NLSQL

Microsoft Entra ID Authentication & Microsoft 365 Integration

Configuration Guide for the NLSQL AI Employee

Connect the AI Employee to each user's OneDrive, Outlook mail and calendar using Microsoft Entra ID app registration and delegated Microsoft Graph permissions.

Document	Post-deployment configuration guide
Audience	Microsoft 365 / Microsoft Entra ID administrators
Product	NLSQL AI Employee (Azure-native deployment)
Version	1.0
Last updated	June 2026

Start here

Complete every step in this guide after you have deployed the NLSQL AI Employee from the Azure Marketplace. You will need administrator access to both Microsoft Entra ID and the Microsoft Teams admin center.

In this guide

Overview

The NLSQL AI Employee can work directly with each team member's Microsoft 365 data — reading and updating OneDrive files, reading and sending Outlook mail, and managing calendar events — acting on behalf of the signed-in user. This guide explains how to enable that access.

Authentication is handled through a **Microsoft Entra ID** (formerly Azure Active Directory) application registration. You register the AI Employee as an app, grant it a set of **delegated Microsoft Graph permissions**, and supply its credentials to the AI Employee. Each user then signs in once and consents; from that point the AI Employee uses **delegated** access, so it can only ever reach data the user is already allowed to see.

What you will configure: an app registration with a sign-in redirect URI, eight Microsoft Graph delegated permissions with tenant-wide admin consent, a client secret, the two environment variables the AI Employee reads, and availability of the app in Microsoft Teams.

> **Good to know**
> The NLSQL AI Employee runs inside your own Azure subscription, so no corporate data leaves your tenant. All Microsoft Graph permissions used here are delegated rather than application-level.

Before you begin

Make sure the following are in place. The full configuration takes about 15 minutes.

- **Administrator role.** You must sign in to Microsoft Entra ID as a Global Administrator (or an Application Administrator together with a Privileged Role Administrator) so you can register the app and grant admin consent.
- **Deployed AI Employee.** The NLSQL AI Employee is already deployed from the Azure Marketplace, and you know its public web app URL (for example `https://<your-ai-employee-app>.azurewebsites.net`).

- **Access to app settings.** You can edit the AI Employee app's environment variables in Azure App Service (Settings › Environment variables / Configuration).
- **Teams admin access.** You can reach the Microsoft Teams admin center at `admin.teams.microsoft.com` to publish and allow the app.

Configuration

Work through the steps in order. Each step shows the corresponding screen in the Azure portal or Microsoft Teams admin center.

Step 1 — Open Microsoft Entra ID

Sign in to the Azure portal at `portal.azure.com`. On the home page, under **Azure services**, select **Microsoft Entra ID** (you can also search for it in the top search bar).

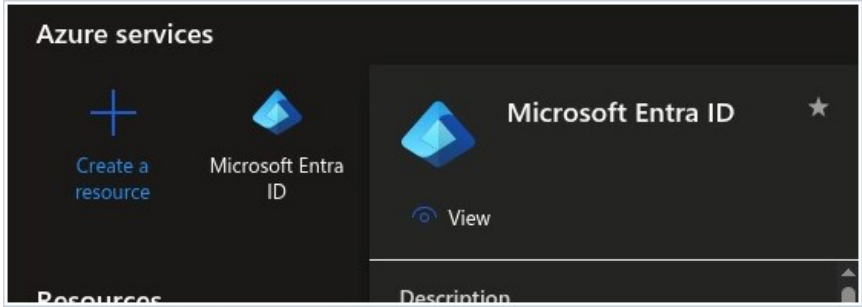

Figure 1. Opening Microsoft Entra ID from the Azure portal home page.

Step 2 — Register a new application

In Microsoft Entra ID, go to **App registrations** › **New registration**, then complete the form:

- **Name** — enter a recognizable name such as `AI-Employee-Coder`.
- **Supported account types** — choose the option that matches your organization. A single-tenant app (*Single tenant only – Default Directory*) is typical when only your own users will sign in.
- **Redirect URI** — set the platform to **Web** and enter your AI Employee web app URL followed by `/callback`, for example `https://<your-ai-employee-app>.azurewebsites.net/callback`.

Select **Register** to create the application.

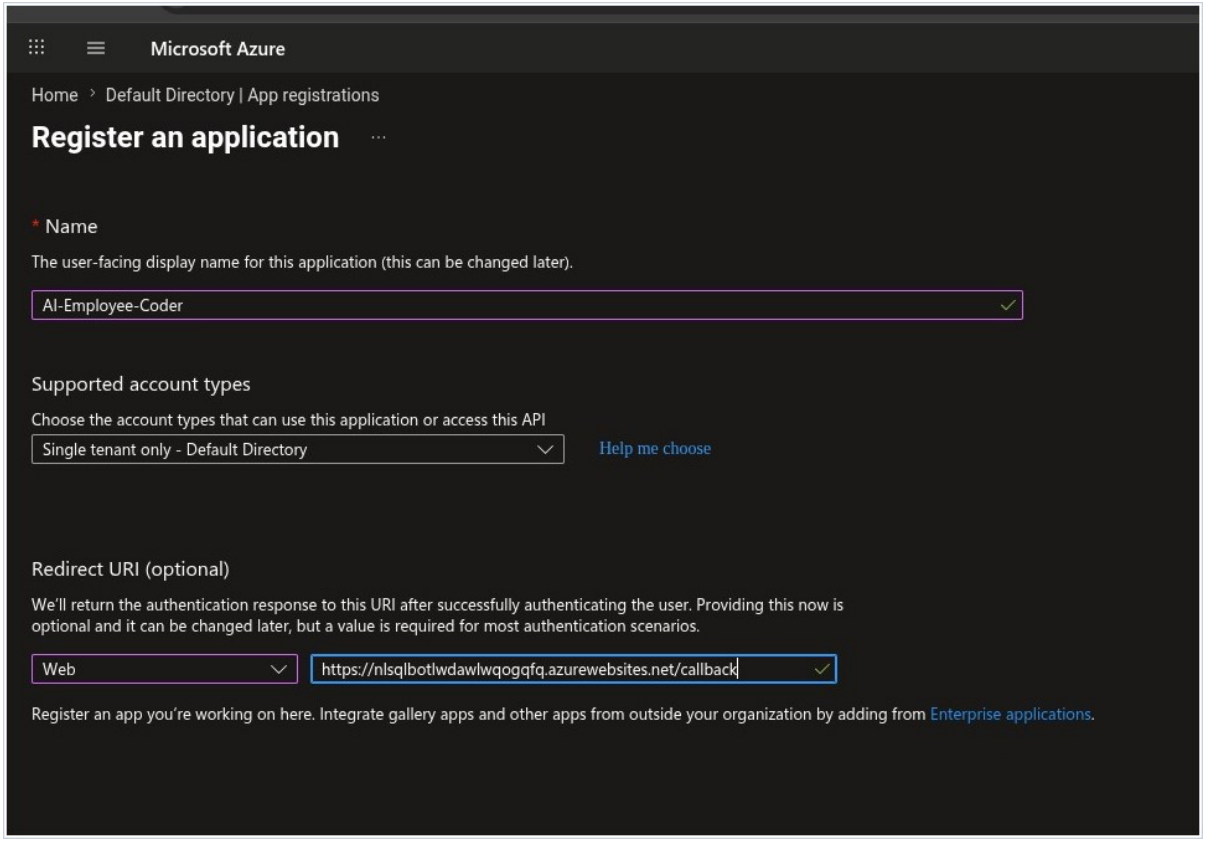

Figure 2. Registering the application. The Web redirect URI must end in /callback.

> **Important**
> The Redirect URI must match the AI Employee's URL **exactly** — same scheme (`https`), host and `/callback` path. A mismatch is the most common cause of sign-in errors (AADSTS50011).

Step 3 — Add Microsoft Graph API permissions

In the new app, open **API permissions** › **Add a permission** › **Microsoft Graph**, then choose **Delegated permissions** (the AI Employee acts as the signed-in user).

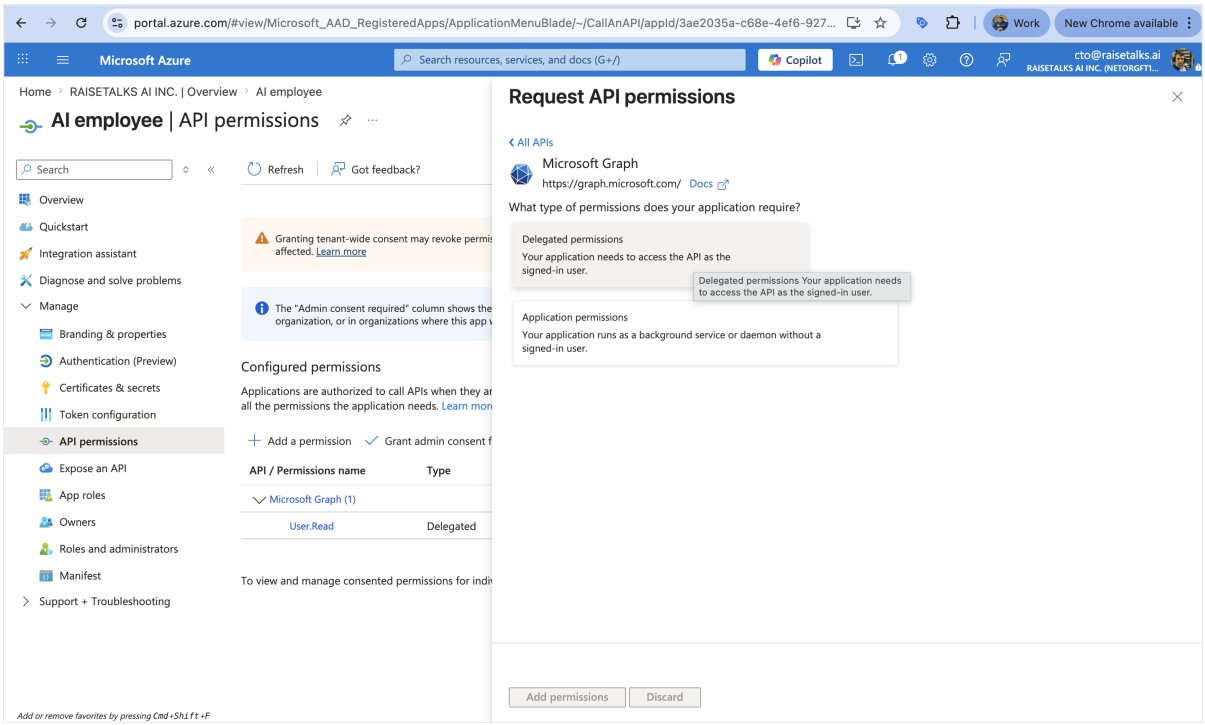

Figure 3. Selecting Delegated permissions under Microsoft Graph.

Add the following eight permissions, then click Add permissions:

- `User.Read` – sign in and read the user profile
- `openid` – sign users in (OpenID Connect)
- `offline_access` – keep access so users need not re-consent each time
- `Files.ReadWrite` – read and write the user's OneDrive files
- `Mail.Read` – read the user's mail
- `Mail.ReadWrite` – read and write the user's mail
- `Mail.Send` – send mail as the user
- `Calendars.ReadWrite` – read and manage the user's calendar

Once added, all eight appear in the Configured permissions list, as shown below.

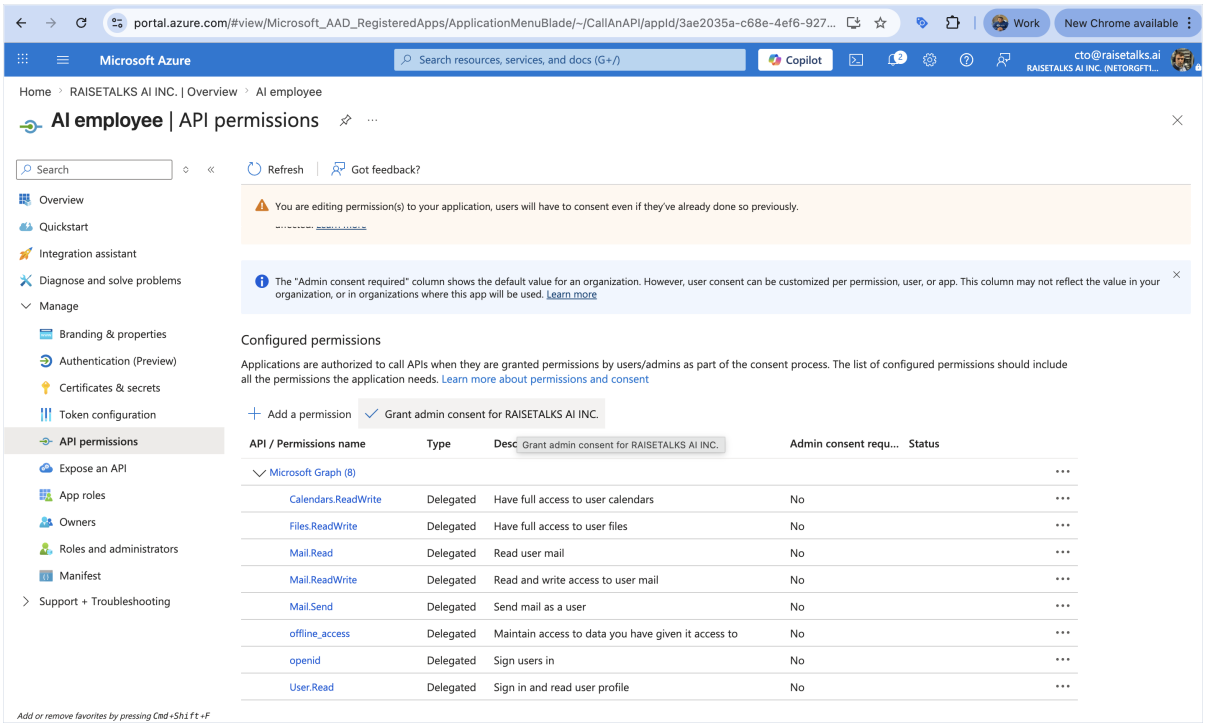

Figure 4. All eight Microsoft Graph delegated permissions configured for the app.

Step 4 — Grant admin consent

Still on the **API permissions** page, click **Grant admin consent for <your tenant>**, then confirm with **Yes**. This consents the permissions once, tenant-wide, so individual users are not prompted to approve each permission themselves.

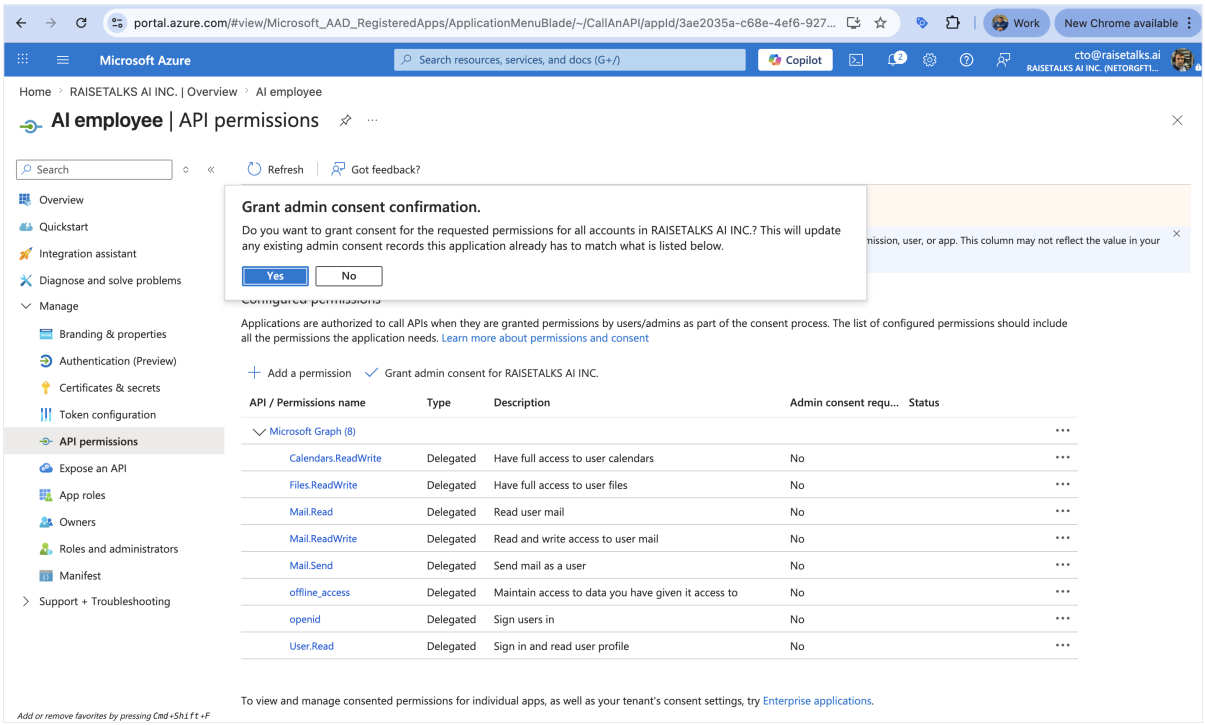

Figure 5. Confirming tenant-wide admin consent for the requested permissions.

> **Verify**
> After consenting, the **Status** column should show a green check (“Granted for <tenant>”) for every permission. If any permission is added later, you must grant admin consent again.

Step 5 — Create a client secret

Open **Certificates & secrets** › **Client secrets** › **New client secret**. Give it a description (for example `user-auth`) and choose an expiry. Click **Add**.

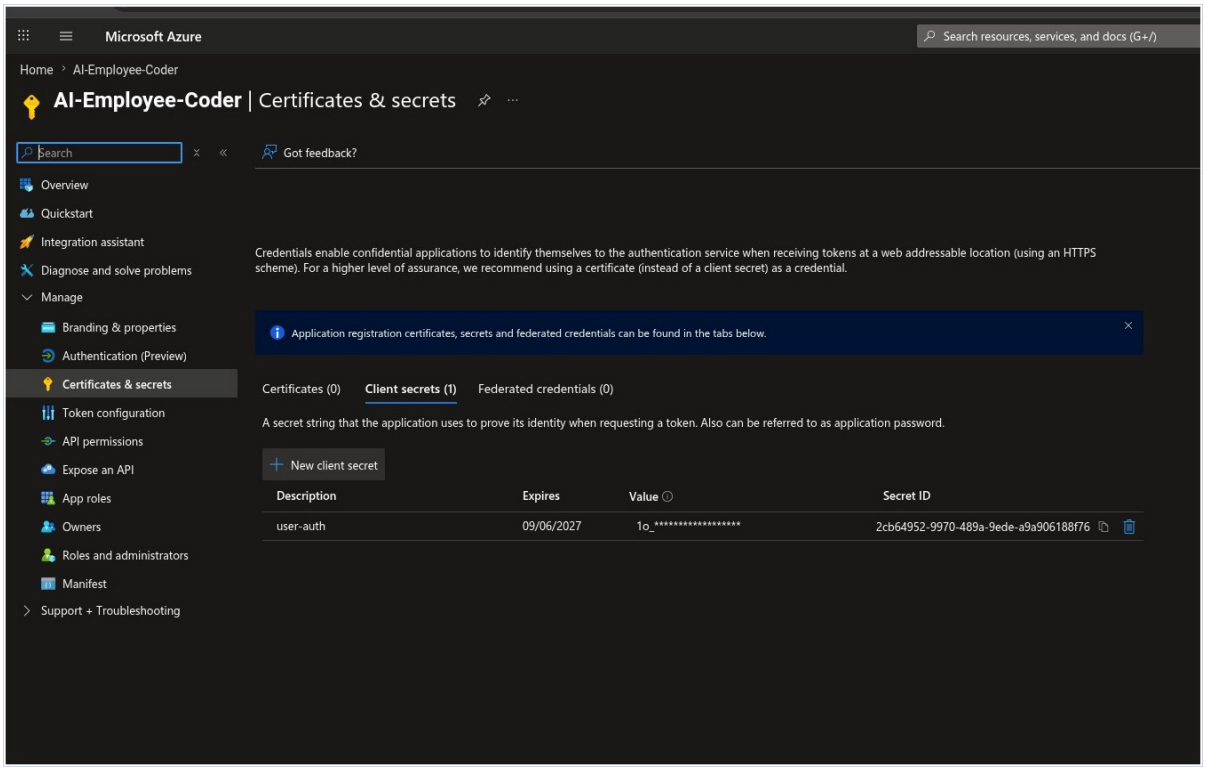

Figure 6. A new client secret created for the application.

> **Important**
> Copy the secret's **Value** immediately — it is shown only once and cannot be retrieved later. Copy the **Value**, not the Secret ID. Note the expiry date and set a reminder to rotate the secret before it expires.

Step 6 — Add the credentials to the AI Employee

Open the AI Employee parent app in Azure App Service and go to its environment variables (Settings › **Environment variables** / Configuration). Add or update these two values:

- `GRAPH_API_ID` — the application's **Application (client) ID**, copied from the app's Overview page.
- `GRAPH_API_SECRET` — the secret **Value** you copied in Step 5.

Save the settings and **restart** the app so the new variables take effect.

Figure 7. The Client ID and secret stored as GRAPH_API_ID and GRAPH_API_SECRET.

Step 7 — Publish and allow the app in Microsoft Teams

Sign in to the Microsoft Teams admin center at `admin.teams.microsoft.com` and go to **Teams apps** › **Manage apps**. Upload the NLSQL AI Employee app package (or open it if already listed), then open the **Users and groups** tab. Under **Installs**, click **Edit installs** and choose who can use it — **Everyone**, specific users or groups — then select **Apply**.

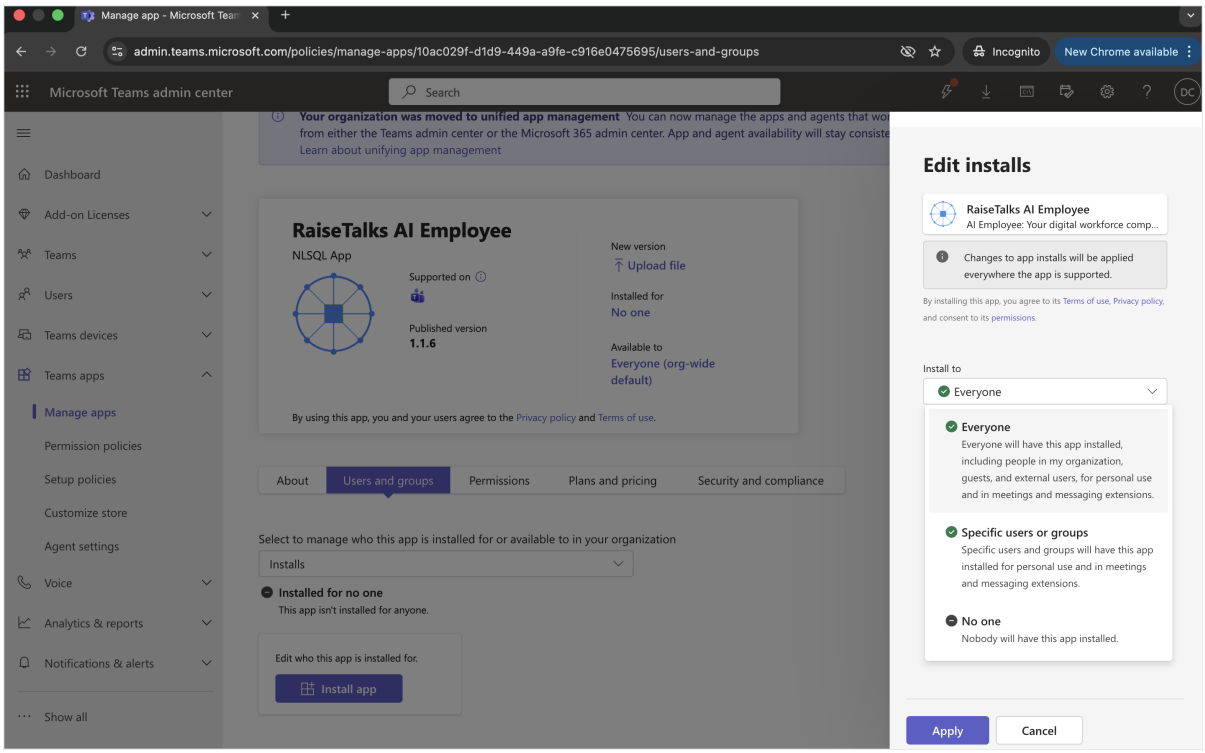

Figure 8. Allowing Microsoft Teams users to install the app in the Teams admin center.

Step 8 — Confirm the app is available

Back on the app's management page, confirm the install status reads **Installed for everyone** (or for the audience you selected). Users can now add the NLSQL AI Employee in Teams and sign in.

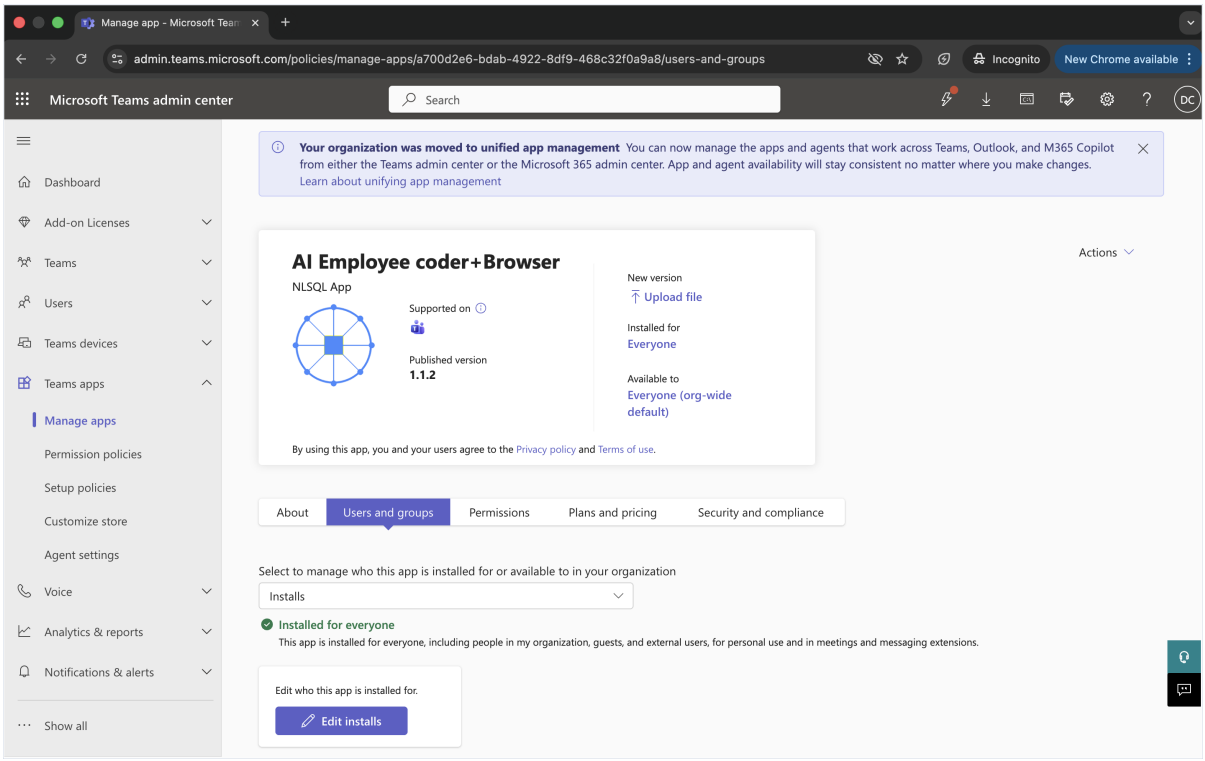

Figure 9. The AI Employee app installed and available to users.

Microsoft Graph permissions reference

Every permission below is delegated, meaning the AI Employee acts only within the signed-in user's own access.

Permission	What it allows	Why the AI Employee needs it
`User.Read`	Sign in and read the user profile	Authenticate the user and read basic profile details
`openid`	Sign users in	Required for OpenID Connect sign-in
`offline_access`	Maintain access to granted data	Refresh tokens so users need not re-consent each session
`Files.ReadWrite`	Full access to user files	Read and update the user's OneDrive files
`Mail.Read`	Read user mail	Read the user's Outlook mail
`Mail.ReadWrite`	Read and write user mail	Draft, organize and update mail
`Mail.Send`	Send mail as the user	Send email on the user's behalf
`Calendars.ReadWrite`	Full access to user calendars	Read and manage calendar events

Securing the client secret

- **Store it safely.** Keep the secret only in the AI Employee's environment variables or in Azure Key Vault. Never place it in source control, email, chat, or this document.
- **Copy it once.** The `Value` is displayed only at creation. If it is lost, create a new secret rather than trying to recover the old one.
- **Rotate before expiry.** Note the expiry date and rotate ahead of it: create a new secret, update `GRAPH_API_SECRET`, restart the app, then delete the old secret.
- **Least privilege by design.** All permissions are delegated, so the AI Employee can never reach data the signed-in user cannot already access.
- **Review before consenting.** Admin consent applies tenant-wide. Review the permission list with your security team before granting it.

> **Rotation reminder**
> Client secrets expire. Add a calendar reminder a few weeks before the expiry date you set in Step 5 so the integration never breaks unexpectedly.

Verify the integration

Confirm each item below once configuration is complete:

- The app registration exists and its Redirect URI ends in /callback, matching the AI Employee URL exactly.
- All eight Microsoft Graph delegated permissions show a green "Granted" status.
- A valid, unexpired client secret exists under Certificates & secrets.
- GRAPH_API_ID and GRAPH_API_SECRET are set on the AI Employee app, and the app has been restarted.
- The app shows "Installed" for the intended audience in the Teams admin center.
- End-to-end test: a user signs in, completes the one-time Microsoft consent prompt, then asks the AI Employee to list recent OneDrive files or draft an email — and it succeeds.

Troubleshooting & FAQ

Symptom	What to check
Sign-in fails with AADSTS50011 (redirect URI mismatch)	The Redirect URI in the app registration must match the app URL plus `/callback` exactly — same scheme, host and path, with no trailing-slash difference.

Symptom	What to check
Users are repeatedly asked to sign in or consent	Confirm that `offline_access` is granted and that admin consent was given, and that the client secret has not expired.
“Need admin approval” / consent is blocked	An administrator must click Grant admin consent. The account used needs Global Administrator or Privileged Role Administrator rights.
A permission shows “Not granted” (no green check)	Return to API permissions and click **Grant admin consent for <tenant>**, then confirm Yes.
Mail, Calendar or Files actions fail with 403	Confirm the specific permission (e.g. `Mail.Send`) is present and consented. Re-grant admin consent after adding any new permission.
The integration worked, then stopped	The client secret most likely expired. Create a new secret and update `GRAPH_API_SECRET`, then restart the app.
The app is not visible to users in Teams	In the Teams admin center, check that the app is allowed in your permission/setup policies and that Installs targets the right audience. Allow a little time for changes to propagate.

Support

For help with deployment or this integration, contact the NLSQL team at nlsql.com/contacts. Product details and documentation are available at nlsql.com.