

# Setting up Delegated Authentication with OpenID on Microsoft Azure

Before setting up the installer, you have to configure Microsoft Azure Active Directory.

## Set up Microsoft Azure Active Directory

- You need to create an `App registration`.
- You have to select `Redirect URI (optional)` and set it to `https://matrix.your-domain.com/_synapse/client/oidc/callback`

### Register an application ...

The user-facing display name for this application (this can be changed later).

#### Supported account types

Who can use this application or access this API?

- ☒ Accounts in this organizational directory only (olivier-doc only - Single tenant)
- ☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- ☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- ☐ Personal Microsoft accounts only

[Help me choose...](#)

#### Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Web

▼

e.g. https://example.com/auth

✓

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Enterprise applications](#).

For the bridge to be able to operate correctly, navigate to API permissions, add Microsoft Graph APIs, choose Delegated Permissions and add

- openid
- profile

Remember to grant the admin consent for those.

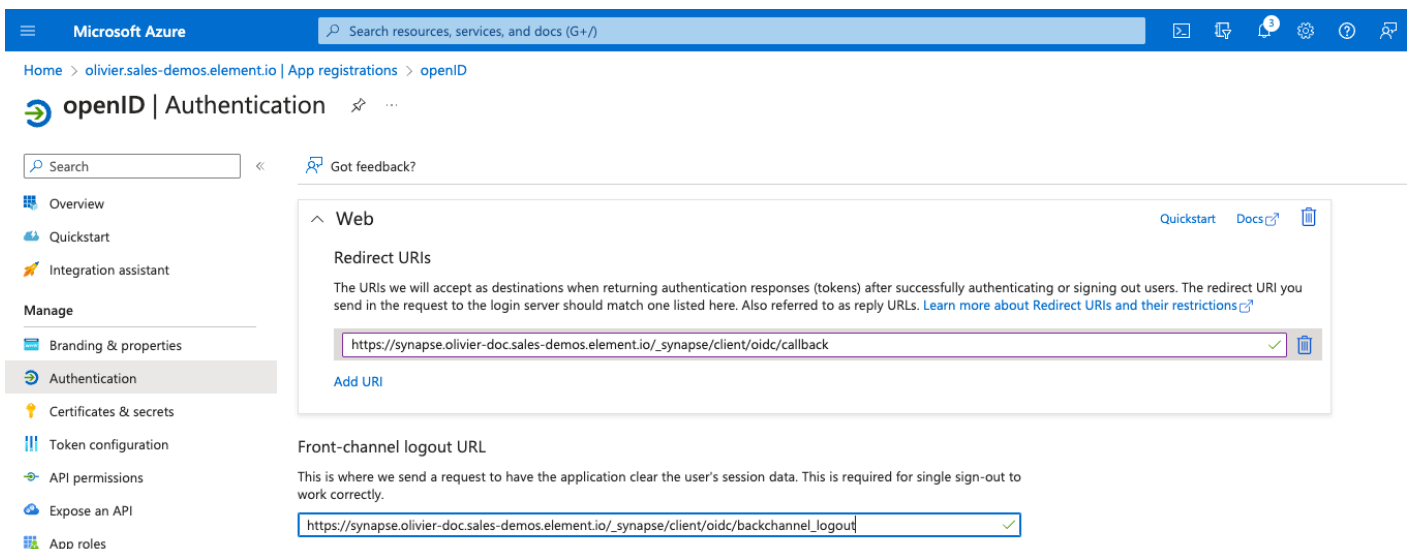
To setup the installer, you'll need

- the `Application (client) ID`
- the `Directory (tenant) ID`
- a secret generated from `Certificates & secrets` on the app.

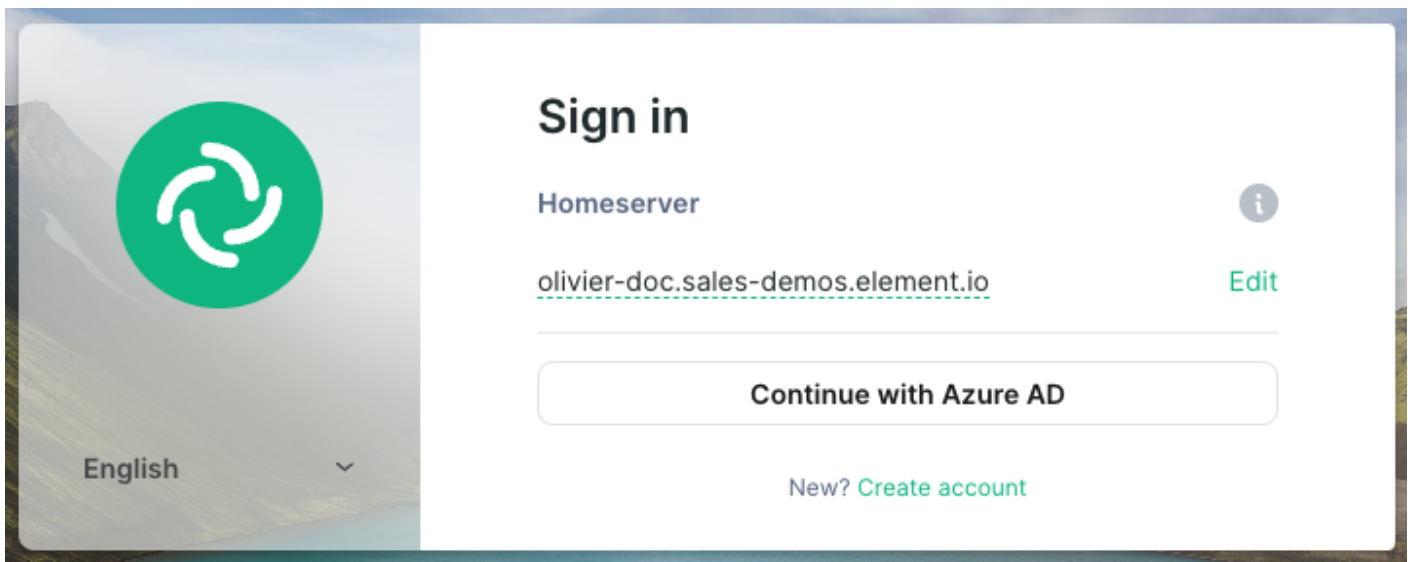
# Configure the installer

Add an OIDC provider in the 'Synapse' configuration after enabling `Delegated Auth` and set the following fields in the installer:

- `Allow Existing Users` : if checked, it allows a user logging in via OIDC to match a pre-existing account instead of failing. This could be used if switching from password logins to OIDC.
- `Authorization Endpoint` : the oauth2 authorization endpoint. Required if provider discovery is disabled.  
`https://login.microsoftonline.com/<Directory (tenant) ID>/oauth2/v2.0/authorize`
- `Backchannel Logout Enabled` : Synapse supports receiving OpenID Connect Back-Channel Logout notifications. This lets the OpenID Connect Provider notify Synapse when a user logs out, so that Synapse can end that user session. This property has to be set to `https://your-domain/_synapse/client/oidc/backchannel_logout` in your identity provider



- `Client Auth Method` : auth method to use when exchanging the token. Set it to `Client Secret Post` or any method supported by your Idp
- `Client ID` : your `Application (client) ID`
- `Discover` : enable/disable the use of the OIDC discovery mechanism to discover endpoints
- `Idp Brand` : an optional brand for this identity provider, allowing clients to style the login flow according to the identity provider in question
- `Idp ID` : a string identifying your identity provider in your configuration
- `Idp Name` : A user-facing name for this identity provider, which is used to offer the user a choice of login mechanisms in the Element UI. In the screenshot below, `Idp Name` is set to `Azure AD`



- **Issuer**: the OIDC issuer. Used to validate tokens and (if discovery is enabled) to discover the provider's endpoints  
https://login.microsoftonline.com/<Directory (tenant) ID>/v2.0
- **Token Endpoint**: the oauth2 authorization endpoint. Required if provider discovery is disabled.
- **Client Secret**: your secret value defined under "Certificates and secrets"

**openID | Certificates & secrets**

Search < Got feedback?

Overview  
Quickstart  
Integration assistant

**Manage**

- Branding & properties
- Authentication
- Certificates & secrets**
- Token configuration
- API permissions
- Expose an API
- App roles
- Owners

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Application registration certificates, secrets and federated credentials can be found in the tabs below.

Certificates (0) **Client secrets (1)** Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.



+ New client secret

Description	Expires	Value ⓘ	Secret ID
Password uploaded on Wed ...	5/2/2025	G0P8Q~.5Yxf3wPcXuYllpJy...	fb62af90-740e-498c-af03-...

- **Scopes**: add every scope on a different line
  - The openid scope is required which translates to the Sign you in permission in the consent UI
  - You might also include other scopes in this request for requesting consent.

## Scopes



A list of scopes requested during the authorization process.



Standard scopes include openid, profile, email.

openid

Edited



Standard scopes include openid, profile, email.

profile

Edited

ADD MORE SCOPES

- User Mapping Provider: Configuration for how attributes returned from a OIDC provider are mapped onto a matrix user.

## User Mapping Provider

### Subject Template

The claim used to identify the subject of the ID token.

### Picture Template

The template used to generate the user's profile picture URL in Matrix.

### Localpart Template

The template used to generate the local part of the user's Matrix ID.

Localpart Template \*

`{{ user.preferred_username.split('@')[0] }}`

The template used to generate the local part of the user's Matrix ID.

### Display Name Template

The template used to generate the user's display name in Matrix.

Display Name Template \*

`{{ user.name }}`

The template used to generate the user's display name in Matrix.

### Email Template

The template used to generate the user's email address in Matrix.

- **Localpart Template** : Jinja2 template for the localpart of the MXID. Set it to `{{ user.preferred_username.split('@')[0] }}` for Azure AD
- **Display Name Template** : Jinja2 template for the display name to set on first login. If unset, no displayname will be set. Set it to `{{ user.name }}` for Azure AD

Other configurations are documented [here](#).