The protocol: how a universal language for AI is unlocking a new era of machine intelligence

The better the prompt.
The better the answer.
The better the world works.

EY
Shape the future with confidence

Contents

Introduction

The integration impasse

In the rapidly accelerating world of artificial intelligence, a quiet paradox has taken root at the very heart of enterprise innovation. On one hand, the raw power of Large Language Models (LLMs) has reached astonishing levels, capable of generating human-like text, writing complex code, and reasoning through abstract problems. Yet, for all their cognitive prowess, these digital minds have largely operated in isolation, powerful but disconnected, their potential constrained by the very systems they are meant to revolutionize. They have been, in the words of one analysis, "trapped behind information silos and legacy systems," unable to easily access the live, contextual data that fuels genuine intelligence.[1] This disconnect has created a frustrating and inefficient reality for both the developers building the future of AI and the organizations seeking to deploy it.

At the core of this challenge lies a debilitating issue known in technical circles as the "NxM" integration problem. Described as a "developer's nightmare," the concept is one of exponential complexity: for every N number of distinct AI models, a unique, custom-built connection is required to integrate with every M number of external tools, databases, or software applications.[1] If a company uses three different AI models and wants them to interact with ten different internal systems – a customer relationship management (CRM) platform, a document repository, a financial database, and so on – it would need to build and maintain thirty separate, resource-intensive integrations. This brittle, bespoke approach has proven to be a formidable barrier to progress, stifling innovation, dramatically inflating development costs, and severely limiting the scalability and versatility of AI applications across the board.

The consequences of this fragmentation are profound and tangible. AI models, cut off from the flow of real-time information, are often hamstrung by "outdated knowledge issues," their responses limited to the static data they were trained on months or even years prior. This forces human users into the inefficient role of a manual bridge, laboriously copying and pasting information between their applications and an AI chat window, a disjointed experience that undermines the very promise of seamless automation.[1] This is more than just a technical or financial burden; it represents a fundamental strategic bottleneck. The inability of AI to easily and reliably connect to the outside world has been the primary obstacle preventing it from evolving beyond a passive "chatbot," capable only of answering questions, into a proactive and truly useful "digital assistant" capable of performing complex, real-world tasks. The integration impasse, born from the NxM problem, has not just been an inconvenience; it has been the architectural roadblock holding back the next generation of intelligent, autonomous AI. It is a problem that demanded a new, foundational solution, a universal standard to finally teach all machines to speak the same language.

Section I

A quiet revolution: the forging of a new standard

In November 2024, a quiet but profound shift began in the world of artificial intelligence when a technology research company introduced an open standard designed to solve the debilitating integration problem at the heart of the AI revolution. Known as the Model Context Protocol (MCP), it aims to become a universal language for AI, promising to do for machine intelligence what the ubiquitous USB-C standard did for consumer electronics.[1] Where once a chaotic tangle of proprietary chargers and cables was needed to connect a myriad of devices, a single, standardized port now reigns. MCP was conceived with a similar, and arguably more ambitious, goal: to create a universal digital connector that allows any AI application to seamlessly "plug and play" with any compliant tool or data source, finally eliminating the need for countless custom adapters.

The protocol was born out of a direct dissatisfaction with the NxM problem that had long plagued AI development.[3] A technology research company recognized that for AI to reach its full potential, it needed a standardized, open-source framework to govern how it integrates and shares data with the outside world. By establishing a common "language" and interface, MCP set out to eliminate the need for bespoke integrations, offering a model-agnostic way for AI to read files, execute functions, and handle contextual prompts from any number of external systems.

This ambition to standardize a chaotic technological landscape draws powerful parallels to other transformative moments in computing history. The protocol's design explicitly takes inspiration from the Language Server Protocol (LSP), which unified the way code editors like VS Code interact with different programming languages, fostering a rich ecosystem of developer tools. Even more fundamentally, MCP mirrors the impact of Open Database Connectivity (ODBC), a standard introduced in the early 1990s that provided a single, standard API for applications to access data from any database management system. Before ODBC, developers had to write custom code for each specific database they wanted to connect to; after ODBC, they could write database-agnostic code once, dramatically simplifying development and fostering an explosion of data-driven applications.

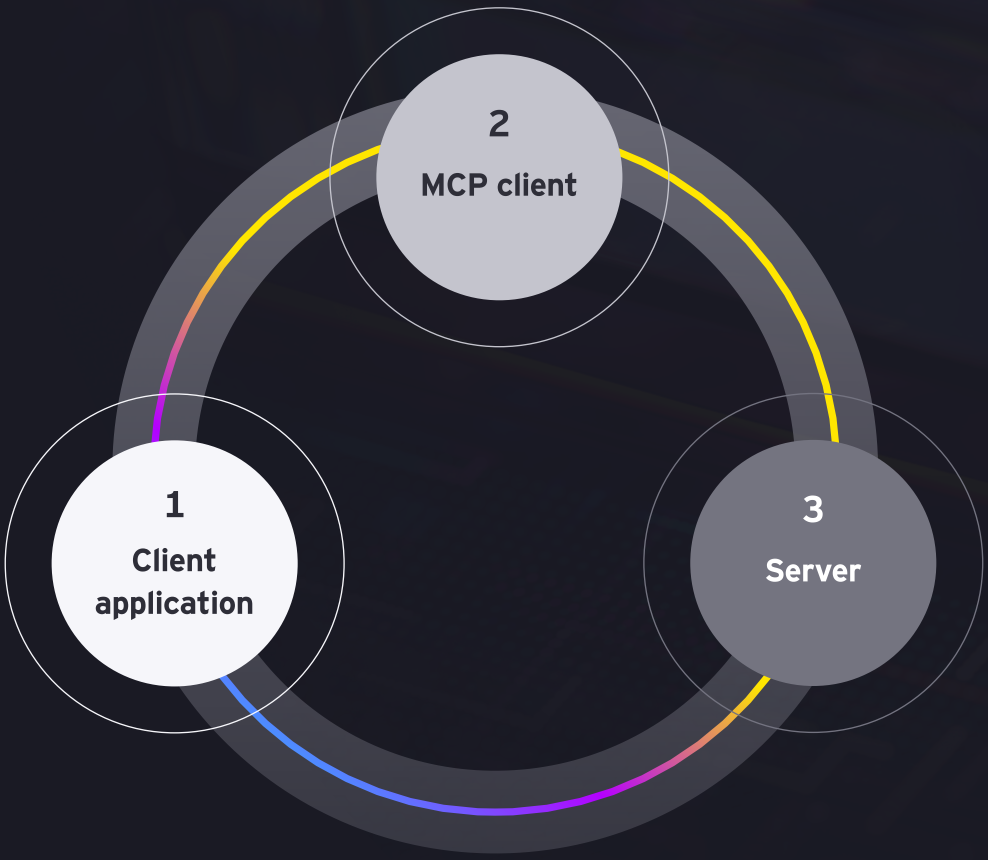
The decision to architect MCP as an open standard, rather than a proprietary plugin system, was a deliberate and strategic one. This approach was designed to foster a truly interoperable AI ecosystem and avoid the vendor lock-in that has fragmented other technology sectors. This open-source strategy is part of a well-understood playbook for establishing a foundational technology. By sacrificing initial proprietary control and lowering the barrier to entry for all developers and companies, a technology can achieve powerful network effects. As more tools are exposed as MCP servers and more AI applications are built as MCP clients, the value of the entire ecosystem increases exponentially for every participant. A new tool instantly becomes available to all existing applications, and a new application can instantly access all existing tools. This creates a virtuous cycle of adoption that, once it reaches a critical mass, establishes a de facto industry standard that becomes difficult for competing, closed-off approaches to challenge. The rapid endorsement of MCP by technology giants active in this market suggests this critical mass is already forming. This positions MCP not as just another product or framework, but as a candidate to become the foundational infrastructure – the digital equivalent of TCP/IP – for the entire agentic AI industry.

Section II

Under the hood: the architecture of connection

To achieve its ambitious goal of universal interoperability, the Model Context Protocol is built upon a robust and elegantly simple client-server architecture. Drawing inspiration from proven standards like the Language Server Protocol (LSP) and using the lightweight JSON-RPC 2.0 message format for communication, this layered design creates a clear separation of concerns that is fundamental to the protocol's security, scalability and flexibility. Rather than a monolithic structure, MCP carefully delineates the roles and responsibilities of each component, clarifying that the "intelligence" of an AI system is kept separate from the "capabilities" it can access.[5] This architecture is composed of three core components that are essential for secure integration in an enterprise.

The three core components



First is **the Client application**, which is the primary, user-facing AI application. This can be broken into two categories:

1. A desktop assistant like Claude Desktop, an AI-enhanced Integrated Development Environment (IDE) like Cursor or VS Code
2. A custom-built enterprise chatbot deployed as a web application running in user's browser.[2]

When deployed as a desktop assistant, it acts as the central "orchestrator" or "container" of the entire operation. It is responsible for managing connection, enforcing security policies, handling user consent for sensitive actions, and coordinating the overall flow of information to and from the Large Language Model at its core.

However, when deployed as a custom-built web application, much of the complexity stated above is delegated to an MCP Client which is now running as a backend service following BFF (Backend for Front-end) pattern. This approach ends up creating a lightweight chat client who has no knowledge of all the dependencies that its backend must track. Key security concerns around LLM API key management or access token handling are all done by at the backend while the chat client maintains a persistent connection for seamless flow of prompts and responses.

Second is **the MCP client**, when running as a desktop assistant this component is embedded within the Client Application but when hosted as a BFF it must run in a backend web service. This functions as a dedicated "translator" or "intermediary."[2] An MCP Client can work with multiple MCP tools spread across multiple MCP Servers while maintaining a secure, one-to-one connection with a single, specific MCP Server. The client's job is to take the AI model's needs or requests and translate them into the standardized MCP format before sending them to the server, and to handle the responses that come back. It serves as the session manager, handling everything from connection interruptions, user authorization, token caching to error handling.

Third is **the Server**, a lightweight and typically independent program that acts as a secure "gateway" or "wrapper" for a specific external system or data source. Each MCP server is designed to be simple and focused, exposing a well-defined set of capabilities for a single integration point, such as a GitHub repository, a PostgreSQL database, a third-party API like Slack, your own custom built Enterprise APIs or even the local file system. The design philosophy is that servers should be extremely easy to build, with the complex orchestration logic handled by the MCP Client.[4]

The architectural separation of these components is a hallmark of a maturing technological field. Early, less-developed systems are often monolithic, with all functions tightly interwoven, making them brittle and difficult to maintain. MCP's modular design, by contrast, formalizes a separation of concerns that is defining the emerging layers of the modern AI agent stack. The Host acts as the "brain," responsible for reasoning and orchestration. The Server acts as the "hands and eyes," providing the connection to external capabilities. And the protocol itself, MCP, serves as the standardized "nervous system" that carries signals between them. This structure allows for specialization and composability; one team can focus on building a brilliant AI brain, while another can build a robust server for a specific enterprise system, and the two can work together seamlessly without custom integration, all thanks to the standardized protocol layer in the middle.

The interaction flow and its primitives

The communication between these components follows a well-defined, standardized process often referred to as a digital handshake.[3] The flow begins with:

Connection establishment, where an MCP Client within a Host application initiates a secure connection to an MCP Server, either locally via standard input/output (STDIO) or remotely over HTTP with Streamable HTTP. This initial handshake also involves negotiating the protocol version to facilitate compatibility.[4]

Once connected, the process moves to **Capability discovery**. The client queries the server to ask, in essence, "What can you do?" The server responds with a detailed, structured list of its available functionalities, which are organized into three distinct primitives. This dynamic discovery allows the AI application to understand at runtime what external capabilities are available to it. Based on a user's request, the LLM within the Host then performs.

Tool/Resource selection, identifying the most appropriate function or data source needed to fulfill the request.

This leads to **Action execution**. The client sends a formal request to the server, specifying the chosen function and any necessary arguments. The server then executes this action—querying its underlying database, calling an external API, or reading a file – and upon completion, it performs the final step: **Result return**. The server sends the result back to the client in a standardized JSON format, and the Host application integrates this new information into the AI's context to formulate its final response to the user.

This entire interaction is governed by the three standardized primitives, which form the core language of MCP and define how context and capabilities are provided. These are:

- **Tools**: These are executable functions that allow the AI model to act upon the world. They are the verbs of the MCP language, enabling the AI to perform a side effect like querying a database, sending a Slack message, or creating a new file. Tools are described as "model-driven," meaning the AI model itself decides when and how to invoke them, giving it the "eyes and hands" to perceive and manipulate its digital environment.
- **Resources**: These represent structured, typically read-only data streams that provide the AI with external context. They are the nouns of MCP, representing entities like files, logs, API responses or database schemas that the model can access to inform its reasoning. Resources are considered "application-driven," as the client application often decides how to fetch and use this data, for instance, for caching or building an analytical dashboard.[7]
- **Prompts**: These are reusable instruction templates that guide the AI's behavior or provide convenient shortcuts for the user. They can be thought of as conversational recipes or pre-defined workflows, such as a "summarize recent issues" command that dynamically pulls data from a project management tool. Prompts are "user-driven," initiated directly by the end-user to trigger a consistent and predictable interaction with the AI.[7]

This granular distinction between Tools (action), Resources (data), and Prompts (guidance) is a deliberate design choice that is foundational to MCP's security and efficiency. It allows developers to precisely define and limit what an AI can do, what it can know, and how it should behave, leading to more controlled, reliable and powerful AI applications.

Section III

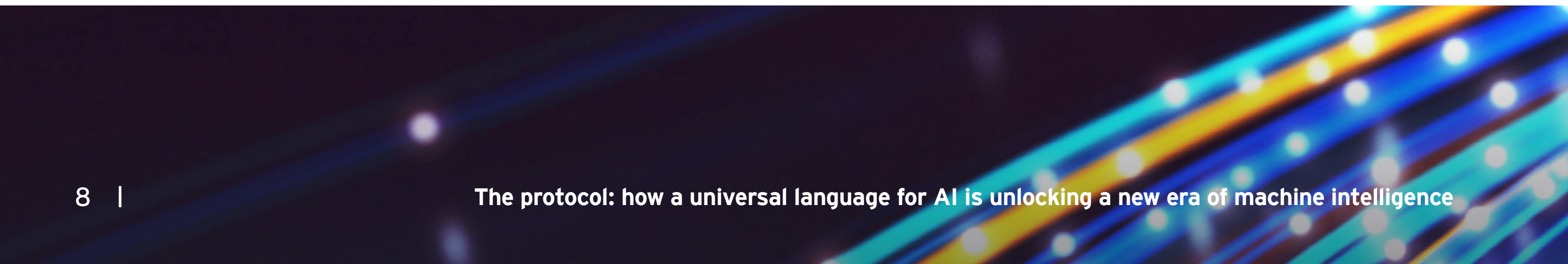
The new competitive edge: from static knowledge to dynamic action

The true value of the Model Context Protocol extends far beyond simplifying code or standardizing connections; it represents a fundamental paradigm shift in the role of artificial intelligence within an organization. By providing a reliable bridge to the outside world, MCP is the key catalyst transforming AI from a passive, knowledge-based "chatbot" into a proactive, "agentic" assistant capable of real-time awareness and dynamic action.[1] This evolution from static knowledge to dynamic capability is where the protocol's profound competitive advantage lies.

For years, the utility of even the most powerful LLMs was constrained by their "outdated knowledge" problem; they could only reason based on the information present in their training data, a static snapshot of the world from months or years ago. MCP shatters this limitation. Through its architecture, it provides AI with what the protocol's creators describe as "eyes and hands" in the digital world. The AI can now use "Resources" to perceive live, real-time data from any connected source, and use "Tools" to manipulate external systems in a controlled and deliberate manner. An AI assistant is no longer just a repository of facts; it is an active participant in live business processes, capable of querying today's sales figures, reading the latest project update, or executing a command in a live production environment.

This leap in capability translates directly into tangible business value and a significant economic advantage. The most immediate impact is a dramatic reduction in development costs and an accelerated time-to-market for new AI-powered features. The protocol effectively transforms the costly MxN integration problem into a simple M+N opportunity; instead of multiplying complexity, each new AI model or tool now adds value to the entire ecosystem with minimal overhead. One analysis suggests that an integration project that might have taken twelve weeks of custom development could be completed in just three weeks using MCP, representing a cost reduction of over 70%.[8]

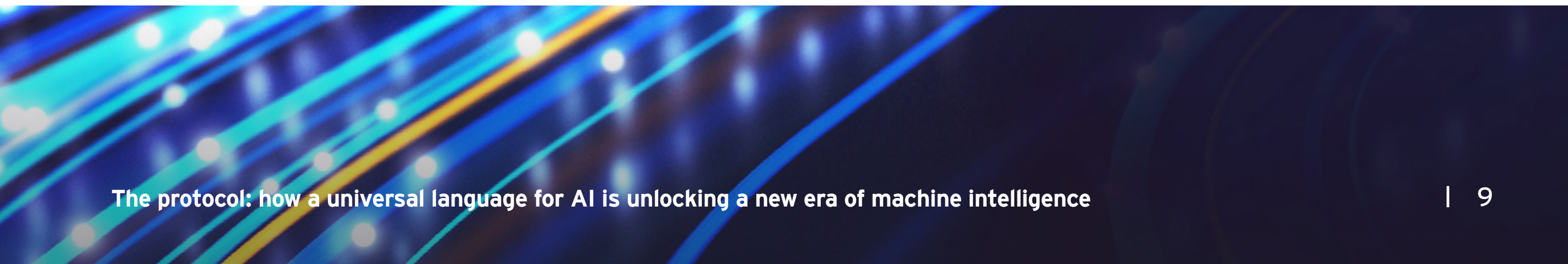
Furthermore, MCP helps to democratize AI integration. By abstracting away the low-level complexities of disparate APIs, authentication mechanisms, and data formats, it lowers the technical barrier for connecting AI to existing enterprise systems. This empowers a broader range of professionals, not just elite coders, to build and deploy AI solutions, shifting the organizational focus from tedious API wrangling to higher-level business logic and strategic innovation.



The most compelling evidence of this value proposition is found in the array of powerful use cases that MCP has already unlocked across industries:

- **Enterprise AI assistants**: Companies like Block and Apollo are leveraging MCP to build sophisticated internal assistants. These AI agents can securely access and reason over proprietary company documents, query live data from Customer Relationship Management (CRM) systems, and tap into internal knowledge bases to provide employees with instant, context-aware answers and automate internal workflows.[1]
- **Software development environments**: In the world of software engineering, MCP is driving a new wave of productivity. AI-enhanced IDEs such as Zed, Cursor, and Windsurf have integrated the protocol to provide coding assistants with real-time access to a project's entire codebase. This allows the AI to offer highly accurate code suggestions, assist in complex debugging tasks, and even execute code directly, all without the developer ever having to leave their primary work environment.
- **Complex multi-agent workflows**: Perhaps the most transformative application is MCP's ability to facilitate complex, cross-system automation involving multiple, collaborating AI agents. An MCP-enabled IT support bot, for example, can autonomously diagnose a user's problem by accessing knowledge bases, then query live device logs for error messages, and finally interact with a ticketing system to create and assign a resolution task—all without human intervention. Similarly, a sales workflow can be orchestrated across multiple agents: a calendar agent schedules a meeting, which triggers an email agent to send a confirmation, which in turn prompts a CRM agent to log the interaction and update the lead status.
- **Regulated and sensitive industries**: The protocol is also finding traction in high-stakes environments. Medical research institutions are exploring MCP-powered workflows for sensitive tasks like data anonymization and diagnostic pattern recognition from complex patient records. In finance, the ability to connect AI to real-time market data and transaction systems is enabling more adaptive and effective fraud detection models.

These examples reveal a deeper strategic implication. MCP is not just a tool for integrating AI; it is the foundational infrastructure for what is known as the "composable enterprise." In this model, an organization's digital capabilities – its databases, applications, and business logic – are exposed as standardized, interchangeable building blocks via MCP servers. Autonomous AI agents can then dynamically discover these building blocks, combine them in novel ways, and orchestrate them to execute new workflows on the fly, without requiring lengthy and expensive software development cycles. An agent could, for instance, combine a tool from the finance system with a tool from the logistics system to answer a complex supply chain query that previously required days of manual work across two separate departments. This marks a fundamental shift in how businesses can build and adapt their digital operations, enabling a new and unprecedented level of agility and intelligent automation.



Section IV

A crowded field: navigating the AI integration landscape

The Model Context Protocol did not emerge in a vacuum. It entered a dynamic and crowded ecosystem of technologies all aiming to enhance the capabilities of Large Language Models. To fully grasp its strategic position, it is crucial to understand how MCP relates to – and differs from – other prominent approaches, particularly OpenAI's Function Calling mechanism and developer-centric frameworks like LangGraph. This comparative analysis reveals a landscape of complementary technologies that are beginning to form the distinct layers of a modern, sophisticated AI stack.

MCP vs. OpenAI Function Calling

While both MCP and OpenAI Function Calling are designed to connect LLMs with external tools, they serve distinct and fundamentally complementary roles. OpenAI Function Calling (now often referred to simply as "Tools" within its API) is primarily a mechanism for **instruction generation**. Its core purpose is to enable an LLM to translate a user's natural language request into a structured, machinereadable JSON object that specifies which function to call and what parameters to use. For example, if a user asks, "What is the weather in London?", the model uses Function Calling to output a directive like {"function": "get_weather", "location": "London"}. It is critical to understand that the model itself does not execute this function; it merely formulates the instruction.

The Model Context Protocol, in contrast, operates at the next layer down: **standardized execution**. MCP provides the universal protocol that takes the structured instruction generated by any LLM's function calling capability and facilitates that it is securely and reliably executed by the correct external tool. It manages the entire process, from discovering which tools are available to invoking the function and handling the response. In this relationship, the AI Host application acts as the intermediary, converting the LLM's specific output format into a standardized MCP request that can be understood by any MCP-compliant server.

This layered architecture, with function calling for instruction generation and MCP for standardized execution, is analogous to the OSI model in computer networking, where different protocols handle distinct layers of communication. Function calling operates at a higher, model-specific layer, translating intent into a structured call. MCP operates at a lower, universal execution layer, supporting that call can be consistently serviced across a diverse and growing ecosystem of tools, regardless of which LLM originated it. This modularity is a sign of increasing maturity in AI systems architecture, moving away from monolithic designs toward composable components essential for enterprise-grade reliability and scale.

MCP vs. LangGraph and other agent development frameworks

Similarly, MCP's relationship with agent development frameworks like LangGraph, AG2, and CrewAI is one of synergy, not competition. These frameworks are comprehensive, developer-centric toolkits designed for building the "brain" of an AI application. They provide the high-level logic for constructing complex chains of thought, managing conversational memory, and orchestrating multi-step tasks that may involve calling multiple tools in a specific sequence. LangGraph is akin to an electronics workbench, providing a versatile set of components for a skilled developer to build a sophisticated, custom robot.

MCP, once again, provides the foundational **standardized plumbing** that this brain connects to. It handles the low-level, standardized connectivity to the outside world, supporting that the agent built with LangGraph can discover and interact with external tools and data sources in a consistent, plug-and-play manner. This creates a powerful division of labor: MCP abstracts away the complexities of individual API integrations, allowing frameworks like LangGraph to focus on what they do best – complex reasoning, planning, and agentic control flows. This synergy has led to the emergence of powerful hybrid architectures, where developers use MCP for robust, standardized tool access and a framework like LangGraph to build the sophisticated orchestration logic that uses those tools to achieve a goal.

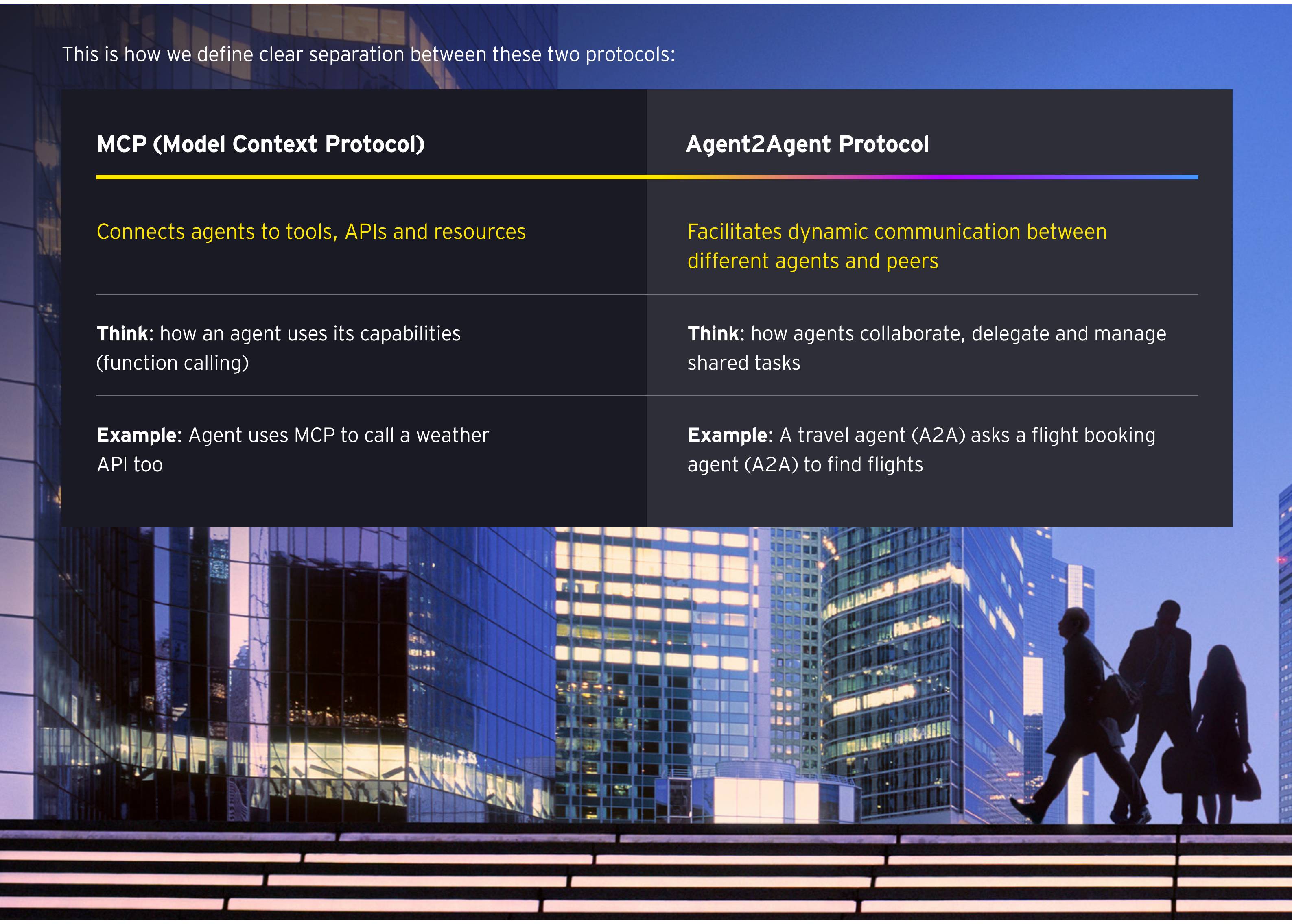
The following table provides a concise comparison of these technologies, clarifying their distinct roles within the AI integration stack.

Aspect	OpenAI Function Calling	Model Context Protocol (MCP)	LangGraph & Agent development frameworks
Primary Role	**Instruction Generation**: Translates natural language into structured, model-specific function calls.	**Standardized execution**: Provides a universal, model-agnostic protocol for discovering and executing tool calls.	**Agent orchestration**: A developer framework for building the logic, memory, and reasoning chains of an AI agent.
Analogy	The thought or decision to act.	The universal nervous system that carries the signal to act.	The brain that coordinates thoughts and actions into complex tasks.
Standardization	Vendor-specific (e.g., OpenAI, Claude, Gemini have different formats).	Universal open standard, aiming for industry-wide adoption.	Framework-specific; provides abstractions but is not a protocol.
Layer in stack	Presentation/translation layer	Session/transport layer	Application/orchestration layer
Best for	Enabling any LLM to express its intent to use a tool.	Robust enterprise integrations, broad interoperability, creating a plug-and-play tool ecosystem.	Building complex, custom AI agents with multi-step reasoning and memory.

MCP vs. A2A vs. other agentic protcols

Comparing MCP with A2A makes a lot more sense as MCP and A2A are essentially protocols and not frameworks. As MCP attempts to standardize tool calling, A2A on the other hand standardizes agent-to-agent communication especially when agents are built using disparate frameworks like AG2, LangGraph or CrewAI. Hence, A2A is not a competing standard but a complementary one. A2A brings four core capabilities to the table:

1. **Discovery**: Agents must advertise their capabilities, so clients know when and how to utilize them for specific tasks.
2. **Negotiation**: Clients and agents need to agree on communication methods that support multimodality.
3. **Task and Statement Management**: Clients and agents need mechanisms to communicate task status, changes and dependencies throughout task execution.
4. **Collaboration**: Clients and agents must support dynamic interaction, enabling agents to request clarifications, information, or sub-actions from client, other agents, or users.



This is how we define clear separation between these two protocols:

MCP (Model Context Protocol)	Agent2Agent Protocol
Connects agents to tools, APIs and resources	Facilitates dynamic communication between different agents and peers
Think: how an agent uses its capabilities (function calling)	**Think**: how agents collaborate, delegate and manage shared tasks
Example: Agent uses MCP to call a weather API too	**Example**: A travel agent (A2A) asks a flight booking agent (A2A) to find flights

Industry adoption as a decisive factor

A protocol's technical merits are only part of its story; its ultimate success is often determined by the breadth and speed of its adoption. On this front, MCP's trajectory has been remarkable. Since its introduction in late 2024, it has seen swift and widespread endorsement from the industry's most influential players, rapidly positioning it as the de facto standard for AI-tool interoperability.

Beyond the giants of AI, a vibrant ecosystem has quickly formed around the protocol. Prominent technology companies active in this market have incorporated MCP into their platforms.[1] By May 2025, a publicly available directory listed over 5,000 active, community-developed MCP servers, and that number grew to over 10,000 by mid-2025.

This rapid convergence around a single, open standard signals a powerful network effect in motion, one that is solidifying MCP's role as the essential connective tissue for the future of AI.

Section V

The price of power: confronting the security frontier

The power to connect autonomous, intelligent systems directly to the world's data and operational tools is a double-edged sword. While the Model Context Protocol unlocks unprecedented capabilities, it simultaneously creates a new and complex security frontier, introducing novel attack surfaces that demand rigorous analysis and mitigation. The protocol, acting as a bridge between powerful AI models and sensitive external systems, requires a shift in security thinking – from protecting isolated applications to securing a dynamic and interconnected ecosystem of agents and tools.

While MCP was designed with security in mind, incorporating a "local-first" philosophy and robust authentication mechanisms like OAuth 2.1, its very function as an interoperability layer introduces distinct challenges.[1]

The successful deployment of MCP in enterprise environments hinges on a clear-eyed understanding of these risks and the implementation of proactive, multi-layered defense strategies.

A comprehensive analysis of the protocol reveals a catalog of critical threats that security professionals must address:

- **Tool poisoning and indirect prompt injection**: This represents one of the most insidious threats. An attacker can manipulate the metadata of a tool – its description or parameters – to include hidden, malicious instructions. Because an AI agent trusts this metadata to understand what a tool does, it can be tricked into performing harmful actions. For instance, a tool description for a simple web search could be "poisoned" with a hidden directive telling the AI to also exfiltrate sensitive files from a connected system. Similarly, an attacker can embed malicious instructions in an external document or webpage (indirect prompt injection), which an AI agent might then process and act upon without the user's knowledge.
- **Malicious server spoofing and tool shadowing**: This is a significant supply chain risk where a malicious MCP server is created to impersonate a legitimate, trusted one. An attacker could create a rogue server named "GitHub-Connector" that perfectly mimics the real one but secretly sends copies of all accessed data to an external location. Without robust validation and a trusted registry, AI agents and their users could be duped into connecting to these "shadow" tools, leading to severe data breaches or system compromise.
- **Privilege abuse and data aggregation risks**: A common vulnerability arises when MCP tools are granted overly broad permissions—for example, requesting full read/write access to a user's email when only read access is needed to perform a specific function. This excessive privilege creates a much larger "blast radius" if the tool is ever compromised. Furthermore, by centralizing access to multiple services, MCP inadvertently creates a powerful potential for data aggregation. An attacker who gains control of an agent could execute sophisticated correlation attacks, combining seemingly innocuous pieces of data from different connected services to piece together a highly sensitive profile of a user or an organization.
- **Sandbox escape**: To operate safely, many MCP servers and the tools they enable, especially those that execute code, are run in a restricted "sandbox" environment. However, any vulnerability in this sandbox implementation could be exploited by an attacker to "escape" into the host system, potentially allowing for arbitrary code execution, unauthorized access to networks, and a complete compromise of the machine running the agent.
- **Confused deputy problem**: Attackers can exploit MCP servers acting as intermediaries to third-party APIs, leading to confused deputy vulnerabilities. By using stolen authorization codes, they can obtain access tokens without user consent. MCP proxy servers using static client IDs MUST obtain user consent for each dynamically registered client before forwarding to third-party authorization servers (which may require additional consent).

The security challenges inherent in MCP highlight a fundamental and permanent tension that will define the future of agentic AI: the trade-off between capability and control. The very features that make an AI agent powerful and valuable – its autonomy, its ability to access diverse tools, and its capacity to act on the world – are the same features that create security risks. To make an agent more capable, one might grant it access to more powerful tools, which inherently increases the potential damage if it is compromised. To make it more secure, one might severely restrict its actions or require constant human approval for every step, which diminishes its autonomy and utility. This is not a problem that can be "solved" once and for all; it is a dynamic equilibrium that must be perpetually managed. The evolution of MCP and the broader field of agentic AI will therefore be a continuous co-evolution, an arms race between expanding capabilities and developing the ever more sophisticated security, governance and oversight frameworks needed to safely contain them.

In response to this new threat landscape, the MCP community and its major adopters are championing a "security by design" philosophy, embedding proactive mitigation strategies directly into the protocol's architecture and best practices. These measures include:

- **Robust technical controls**: A primary line of defense involves implementing strong technical isolation. This includes enforcing strict sandboxing for server processes with network restrictions and syscall filtering, ensuring that a compromised server cannot affect the wider system. Mandatory code signing to establish the provenance of tools and enable revocation of compromised components is also becoming a critical best practice.
- **The principle of least privilege and granular permissions**: To counter privilege abuse, the ecosystem is moving towards enforcing the principle of least privilege, where every component is granted only the absolute minimum permissions required to perform its function. This involves moving away from broad access scopes to fine-grained authorization for specific resources and actions.
- **User-in-the-loop and explicit consent**: A cornerstone of MCP security is keeping the human user in control. This means implementing clear user interfaces that transparently surface any sensitive actions an AI agent wishes to perform and requiring explicit user consent before accessing private data or executing potentially impactful tools.
- **Comprehensive auditing and monitoring**: To detect and respond to threats, robust auditability is essential. This requires comprehensive logging of all AI interactions, tool invocations, and data access events, allowing security teams to trace the actions of every agent and investigate any suspicious activity. Continuous security testing of all exposed MCP interfaces is also a crucial practice.

By embedding these security principles from the ground up, the architects of the MCP ecosystem are working to build a safer and more reliable foundation for an increasingly autonomous AI future.

Section VI

The road ahead: forging the standard for an agentic future

The Model Context Protocol is not a static specification but a rapidly evolving standard, with a clear and ambitious development roadmap designed to guide its transition from a promising protocol into ubiquitous, enterprise-grade infrastructure. The official roadmap outlines a series of strategic priorities focused on hardening the protocol, expanding its capabilities, and fostering a mature, trustworthy ecosystem. These developments are crucial steps toward realizing the long-term vision of a truly interconnected and autonomous AI landscape.

The near-term priorities for the next six to twelve months are centered on several key areas that address the most immediate needs of developers and enterprises:

- **Validation and compliance**: To build developer confidence and promote reliability across the ecosystem, a significant investment is being made in validation. This includes the development of official **Reference Implementations** for both clients and servers to showcase best practices, and the creation of automated **Compliance Test Suites**. These suites will allow developers to verify that their implementations correctly adhere to the MCP specification, guaranteeing consistent behavior and interoperability.

- **Registry and service discovery**: For MCP to reach its full potential, AI agents need a streamlined way to find and connect to relevant tools. To solve this, the roadmap includes the development of an official **MCP Registry**. This will function as a centralized API layer for server discovery and metadata management, acting as a kind of DNS for AI tools. Third-party marketplaces and discovery services will be able to build on top of this registry, allowing agents to dynamically locate, understand, and interact with trusted servers in real time.

- **Enhanced agentic behavior and orchestration**: As MCP becomes more deeply integrated into complex workflows, the protocol is being enhanced to better support multi-agent systems. This includes the development of **Agent Graphs**, a concept that will enable more complex agent topologies through namespacing and graph-aware communication patterns. These improvements aim to facilitate more sophisticated Agent-to-Agent (A2A) communication, where multiple specialized AI agents can collaborate on a single task. The roadmap also focuses on improving human-in-the-loop experiences through more granular permissioning and standardized patterns for eliciting user feedback or approval.

- **Enterprise-grade authentication and security**: Recognizing that security is paramount for enterprise adoption, the roadmap places a strong emphasis on refining its authentication and authorization mechanisms. This includes exploring alternatives to Dynamic Client Registration (DCR) to address operational challenges, developing guidelines for more **fine-grained authorization** on a per-primitive basis, and adding support for enterprise Single Sign-On (SSO) to simplify authorization management.[9]

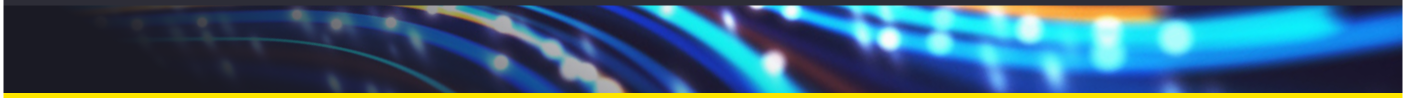
- **Multimodality**: To ensure the protocol remains relevant as AI models evolve, the roadmap includes plans to expand MCP's capabilities beyond text. This involves adding support for additional modalities such as **video and other rich media types**, as well as implementing streaming capabilities beyond text for more interactive, real-time experiences.

This roadmap is more than a simple feature list; it is a strategic blueprint designed to systematically build the pillars of a successful and enduring open standard: trust, accessibility, and governance. The intense focus on validation, compliance suites, and reference implementations is engineered to build **trust** among enterprise adopters who require guarantees of reliability and consistency. The development of a server registry and improved developer tooling is designed to enhance **accessibility**, lowering the barrier to entry and accelerating the network effects of adoption. Finally, the stated commitment to community-led development, shared governance, and collaboration with formal standards bodies is a crucial step toward establishing legitimate, neutral **governance**. This is essential for establishing that MCP is perceived not as a single company's project, but as critical global infrastructure that all major players can safely build upon.

Looking further ahead, the long-term vision for MCP over the next three to five years is to become as fundamental to the AI ecosystem as TCP/IP is to the internet. The protocol is expected to evolve to address the specific requirements of different industries, with tailored extensions for regulated sectors like healthcare and finance. This trajectory points toward a future defined by composability, where highly autonomous AI workforces can dynamically orchestrate complex tasks across a vast, interoperable, and secure ecosystem of tools and services, all communicating through the universal language of the protocol.

The recent update to the Model Context Protocol (MCP) introduces three new, crucial features: **Roots**, **Sampling**, and **Elicitation**. These additions elevate the protocol from a simple tool-calling mechanism to a sophisticated framework for building truly interactive and secure AI agents.

- **Roots** provides a standardized way for clients to define filesystem boundaries, facilitating that servers can only access the directories and files they've been explicitly granted permission for, thereby preventing path traversal attacks and strengthening data security.
- **Sampling** allows a server to request a language model to generate a response on its behalf, enabling nested agentic behaviors where an AI can reason and act dynamically, all without the server needing its own API keys.
- Finally, the **Elicitation** primitive enables servers to dynamically request structured information from the human user through the client's interface, allowing for complex, multi-step workflows that can adapt to missing information or user input in real-time.



Combined example: An AI-powered project migration server could:

1. Ask for access to your old and new project folders (Roots)
2. Request user input about migration preferences (Elicitation)
3. Use AI to analyze code patterns and suggest changes (Sampling)
4. Only work within the approved directories (Roots enforcement)

These new capabilities, combined with the continued emphasis on security and a human-in-the-loop design, highlight MCP's rapid evolution from a promising idea into a hardened, production-ready standard. The specification is moving from merely connecting tools to orchestrating complex, user-centric, and auditable interactions, solidifying its position as the foundational infrastructure for the next generation of intelligent systems.

Conclusion

The dawn of the composable age

The era of powerful but isolated artificial intelligence, defined by the frustrating and costly "Integration Impasse," is drawing to a close. The chaotic landscape of bespoke connectors and fragmented systems is giving way to a new paradigm of standardized, seamless interoperability. The Model Context Protocol stands as the primary catalyst for this transformation, providing the architectural blueprint for a future where intelligent systems can securely and efficiently connect to the world's vast reserves of data and digital tools.

By establishing a universal, open standard, MCP has systematically addressed the core challenges that have long constrained the potential of AI. Its elegant client-server architecture and well-defined primitives – tools, resources, and prompts – have replaced the brittle complexity of the NxM problem with a flexible and scalable plug-and-play model. The protocol's rapid adoption by the industry's most influential players active in this market, is not merely a trend; it is a decisive convergence around a foundational technology, signaling the emergence of a new, unified standard for AI communication.

However, the journey ahead is one of careful navigation. The immense power unlocked by MCP brings with it a new frontier of security challenges that demand a steadfast commitment to "security by design" principles, robust technical controls, and unwavering user-centric governance.

The protocol's evolution will be a continuous balancing act,a co-evolution of expanding agentic capabilities and the sophisticated oversight frameworks required to manage them responsibly. The official roadmap, with its strategic focus on building trust, accessibility, and legitimate governance, provides a clear and promising path forward.

The Model Context Protocol is more than just a technical specification; it is the foundational layer for the next generation of artificial intelligence. It provides the common language necessary for AI to move beyond the confines of passive information retrieval and become an active, dynamic participant in our digital lives. In doing so, MCP is not merely connecting systems – it is unlocking the potential for a truly composable and autonomous age of machine intelligence, the full implications of which are only just beginning to be understood.

References

1. Model Context Protocol (MCP): What problem does it solve? – Collabnix, https://collabnix.com/model-context-protocol-mcp-what-problem-does-it-solve/
2. Solving the AI Integration Puzzle: How Model Context Protocol (MCP) is Transforming Enterprise Architecture | by Rick Hightower | Medium, https://medium.com/@richardhightower/solving-the-ai-integration-puzzle-how-model-context-protocol-mcp-is-transforming-enterprise-8d134f291577
3. The Model Context Protocol (MCP) | Medium – Dirk Steynberg, https://bytemedirk.medium.com/the-model-context-protocol-57d0f6f04a47
4. The Model Context Protocol (MCP) – A Complete Tutorial | by Dr. Nimrita Koul – Medium, https://medium.com/@nimritakoul01/the-model-context-protocol-mcp-a-complete-tutorial-a3abe8a7f4ef
5. Model Context Protocol: A Technical Overview | by Khushiyant - Medium, https://khushiyant.medium.com/model-context-protocol-a-technical-overview-96c117a8736f
6. Exploring MCP Primitives: Tools, Resources, and Prompts | CodeSignal Learn, https://codesignal.com/learn/courses/developing-and-integrating-a-mcpserver-in-python/lessons/exploring-and-exposing-mcp-server-capabilities-tools-resources-and-prompts
7. Beyond Tool Calling: Understanding MCP's Three Core Interaction Types, https://devcenter.upsun.com/posts/mcp-interaction-types-article/
8. The Power of MCP: How Model Context Protocol Is Fueling the Future of AI-Driven Business, https://www.tribalscale.com/insights/what-mcp-model-context-protocol-means
9. Roadmap - Model Context Protocol, https://modelcontextprotocol.io/development/roadmap

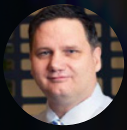
Contacts

Pablo Cebro

Partner, EY Global Head of Technology Platforms, Client Technology,
EY GDS (CS) Argentina S.R.L.
pablo.cebro@gds.ey.com

Deborah Berebichez

Partner, AI Products Leader, Client Technology,
Ernst & Young U.S. LLP
deborah.berebichez@ey.com



Martin Stutchbury

Executive Director,
Product Director for Smart Automation and Assurance Start-up,
Client Technology, Ernst & Young U.S. LLP
martin.stutchbury@ey.com

Matt Perrins

Director, EY Global Lead Architect, Client Technology,
Distinguished Technologist,
Ernst & Young U.S. LLP
matt.perrins@ey.com

Abdul Rehman

Director, EY Technology, Client Technology,
Ernst & Young U.S. LLP
abdul.rehman@ey.com

BMC Agency
GA 132936172

EYG no. 009918-25Gbl
ED None