# Exercise 4: Understanding messaging extensions in Microsoft Teams

## Task 1: Create a messaging extension with action-based commands

### Get sample project files

1. Open PowerShell and navigate to **C:/LabFiles/Teams**
2. Clone the repository, execute the following command:

* git clone https://github.com/Microsoft/botbuilder-samples.git

1. In a terminal, navigate to **samples/javascript\_nodejs/51.teams-messaging-extensions-action**
2. Install modules:

* npm install

1. Run ngrok - point to port 3978:

* ngrok - point to port 3978

### Create Bot Framework registration resource

You need to create an Azure Bot Service:

1. Go back to the Azure Portal, [https://portal.azure.com](https://portal.azure.com/)
2. Search for **Bot Services**.
3. In the Bot Service creation page, select **Web App Bot**.
4. From the **Web App Bot** page, select **Create**.
5. Fill in the required fields:
   * **Bot** **handle**: type a name for your bot. The bot name must be unique and will give you a message when it’s not.
   * **Subscription**: ensure your Azure subscription is selected.
   * **Resource group**: select **Create new** and give it a name such as **WebAppBotResource**.
   * **Location**, will default to your current tenant location. You can leave this as the default or select a different location.
   * **Pricing tier**: select **F01 (10K Premium Messages)**
   * **App name**: type a unique name for your app. This name you provide will be the prefix for the URL of your bot. You bot will have it’s own unique URL such as botname.azurewebsites.net\*\*\*\*
   * **Bot template**: select the **>** arrow and then proceed with the following steps.\*\*\*\*
     1. For the **SDK Language**, select **Node.js**.
     2. Select **Echo Bot**
     3. Select **OK**.
6. Select **Create**.
7. Create Bot Framework registration resource in Azure: <https://docs.microsoft.com/en-us/azure/bot-service/bot-service-quickstart-registration>
   * Use the current https URL you were given by running ngrok. Append with the path /api/messages used by this sample
8. Update the **.env** configuration for the bot to use the Microsoft App Id and App Password from the Bot Framework registration. (Note the App Password is referred to as the “client secret” in the azure portal and you can always create a new client secret anytime.)
9. This step is specific to Teams.
   1. Edit the **manifest.json** contained in the **teamsAppManifest** folder to replace your Microsoft App Id (that was created when you registered your bot earlier) everywhere you see the place holder string **YOUR-MICROSOFT-APP-ID** (depending on the scenario the Microsoft App Id may occur multiple times in the manifest.json)
   2. Zip up the contents of the **teamsAppManifest** folder to create a manifest.zip
   3. Upload the manifest.zip to Teams (in the Apps view select **Upload a custom app**)
10. Run your bot at the command line:

* npm start

### Add registered bot to Teams channel

1. To add Microsoft Teams channel, open the registered bot in the Azure portal.
2. Select the **Channels** in the left navigation menu, and then select **Teams**.

* Channels panel in Azure portal.
* Channels panel in Azure portal.

1. Select **Save**.

* Configure MSTeams in Azure portal.
* Configure MSTeams in Azure portal.

1. After adding the Teams channel, go to the **Channels** page and select on **Get bot embed code**.

* Get bot embed codes.
* Get bot embed codes.

1. Copy the *https* part of the code that is showvin in the **Get bot embed code** dialog. For example, https://teams.microsoft.com/l/chat/0/0?users=28:b8a22302e-9303-4e54-b348-343232.
2. In the browser, paste this address and then choose the Microsoft Teams app (client or web) that you use to add the bot to Teams. You should be able to see the bot listed as a contact that you can send messages to and receives messages from in Microsoft Teams.

### Create your app manifest using App Studio

You can use the App Studio app from within the Microsoft Teams client to help create your app manifest.1. In the Teams client, open App Studio from the **…** overflow menu on the left navigation rail. If it isn’t already installed, you can do so by searching for it.

1. On the **Manifest editor** tab select **Create a new app** (or if you’re adding a messaging extension to an existing app, you can import your app package)
2. Add your app details.
3. On the **Messaging extensions** tab select the **Setup** button.
4. You can either create a new web service (bot) for your messaging extension to use, or if you’ve already registered one select/add it here.
5. If necessary, update your bot endpoint address to point to your bot. It should look something like **https://someplace.com/api/messages**.
6. The **Add** button in the **Command** section will guide you through adding commands to your messaging extension.
7. The **Message Handlers** section allows you to add a domain that your messaging will trigger on.
8. From the **Finish => Test and distribute** tab you can **Download** your app package (which includes your app manifest as well as your app icons), or **Install** the package.

### Interacting with the bot in Teams

Per the manifest, you can call this bot from the compose and message areas in Teams.

* Select the **Create Card** command from the Compose Box command list. The parameters dialog appears, and you submit it to initiate card creation in the Messaging Extension code.
* —OR—
* Select the **Share Message** command from the Message command list.

## Review

In this exercise, you:

* Created an action-based messaging extension.
* Both extensions involved editing the manifest and uploading app packages to Teams.

You use the skills learned here whenever you build a Teams app.