

AI agent governance on SharePoint data (PDF)

What Copilot's SharePoint controls cover, why they miss an agent your developers built with its own Graph app, and how a database login, views, row-level security and the database audit log bound it.

The security review usually starts with one line in a ticket. "We have locked Copilot down with Restricted Content Discovery and Purview. What stops the procurement team's new agent reading the HR site?" The agent in question is a Python service, or an n8n flow, or a LangChain app on a VM, with an Entra app registration and a client secret. It answers questions about the Supplier Contracts list, and nobody in the security team can say what else it could read, or show what it did read last Tuesday.

On this page a custom agent means exactly that: software your own people built, holding its own credential, calling Microsoft Graph or a database itself. Copilot, Copilot Chat and agents built in Copilot Studio are Microsoft's own surfaces, and Microsoft's controls are built around them. This resource sets out which of those controls reach a custom agent and which do not, then how to bound one with a replica it reads from. Forward the PDF to whoever signs off the review.

What Copilot's SharePoint controls cover

Every control below was checked against Microsoft Learn on 23 October 2026.

Control | What it does | What it acts on

Restricted SharePoint Search | An allow list of up to 100 sites for organisation-wide search and Copilot. Microsoft calls it temporary, "not a security boundary", and blocks new enablement from 31 July 2026 | Search and Copilot chat and agentic experiences

Restricted Content Discovery | Keeps a site's content out of organisation-wide search and Copilot responses. "Doesn't change existing permissions" | Discovery, per site

Agent access insights | A report of how registered and activated agents access SharePoint and OneDrive, built from the unified audit log, over 1, 7, 14 or 28 days | Agents registered in Microsoft 365

Purview audit for Copilot and AI apps | Records each user interaction with Copilot or an AI application, with the files, sites and list items it accessed | Interactions with Copilot and AI applications Purview knows about

Purview DLP for Copilot | Stops Microsoft 365 Copilot, Copilot Chat and Cowork processing sensitive prompts or labelled files | Those Copilot experiences

Read the right-hand column again. Two of the five act on search and discovery. Two act on Copilot by name. The agent report works from agents Microsoft 365 has registered. All five are worth switching on for Copilot, and none of them changes what an app registration with Sites.Read.All is allowed to read.

Do Copilot's controls apply to an AI agent that is not Copilot?

Mostly they do not, and the reason is how the agent reaches the data. Restricted SharePoint Search and Restricted Content Discovery change which sites appear when someone searches or asks Copilot. A custom agent that calls `/sites/{site-id}/lists/{list-id}/items` is not searching. It asks Graph for the items of a list it already knows, and Graph answers according to the app's permission. Learn says of both controls that permissions stay as they were.

Purview DLP for Copilot names its locations, and a Python service is not one of them. Purview audit does extend past Microsoft's own apps. Agents built in Copilot Studio and Microsoft Foundry are logged under Audit (Standard), and interactions with non-Microsoft AI applications are logged on pay-as-you-go billing, retained for 180 days. Those records describe a person's interaction with an AI application that Purview recognises. A scheduled job that reads a list with an app token and passes rows to a model API is not that, so do not assume it shows up.

Agent access insights lists agents "registered and activated" in Microsoft 365, with custom agents among them. Your LangChain service is an Entra app registration. Check whether the report shows it before you rely on the report for it.

AI agent SharePoint permissions are whatever the app was granted

A custom agent with its own app registration sees everything its Graph permission covers, with no per-user trimming. With the application permission `Sites.Read.All` that is every site in the tenant. The agent's code may only ever ask about Supplier Contracts, but the credential can read the board papers, and a prompt injection in a list field or a bug in a tool can send it there.

The narrow version exists. With Selected permissions an app consented for `Sites.Selected` "would initially have no access", and an admin grants it a role on each named site. Two limits remain. A read grant covers "the metadata and contents of the resource", so the agent can open every file on that site. And every agent your organisation builds needs its own registration, grant, secret rotation and review.

There is a second gap. When the agent runs with application permissions, SharePoint sharing settings no longer decide what reaches a given user. The agent reads as itself, and whoever types into its chat box gets answers drawn from everything the app can see.

How do you audit what an AI agent read in SharePoint?

On the Microsoft side you have two sources. Entra's sign-in logs include service principal sign-ins, which show that the app obtained tokens. Microsoft Graph activity logs record every HTTP request Graph receives for the tenant, with the request URL, status and response size. They need a Microsoft Entra ID P1 or P2 licence, a Security Administrator to set up diagnostic settings, and an Azure Log Analytics workspace, storage account or Event Hubs namespace to send them to, which you pay for.

The activity log tells you the agent requested the items of list X at 14:02. It does not tell you which rows came back, which of them went into a prompt, or what the person asked. Rebuilding "which supplier records did the agent read last Tuesday" from request URLs is slow work, and it falls to the admin who approved the app.

Bounding the agent with a replica it reads from

The route we recommend takes Graph out of the agent's hands. SPSync reads the lists you choose into a database you run, and the agent queries that database with a login of its own. The agent then holds no Microsoft Graph permission at all.

On the SharePoint side there is one app and one decision for the admin. SPSync asks for read-only Sites.Selected on each site it replicates, or Lists.SelectedOperations.Selected on single lists, and has no code path that writes to SharePoint. It copies list items and library metadata, never file contents, so the contracts themselves stay where they are. The admin permissions pack has the grant commands.

On the database side, every agent is a login your DBA creates and can disable with one statement. What the agent can read becomes a list of tables, views and policies you can print and attach to the review. The steps below use Postgres and SQL Server; Azure SQL follows SQL Server except where noted.

Start with a view per agent, so the agent never sees the replicated table directly:

```
-- Postgres, run as the role SPSync writes with
create schema agent_procurement;
create view agent_procurement.contracts as
select sp_item_id as contract_id, title, supplier_value as supplier,
       region, renewal_date, annual_value, sp_synced_at as synced_at
from spsync.proc_supplier_contracts
where sp_deleted_at is null;
create role procurement_agent login password 'change-me';
alter role procurement_agent set default_transaction_read_only = on;
alter role procurement_agent set statement_timeout = '15s';
grant usage on schema agent_procurement to procurement_agent;
grant select on agent_procurement.contracts to procurement_agent;
```

The view leaves out the columns the agent has no business reading, such as the owner's email address and the internal notes. A read-only session and a statement timeout stop a bad query becoming an outage.

Row-level security for an AI agent

A view filters columns well. When the same table must show different rows to different agents, use the database's row-level security, which applies to every query the login runs, including ones the model writes for itself.

In Postgres, a policy per role does it, and the Postgres starter kit has a worked example with the two details that catch people out. In SQL Server and Azure SQL, row-level security is a predicate function plus a security policy:

```
-- SQL Server or Azure SQL; spsync_writer is the login SPSync writes with
create schema agent;
go
create table agent.region_access (db_user sysname, region nvarchar(50));
insert agent.region_access values (N'procurement_agent', N'EMEA');
go
create function agent.region_filter(@region nvarchar(50))
returns table with schemabinding as
return select 1 as allowed
        where user_name() = N'spsync_writer'
        or exists (select 1 from agent.region_access a
        where a.db_user = user_name() and a.region = @region);
go
create security policy agent.contracts_by_region
add filter predicate agent.region_filter(region) on spsync.proc_supplier_contracts
with (state = on);
```

Keep the line for the login SPSync writes with. A filter predicate also applies to that login's own updates, and without it the sync cannot see the rows it needs to change.

Auditing the agent's queries in the database

Each engine has a log your DBA already knows how to keep, and each one records the agent's login against every statement.

Database | How to log every read by the agent

PostgreSQL | alter role procurement_agent set log_statement = 'all' sends each statement the role runs to the server log. For structured audit records, install the pgAudit extension and set pgaudit.log = 'read' for the role

Azure Database for PostgreSQL | pgAudit is available as an extension, and its logs can go to Azure Monitor (Learn)

SQL Server | A server audit plus a database audit specification for SELECT ON SCHEMA::agent_procurement BY procurement_agent (Learn)

Azure SQL Database | Built-in auditing to a storage account, Log Analytics or Event Hubs (Learn)

Each log records the statement, the login and the time. None records the rows returned, but with a narrow view the statement says most of what a data protection officer needs. Learn adds one caveat for Azure SQL: under very high load, auditing may let transactions through without recording every event, so size the database for the agent's traffic.

Compare that with the Graph route. The question "what did the procurement agent read last Tuesday" becomes a filter on one login in one log, and revoking the agent is alter role procurement_agent nologin, with nothing to change in Entra.

Where does the agent's data go?

Microsoft 365 Copilot is a covered workload in Microsoft's data residency commitments, and for EU customers it is an EU Data Boundary service (Learn). Those commitments belong to Copilot. A custom agent's prompts go to whichever model endpoint your developers configured, in whichever region they picked.

The replica gives you one fixed point. It sits in a database whose region you chose, and SPSync's desktop and command-line editions run on your own machine or server. Where the rows go after the agent reads them is a decision for whoever built the agent, so put the model endpoint and its region in the review alongside the database. Our post on keeping SharePoint data in region for AI goes through each hop, including Azure OpenAI deployment types and what Azure AI Search logs.

What the replica does not carry

SharePoint item permissions are not carried into the replica. Each replicated list arrives as a table holding every row, whoever could open each item in SharePoint. Views and roles are therefore the control. If a list hides rows from some people with item-level permissions, rebuild that rule as a policy before any agent reads the table. Carrying permissions across, including filling the `permitted_groups` field SPSync already creates in Azure AI Search and leaves empty, is under consideration, with no plan or date.

File contents are not replicated either, so an agent that must read the documents themselves needs a separate route with its own Graph permission. SPSync ships no agent of its own. Point any Postgres or SQL Server MCP server at the replica with the agent's login, or run `spsync mcp serve`, SPSync's own read-only MCP server over the replica. Write-back to SharePoint is designed, not shipped, so a change the agent suggests goes to a staging table a person reviews.

A checklist for the security review

1. List every custom agent, the credential it holds and the Graph permission behind it.
2. Replace tenant-wide `Sites.Read.All` with a `Selected` grant, or move the agent onto a replica so it holds no Graph permission.
3. Give each agent its own database login, read only, with a statement timeout.
4. Grant each login select on views written for its job.
5. Add row-level security where two agents share a table but not its rows.
6. Turn on the database audit for each agent login and agree how long to keep the log.
7. Record the model endpoint and region each agent sends rows to.
8. Rebuild any item-level permission rule from SharePoint as a database policy.

For the architecture behind steps 3 to 6, see the agent data layer use case. For an admin whose ticket says no to Graph access for AI tools, read agents on SharePoint data without a Graph app registration.

Frequently asked questions

Does Restricted SharePoint Search stop a custom AI agent reading a site?

No. Microsoft describes it as a temporary measure that is not a security boundary and does not change permissions. It shapes organisation-wide search and Copilot results. An app that reads list items through Graph with a permission covering that site still reads them, and new enablement has been blocked since 31 July 2026.

Does SPSync give the agent SharePoint's item-level permissions?

No. SPSync does not read item permissions, so every row of a replicated list reaches the database. Decide what each agent sees with its database login, views and row-level security. Carrying permissions into the replica is under consideration, with no plan or date.

Can an agent change SharePoint data through the replica?

No. Every grant SPSync asks for is read only and the product has no call that writes to SharePoint. A change the agent proposes can go to a staging table that a person reviews. Write-back to SharePoint is designed and not shipped.

Does SPSync include an MCP server or an agent?

SPSync ships no agent. It replicates SharePoint lists into a database, and it has a read-only MCP server of its own, spsync mcp serve. To give an agent SQL tools, you can also point any Postgres or SQL Server MCP server at the replica with the agent's own read-only login.

Full, up to date version: <https://spsync.io/resources/governing-custom-agents-on-sharepoint-data/>