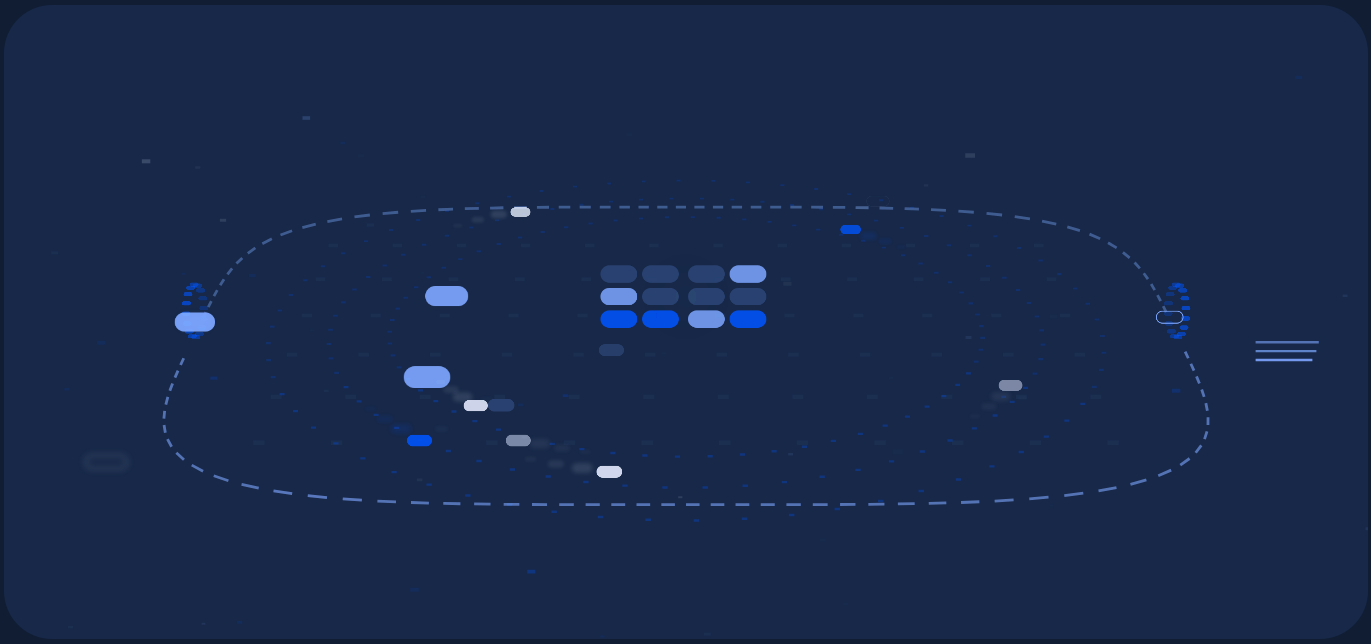
Rillor

EXPLAINER

MCP access control: scoped credentials, allow-lists and audit logs

How the Model Context Protocol connects superintelligence (SI) assistants to business tools, and the controls that keep that access governed: tenant isolation, scoped credentials, allow-lists and audit logs.



Rillor

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

- The Model Context Protocol (MCP) standardizes how SI applications connect to tools and data. It was introduced in November 2024 and has been a project of the Agentic AI Foundation, a directed fund under the Linux Foundation, since December 2025 [1, 14].
- The specification sets firm rules for authorization over HTTP: an MCP server is an OAuth 2.1 resource server, tokens are bound to the server they were issued for, and passing a client's token through to other services is forbidden [4, 7]. It states consent, privacy and tool safety as principles and says plainly that the protocol "cannot enforce" them [2].
- Published attacks fall into four groups: poisoned or changed tool descriptions, harmful combinations of tools in one session, misuse of credentials, and isolation failures in servers shared by many customers [7, 17, 18, 19, 20].
- Five controls answer them: tenant isolation (an isolated server per tenant, or per-call credential scoping on a shared server), scoped and audience-bound credentials, allow-lists of servers and tools, human approval for consequential actions, and audit logs with defined content. Each maps to MCP requirements and to general security standards: NIST SP 800-53 and the OWASP guidance for LLM applications [21, 22].
- Revision 2026-07-28 makes several of these controls easier to enforce: every request is self-contained, tool names travel in HTTP headers that gateways can read, a server's tool list may vary with the caller's authorization, credentials are bound to their issuer and trace context is documented [3, 5, 6].

Background

The problem

An assistant that can only answer questions from its training data is of limited use at work. An assistant that can read a ticket, query a database or update a customer record is useful, and it is also a new kind of user of business systems. It acts on instructions it reads, some of which come from content it was asked to process. Every connection between an assistant and a tool is therefore an access decision: which data the assistant can reach, which actions it can take, on whose behalf, and with what record left behind.

Before MCP, each connection was custom. Anthropic's announcement described the result: "Every new data source requires its own custom implementation, making truly connected systems difficult to scale" [1]. A common protocol reduces that cost. It also makes it easy to connect many tools quickly, which raises the stakes of getting governance right.

The protocol

The specification defines three roles that exchange JSON-RPC 2.0 messages [2].

- **Hosts**, the SI applications that start connections.
- **Clients**, the connectors inside a host.
- **Servers**, the services that provide context and capabilities.

Servers offer resources (context and data), prompts (templated messages and workflows) and tools ("functions for the AI model to execute") [2]. The specification defines a transport over standard input and output, for servers run as a local process, and Streamable HTTP, for servers reached over a network [4, 6].

Authorization is optional in the specification. When it is supported, implementations on an HTTP transport "SHOULD conform" to the authorization chapter, and implementations on standard input and output "SHOULD NOT follow this specification, and instead retrieve credentials from the environment" [4].

The specification's security section sets out principles: users "must explicitly consent to and understand all data access and operations"; hosts "must obtain explicit user consent before invoking any tool"; and "Tools represent arbitrary code execution and must be treated with appropriate caution" [2]. It then says: "While MCP itself cannot enforce these security principles at the protocol level, implementors SHOULD" build consent and authorization flows, document security implications, implement access controls and data protections, and follow security best practices [2].

How the specification has changed

Revision	Changes relevant to governance	Source
2024-11-05	The revision that 2025-03-26 replaced; defined the HTTP+SSE transport	[6, 8]
2025-03-26	Authorization framework based on OAuth 2.1; Streamable HTTP transport; tool annotations describing behavior, such as read-only or destructive	[8]
2025-06-18	MCP servers classified as OAuth resource servers with protected resource metadata; clients required to implement resource indicators "to prevent malicious servers from obtaining access tokens"; new security best practices page	[9]
2025-11-25	Incremental scope consent through `WWW-Authenticate`; Client ID Metadata Documents as a recommended registration mechanism; experimental tasks; formal governance structure	[10]
2026-07-28	Protocol-level sessions removed and the initialize handshake dropped; required `Mcp-Method` and `Mcp-Name` request headers; issuer validation under RFC 9207; credentials bound to their issuer; Dynamic Client Registration deprecated in favor of Client ID Metadata Documents; trace context conventions; Roots, Sampling and Logging deprecated	[3]

Adoption

Date	Milestone	Source
25 November 2024	Anthropic introduces MCP with a specification, SDKs, local server support in its desktop apps and an open-source repository of servers	[1]
21 May 2025	OpenAI's Responses API supports remote MCP servers; OpenAI "has also joined the steering committee for MCP"	[12]
8 September 2025	The MCP Registry opens in preview as "an open catalog and API for publicly available MCP servers"	[11]
9 December 2025	MCP donated to the Agentic AI Foundation, co-founded by Anthropic, Block and OpenAI. The announcement reports "more than 10,000 active public MCP servers", "97M+ monthly SDK downloads across Python and TypeScript" and adoption by popular SI products and developer tools	[14]

Date	Milestone	Source
28 July 2026	Revision 2026-07-28 published. Across Tier 1 SDKs, the maintainers report "close to half-a-billion downloads a month", with the TypeScript and Python SDKs each past one billion total downloads	[15]

Method

This is an explainer. We read revision 2026-07-28 of the MCP specification, including its overview, tools, authorization and Streamable HTTP chapters [2, 4, 5, 6], its changelog and those of the three prior revisions [3, 8, 9, 10], and the project's security best practices [7]. We read published research on attacks against MCP deployments [17, 18, 19], one public incident report [20], one model provider's guidance on connecting to MCP servers [13], the OWASP entry on excessive agency [21], the access control and audit controls in NIST SP 800-53 Rev. 5 [22], the Kubernetes guidance on multi-tenancy [23] and RFC 8707 [24].

From these we grouped the documented threats, derived a set of controls that answers them, and mapped each control to the specification's own requirements and to the general standards. We did not test any MCP client, server or registry.

Findings

1. What the specification requires

The authorization and transport chapters contain firm requirements. The rest of the security model is guidance.

Area	Requirement	Strength	Source
Server role	A protected MCP server acts as an OAuth 2.1 resource server and implements OAuth 2.0 Protected Resource Metadata (RFC 9728)	MUST	[4]
Token audience	Clients include a `resource` parameter naming the server in authorization and token requests (RFC 8707)	MUST	[4, 24]
Token validation	Servers "validate that access tokens were issued specifically for them as the intended audience" and "MUST NOT accept or transit any other tokens"	MUST	[4]
Token transport	The `Authorization` header on every HTTP request; never a token in the URI query string	MUST	[4]
Issuer binding	Clients validate a present `iss` against the recorded issuer before redeeming a code; credentials are keyed by issuer and not reused with another authorization server	MUST	[3, 4]
Scopes	Servers indicate required scopes in challenges; clients request only the scopes needed, following least privilege	SHOULD	[4]
Origin	Servers validate the `Origin` header on all incoming connections; local servers bind only to localhost	MUST; SHOULD	[6]
Request headers	`Mcp-Method` on all requests and `Mcp-Name` on tool calls, resource reads and prompt requests; servers reject requests whose headers and body disagree	REQUIRED; MUST	[6]
Tool servers	"Validate all tool inputs", "Implement proper access controls", "Rate limit tool invocations", "Sanitize tool outputs"	MUST	[5]
Tool clients	Prompt for confirmation on sensitive operations, show inputs before calling, validate results, set timeouts, "Log tool usage for audit purposes"	SHOULD	[5]

Area	Requirement	Strength	Source
Human in the loop	"there SHOULD always be a human in the loop with the ability to deny tool invocations"	SHOULD	[5]
Annotations	Clients "MUST consider tool annotations to be untrusted unless they come from trusted servers"	MUST	[5]
State handles	Servers "MUST NOT treat possession of a state handle as authentication"	MUST	[7]

Two features of this table matter for governance. First, the firm requirements protect the connection: who holds a token, which server it is for, whether a request is what its headers say. They say little about which tools a given user should reach or which actions need a person's approval. Those are operator decisions. Second, the authorization chapter applies to HTTP transports. A server run as a local process takes its credentials from the environment [4], so its access is whatever the process can reach.

2. Threats documented in published work

Threat	What happens	Published account	Controls that answer it
Tool poisoning	Instructions hidden in a tool description steer the model; users "have no visibility into the full tool descriptions"	Invariant Labs, April 2025 [17]	Allow-lists; pinned descriptions; approval
Rug pull	"a malicious server can change the tool description after the client has already approved it"	Invariant Labs [17]	Descriptions pinned by hash; re-approval on change
Shadowing	One server's descriptions alter how the agent uses another, trusted server	Invariant Labs [17]	Per-session limits on which servers combine; isolation
Toxic agent flow	A malicious issue in a public repository leads an agent to pull private repository data into context and publish it	Invariant Labs, May 2025 [18]	Per-session allow-lists, such as one repository per session; least privilege
Lethal trifecta	An agent with private data, untrusted content and a way to communicate externally can be tricked into sending the data out	Simon Willison, June 2025 [19]	Allow-lists that keep the three apart; egress limits; approval

Threat	What happens	Published account	Controls that answer it
Confused deputy	A proxy server with a static client ID and dynamic client registration lets an attacker obtain an authorization code without the user's consent	MCP security best practices [7]	Per-client consent; exact redirect URI matching
Token passthrough	A server accepts and forwards tokens issued for other services, bypassing controls and muddying audit trails	MCP security best practices [7]	Audience-bound tokens
Server-side request forgery	A malicious server points OAuth discovery URLs at internal addresses or cloud metadata endpoints	MCP security best practices [7]	Egress proxies; blocked private address ranges
State handle hijacking	An attacker guesses or obtains a handle and acts on another user's state	MCP security best practices [7]	Handles bound to the verified user
Cross-tenant exposure	A logic flaw in a shared MCP server exposes one customer's data to other customers' users	Asana, disclosed June 2025 [20]	Tenant isolation: an isolated server per tenant, or per-call credential scoping
Over-broad scopes	A stolen token with wildcard scopes opens every tool; one omnibus scope "masks user intent per operation"	MCP security best practices [7]	Scope minimization; step-up authorization

The Asana case is a public example of an isolation failure. BleepingComputer reported that Asana released its MCP server on 1 May 2025, discovered "the logic flaw that created this exposure on June 4", and said the issue "impacts roughly 1,000 customers"; exposure was limited to each user's access scope, and the company described it as "a logic flaw in the MCP system and not the result of a hack" [20].

One model provider's documentation adds a blunt summary of the risk from untrusted servers: "A malicious server can exfiltrate sensitive data from anything that enters the model's context" [13].

3. Five controls

Control 1: tenant isolation. No request made for one customer reaches another customer's data or credentials. Two designs meet it.

- An isolated server per tenant. Each customer gets its own MCP server instance with its own runtime, its own credentials store and its own network policy. No tool runtime is shared between customers. The logic flaw behind a cross-tenant exposure then has no other tenant's data within reach.

- Per-call credential scoping on a shared server. One server serves many customers, and every tool call is scoped to one of them. The server takes the customer from the verified token on the request, never from a value the client supplies [7]. It loads only that customer's credentials for that call, and keeps every customer's credentials encrypted and out of the model's context. Tokens are accepted only by the server they were issued for [4]. The boundary then sits in each call's authorization, so the code that maps a verified token to a customer and loads that customer's credentials is the part to review and test.

On Kubernetes, the documentation describes namespaces as "a mechanism for isolating groups of API resources within a single cluster" and notes that role-based access control rules and network policies are namespace-scoped [23]. Its network guidance says that where strict isolation between tenants is required, "starting with a default policy that denies communication between pods is recommended", with a rule that allows DNS, and narrower allow rules added from there [23]. It also warns that network policies require a network plugin that implements them, "Otherwise, NetworkPolicy resources will be ignored", and that containers "offer a weaker isolation boundary than virtual machines" [23]. The choice of boundary, namespace, node or dedicated cluster, follows from how far tenants trust each other; the documentation describes "hard" multi-tenancy as the case "where the tenants do not trust each other" [23].

Control 2: scoped, audience-bound credentials. Credentials are issued per tenant and per server, carry only the scopes that the work needs and name the server they are for. The specification supplies the mechanics: resource indicators in every authorization and token request, audience validation at the server and a ban on token passthrough [4, 24]. Downstream credentials, the keys the server uses to reach a business system, stay on the server inside their tenant's boundary and never enter the model's context.

Scopes start small. The security guidance recommends a "minimal initial scope set" of low-risk discovery and read operations, with "incremental elevation via targeted `WWW-Authenticate scope="..."` challenges when privileged operations are first attempted" [7]. It lists as common mistakes "Publishing all possible scopes in `scopes_supported`", "Using wildcard or omnibus scopes (`*`, `all`, `full-access`)" and "Bundling unrelated privileges to preempt future prompts" [7]. Revision 2026-07-28 adds that clients "MUST NOT reuse" credentials with a different authorization server [3].

Control 3: allow-lists of servers and tools. Each tenant, and where needed each user role, gets an explicit list of servers and of tools on each server. Three mechanisms support this.

- At the server, revision 2026-07-28 lets the tool list "vary by the authorization presented on the request", for example by returning only the tools the caller's granted scopes permit [5].
- At the network, every HTTP request carries its method in a header, and every tool call carries the tool name, mirrored "so that intermediaries (load balancers, gateways, observability tooling) can route and inspect requests without parsing the body" [6]. A gateway can enforce the list from the header. Servers must reject a request whose headers and body disagree, and intermediaries that enforce policy on mirrored headers "SHOULD verify" that the protocol version requires that check [6].
- At the client, a model provider's guidance shows an `allowed_tools` parameter that imports only named tools from a server [13].

Allow-lists also limit composition. The toxic agent flow was stopped in the researchers' example by a policy that allows access to "only one repo per session" [18]. The same idea keeps private data, untrusted content and outbound communication out of a single session [19]. Tool descriptions on the list are pinned by hash, so a changed description is caught and sent for review before the model sees it [17].

For larger organizations, the Enterprise-Managed Authorization extension moves the server list to the identity provider, which "controls which MCP servers employees can access, and under what conditions" [16]. Access policies sit in one place, and revoking an employee's access "happens at the IdP level, taking effect immediately across all MCP clients" [16]. The extension's own documentation notes that support "varies by client" [16].

Control 4: approval for consequential actions. The specification says there "SHOULD always be a human in the loop with the ability to deny tool invocations", and asks applications to "Present confirmation prompts to the user for operations" [5]. The operator decides which tools need approval, by policy. A tool's own annotations are hints: clients "MUST consider tool annotations to be untrusted unless they come from trusted servers" [5]. A model provider's guidance shows the same pattern in its API: approval is requested by default before data is shared with a remote MCP server, and "Each approval applies to one tool call" [13]. OWASP lists "Require user approval" for high-impact actions and "complete mediation", implementing authorization "in downstream systems rather than relying on an LLM to decide if an action is allowed" [21].

Control 5: audit logs with defined content. Every tool call produces a record that answers the questions in NIST SP 800-53 AU-3: what type of event occurred, when, where, its source, its outcome and the identity of the individuals or entities involved [22]. For a tool call that means the tenant, the user, the client, the server instance, the tool name, a digest of the arguments, the scopes on the token, the result or error and the time in UTC. Use of privileged functions is logged (AC-6(9)), and the logs are protected from unauthorized access, modification and deletion (AU-9) [22].

The MCP documents point the same way. Clients "SHOULD" log tool usage for audit purposes [5]. Servers should "Log elevation events (scope requested, granted subset) with correlation IDs" [7]. Token passthrough is forbidden partly because it breaks the audit trail: the server "will be unable to identify or distinguish between MCP Clients", and downstream logs "may show requests that appear to come from a different source" [7]. Revision 2026-07-28 documents trace context propagation through `traceparent`, `tracestate` and `baggage` in request metadata, so a call can be followed from the assistant to the business system [3]. The same revision deprecates the protocol's Logging feature and suggests logging to standard error or using OpenTelemetry instead [3]; audit records belong in the operator's own systems in any case. A model provider's guidance also recommends "logging any data sent to MCP servers" [13].

4. Control map

Control	MCP specification	NIST SP 800-53 Rev. 5 [22]	OWASP LLM06:2025 [21]
Tenant isolation	Servers implement access controls [5]; sandbox local servers, restrict egress [7]; bind state to the user in the verified token [7]	SC-39 Process Isolation (isolated servers); AC-4 Information Flow Enforcement; AC-6 Least Privilege (per-call credentials)	No direct entry; supports minimizing extension permissions
Scoped, audience-bound credentials	Resource indicators, audience validation, no passthrough, scope selection, step-up [4, 7]	AC-6 Least Privilege	Minimize extension permissions; execute extensions in the user's context
Allow-lists	Tool list may vary by authorization [5]; mirrored headers for gateways [6]; enterprise-managed server access [16]	AC-6 Least Privilege; AC-4 Information Flow Enforcement	Minimize extensions; minimize extension functionality; avoid open-ended extensions
Approval for consequential actions	Human in the loop; annotations untrusted [5]	No single control; supports AC-6	Require user approval; complete mediation
Audit logs	Log tool usage [5]; log scope elevation with correlation IDs [7]; trace context [3]	AU-3 Content of Audit Records; AU-9 Protection of Audit Information; AC-6(9) Log Use of Privileged Functions	Log and monitor extension and downstream activity

The mapping is ours. NIST SP 800-53 controls are written for information systems in general, and OWASP's mitigations for LLM applications in general; neither names MCP.

5. What revision 2026-07-28 changes for governance

The 2026-07-28 revision removed protocol-level sessions and the initialize handshake. Every request now carries its protocol version and client capabilities [3]. The maintainers note that "any request can land on any instance behind a plain round-robin load balancer" [15]. For governance this has two effects. Authorization is checked on every request, with nothing inherited from a connection. And any state a server keeps across calls lives behind explicit handles, which the security guidance says must be bound server-side to the authenticated user, "where the user ID is derived from the verified token rather than supplied by the client" [7].

The required request headers give gateways a reliable place to enforce allow-lists and per-tenant rate limits, provided the server validates that headers match the body [6]. The tool list's ability to vary by authorization lets a server show each caller only what it may use [5]. Issuer validation and issuer-bound credentials close an authorization-server mix-up path, and Dynamic Client Registration is deprecated in favor of Client ID Metadata Documents [3, 15]. The deprecated features remain usable for at least twelve months under the new deprecation policy [3].

6. Tenant isolation on Rillor's platform

Rillor builds and secures agentic SI systems. The tenant isolation tests on Rillor's platform, and what an agentic security engagement covers, are on our agentic security (https://rillor.com/si/agentic-security) page.

Implications

For security and procurement teams. Ask a provider of MCP-based tool access these questions:

1. Does each customer get its own server and runtime? If customers share a server, is every call scoped to one customer's credentials, with the customer taken from the verified token?
2. Are tokens bound to a single server, and does the server reject tokens issued for anything else?
3. Which scopes does the integration request at first use, and how does it ask for more?
4. Can we set the list of servers and tools per user role, and is it enforced outside the model?
5. Which actions require a person's approval, and who sets that policy?
6. What does an audit record of a tool call contain, where is it kept, and who can change it?
7. How are changes to a tool's description detected and reviewed?

For builders of MCP servers. Follow the firm requirements first: audience validation, no passthrough, Origin checks, header and body validation. Then publish small scope sets, return only the tools a caller may use and log scope elevation. Treat any state handle as a name to authorize on every call.

For platform operators. Start tenant networks from default deny, keep each tenant's credentials in that tenant's boundary and choose the isolation boundary by how far tenants trust each other [23]. Keep audit records outside the tenant boundary, where a compromised server cannot alter them.

Limits of this analysis

This explainer is based on published documents. We did not test any MCP client, server, gateway or registry, and the requirements quoted here describe what the specification asks of implementations, not what any implementation does.

The specification changes between revisions. Requirements are quoted from revision 2026-07-28 unless a different revision is named. Earlier revisions remain in use, and several features deprecated in 2026-07-28 stay available during the deprecation window [3].

Adoption figures are as reported by Anthropic and by the MCP maintainers [14, 15]. The account of the Asana incident relies on press reporting of the company's disclosure [20].

The mapping of controls to NIST SP 800-53 and OWASP is our own. It is not a compliance assessment and does not establish that any system meets either document.

The tenant isolation tests on the Rillor platform are Rillor's own.

Sources

1. "Introducing the Model Context Protocol". Anthropic, 25 November 2024. https://www.anthropic.com/news/model-context-protocol

2. Model Context Protocol Specification, revision 2026-07-28: Overview and Security and Trust & Safety. Model Context Protocol project, retrieved 9 October 2026. https://modelcontextprotocol.io/specification/2026-07-28

3. Model Context Protocol Specification, revision 2026-07-28: Key Changes. Model Context Protocol project, retrieved 9 October 2026. https://modelcontextprotocol.io/specification/2026-07-28/changelog

4. Model Context Protocol Specification, revision 2026-07-28: Authorization. Model Context Protocol project, retrieved 9 October 2026. https://modelcontextprotocol.io/specification/2026-07-28/basic/authorization

5. Model Context Protocol Specification, revision 2026-07-28: Tools. Model Context Protocol project, retrieved 9 October 2026. https://modelcontextprotocol.io/specification/2026-07-28/server/tools

6. Model Context Protocol Specification, revision 2026-07-28: Streamable HTTP transport. Model Context Protocol project, retrieved 9 October 2026. https://modelcontextprotocol.io/specification/2026-07-28/basic/transports/streamable-http

7. Security Best Practices (documentation for revision 2026-07-28). Model Context Protocol project, retrieved 9 October 2026. https://modelcontextprotocol.io/docs/2026-07-28/tutorials/security/security_best_practices

8. Model Context Protocol Specification, revision 2025-03-26: Key Changes. Model Context Protocol project. https://modelcontextprotocol.io/specification/2025-03-26/changelog

9. Model Context Protocol Specification, revision 2025-06-18: Key Changes. Model Context Protocol project. https://modelcontextprotocol.io/specification/2025-06-18/changelog

10. Model Context Protocol Specification, revision 2025-11-25: Key Changes. Model Context Protocol project. https://modelcontextprotocol.io/specification/2025-11-25/changelog

11. David Soria Parra, Adam Jones, Tadas Antanavicius, Toby Padilla and Theodora Chu, "Introducing the MCP Registry". Model Context Protocol blog, 8 September 2025. https://blog.modelcontextprotocol.io/posts/2025-09-08-mcp-registry-preview/

12. "Introducing support for remote MCP servers, image generation, Code Interpreter, and more in the Responses API". OpenAI Developer Community, Announcements, 21 May 2025. https://community.openai.com/t/introducing-support-for-remote-mcp-servers-image-generation-code-interpreter-and-more-in-the-responses-api/1266973

13. "MCP servers" (connectors and MCP guide, API documentation). OpenAI, retrieved 9 October 2026. https://developers.openai.com/api/docs/guides/tools-connectors-mcp

14. "Donating the Model Context Protocol and establishing the Agentic AI Foundation". Anthropic, 9 December 2025. https://www.anthropic.com/news/donating-the-model-context-protocol-and-establishing-of-the-agentic-ai-foundation

15. David Soria Parra and Den Delimarsky, "The 2026-07-28 Specification". Model Context Protocol blog, 28 July 2026. https://blog.modelcontextprotocol.io/posts/2026-07-28/

16. Enterprise-Managed Authorization (extension `io.modelcontextprotocol/enterprise-managed-authorization`). Model Context Protocol project, retrieved 9 October 2026. https://modelcontextprotocol.io/extensions/auth/enterprise-managed-authorization

17. Luca Beurer-Kellner and Marc Fischer, "MCP Security Notification: Tool Poisoning Attacks". Invariant Labs, 1 April 2025. https://invariantlabs.ai/blog/mcp-security-notification-tool-poisoning-attacks

18. Marco Milanta and Luca Beurer-Kellner, "GitHub MCP Exploited: Accessing private repositories via MCP". Invariant Labs, 26 May 2025. https://invariantlabs.ai/blog/mcp-github-vulnerability

19. Simon Willison, "The lethal trifecta for AI agents: private data, untrusted content, and external communication". simonwillison.net, 16 June 2025. https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/

20. Bill Toulas, "Asana warns MCP AI feature exposed customer data to other orgs". BleepingComputer, 18 June 2025. https://www.bleepingcomputer.com/news/security/asana-warns-mcp-ai-feature-exposed-customer-data-to-other-orgs/

21. "LLM06:2025 Excessive Agency". OWASP Top 10 for LLM Applications 2025, OWASP Gen AI Security Project, 2025. https://genai.owasp.org/llmrisk/llm062025-excessive-agency/

22. NIST SP 800-53 Rev. 5, Security and Privacy Controls for Information Systems and Organizations (release 5.2.0, OSCAL catalog): controls AC-4, AC-6, AC-6(9), AU-3, AU-9 and SC-39. National Institute of Standards and Technology, catalog last modified 11 May 2026. https://github.com/usnistgov/oscal-content/blob/main/nist.gov/SP800-53/rev5/json/NIST_SP-800-53_rev5_catalog.json

23. "Multi-tenancy". Kubernetes documentation, The Kubernetes Authors, retrieved 9 October 2026. https://kubernetes.io/docs/concepts/security/multi-tenancy/

24. B. Campbell, J. Bradley and H. Tschofenig, "Resource Indicators for OAuth 2.0", RFC 8707. Internet Engineering Task Force, February 2020. https://www.rfc-editor.org/rfc/rfc8707

About this report

Published by Rillor on 7 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. MCP access control: scoped credentials, allow-lists and audit logs. 7 November 2026. https://rillor.com/insights/mcp-access-control

Notices

NVIDIA, AMD, Ceph, Kubernetes and related product names are trademarks of their owners and are used only to identify products. No affiliation or endorsement is implied.

Superintelligence (SI) outputs can be wrong. Important decisions should include human review.

Work with Rillor

Send us the agents, the tools and data they reach, where they run and what you want tested. Contact us at rillor.com/contact?type=security or contact@rillor.com. More on agentic security: rillor.com/si/agentic-security