



GUIDE 01 OF 03 · UNDERSTAND

Understanding MCP

The Model Context Protocol in Plain Language

What the Model Context Protocol is, where it came from, how it works, and what it means for organizations like yours — no technical background required.

A deep research guide from MCP.NGO · Reading time: about 10 minutes · Last reviewed: September 2026

IN THIS SERIES

Guide 01 — Understanding MCP (this guide)

Guide 02 — Putting MCP to Work

Guide 03 — Connecting with Care

The short version

The Model Context Protocol (MCP) is an open, free standard that lets AI assistants connect to the software your organization already uses — your donor database, shared drives, spreadsheets, project boards, and communication tools. Instead of copying and pasting information into a chat window, your team can ask an AI assistant questions and delegate tasks that draw on your real, current data, with your permission and within limits you set.

MCP was introduced by the AI company Anthropic in November 2024, was adopted within months by essentially every major AI platform, and moved to neutral, vendor-independent governance under the Linux Foundation in late 2025. It is not a product you buy. It is a shared language that products speak — which is exactly why it matters for organizations that can't afford custom software.

The problem MCP solves: your AI can't see your work

By now, AI assistants are everywhere in the nonprofit sector. A 2026 benchmark study of 346 organizations found that 92 percent of nonprofits use AI in some capacity — mostly for drafting, summarizing, and research. Yet only 7 percent said AI had meaningfully expanded what their organization can accomplish.

One big reason for that gap is disconnection. A general-purpose AI assistant is like a brilliant new volunteer on their first day: capable, articulate, and completely unaware of your donors, your programs, your files, or your history. So staff act as couriers — exporting a report here, pasting an email thread there, retyping numbers from the CRM into a chat box. The assistant helps with each fragment, but someone has to carry every fragment to it. The time saved on writing gets spent on copying.

Before MCP, fixing this required custom integrations: one piece of connective software for each pairing of AI tool and data source. Ten tools and three AI platforms meant up to thirty separate integrations to build and maintain — realistic for a bank, not for a food pantry.

What MCP actually is

MCP replaces all of those one-off connections with a single, standardized way for AI applications and other software to talk to each other. The comparison you'll hear most often is USB-C: rather than a different cable for every device, one port works across all of them. In the same way, a tool that offers an MCP connection can work with any AI assistant that supports MCP — and every major one now does.

A few facts worth knowing:

- **It's open and free.** MCP is an open standard with open-source software behind it. No license fees, no single company controlling access.
- **It's neutrally governed.** Anthropic created MCP, but in December 2025 the project was donated to the Linux Foundation's Agentic AI Foundation, which now provides vendor-neutral stewardship — the same

kind of arrangement that underpins much of the internet's shared infrastructure. Rival AI companies contribute to it side by side.

- **It's widely adopted.** OpenAI adopted MCP in March 2025, followed by Google DeepMind, Microsoft, Amazon Web Services, and others. Thousands of MCP servers (connectors) now exist for popular software, and an official registry helps people find them.
- **It's actively maintained.** The specification is updated on a regular cadence — the current release came out in November 2025 — with a public roadmap and formal community governance.

For a nonprofit, the practical meaning of all this is simple: MCP is not a bet on one vendor's ecosystem. Learning it, and building workflows around it, is more like learning email than like learning one company's proprietary app.

How it works, minus the jargon

You will encounter three technical terms. Here they are, translated:

The host is the AI application your team already uses — a chat assistant on the web, desktop, or phone. It's where you type your request.

The server (often called a **connector** in consumer products) is a small piece of software that represents one of your tools — your CRM, your cloud drive, your calendar. It knows how to look things up and take actions in that tool, and it advertises what it can do in a standard format the AI understands.

The client is the piece inside the host that manages the connection between the two. You'll almost never think about it.

Here's what that looks like in practice. Suppose your development director connects the organization's CRM to an AI assistant through an MCP connector, granting it read-only access. She then types: "List donors who gave last year but haven't given this year, and draft a personalized check-in note for each."

Behind the scenes, the assistant sees that the CRM connector offers a "search donors" capability, asks it to run that search, receives the current, real results, and then does what AI assistants are good at — drafting warm, specific notes using each donor's actual giving history. No export, no pasting, no stale spreadsheet. And because the connection was set to read-only, the assistant can look but cannot change a single record.

Connectors can offer the AI three kinds of things: **tools** (actions it can take, like "search records" or "create a task"), **resources** (information it can read, like files or database entries), and **prompts** (pre-written instruction templates for common jobs). You don't need to remember the distinctions — just know that every capability a connector exposes is something you can see, permit, or refuse.

One more distinction worth a sentence: some connectors run **remotely** (hosted by the software vendor — you just click "Connect" and sign in, much like linking your calendar to a scheduling app), while others run **locally** on a staff computer (typically set up by more technical users). Most nonprofits will start, and often stay, with remote connectors from official sources.

What this means for organizations like yours

The honest case for nonprofit attention to MCP is an equity argument. Integrated, "AI that knows our data" workflows used to require enterprise budgets and engineering teams. MCP collapses that requirement to a subscription your organization may already hold and an afternoon of setup. Concretely, that can look like:

- A grants manager asking the assistant to pull program outcomes from the shared drive and draft the narrative section of a proposal in the funder's format.
- An operations lead having meeting notes turned into assigned tasks on the project board, automatically.
- An executive director getting a board-packet first draft assembled from the finance spreadsheet, the program dashboard, and last quarter's minutes.
- A volunteer coordinator asking, in plain English, which weekend shifts are still unfilled — and getting an answer from the live schedule.

The same study cited earlier found that most nonprofits are stuck at an "efficiency plateau": faster drafts, same outcomes. Connection is one of the main bridges past that plateau — but only when it's paired with intention. Which brings us to what MCP is not.

What MCP is not

It is not an AI model. MCP doesn't think or write. It's the wiring between an AI assistant and your tools. The quality of answers still depends on the assistant you use and the quality of your data.

It is not a product or an app. You won't "download MCP." You'll use features — usually called connectors or integrations — inside AI products that speak MCP under the hood.

It is not automatic. Someone at your organization must decide what to connect, with what permissions, for which staff. Those decisions matter more than the technology.

It is not a data-cleanup fairy. If your CRM is full of duplicates and your file naming is chaos, a connected assistant will faithfully retrieve the chaos. Connection rewards organizations with reasonable data hygiene.

It is not risk-free. Anything that gives software access to donor and community data deserves care. The risks are manageable and well understood — least-privilege access, read-only defaults, and human sign-off on actions handle most of them — but they are real. We wrote a whole guide about it (Guide 03: *Connecting with Care*), and we mean it when we say to read that one before you connect anything sensitive.

It is not a replacement for judgment. A connected assistant makes staff faster; it doesn't make their review optional. Keep people in the loop, especially for anything a donor or community member will see.

A five-minute glossary

- **MCP (Model Context Protocol):** The open standard described in this guide; a common language that lets AI applications and other software exchange information and capabilities.
- **Connector / MCP server:** The adapter for one specific tool (e.g., your CRM). "Connector" is the friendly product term; "MCP server" is the technical one. Same idea.
- **Host / AI assistant:** The AI application you chat with, which uses connectors on your behalf.
- **Tool:** An action a connector lets the assistant take, such as searching records or creating a calendar event.
- **Resource:** Information a connector lets the assistant read, such as a document or database entry.
- **Registry:** A public directory of available MCP servers, including an official one maintained by the project, that helps users find and vet connectors.
- **OAuth:** The standard "sign in and grant permission" flow you already know from linking apps to your Google or Microsoft account; the common, safer way connectors get access without you handing over passwords.
- **Read-only access:** A permission level that lets the assistant look things up but never change, add, or delete anything. The right default for nonprofits starting out.
- **Agent / agentic AI:** An AI assistant that can carry out multi-step tasks using tools, not just answer questions. MCP is the plumbing that makes much of this possible.
- **Prompt injection:** A trick where malicious instructions hidden in content (an email, a web page, a document) try to manipulate an AI assistant. Covered thoroughly in Guide 03.

Where to go next

If you're evaluating whether this is worth your team's time, continue to Guide 02, *Putting MCP to Work*, for concrete nonprofit use cases and a 30-day pilot plan that requires no developer. If you're the person responsible for data, privacy, or IT, go straight to Guide 03, *Connecting with Care*, for the safety and governance side.

Sources & further reading

- Model Context Protocol — official documentation and specification: <https://modelcontextprotocol.io>
- Model Context Protocol blog — "The 2026 MCP Roadmap": <https://blog.modelcontextprotocol.io/posts/2026-mcp-roadmap/>
- Wikipedia — "Model Context Protocol" (history and adoption timeline): https://en.wikipedia.org/wiki/Model_Context_Protocol
- WorkOS — "Everything your team needs to know about MCP in 2026": <https://workos.com/blog/everything-your-team-needs-to-know-about-mcp-in-2026>
- Virtuous & Fundraising.AI — "The 2026 Nonprofit AI Adoption Report" (source of the 92% / 7% figures): <https://virtuous.org/blog/the-2026-nonprofit-ai-adoption-report-blog/>

This guide is for general information only and is not technical, legal, or security advice. Product capabilities change quickly; verify details against official documentation before making decisions.