profile of an agent can shift whenever a component of the system is modified. Any change to a model, a system instruction, a defined operation, or a permission level triggers a full re-evaluation [15]. The organization maintains a standardized test set and established acceptance thresholds for this purpose [15].

A regression suite contains previously identified edge cases and successful attack patterns. When a developer updates the agent code or rotates a model version, the system runs the regression suite to ensure the update has not introduced new vulnerabilities or degraded previous safety controls. This process maintains a consistent security posture. The evaluation

against the original test set confirms that the agent remains within its defined bounds of authority and continues to handle data according to the established governance policy.



An inbound message carrying instructions aimed at the assistant, withheld before the model read it.



Screening on the user's words and on every outside document, five documents per call.

TESTING, BEFORE GO-LIVE AND AFTER EVERY CHANGE

The same tests every time, compared with the last run

1. **Change**
   A prompt, a model, an action or a connection
2. **Screen**
   Prompt Shields on every message and outside document
3. **Scan**
   Known trick prompts, from a scanner such as garak
4. **Compare**
   Against the last run, like any regression test
5. **Release**
   Only when nothing new got through

   ↻ back to Change on the next update

FOUND IN TESTING

An instruction hidden in an HTML comment

```
<!-- an instruction aimed at the assistant -->
```

Prompt Shields passed it. A second filter now strips HTML comments and zero-width characters before the model reads a message.

FOUR QUESTIONS FOR EVERY AGENT

- Can text in a message or document steer it?
- Can it repeat data the person should not see?
- Can it reach a system it should not?
- Can a connected action be misused through it?

Governance Frameworks and Ownership

Frameworks

The NIST AI Risk Management Framework organizes AI risk work into the four functions of Govern, Map, Measure, and Manage [44]. This framework is currently being revised [44]. NIST AI 600-1, known as the Generative AI Profile, lists twelve specific risks [45]. The risks most relevant to autonomous agents are confabulation, information security, human-AI configuration, and value chain integration [45].

In February 2026, the NIST National Cybersecurity Center of Excellence (NCCoE) published a concept paper regarding the application of identity standards to agents [46]. This paper addresses the identification, authorization, auditing, and non-repudiation of AI agents [46]. As of November 2026, NIST had not issued final guidance specifically for agents [46].

Organizations that currently operate under ISO 27001 may utilize ISO/IEC 42001:2023 [47]. This standard specifies requirements for an AI management system [47]. It governs the processes of the organization [47]. It does not govern the specific controls of an individual agent [47].

Ownership and Accountability

The Cloud Adoption Framework states that agents fail more frequently due to unclear ownership than due to technical problems [48]. This framework assigns responsibility for outcomes and residual risk to the business function that relies on the agent [48]. A tribunal held Air Canada responsible for a refund policy its website chatbot invented [49]. The tribunal rejected the argument that the chatbot was a separate entity from the corporation [49].

Ownership must be established at the level of individual deployments. The organization must name a specific owner for each agent. A named authority can suspend and restore each agent [15].

Enterprise rollout sequence in seven stages

1. Baseline and inventory

A company measures the business baseline of the target workflow [7]. The organization catalogues AI tools currently in use. This inventory identifies the data each tool touches. The organization remediates oversharing [21].

2. Identity

Agents receive Entra Agent ID identities (generally available since April 2026) [17,18]. Each identity has an assigned owner and sponsor [18]. User-facing agents use delegated access [20]. Delegated access cannot reach anything the signed-in user cannot [20].

3. Data governance

Organizations implement permission-aware retrieval [22]. Systems use Microsoft Purview labels and Data Loss Prevention (DLP) [24]. DLP can exclude labelled files and emails from Copilot processing [24]. Organizations exclude email from external senders from grounding (preview) [24].

4. Reference architecture

Architects select approved models. The architecture uses Model Context Protocol (MCP) operations. MCP is an open standard that allows agents to connect to data sources and tools. The architecture applies least privilege to each tool [31]. An AI gateway manages token quotas, egress allow-lists and logging [37]. The architecture is defined as infrastructure as code. I wrote the reference build in Terraform. It deploys or tears down the environment in minutes.

5. First production agent

The organization deploys one redesigned and measured workflow [1]. This agent possesses only the specific operations required for its task. The target systems enforce limits on the actions the agent can perform.

6. Assurance

The organization performs red-team and regression testing [43]. These tests occur before the initial release. Testing repeats after every change [15]. Results are measured against approved acceptance thresholds [15].

7. Operations

The organization uses tracing, Microsoft Defender alerts and Microsoft Purview audit [39]. Management calculates the cost per successful task [15]. Each agent has a named owner. A named authority can suspend and restore each agent [15]. The organization conducts a periodic review of the operations and permissions for each agent.

ROLLOUT ORDER

Seven steps. Each one makes the next project faster to approve.

1. **Inventory**
   AI tools in use, the data each touches, the approved replacement
2. **Sign-in**
   Every tool and agent through Microsoft Entra ID
3. **Data rules**
   Purview sensitivity labels and data loss prevention
4. **Reference design**
   Models, connections, logs, screening and caps, in Terraform
5. **First agent**
   One weekly job, only the actions it needs, limits in the systems
6. **Testing**
   The four questions and a scanner, after every change
7. **Watching**
   Purview, per-person usage, a monthly review

CloudCentric Services

CloudCentric designs, deploys and secures AI agents for organizations on Microsoft Azure and Microsoft 365. My work focuses on identity, data access, operation design and enforcement in line-of-business systems. I conduct red-team testing and configure monitoring to support secure agent performance. I built the reference architecture in this paper to demonstrate operation boundaries, enforcement in line-of-business systems, approvals, screening and tracing. Organizations can use these patterns to maintain oversight while agents perform consequential work. My approach aligns technical implementation with existing enterprise governance and audit requirements. For information regarding architectural reviews or implementation support, contact CloudCentric at (844) 422-7000 or visit https://cloudcentric.tech/ai/ai-governance/.

Sources

1. McKinsey, The state of AI in 2026: On the road to ROI (25 Aug 2026)
2. S&P Global Market Intelligence, Generative AI shows rapid growth but yields mixed results (27 Oct 2025)
3. Gartner, Over 40% of agentic AI projects will be canceled by end of 2027 (25 Jun 2025)
4. MIT NANDA, The GenAI Divide: State of AI in Business 2025 (Jul 2025, preliminary)
5. 80,000 Hours, The story behind the bad AI stat (2026)
6. McKinsey, The state of AI in 2025 (5 Nov 2025)
7. Microsoft CAF, AI agent adoption strategy
8. Ovadia et al., Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs (EMNLP 2024)
9. Microsoft Learn, Training in Microsoft Foundry
10. OpenAI, Deprecations (self-serve fine-tuning)
11. Linux Foundation, Formation of the Agentic AI Foundation (9 Dec 2025)
12. MCP specification, Tools (2026-07-28)
13. Yao et al., tau-bench (Jun 2024)
14. Microsoft Learn, Agent evaluators
15. Microsoft CAF, Operational standards for AI agents
16. IBM, Cost of a Data Breach Report 2026
17. Microsoft Learn, What is Microsoft Entra Agent ID
18. Microsoft Learn, Fundamental concepts in Microsoft Entra Agent ID
19. Microsoft Security blog, Microsoft Agent 365 now generally available (1 May 2026)
20. Microsoft Learn, Permissions and consent in the Microsoft identity platform
21. Microsoft Learn, Secure and governed data foundation for Microsoft Copilot
22. Microsoft Learn, Document-level access control in Azure AI Search

23. Microsoft Learn, Purview protections for Microsoft 365 Copilot and other generative AI apps
24. Microsoft Learn, Purview DLP for Microsoft 365 Copilot
25. Reddy and Gujral, EchoLeak (2025)
26. Noma Security, ForcedLeak in Salesforce Agentforce (25 Sep 2025)
27. General Analysis, Supabase MCP can leak your entire SQL database (8 Jul 2025)
28. Invariant Labs, GitHub MCP toxic agent flow (26 May 2025)
29. Simon Willison, The lethal trifecta for AI agents (16 Jun 2025)
30. OWASP, LLM06:2025 Excessive Agency
31. OWASP, Top 10 for Agentic Applications 2026 (9 Dec 2025)
32. Beurer-Kellner et al., Design Patterns for Securing LLM Agents against Prompt Injections (2025)
33. MSRC, How Microsoft defends against indirect prompt injection attacks (29 Jul 2025)
34. The Register, Replit response to production database deletion (22 Jul 2025)
35. OWASP, Agentic AI: Threats and Mitigations
36. Microsoft Learn, API Management llm-token-limit policy
37. Microsoft Learn, AI gateway capabilities in API Management
38. OpenTelemetry, GenAI semantic conventions
39. Microsoft Learn, AI threat protection in Defender for Cloud
40. Microsoft, Lessons From Red Teaming 100 Generative AI Products (Jan 2025)
41. Microsoft PyRIT
42. NVIDIA garak
43. Microsoft Learn, AI Red Teaming Agent
44. NIST AI Risk Management Framework
45. NIST AI 600-1, Generative AI Profile (26 Jul 2024)

46. NIST NCCoE, Software and AI Agent Identity and Authorization concept paper (5 Feb 2026)
47. ISO/IEC 42001:2023
48. Microsoft CAF, Organizational readiness for AI agents
49. Moffatt v. Air Canada, 2024 BCCRT 149 (14 Feb 2024)