api0

GUIDE 06

Connect Azure Services to api0: Azure DevOps, SharePoint, Azure SQL and Logic Apps

How to expose business data, content and processes from Azure through api0, using only API endpoints: who each call runs as, how to set up each Microsoft component, and the errors you'll hit along the way.

Updated 11 November 2026 · Read online: api0.ai/guides/connect-azure-services

api0 calls HTTP endpoints. There is no Azure connector to install: for each service, you declare its endpoints in api0 and say how api0 gets the token that signs them.

The first decision is who each call runs as. This guide recommends each person's own Microsoft account wherever the service knows people, and one service identity only where it does not.

Placeholder	What it is	Where to find it
`<tenant-id>`	Your Microsoft directory ID	Entra admin center → **Overview** → *Tenant ID*
`<client-id>`	Your Entra application's ID	**App registrations** → your app → *Application (client) ID*
`<site-id>`	The SharePoint site's Graph ID	`GET https://graph.microsoft.com/v1.0/sites/<host>:/sites/<name>?$select=id`

1. Who does each call run as?

	Each person signs in	One service identity
api0 mode	`per_user_oauth`	`client_credentials`
Microsoft permission type	Delegated	Application
What the service sees	The person who asked	The application
What the person can reach	Only what their own account can reach	Everything the application can reach
Audit trail in Azure	Each person's name	The application's name
First use	Each person clicks **Connect** once	Nothing

Use each person's sign-in for Azure DevOps, SharePoint, OneDrive, Outlook and anything else in Microsoft Graph. With one service identity, anyone who can talk to the assistant acts with the application's rights, and a name written in a request is only text.

Keep a service identity only where the service has no notion of people, or where everyone must see the same thing: a read-only reporting database, or a Logic App that runs a process with its own identity.

In Teams, each person always has their own one-to-one chat and history with the bot, whichever mode you pick. The mode only decides whose rights the calls use.

2. Does the service have an HTTP API?

Service	HTTP API	What to add
Azure DevOps	Yes	Nothing
SharePoint	Yes, Microsoft Graph	Nothing
Logic App	Yes, its Request trigger	Nothing
Azure SQL	No, it speaks TDS	An API in front of it, such as Data API builder

If a service has no HTTP API, api0 cannot reach it directly: put an API in front of it first.

3. Create one api0 connection per token audience

A Microsoft token is valid for one API only, its audience. A token for Microsoft Graph is refused by your own API, and the reverse. So you need one connection per audience, even with a single Entra application.

On app.api0.ai, open the **Connections** card on the **API** page, click **Add a connection**, or **Edit** on an existing one.

3.1 Each person signs in

Field	Value
Name	A short name, for example `graph`
Sign in as	Each person signs in (OAuth)
Authorize URL	`https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/authorize`
Token URL	`https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/token`
Client ID	`<client-id>`
Client secret	The secret's Value
Scope	See the table below

Calling	Scope
Azure DevOps	`499b84ac-1321-427f-aa17-267ca6975798/user_impersonation offline_access`
Microsoft Graph, SharePoint read and write	`https://graph.microsoft.com/Sites.ReadWrite.All offline_access`
Microsoft Graph, SharePoint read only	`https://graph.microsoft.com/Sites.Read.All offline_access`
Your own API or Logic App	`api://<client-id>/access_as_user offline_access`

Name each delegated permission, and add `offline_access` so api0 can renew access without asking the person to sign in again. `https://graph.microsoft.com/.default offline_access` also works: it asks for every delegated Graph permission granted to the application.

Saving replaces the whole connection, and api0 never sends the stored secret back to the browser: enter the secret again each time you edit it.

3.2 One service identity

Field	Value
Name	A short name, for example `demo-app`
Sign in as	One service identity (client credentials)
Token URL	`https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/token`
Client ID	`<client-id>`
Client secret	The secret's Value
Scope	See the table below

Calling	Scope
Microsoft Graph	`https://graph.microsoft.com/.default`
Your own API or Logic App	`api://<client-id>/.default`
Azure Resource Manager	`https://management.azure.com/.default`

The scope must be one value ending in `/.default`. Client credentials uses the application permissions already granted to the application.

4. Microsoft components, one by one

Each section says what the component is for, how to set it up, and how to check it.

4.1 The Entra application

One application can serve every connection. It holds the permissions, the secret and, for per-person sign-in, the address Microsoft sends people back to.

1. In the Entra admin center, open **App registrations** → **New registration**, name the application and register it.
2. Open **Certificates & secrets** → **New client secret**. Copy the secret's *Value*, not its *Secret ID*. A new secret can take a few minutes to be accepted. Type it only into app.api0.ai.
3. For per-person sign-in, open **Authentication** → **Add a platform** → **Web** and add the redirect URI `https://gateway.api0.ai/oauth/downstream/callback`. Leave *Access tokens* and *ID tokens* unchecked.
4. Open **API permissions** → **Add a permission** and add what each service needs: **Delegated permissions** for per-person sign-in, **Application permissions** for a service identity. Sections 4.2 to 4.5 list them.
5. Click **Grant admin consent for** . Every line must show a green *Granted*.

Admin consent matters for delegated permissions too. Many directories do not let people consent to applications themselves: without the admin's consent, each person sees *Need admin approval* at Connect.

Check: on **API permissions**, every permission shows *Granted*, with the type you meant, *Delegated* or *Application*.

The secret expires, after one or two years. When it does, create a new one and update every api0 connection that uses it.

4.2 Azure DevOps

Azure DevOps knows each person: use per-person sign-in, so every work item is created in their name and with their rights.

1. In the Entra application, add the **Azure DevOps** → `user_impersonation` permission, as **Delegated permissions**, and grant admin consent.
2. The Azure DevOps organization must be connected to the same directory as the application: **Organization settings** → **Microsoft Entra**.
3. In api0, import the Azure DevOps endpoints after replacing every `YOUR-ORGANIZATION` with your organization's name, and bind them to a section 3.1 connection with the scope `499b84ac-1321-427f-aa17-267ca6975798/user_impersonation offline_access`.
4. Each person opens `https://dev.azure.com/<organization>` once in a browser: Azure DevOps refuses API calls from an account that has never signed in there.

Never click **Connect** with an account that holds an admin role: Microsoft then demands a multi-factor sign-in that background renewal can't provide.

The full step-by-step, with api0's wizard and every error along the way, is in guide 05: each person's Microsoft sign-in.

Check: a person clicks **Connect**, then asks Claude or Teams to create an issue. In Azure DevOps, the work item shows their name under **Created By**.

4.3 SharePoint through Microsoft Graph

1. In the Entra application, add the Microsoft Graph permission `Sites.ReadWrite.All`, or `Sites.Read.All` to read only. Pick **Delegated permissions** for per-person sign-in, the recommended choice, or **Application permissions** for a service identity. With application permissions, `Sites.Selected` limits access to chosen sites.
2. Grant admin consent.
3. In api0, use the base URL `https://graph.microsoft.com/v1.0/sites/<site-id>` and the connection from section 3.1.

With per-person sign-in, Graph applies each person's SharePoint rights. Give people access to the site, and restrict who can edit a list where an edit means a decision, such as an approval column: in the list, open **Settings** → **List settings** → **Permissions for this list**, stop inheriting permissions, and give *Edit* only to the people who decide.

Example endpoints:

Purpose	Verb	Path
Read a list	GET	`/lists/<list>/items?$expand=fields&$select=id`
Update an item	PATCH	`/lists/<list>/items/{item_id}/fields`
Search documents	GET	`/drive/root/search(q='{query}')`
Read a text document	GET	`/drive/items/{item_id}/content`

Name a parameter `$filter` to let the agent filter a list. Only indexed columns can be filtered reliably: in **List settings** → **Indexed columns**, index the columns people filter on.

Check: someone with access to the site clicks **Connect**, then asks for a list in Teams or Claude.

4.4 Azure SQL through Data API builder

Data API builder turns tables and views into REST endpoints with `$filter`, `$orderby` and `$first`.

1. Create the database. The Azure SQL free offer is enough for a demo: set it to stop at its free limit, not to bill beyond it.
2. Run Data API builder where it can reach the database, for example in an Azure Container App, with the database's connection string as a secret. Allow Azure services through the SQL firewall.
3. In the Entra application, open **Expose an API** and set the Application ID URI to `api://<client-id>`.
4. For a service identity, open **App roles** and create a role whose allowed member type is *Applications*, then in **API permissions** → **My APIs**, add that role to the application and grant admin consent. For per-person sign-in, add a scope named `access_as_user` under **Expose an API** instead.
5. In **Manifest**, set the access token version to 2.
6. In `dab-config.json`, use provider `EntraId`, audience `<client-id>` and issuer `https://login.microsoftonline.com/<tenant-id>/v2.0`.
7. Give the `authenticated` role only the actions you want. Read only is a good default.

A shared, read-only reporting database is a sound use of a service identity.

The database, and a Container App scaled to zero, may sleep when idle. Expect the first call to be slow.

Check: ask for one table's rows in Claude or Teams. If Data API builder refuses, see the error recap.

4.5 A Logic App through its Request trigger

1. Use a Consumption Logic App with a **When an HTTP request is received** trigger, and end it with a **Response** action.
2. In **Settings** → **Authorization**, add a policy with issuer `https://login.microsoftonline.com/<tenant-id>/v2.0` and audience `<client-id>`.
3. Disable SAS on the trigger, so that the URL carries no signature.
4. In api0, use the trigger URL up to `/paths`, with the path `/invoke?api-version=2016-10-01`.
5. If the Logic App calls another API, give it a managed identity and grant that identity the API's permission.

The application needs the same setup as in steps 3 to 5 of section 4.4, without Data API builder.

A Logic App acts with its own managed identity, whoever calls it. If the person matters, for example who requested something, send it in the request and check it in the workflow; do not rely on a name the person types.

Check: open the Logic App's **Run history** after a call. Each call is a run, with its input and each action's result.

4.6 The Teams bot

The Teams guide (guide 07) covers the Teams licence, the Azure Bot, connecting it in api0 and installing the app. Once it works for one API, it works for every API of the tenant: nothing changes in Teams when you add Azure DevOps, SharePoint, SQL or a Logic App.

Each person links their Teams account to their api0 account once, by sending the bot the code shown on the linking page. The bot answers only in one-to-one chats.

5. Bind each endpoint to its connection

Import your endpoints on the **API** page. Then, in the **Connections** card, use **Which connection each tool uses** to pick the right connection for each one. An endpoint that names no connection uses `default`.

Importing a group again can reset that choice: check it after each import.

6. Check and try

Ask the setup assistant for `setup_status` with the connection's name. It reports missing settings, a scope that does not match the API called or the mode, and connections no endpoint uses.

For a connection where each person signs in, each person then opens **Settings** → **My access tokens** on app.api0.ai and clicks **Connect** next to it. They must be signed in to app.api0.ai with the same api0

account their Teams account is linked to: the access is stored under that account.

Then ask a read-only question in Claude or Teams. Azure settings are not visible to api0: if a call fails, give the exact message to `explain_error`.

Error recap

Message	Cause	Fix
`Need admin approval` at Connect, or `AADSTS90094` / `AADSTS65001`	No admin consent for the delegated permissions, and people may not consent themselves	An admin opens **API permissions** and clicks **Grant admin consent**, until every line shows *Granted*, then the person clicks Connect again
`your access is not set up, or has expired` from a tool	The person has not clicked Connect, or connected from another api0 account than the one linked in Teams	Open Settings → My access tokens with the api0 account linked in Teams, and click Connect
`AADSTS50011` redirect URI mismatch	The application lacks api0's redirect URI	Add `https://gateway.api0.ai/oauth/downstream/callback` under **Authentication** → **Web**
`AADSTS7000215` invalid client secret	The Secret ID was pasted, or the secret is brand new	Wait a few minutes, then paste the secret's Value again
`AADSTS7000222` client secret keys are expired	The secret expired	Create a new secret and update every connection using it
`AADSTS1002012` client credential flows must have a scope value with `/.default`	A service identity's scope does not end in `/.default`	Use a scope from section 3.2

Message	Cause	Fix
`AADSTS500011` resource principal not found	The scope names an API this directory does not know	Set the Application ID URI, and check the `<client-id>` in the scope
`Either scp or roles claim need to be present in the token`	The application has no permission on Microsoft Graph	Add the permission and grant admin consent
`accessDenied` or `403` from Microsoft Graph with per-person sign-in	The person has no access to that site or list	Give the person access in SharePoint
`Access Not Allowed` `AuthorizationCheckFailed` from Data API builder	No valid token, or a role without that action	Bind the endpoint to the right connection, then check audience, issuer and token version 2
`Field cannot be referenced in filter or orderby as it is not indexed`	The SharePoint column has no index	Index the column, or filter on one that has an index
`The server did not receive a response from an upstream server` from a Logic App	The workflow failed before its Response action	Open the Logic App's **Run history** and read the failed action
Logic App refuses a call carrying both a signature and a bearer token	SAS and OAuth were used together	Disable SAS on the trigger and remove `sp`, `sv` and `sig` from the URL
`The selected region is currently not accepting new customers`	That region is closed to your subscription	Pick another region. A resource group stays in its region, so create a new one
`Client with IP address is not allowed to access the server`	The SQL firewall blocks that address	Add a firewall rule. Behind a proxy, the address can vary, so allow its range temporarily

Message	Cause	Fix
`Database is not currently available` retry the connection later	The serverless database is waking up	Retry after a minute