JUNE 24, 2026

What Microsoft Graph actually is, and why it decides whether Copilot works

Copilot doesn't read your files. It asks Microsoft Graph, and Graph returns what you're allowed to see. Get Graph right and Copilot works.

By **Dave Taylor**

Microsoft Graph is the single data and permissions layer that sits across all of Microsoft 365 — SharePoint, OneDrive, Outlook, Teams, your calendar, and the Entra ID directory that decides who can see what. Copilot never reads your files directly. It asks Graph, and Graph returns only what that specific user is allowed to open. So Copilot is exactly as good as what Graph can see and what Graph lets through — which is why early users completed tasks 29% faster, per Microsoft's 2023 Work Trend Index, only once Graph was pointed at the right material. In Part 1 we said garbage Graph means garbage Copilot. This is the level below that: what Graph actually is, and how to extend it.

What is Microsoft Graph, in plain language?

Think of Graph as both the librarian and the bouncer for your tenant. As librarian, it knows what content exists across Microsoft 365 and what each document says — it maintains a semantic index, a searchable map of every file, email, and message a user could reach. As bouncer, it enforces who's allowed in: the permissions Copilot honours are Entra ID (the Microsoft identity service that holds your users and access rules) permissions, read straight through Graph. There is no second set of rules. Microsoft describes Graph as the single gateway to Microsoft 365 data, and that design is the point: when Copilot answers, one component does both jobs — find the relevant material, then filter it to what this person is allowed to see.

That dual role is why Graph decides whether Copilot is useful or dangerous. A clean permission estate means Copilot answers from the right documents and refuses the wrong ones. A messy one means it surfaces a salary sheet to someone who was never meant to open it — not a Copilot bug, but a Graph permission someone set too wide years ago, now read back faithfully by an assistant that has no idea it was a mistake. This is the same exposure Part 1 covered: the assistant didn't leak anything Graph wasn't already willing to hand over to that user. So the first engineering question of any rollout isn't about the model at all. It's whether Graph's view of your tenant is clean enough to trust as the thing standing between a curious employee and every file the company owns, because once Copilot is live, that view is exactly what every staff member is querying all day.

How does Copilot actually use Graph to answer?

The mechanism is retrieval-augmented generation, usually shortened to RAG (the model looks things up before it answers rather than relying on memory). When you ask Copilot something, it doesn't reach into the model's training. It queries Graph's semantic index over your tenant, pulls back the handful of documents most relevant to your question and your permissions, and only then does the model write an answer grounded in that retrieved material. We walk through this pantry-and-chef picture in our primer on how AI works and breaks — Graph is the pantry, and the chef can only cook with what's in it.

The consequence is unforgiving and worth saying plainly: garbage in the index produces a garbage answer. If Graph indexes three versions of last year's pricing and none of this year's, Copilot will confidently quote the wrong number, because it's grounding faithfully on bad source material. The model isn't wrong; the pantry is. This is the same retrieval-quality problem behind structural hallucinations — an assistant grounded in stale or duplicated documents doesn't hallucinate from nowhere, it repeats what it was given, with the same confident tone it uses for the right answer. So the real readiness work in a Copilot rollout

isn't prompt-writing or model choice. It's deciding what Graph indexes, retiring the duplicates and dead drafts that have piled up in SharePoint for a decade, and making sure the current version of anything that matters is the one Graph finds first. Skip that step and you haven't bought an assistant, you've bought a very fast way to quote your own outdated files back at yourself.

What about data that lives outside Microsoft 365?

Here's where most rollouts hit the wall: the firm's most valuable data usually isn't in Microsoft 365 at all. In professional services it's the bespoke practice-management ledger, the tax database, the line-of-business app that runs the actual work. Copilot can't ground on what Graph can't see, so out of the box it's blind to exactly the systems that hold the answers people most want to ask about. A solicitor's Copilot that can read every email but not the matter-management system is answering with one eye shut.

There are three honest options, and which one is right depends almost entirely on what that outside system's API will allow (the API, or Application Programming Interface, is the set of hooks that let other software read a system's data). The first is to build a **Graph connector** — Microsoft's mechanism for pulling external content into the Graph index so Copilot can ground on it natively, the same way it grounds on a SharePoint file. That's the right call when you want the data fully searchable inside Copilot and the source has a usable API to pull from. It's the most capable option and the most engineering to stand up, so it earns its place when the data is central and queried often — a tax database a team hits twenty times a day is worth a connector. The connector copies the content into the index, honours the source's permissions where it can, and refreshes on a schedule.

When is a Graph connector the wrong answer?

Sometimes the cleaner answer is cruder. A **scheduled export to SharePoint** pushes the key reference data — the rate card, the client roster, the standard procedures — into a curated SharePoint site on a timer, where Graph already indexes it for free. It's less elegant than a live connector and the data is only as fresh as the last export, but for reference material that changes weekly rather than hourly, it's often good enough and a fraction of the cost to build. We reach for this far more often than clients expect, because most of the value is in a small set of stable documents, not the whole live database.

The third option is the one nobody offers: **keep the system out of Graph deliberately**. If the data is too sensitive to widen Copilot's reach into, too volatile to keep in sync, or the source API is too poor to pull from reliably, the right move is to serve those workflows with separate, purpose-built tooling instead of forcing them through Graph. That's not a

weakness in the rollout — it's a scoping decision, and it sets up where this series goes next. Not every workflow belongs inside the Copilot grounding layer, and pretending otherwise is how teams end up with a connector that breaks every Tuesday and a database that's perpetually one sync behind reality. We'd rather tell a client "this system stays out of Graph, and here's the separate tool we'll build for it" than wire in a fragile connector that quietly serves wrong answers — because a confident wrong answer from Copilot does more damage than an honest "I can't see that."

Which of your systems can Graph even reach?

The deciding factor in all three options is the same: the API of the source system. A clean, documented API makes a Graph connector straightforward. A brittle or undocumented one pushes you toward scheduled exports or out of Graph entirely. So a genuine week-one question for any Copilot rollout is the unglamorous one: which of our systems have APIs Copilot can reach, and which don't? That single audit decides more about whether Copilot feels useful than any setting in the admin centre.

Run it before you buy licences, not after. The pattern we see across our work with Irish SMBs is that the answer divides neatly — the modern cloud apps have APIs and join the Graph index without drama, while the older line-of-business systems, the ones holding the most valuable data, are exactly the ones with the worst APIs. That's not a coincidence; the systems that have run the business for fifteen years were never built to be read by anything else. Knowing which side of that line each system falls on tells you, before a single user logs in, where Copilot will feel magic and where it'll feel deaf. It turns "roll out Copilot" from a licence purchase into a short, honest list of what's reachable and what needs a bridge built first.

What it adds up to

Copilot is not the product you're really configuring — Graph is. The model is a commodity; what separates a Copilot that earns its licence from one that quietly gets abandoned is whether Graph can see the right material and whether its permissions are clean enough to trust. That's why the same early users completed tasks 29% faster and felt 70% more productive in Microsoft's 2023 study — the payoff was downstream of the grounding layer. Clean what Graph indexes, fix the permissions, and decide system by system whether each gets a connector, a scheduled export, or stays out by design. Once Graph can see the right data, you're ready for Part 3, where we build the automations and agents on top of a Copilot that finally knows where to look. If you want a clear read on which systems Copilot can reach and which need a bridge first, book a 30-minute working session and we'll map it together.