Channel Meeting Bot

This is a proof of concept used to show how to install a bot in Microsoft Teams that can proactively install himself for the users of a Team/Group, allowing it them to schedule meetings on behalf of an user for all members of a channel. Later on, the bot also sends an e-mail notifying every user of the channel about the scheduled meeting. Finally, the bot also sends an 1:1 message to each user in the channel notifying them of the meeting.

You can find here a video demonstrating this bot:

The base template for this bot was the Proactive Installation Bot Sample provided by Microsoft: [Microsoft-Teams-Samples/samples/graph-proactive-installation/csharp at main · OfficeDev/Microsoft-Teams-Samples (github.com)](https://github.com/OfficeDev/Microsoft-Teams-Samples/tree/main/samples/graph-proactive-installation/csharp). All credits to this sample’s code used in my bot’s code goes to the Microsoft Teams Samples GitHub community. Most of the installations instructions below also belongs to the Proactive Installation Bot Sample’s documentation, so thank you Microsoft Community for the great documentation.

Pre requisites to install the bot:

* Microsoft Teams is installed and you have an account.
* .NET Core SDK version 6.0.
* ngrok or equivalent tunnelling solution.

Setup:

1. Create a new Team in your Microsoft Teams environment or use an existing one. Ensure it has at least one channel. Make it have 2 or more members. Get the **Object ID (User ID)** of the Team´s owner (current limitation: please ensure that the Team has only 1 owner) by accessing Azure AD -> Users. Keep it somewhere.
2. Register a new Multitenant application in the Azure Active Directory – App Registrations portal.
   1. Required Microsoft graph Application level permissions to run this sample app:
      1. TeamsAppInstallation.ReadWriteForUser.All
      2. TeamsAppInstallation.ReadWriteForChat.All
      3. TeamsAppInstallation.ReadWriteForTeam.All
      4. Group.ReadWrite.All
      5. OnlineMeetings.ReadWrite.All
      6. Mail.Send
   2. Please note that these permissions may expose vulnerabilities to your tenant that would have to be dealt with it if it was a production tenant (such as the Mail.Send permission allowing the bot to impersonate any user when sending an email). As this is only a sample, we can use the permissions freely.
   3. Grant Admin consent to the permissions mentioned above.
3. Go to the Overview tab of your App Registration, copy the **App ID (Application Client ID)** and keep it somewhere. Go to Certificates & Secrets, create a new client secret. Copy the **Secret Value** (not the Secret ID) and keep it somewhere.
4. Grant Online Meeting Application Access Policy to your App. This is necessary for the bot application to be allowed to impersonate an user when creating a new meeting. Follow the [instructions here](https://learn.microsoft.com/en-us/graph/cloud-communication-online-meeting-application-access-policy). Use the **App ID** and **User ID** collected earlier. Please wait up to 60 minutes before testing the bot in Teams as this configuration may take a while to have effect on the tenant.
5. Run ngrok: ngrok http -host-header=rewrite 3978.
   1. Copy the URL for the ngrok tunnel generated.
6. Go to Azure Bot Services and create a new Azure Bot. Follow the instructions [here](https://docs.microsoft.com/azure/bot-service/bot-service-quickstart-registration?view=azure-bot-service-3.0). When selecting the option “Creation type”, be sure to select “Use existing app registration” and provide the **App ID** that you copied earlier.
   1. Ensure that you have enabled the [Teams Channel in the bot](https://learn.microsoft.com/en-us/azure/bot-service/channel-connect-teams?view=azure-bot-service-4.0).
   2. Go to the Configuration tab in the bot and use your ngrok tunnel URL as the messaging endpoint: https://<your\_ngrok\_url>/api/messages.

MS Teams Manifest Setup:

* Edit the manifest.json file contained in the ./TeamsAppManifest folder to replace your Microsoft App Id (the **App Id** collected ealier) everywhere you see the place holder string **{{Microsoft-App-Id}}** (depending on the scenario the Microsoft App Id may occur multiple times in the manifest.json).
* Zip up the contents of the TeamsAppManifest folder (**only the contents of the folder, not the folder itself**) to create a manifest.zip file.
* Upload the manifest.zip file to Teams:
  + Go to Microsoft Teams. From the lower left corner, select Apps.
  + From the lower left corner, select Manage your apps.
  + Select Upload an app.
  + Select Upload an app to your org’s app catalog. Be sure to be with an admin account, otherwise you will need to approve the app in the Teams Admin Center.
  + Go to your project directory, go to the ./TeamsAppManifest folder, select the zip file, and choose Open.
  + Select Add in the pop-up dialog box. Your app is uploaded to Teams.
* You can also upload the app directly through the Teams Admin Center.

Code setup:

1. Clone the Git Repository: <https://github.com/fsdp15/Microsoft-Teams-Channel-Meeting-Email-Sample.git>.
2. Modify the appsettings.json file and fill in the following details:
   1. {{YOUR-MICROSOFT-APP-ID}} – The **App Id** collected before.
   2. {{ YOUR-MICROSOFT-APP-PASSWORD}} – Generated before as **Client Secret Value**.
   3. {{ YOUR-APPCATALOG-TEAM-ID }} – You can get the TeamsappcatalogAppId for your app by navigatin to the App page in the Teams Admin Center (App Id field) or by using the [following query in the Graph Explorer](https://developer.microsoft.com/graph/graph-explorer?request=appCatalogs%2FteamsApps%3F%24filter%3DdistributionMethod%20eq%20%27organization%27&method=GET&version=v1.0&GraphUrl=https://graph.microsoft.com) (teamApp.Id).
3. Run the bot from Visual Studio:
   1. Launch Visual Studio.
   2. File -> Open -> Project/Solution.
   3. Select ChannelMeetingBot.sln file.
   4. Press F5 to run the project.

The bot is running. Add it to a Team using Microsoft Teams and then mention it with @Channel Meeting Bot. Send a hello message to get the Adaptive Card message with the available options to choose.