api0

GUIDE 02

Put Your Product in Teams Without Building a Bot

Your customers live in Microsoft Teams. Here's how to put your product there — in conversation, with each user's own permissions — without writing, hosting or maintaining a bot.

Updated 7 November 2026 · Read online: api0.ai/guides/product-in-teams-without-a-bot

When a B2B customer asks "do you have anything in Teams?", the honest answer is often no. Because a Teams bot is not a small project:

- an Azure Bot resource, an app registration, secrets that expire;
- a web service that receives Bot Framework messages and replies;
- a conversation engine that understands requests and calls your API;
- per-user authentication, so each person acts only with their own permissions;
- a Teams app manifest, icons, approval by the customer's admin;
- and maintenance for all of it.

With api0, you keep the one part Microsoft requires — the Azure Bot resource — and the rest is provided.

How it works

```
Teams user ──► Azure Bot (yours) ──► api0 ──► your API
                                       │
                          tools, identity, log, credits
```

1. The user messages the bot in a private Teams chat.
2. Microsoft forwards the message to the **messaging endpoint** api0 provides.
3. api0's assistant understands the request and calls the **tools** built from your API — as that user.
4. The answer comes back in Teams.

These are the same tools as in Claude or WhatsApp: you set up your API once.

This guide covers the approach. The full step-by-step, with the Teams admin approval and the common errors, is in the Teams guide.

Before you start

- Your API imported into api0 (**Tools** tab) — see the main guide.

- Access set up (**Access** tab): ideally each person's own credential, so your application applies their permissions.
- Access to the **Azure portal** with the right to create an *Azure Bot* resource.
- On the customer side: a Teams admin who approves the app, or custom app upload allowed.

Step 1 — Create the Azure Bot resource

1. Azure portal → *Create a resource* → **Azure Bot**.
2. Bot name, subscription, resource group. Pricing tier: **F0**.
3. **App type: Single Tenant.** Note the **App ID** (a GUID) and the **Microsoft tenant ID**.
4. In the resource → *Configuration* → *Manage Password* → **New client secret**. Copy the value right away: Azure only shows it once.

Step 2 — Register it in api0

app.api0.ai → **Channels** → **Teams**:

- App ID, secret, Microsoft tenant ID;
- bot name (the one your users will see);
- optional: a system prompt (tone, answer length, confirmations).

api0 immediately asks Microsoft for a token with these values. If the App ID and secret don't belong together, you know right away, with a clear message — not three days later in production.

In return, api0 shows your bot's **messaging endpoint**.

Step 3 — Point Azure at api0

In the Azure Bot resource:

1. *Configuration* → **Messaging endpoint** → paste the URL from api0 → Apply.
2. *Channels* → **Microsoft Teams** → accept the terms → Apply.

Step 4 — The Teams app package

A bot reaches nobody until a Teams app naming it is installed. That app is a zip: a `manifest.json` and two icons. This is often where projects stall.

In Channels → Teams, api0 **generates the package** from what it already knows: App ID, bot name, your tenant's Get started page. You download it, and that's it.

The app is **personal** scope: the bot answers in one-to-one chats, where it makes sense.

Prefer to build it yourself (branded icons, your own text)? A manifest template is provided.

Step 5 — Install and test

- For yourself: Teams → *Apps* → *Manage your apps* → **Upload an app**.
- For an organization: the Teams admin publishes it from the *Teams admin center*.

In api0, the **Test** button checks the chain. Then message the bot: on the first message, it sends the person a link to **link their account**. Once linked, every action is taken with their identity.

Software vendors: one bot per customer

If you sell software to several companies, the cleanest model is **one api0 tenant per customer**:

- each customer has its own tools, users, credits and log;
- its bot lives in **its** Azure tenant (Single Tenant), which its security team prefers;
- you bill usage customer by customer.

You can do the setup for them (with an Azure account in their Microsoft tenant) or send them the steps above: about half an hour on their side.

What your customer gets

- Your product in the tool they already have open all day.
- Answers and actions in plain language, no training needed.
- Every action with the person's own permissions, nothing more.
- A log of every call.

What you get

- No bot to write, host or maintain.
- The same setup serves Claude, WhatsApp and Telegram.
- A concrete answer to your customers' "AI" requests.

Common problems

Symptom	Likely cause
"Invalid App ID" in api0	The secret was pasted into the App ID field. The App ID is a GUID.
Microsoft rejects the App ID / secret pair	Expired secret, or wrong Microsoft tenant ID.
The bot doesn't answer	Messaging endpoint not pasted in Azure, or Teams channel not enabled.
The app doesn't show up in Teams	Custom app upload is disabled: go through the Teams admin.