

One Year of MCP: November 2025 Spec Release

November 25, 2025

Today, MCP turns **one year old**. You can check out the [original announcement blog post](#) if you don't believe us. It's hard to imagine that a little open-source experiment, a **protocol to provide context to models**, became the de-facto standard for this very scenario in less than twelve months.

But not only do we hit the first anniversary milestone today - we're also releasing a brand-new MCP specification version. Before we get to the details of what's new, let's do a bit of a retrospective.

A Year In

With all the changes that we've made in the past year, it feels like a decade flew by. The protocol has grown leaps and bounds since its inception and has been adopted by a *huge* number of developers and organizations. We went from a little open source experiment to becoming *the* standard for connecting data and applications to Large Language Models (LLMs).

But adoption can only grow as long as there are MCP servers to actually use and clients which are capable of communicating with them. Within the same timeframe, we saw the number of active MCP servers go from just a few experimental ones to *thousands*. If you think about a scenario, it's likely there's an MCP server for it.

Here are just a few of many (very many) MCP servers that you can try *today*:

- Notion [built an MCP server](#) to help you manage your notes.
- Stripe has a [pretty extensive MCP server](#) to manage all kinds of payment workflows.
- GitHub [built their own MCP server](#) to help developers automate their engineering processes.
- Hugging Face [created an MCP server](#) to make model management and dataset search a breeze.
- Postman [built their MCP server](#) to help automate API testing workflows.

And there's so much more to discover in the MCP ecosystem! That's why we also launched the [MCP Registry](#) earlier this year. It's the central index for all available MCP servers that now has close to two thousand entries since its announcement in September. That's a **407% growth** from the initial batch of servers we onboarded that same month.

The ecosystem is blooming, adoption is growing, but what's underpinning all of this?

Community & Governance

MCP's growth was never a one-company effort. Students, hobbyists, startup engineers, and enterprise architects all shaped the protocol - submitting Specification Enhancement Proposals (SEPs), shipping SDKs in new languages, and stress-testing some of the early assumptions we had about MCP in production. MCP servers became a staple of many products, official and unofficial (there's even a [Blender MCP server](#)). That kind of organic adoption isn't something you can just come up with, no matter how ambitious your aspirations are with an open source project.

David Soria Parra presenting on MCP in May of 2025

From the start, we believed that it was all about the **MCP community**. Our community rallied around the protocol, organizing events like [MCP Dev Summit](#), [MCP Night](#), [MCP Dev Days](#), and showing up at other marquee events like [AI Engineer World's Fair](#) to share what they learned and built.

Presentation on GitHub MCP server at the MCP Night event

We also nurtured large contributor communities on [Discord](#) and on [GitHub](#), helping us debug issues, build amazing tools like the [MCP Inspector](#), propose changes, and assist each other in shipping great MCP experiences. That kind of daily collaboration got us further than any single individual or company ever could.

Audience at MCP Night in San Francisco

Really, the success of MCP in the past year is entirely thanks to the broad community that grew around the project - from transports, to security, SDKs, documentation, samples, extensions, and developer tooling, it was all significantly evolved by and for the community.

Kent C. Dodds talking about his vision for MCP at MCP Dev Summit

To keep this pace sustainable, we spent some time thinking through and putting together a [governance structure](#). Through it, community leaders and Anthropic maintainers were and continue to work together to figure out what needs fixing and how to get the right changes into the spec. Our maintainer team isn't there to gatekeep; they help surface problems, align on solutions, and turn rough ideas into actual protocol updates.

MCP maintainers collaborating during a writing session in New York City

Our approach to governance, while still evolving, proved itself to be extremely valuable. We've been able to move faster on critical improvements without breaking existing implementations. Potential contributors now also know how to jump in through formal **Working and Interest Groups** ([SEP-1302](#) set the stage for this).

Group photo from an MCP maintainers meetup

Even though this is a significant improvement, we know that we're not done. There's still work ahead for us to make this process even better - improved transparency, decision timelines, broader platform coverage, and so much more to help the ecosystem. We are *incredibly thankful* for everyone who's been part of this journey and helped us navigate so many changes in such a short time span.

 Developing on the bleeding edge of MCP servers presentation

What Others Have To Say

As we called out above, the success of MCP *would not be possible* without the broader community of adopters. We're delighted that the protocol enabled so many scenarios across the industry. Here are some thoughts from a few of our key partners and supporters.

"In just one year, MCP has evolved from an experiment to a widely adopted industry standard, highlighting the impact of open collaboration – something we deeply believe in at GitHub. Developers across our community, customers and own teams are using our GitHub MCP Server, Registry, and enterprise controls like the MCP allowlist to unlock real benefits of agentic development in production workflows. We're excited to keep building with the broader community to push this standard forward."

✦ **Mario Rodriguez**, CPO, [GitHub](#)

"We believe open standards are an important part of an agentic web—helping models work with tools and platforms more seamlessly. OpenAI has been contributing to the MCP ecosystem since early on, and it's now a key part of how we build at OpenAI, integrated across ChatGPT and our developer platform. We're excited to keep working with the community to strengthen the protocol as it evolves."

✦ **Srinivas Narayanan**, CTO of B2B Applications, [OpenAI](#)

"In the year since its launch, MCP has become an incredibly impactful open standard in the industry," said Dhanji R. Prasanna, CTO of Block. "It has quickly moved to unlocking an enormous amount of value from existing systems and made applied AI real like few anticipated. MCP has been key to building AI-powered solutions like Square AI and Moneybot, saving our customers time and delivering powerful insights as well as our internal AI systems. It sits at the heart of open source projects like goose, proving that open standards fuel innovation across the board. We are excited to see the protocol and AI agents evolve to unlock ever more productivity in the enterprise."

✦ **Dhanji Prasanna**, CTO, [Block](#)

"Having an open source protocol that unlocks real interoperability has made agents truly useful. In one year, Foundry went from a small set of tools to thousands because MCP let tools from GitHub, Azure, and M365 show up wherever agents run. It made write once integrate everywhere real and gives agents the ability to work across any system and any cloud with the full power of Microsoft behind them."

✦ **Asha Sharma**, President, CoreAI, [Microsoft](#)

"MCP has become the natural language for AI integration - connecting everything from model discovery to inference APIs to chat applications. The community has created thousands of MCP applications with Gradio and our HF-MCP server. Having an Open Source protocol that unlocks this seamless interoperability has been a game changer in the past year."

✦ **Julien Chaumond**, CTO, [Hugging Face](#)

"The enterprise promise of AI is being realized by MCP's ability to unify data, tools, and workflows across previously siloed systems. As agentic AI is more rapidly adopted, we're excited to see identity and authorization at the core of a security framework. By formally incorporating Cross App Access as an MCP authorization extension, organizations can have the necessary oversight and access control to build a secure and open AI ecosystem."

✦ **Harish Peri**, SVP & GM, AI Security, [Okta](#)

"We're hearing great things from customers who have embraced MCP as their standard for connecting generative AI agents with external systems. Open source is incredibly important to our mission at AWS, which is why we started and continue contributing to MCP— building improvements on authorization, human in the loop interactions, and asynchronous execution. We have also built MCP into offerings like Amazon Bedrock, Kiro, Strands, AgentCore and Amazon Quick Suite. We're excited to continue to collaborate with this community to make agent interoperability seamless for developers."

✦ **Swami Sivasubramanian**, VP, Agentic AI, [AWS](#)

"In just one year, the Model Context Protocol has proven to be a critical standard that connects models to data and applications, solving the fragmentation that held agents back. We're proud to support MCP across Gemini, from our models to our agentic software development tools like Gemini CLI, as well as provide open source MCP servers such as for Google Maps and Google Cloud databases. These are the very tools our own teams use, and we're thrilled to celebrate MCP's first birthday by continuing to build this foundation together."

✦ **Anna Berenberg**, Engineering Fellow, [Google Cloud](#)

"MCP has changed everything for us at Obot AI. A standard, open protocol for connecting AI with apps, data, and systems is the biggest shift since LLMs. We're all-in on secure MCP management because we believe it's going to be foundational infrastructure for every organization"

✦ **Shannon Williams**, President and Co-Founder, [Obot AI](#); Organizer, [MCP Dev Summit](#)

Of course, we would be remiss not to mention the massive effort it takes to coordinate the MCP community engagement. We asked some of our most prolific community managers about what MCP meant to them. Here are their stories.

"As a community moderator and maintainer, I keep coming back to something [Donella Meadows](#) wrote: "Systems can't be controlled, but they can be designed and redesigned... We can listen to what the system tells us, and discover how its properties and our values can work together to bring forth

something much better than could ever be produced by our will alone.”

What made this year's growth possible was embracing messiness as a feature, and doing our best to solve real problems emerging from that messiness while leaving room for systems and models to advance and adapt. The looseness created velocity, which I watched unfold in a flood of discussions, PRs, and issues.

As someone who's merged hundreds of pull requests across MCP repos, I still feel like I'm barely keeping up with this velocity. I mean that in the most positive way. Better patterns and practices are emerging from the sheer volume of contributions and breadth of experience and expertise represented in the contributor community. As a random person from the Internet, I appreciate that pretty much anyone can bring something to the table. This includes standout maintainers like [Cliff Hall](#), who raises the bar for reviewing, testing, and giving feedback, and [Jonathan Hefner](#), who's done the same for documentation.

As Darren Shepard [recently put it](#):

‘People think the value of MCP is the protocol. The value is getting people to agree and do something.’

MCP gives people a reason to coordinate and talk about the same thing. Helping to enable that coordination and discussion has been a lot of fun, and it keeps me coming back.”

♦ **Ola Hungerford**, Principal Engineer, [Nordstrom](#): MCP Maintainer and Community Lead

“Watching the MCP community start small and grow up across the past year has been a joy to watch. No matter whether someone has been an independent contributor, member of a small startup, or a leader at a big enterprise: everyone has had and continues to have a voice and a role to play.

I think this is largely due to how pragmatic and use-case oriented the MCP community has been from the get-go. There is a focus on not overcomplicating the specification, and not designing ahead of need. When that's the ethos driving decision-making, everyone's voice matters. The hobbyist that has something working in production might have a practical opinion to contribute, that the big tech engineer can pick up and confidently deploy to a large userbase. And vice-versa: big tech can foresee problems around critical issues like security and governance that the hobbyist might have missed designing for.

*That ethos has translated to community governance, too. There's no layers of bureaucracy: just a lightweight and still-distributed structure for making decisions that keep us all marching to the beat of the same drum. We are now **58 maintainers** supporting the **9 core/lead maintainers** in the MCP steering group, with **2,900+ contributors** in the MCP contributor community on Discord, and **100+ new contributors** joining every week. We're successfully maintaining long-running projects like the Registry, the Inspector, and a host of SDKs with that distributed group of leaders and contributors - and managed to collaborate through 17 (!) SEPs in about a quarter's worth of time.*

That doesn't even begin to touch on the thousands of MCP implementors or millions of MCP end-users. It's inspiring to see the often-competitive AI community rally around a foundational piece of infrastructure; I'm thankful for that willingness to collaborate and look forward to seeing where our second year takes us.”

♦ **Tadas Antanavicius**, Co-Creator, [PulseMCP](#): MCP Maintainer and Community Lead

We are immensely grateful for our partners and community for helping us bring the protocol to where it is today. Let's now jump into the latest big release - the 2025-11-25 version of the MCP specification.

The November 2025 Release

The latest release of the MCP specification ships with a number of highly-anticipated features that came directly from our community deploying and using MCP for production scenarios. People told us what wasn't working, what was missing, and what papercuts prevented them from being able to use MCP. We listened and worked together with community experts to deliver a number of enhancements that make MCP even more scalable and reliable.

Support for Task-based Workflows

SEP: [1686](#)

Tasks provide a new abstraction in MCP for tracking the work being performed by an MCP server. Any request can be augmented with a task that allows the client to query its status and retrieve its results up to a server-defined duration after the task is created.

Tasks support a variety of states including `working`, `input_required`, `completed`, `failed`, and `cancelled`, allowing clients to effectively manage multi-step operations.

Some noteworthy capabilities that this feature enables:

- **Active polling**: Clients can check the status of ongoing work at any time.
- **Result retrieval**: Results of completed tasks are accessible after the request has completed.
- **Flexible lifecycle management**: Support for `working`, `input_required`, `completed`, `failed`, and `cancelled` states.
- **Task isolation**: Proper security boundaries with session-based access control.

From the multitude of MCP servers that we've seen out there, this is particularly helpful for scenarios such as the ones below.

- Healthcare & life sciences data analysis that processes hundreds of thousands of data points
- Enterprise automation platforms with complex multi-step workflows
- Code migration tools that run for minutes or hours

- Test execution platforms that need to stream logs from long-running suites
- Deep research tools that spawn multiple agents internally
- Multi-agent systems where agents can work concurrently

Tasks are launching as an **experimental capability**, meaning that it's part of the core protocol but it's not yet finalized. Task-based workflows are a tough problem to solve at scale, so we want to give some time to the specification to be battle-tested in real-world scenarios. We'll work closely with the community, SDK developers, as well as client and server implementers to get this right.

Simplified Authorization Flows

One of the top painpoints from the community when it comes to authorization has been [Dynamic Client Registration](#), or DCR. This capability is needed because in the MCP world there is an unbounded number of clients and servers, so doing standard client pre-registration is not always feasible. You wouldn't expect every MCP client in the world to also have a client registration with every Authorization Server (AS) out there, so DCR was used as a solution to this problem. You can learn more about the current approach in our [authorization guide](#).

To use DCR, however, an MCP server developer would need to rely on an AS that allows clients to register themselves via a public API. If the AS doesn't support this capability, developers would now need to build an OAuth proxy that would be manually registered with the AS, and support Dynamic Client Registration itself, mapping its own issued tokens to tokens issued from the downstream AS. This is a complex, time-consuming, and error-prone task, and doesn't actually solve the fundamental problems with Dynamic Client Registration.

The alternative would be for every customer or end user to provide *their own* client registration, but that's just trading one complex task for another. In that model, when a user connects to an MCP server, they need to go through their IT team to create a registration, assign it the right permissions, and then configure the MCP client to use it.

[SEP-991](#) introduced a much more elegant solution to the problem - URL-based client registration using [OAuth Client ID Metadata Documents](#) (you might've already seen our [blog post on this change from earlier this year](#)). Clients can now provide their own client ID that is a URL pointing to a JSON document the client manages that describes properties of the client.

You can learn more in the [Client ID Metadata Documents Flow](#) section of the [MCP authorization specification](#).

Security and Enterprise Features

As the protocol matures, we also can't ignore the myriad of security and authentication/authorization needs. MCP is not just a hobby protocol - we've seen it adopted in some of the most mission-critical workloads. This translates into a direct need to ensure that all data is protected and access is properly managed.

Working with security and authentication experts from across the community, we've developed a number of enhancements shipping with this release:

- [SEP-1024](#): Client security requirements for local server installation
- [SEP-835](#): Default scopes definition in authorization specification

We also hear loud and clear from the industry that discovery and management of internal registries is an important component to the MCP story. With the help of the MCP Registry team, we've also established a [vision for the ecosystem](#) that will help enterprises adopt *their own* MCP registries, with self-managed governance controls and security coverage.

To learn more about other upcoming auth and security improvements you can follow the [auth](#) and [security](#) tags in the specification repository.

Extensions

As MCP continues to evolve, we *constantly* hear from developers who want to extend the protocol with specialized capabilities, whether for UI interactions, custom authentication flows, or other environment-specific logic. While these additions could be valuable, incorporating them directly into the core specification isn't always practical from the get-go, especially when a feature hasn't yet achieved broad adoption or proven its universal applicability.

To address this, we're introducing **extensions** in the protocol. Extensions are components and conventions that operate outside the core specification, providing a flexible way to build scenario-specific additions that follow MCP conventions without requiring full protocol integration. This approach allows for experimentation and specialized use cases while keeping the core protocol focused and stable. With extensions, we can move faster and enable developers to *test out* protocol capabilities before they become part of the specification.

Extensions are:

- **Optional**. Server and client implementors can choose to adopt these extensions.
- **Additive**. Extensions do not modify or break core protocol functionality; they add new capabilities while preserving core protocol behavior.
- **Composable**. Extensions are modular and designed to work together without conflicts, allowing implementations to adopt multiple extensions simultaneously.
- **Versioned independently**. Extensions follow the core MCP versioning cycle but may adopt independent versioning as needed.

You might've already seen our announcement of the [MCP Apps Extension](#) proposal. In this specification release, we're introducing a couple of other extensions that should help developers further.

Authorization Extensions

To make MCP better suited for environments that require specific levels of control over the authorization process, we've officially introduced the concept of [authorization extensions](#) (building on the broader [MCP Extensions](#)). As with all other extensions, authorization extensions build on the core protocol and define additional authorization mechanisms that can be implemented by both server and client developers.

The first two authorization extensions came on the heels of community feedback regarding some of the most-used authorization flows:

- [SEP-1046](#): OAuth client credentials support for machine-to-machine authorization
- [SEP-990](#): Enterprise IdP policy controls for MCP OAuth flows (Cross App Access). This enables users within an enterprise to sign in to the MCP client once, and immediately get access to every authorized MCP server without additional authorization prompts.

As we engage closer with the community, we expect the number of authorization extensions to grow as well - after all, there are more than just a few ways for a system to acquire and manage credentials.

URL Mode Elicitation: Secure Out-of-Band Interactions

SEP: [1036](#)

Asking users for their API keys, tokens, or any other credentials directly through the MCP client might seem like quite a scary proposition. This is especially critical when you need to connect an MCP server to an array of *other* APIs, where the traditional client-to-server authorization flow doesn't quite work. Until now, there wasn't a good alternative - you either had to trust the client to handle the user's credentials directly, or implement a bunch of custom authorization logic to be used from the start.

URL mode elicitation lets you send users to a proper OAuth flow (or any credential acquisition flow, for that matter) in their browser, where they can authenticate securely without your client ever seeing the entered credentials. The credentials are then directly managed by the server and the client only needs to worry about its own authorization flow to the server.

We are excited about including this feature in addition to capabilities that we already have, like elicitations, because it allows the protocol to be used for a few scenarios that were quite hard to get right, such as:

- **Secure credential collection**: API keys and passwords never transit through the MCP client
- **External OAuth flows**: MCP servers have a path to obtain third-party authorization without token passthrough
- **Payment processing**: PCI-compliant financial transactions with secure browser contexts can now be done outside the client

All the server does is send a URL that the client will provide an affordance for. When the user completes the flow in their browser, the server will get the necessary tokens *directly*, avoiding sharing credentials with the client or other manual steps. Simple!

Sampling with Tools: Agentic Servers

SEP: [1577](#)

This functionality allows MCP servers to run their own agentic loops using the client's tokens (still under the user's control, of course), and reduces the complexity of client implementations, context support becoming explicitly optional. This came from the fact that [sampling](#) doesn't support tool calling, although it's a cornerstone of modern agentic behaviour. With the new spec release, this is no longer a gap!

Now that sampling with tools is available, this also means that all of the scenarios below are possible!

- **Tool calling in sampling requests**: Servers can now include tool definitions and specify tool choice behavior
- **Server-side agent loops**: Servers can implement sophisticated multi-step reasoning
- **Parallel tool calls**: Support for concurrent tool execution
- **Better context control**: The ambiguous `includeContext` parameter is being soft-deprecated in favor of explicit capability declarations

As an example, a research server can spawn multiple agents internally, coordinate their work, and deliver a coherent result while using nothing other than standard MCP primitives without custom scaffolding or complex orchestration code.

Developer Experience Improvements

One of the core tenets of MCP is *simplicity* - we want to make the developer and integration experience as intuitive and easy as possible. To help achieve this, the latest spec release also adds a few minor changes that help make the protocol easier to use for developers.

- [SEP-986](#): Standardized format for tool names
- [SEP-1319](#): Decoupled request payload from RPC methods definition
- [SEP-1699](#): SSE polling via server-side disconnect for better connection management
- [SEP-1309](#): Improved specification version management for SDKs

Looking Forward

This release is backward compatible. Your existing implementations keep working. The new features are there when you need them.

Looking ahead, we're excited about what's coming next for MCP. The protocol is entering a new phase, one where it's not just about connecting LLMs to data, but about enabling entirely new categories of AI-powered applications.

We're seeing early signals of this transformation already. Developers are building multi-agent systems that coordinate across dozens of MCP servers. Enterprise teams are deploying MCP at scale with sophisticated security and governance controls. Startups are launching products where MCP is the

core architectural pattern. MCP servers are even being transformed into executable code, to create sandboxed agent workflows.

The [roadmap ahead](#) includes deeper work on reliability and observability, making it easier to debug and monitor complex MCP deployments. We're exploring better patterns for server composition, allowing you to build sophisticated capabilities by combining simpler building blocks. And we're continuing to refine the security model to meet the needs of the most demanding enterprise environments.

What excites us most isn't what *we're* planning to build but what *our community* is going to build. Every week we see MCP servers designed, developed, and deployed in novel ways. Every conversation in Discord reveals new use cases and patterns. The protocol has become a canvas for AI innovation, and we can't fill it alone.

The next year of MCP will be shaped by more production deployments, more real-world feedback, amplified by the creativity of thousands of developers worldwide. We're here to support that growth, to ensure the protocol evolves thoughtfully, and to keep MCP stable, secure, and simple as it scales.

Get Started

To get started with all the new goodness in the latest MCP specification release, check out the following resources:

- **Read the changelog:** All major changes are captured in our [Key Changes document](#)
- **Get to know our docs:** The [MCP documentation](#) is the source of truth for the all the inner workings of the protocol
- **Join the discussion:** If you would like to contribute or engage with other MCP maintainers, start with our [GitHub repo](#) and [Discord](#)

Model Context Protocol Hub MCP A2A Protocol AP2 Lab ACP Protocol Agent2Agent AI to Sora ChatGPT UCP Clawd Bot moltBot Lab

Powered by ModelContextProtocol

© 2024 ModelContextProtocol.Info.