Microsoft OAuth 2.0 Quick Start Guide

This guide will show you how to obtain OAuth 2.0 bearer tokens (also called access tokens) for use with Microsoft 365 mail accounts. To get started, you will need the following:

1. A Microsoft Azure account which you will need to register your application. You can create this for free at https://azure.microsoft.com/en-us/free/

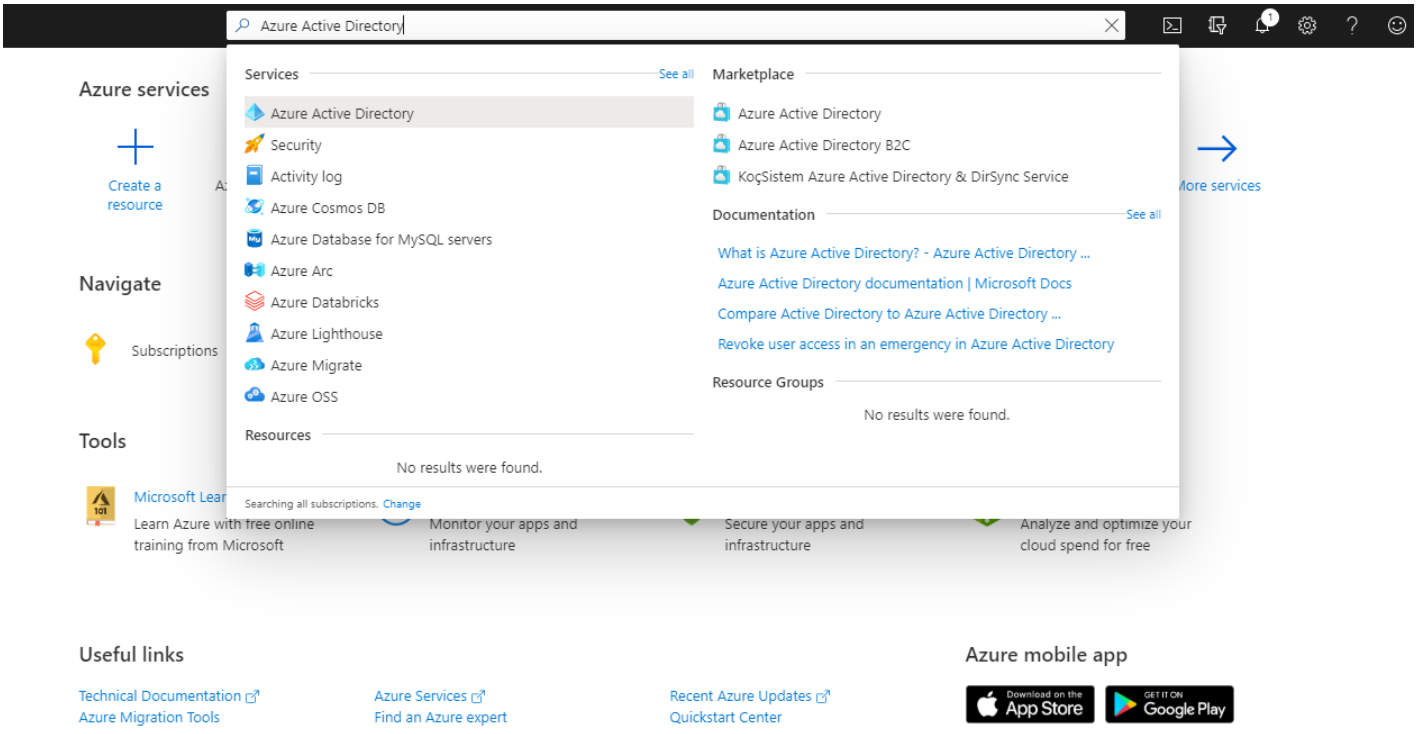
2. A Microsoft 365 account which you can use for testing purposes. You can create a free mail account at https://outlook.live.com/

3. A copy of the Postman utility which can be downloaded for free from https://www.postman.com. We will use this to compose the queries to Microsoft's servers to obtain the authorization code, bearer, and refresh tokens.

4. SocketTools 10 Build 1245 which supports authentication using OAuth 2.0 bearer tokens. If you have a current development license, this is a free update. If you have an older version of SocketTools, you will need to upgrade to the current version.

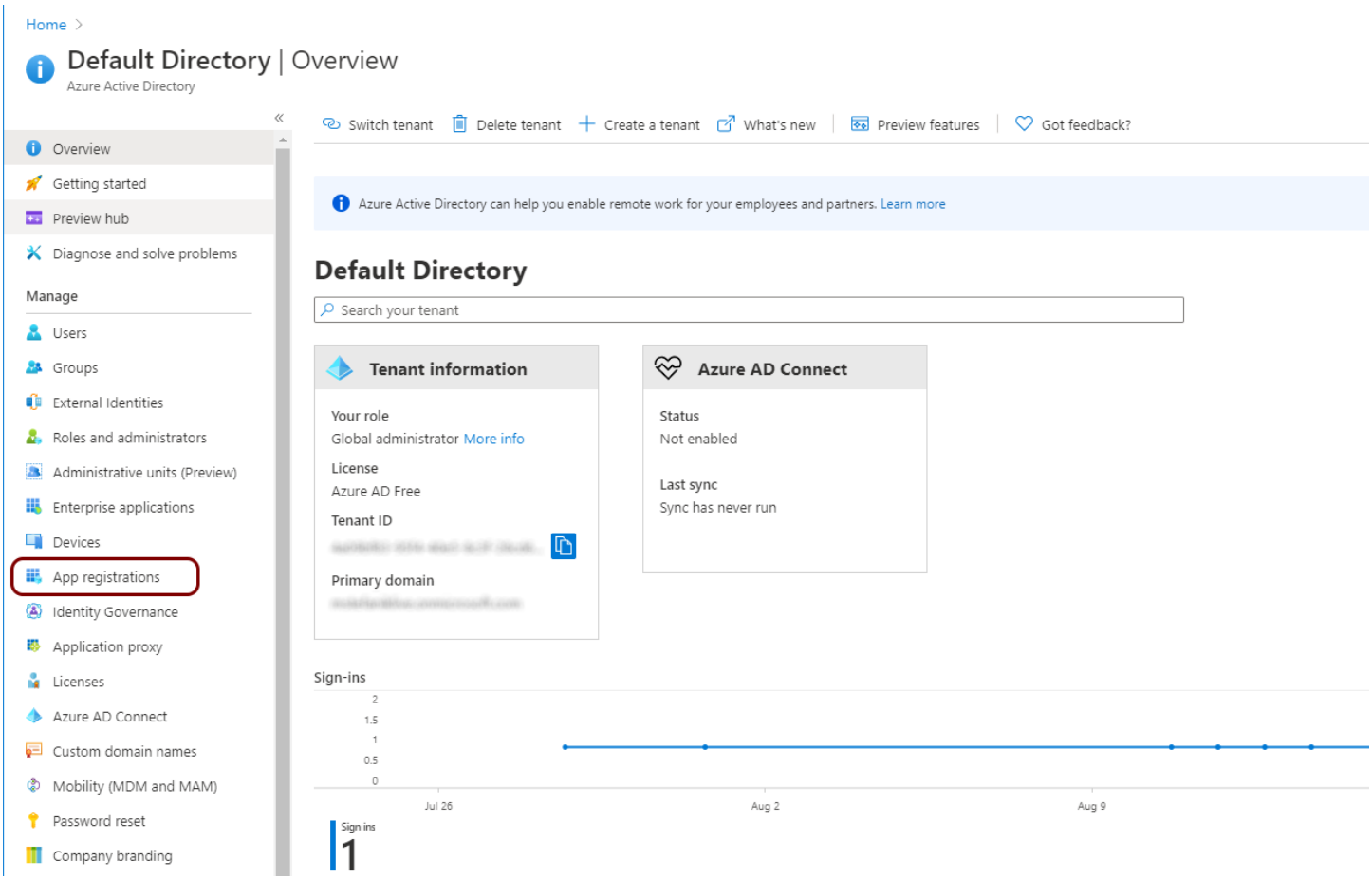
If you already have a Microsoft account, but do not have an Azure account, you can create the Azure account using your current Microsoft account. However, it's not required they be linked.

Once you have created your Microsoft Azure and Microsoft 365 accounts, the first step is to use your browser to open the Azure portal at https://azure.microsoft.com/ and login and select **Portal** from the menu.

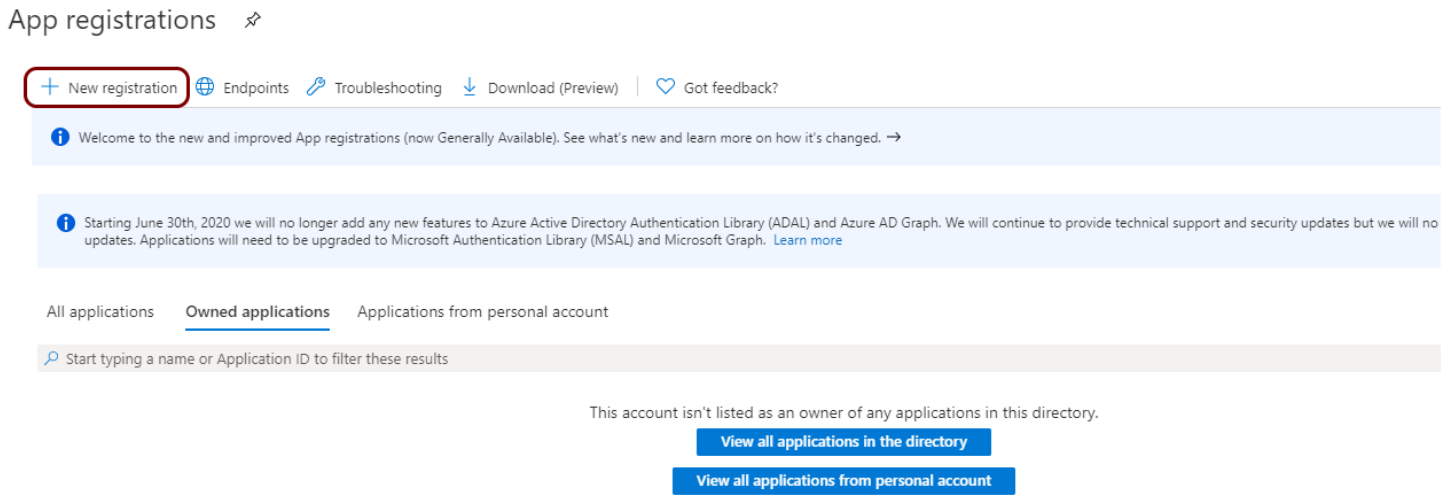


In the search bar, enter **Azure Active Directory** and select the search result.

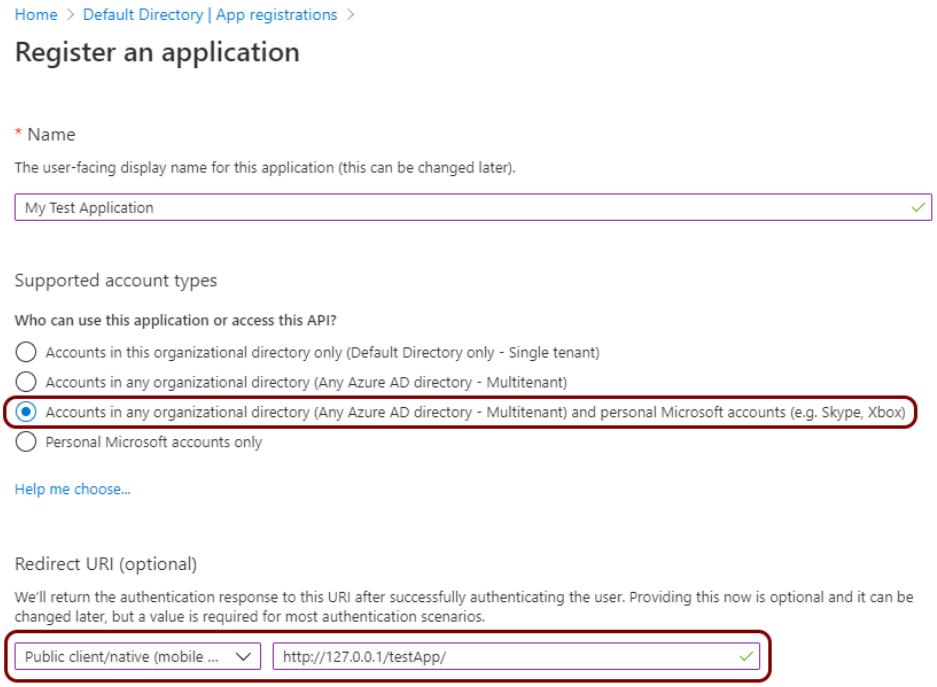
Registering Your Application



You will be in the Default Directory Overview. Select **App registrations** from the left sidebar. This is how you will register your application with Microsoft and is required to obtain your client ID.



Select **New registration** from the menu at the top. This will begin the process of registering a new application associated with your Azure account.



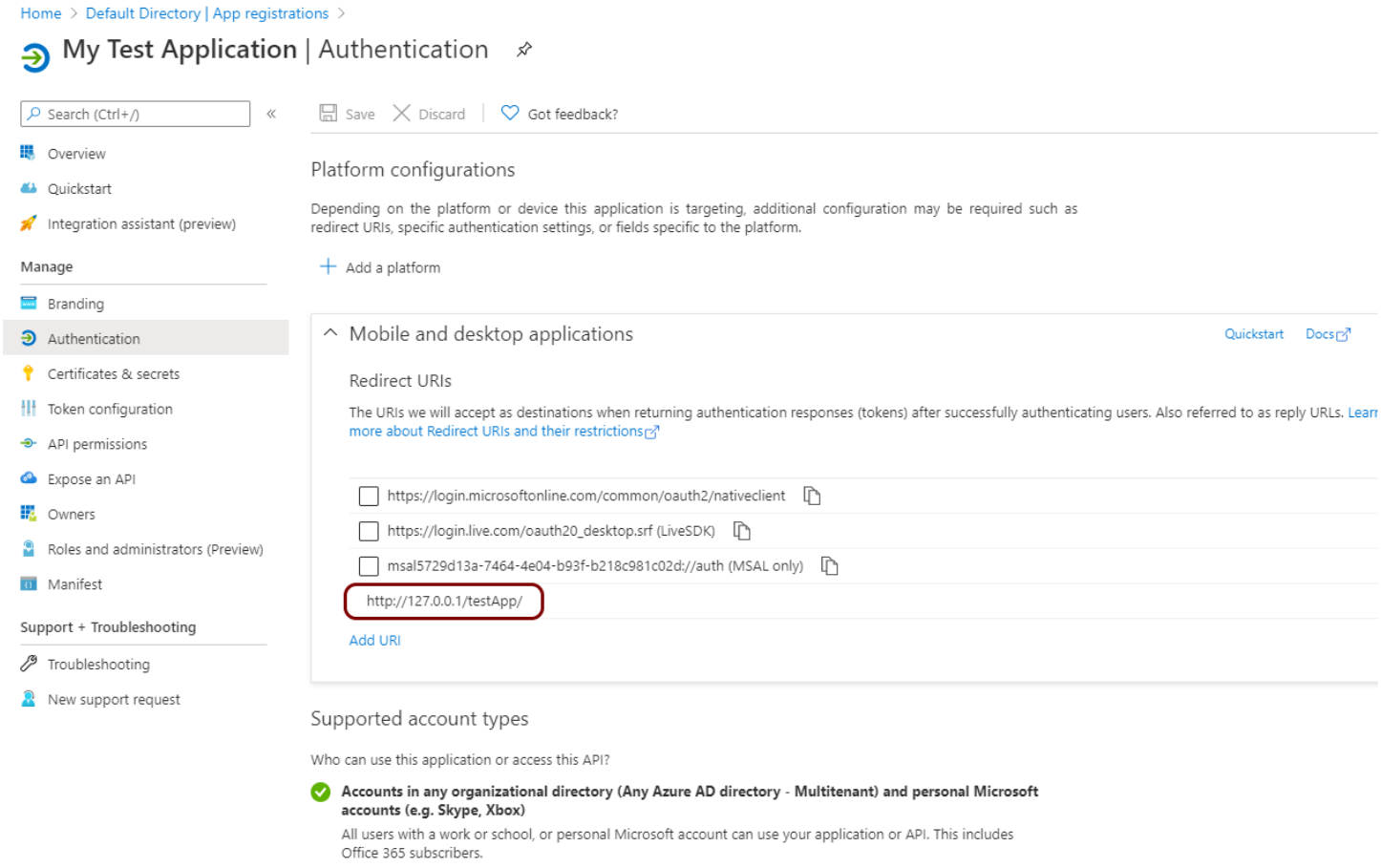
You can choose whatever name you wish for your application. This will be displayed to the user and can be changed later. For this guide, we will use “My Test Application” for the name.

For the supported account types, select **Accounts in any organizational directory**. For the redirect URI, select **Public client/native** from the dropdown list, and use a local loopback IP address for the redirect URI. This tells Microsoft you’re developing a native desktop application, and how to provide you with the authorization code needed to access the mail account.

You should be sure to use a unique address for the redirect URI. We recommend following it with an abbreviated name for your application. It needs to be a valid URI and you should keep it short and concise. For this guide, we will use the redirect URI `http://127.0.0.1/testApp/`

You may wonder if using a non-secure connection is safe. Because 127.0.0.1 is the local loopback IP address for your device, a connection will never be attempted beyond your desktop system. To prevent possible conflicts with misconfigured firewalls, we recommend using the IP address and not `http://localhost/testApp`

Once you have completed the registration of your application, Microsoft will provide you with your client ID. This is the first critical piece of information you’ll need to obtain OAuth 2.0 bearer tokens.

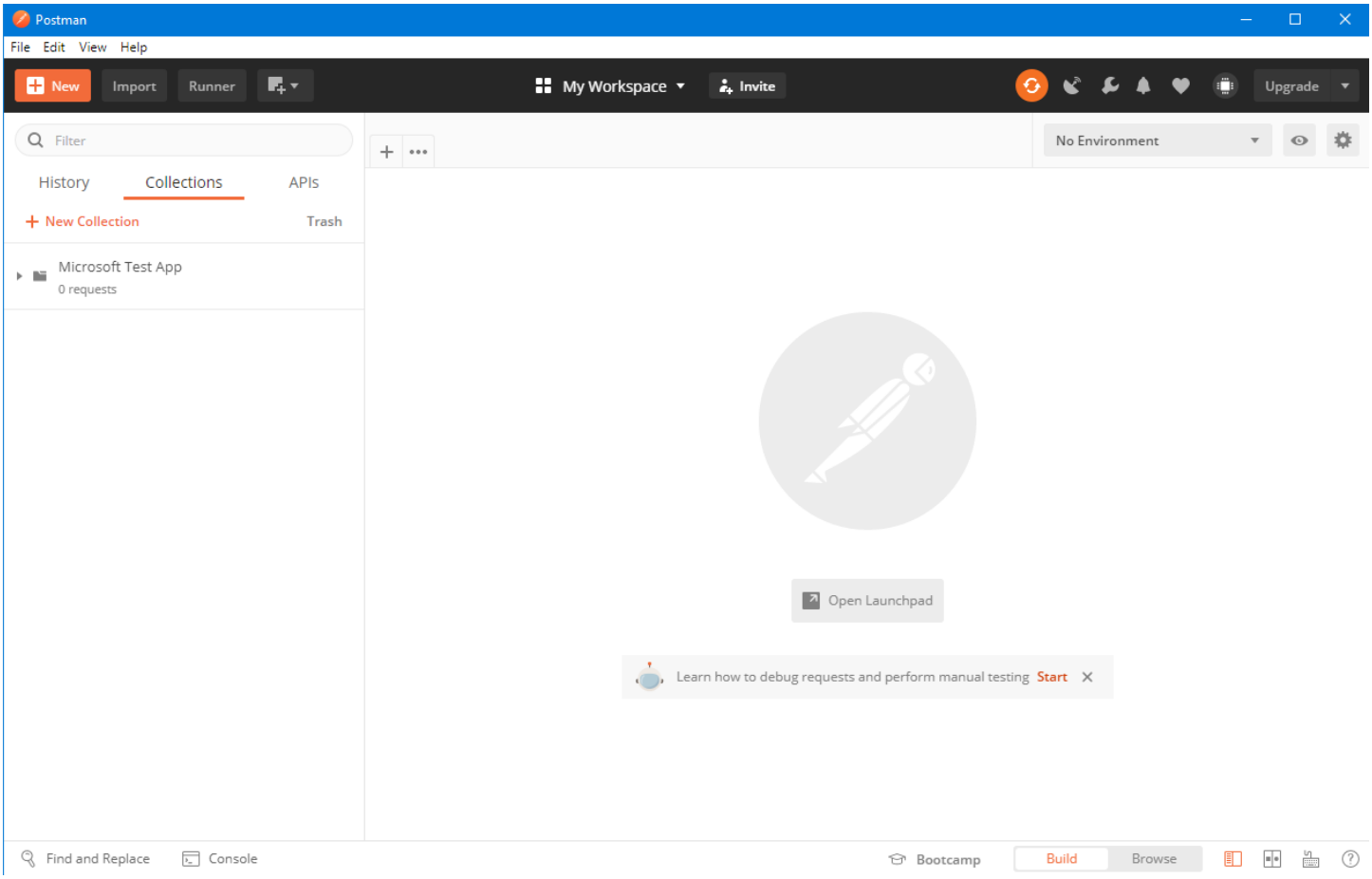


Although you can always return to this screen to get your ID, we recommend you copy down your client ID and keep in a safe place. With Microsoft, the client IDs are UUIDs and are unique to every application registered. The other IDs are not needed for our purposes.

Select **Authentication** on the left sidebar and this will display more information about your application. You can use this to confirm that the redirect URI and supported account types are correct.

Now that we've registered our application and have the client ID, the next step is to use Postman to create the queries we will send to Microsoft's servers to obtain the authorization code. And with that authorization code, we can request the bearer (access) token and refresh token which are used to access the Microsoft 365 mail account.

Creating Postman Collections



Open Postman and create a new collection by selecting **New Collection** and give it a name. For this example, we will name the collection "**Microsoft Test App**". You can also add a description of it if you wish, then press **Create**.

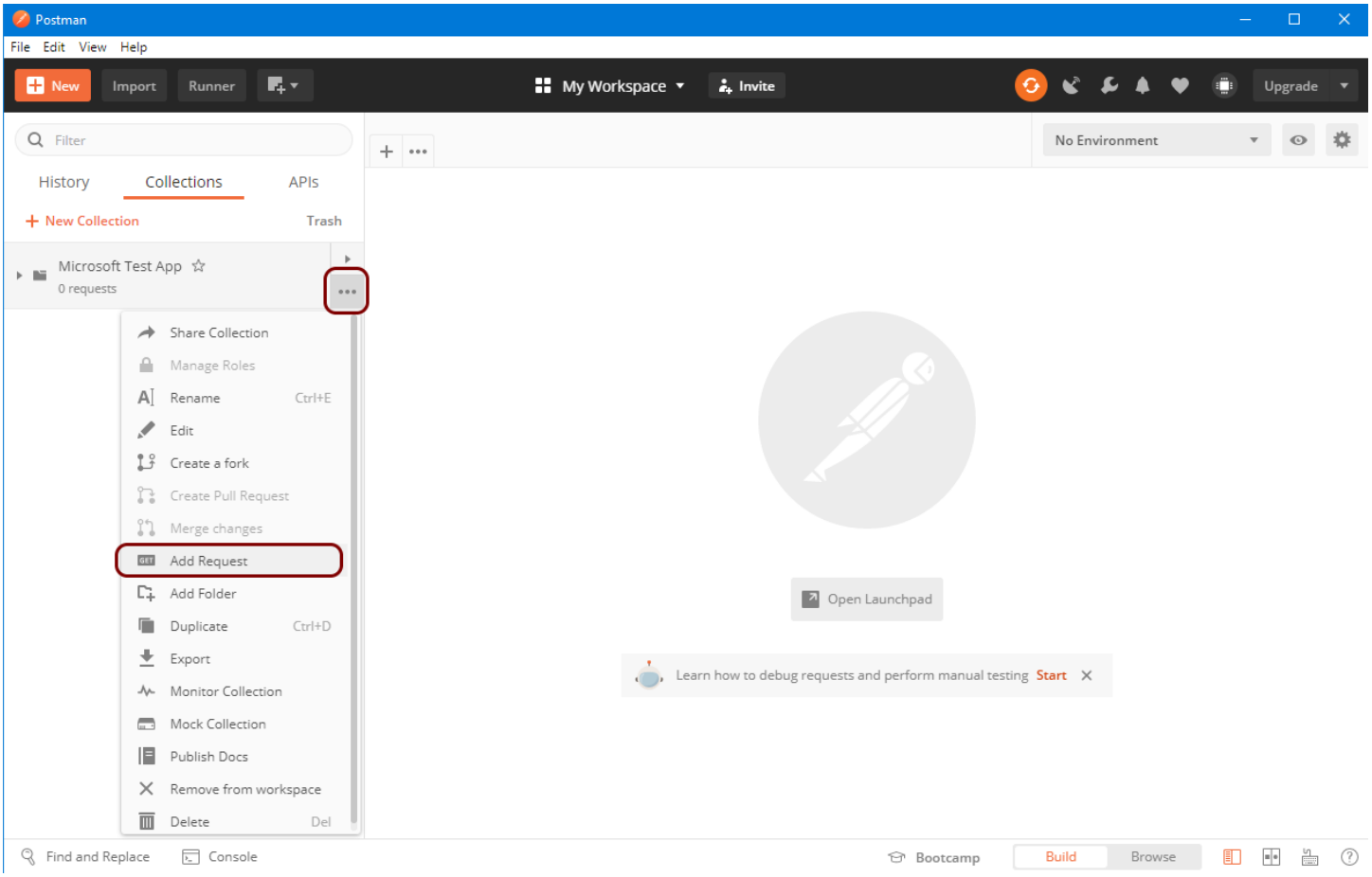
These collections are where you will store the requests we will send to Microsoft's servers. We will create the requests first, and then we'll begin the process of obtaining the tokens we need.

The three requests we'll create in Postman are:

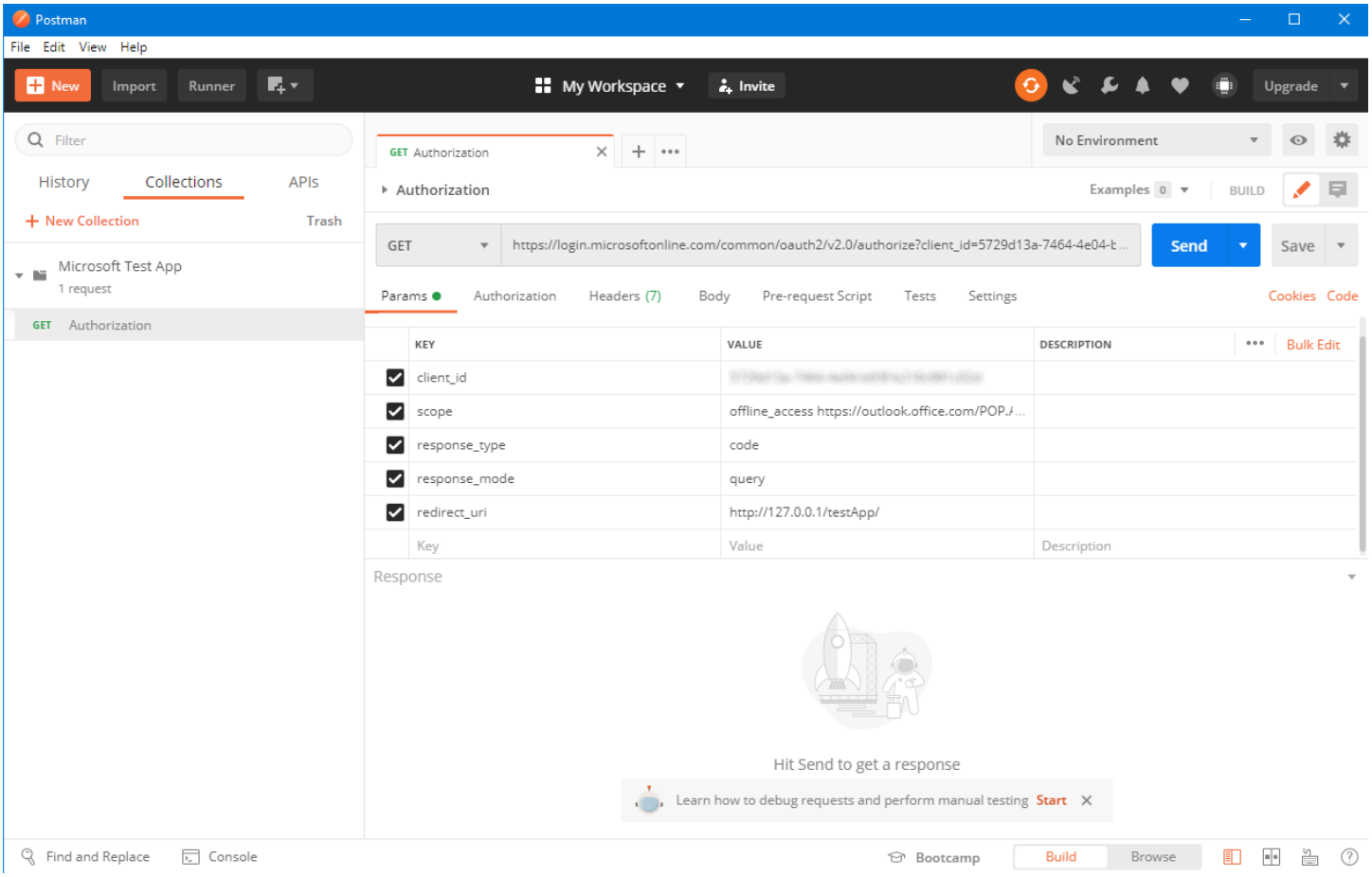
Authorization
This request is used to obtain the authorization code and is the first step towards getting the bearer token. This request is sent with GET to https://login.microsoftonline.com/common/oauth2/v2.0/authorize and the query parameter will provide your client ID and other needed information.

Request Token
This request is used to obtain the bearer and refresh tokens needed to access the user's account. This is done using a POST to https://login.microsoftonline.com/common/oauth2/v2.0/token and the post data will contain the authorization code provided in the previous step.

Refresh Token
This request is used to refresh the bearer token after it has expired. Bearer tokens have a short lifespan, typically an hour, and will expire. Refresh the token is also done using a POST to https://login.microsoftonline.com/common/oauth2/v2.0/token with the refresh token provided in the post data.



Create the first request, which is Authorization. Select **Add Request** from the menu on the left side of the display, and name it. You can also add a description if you wish.



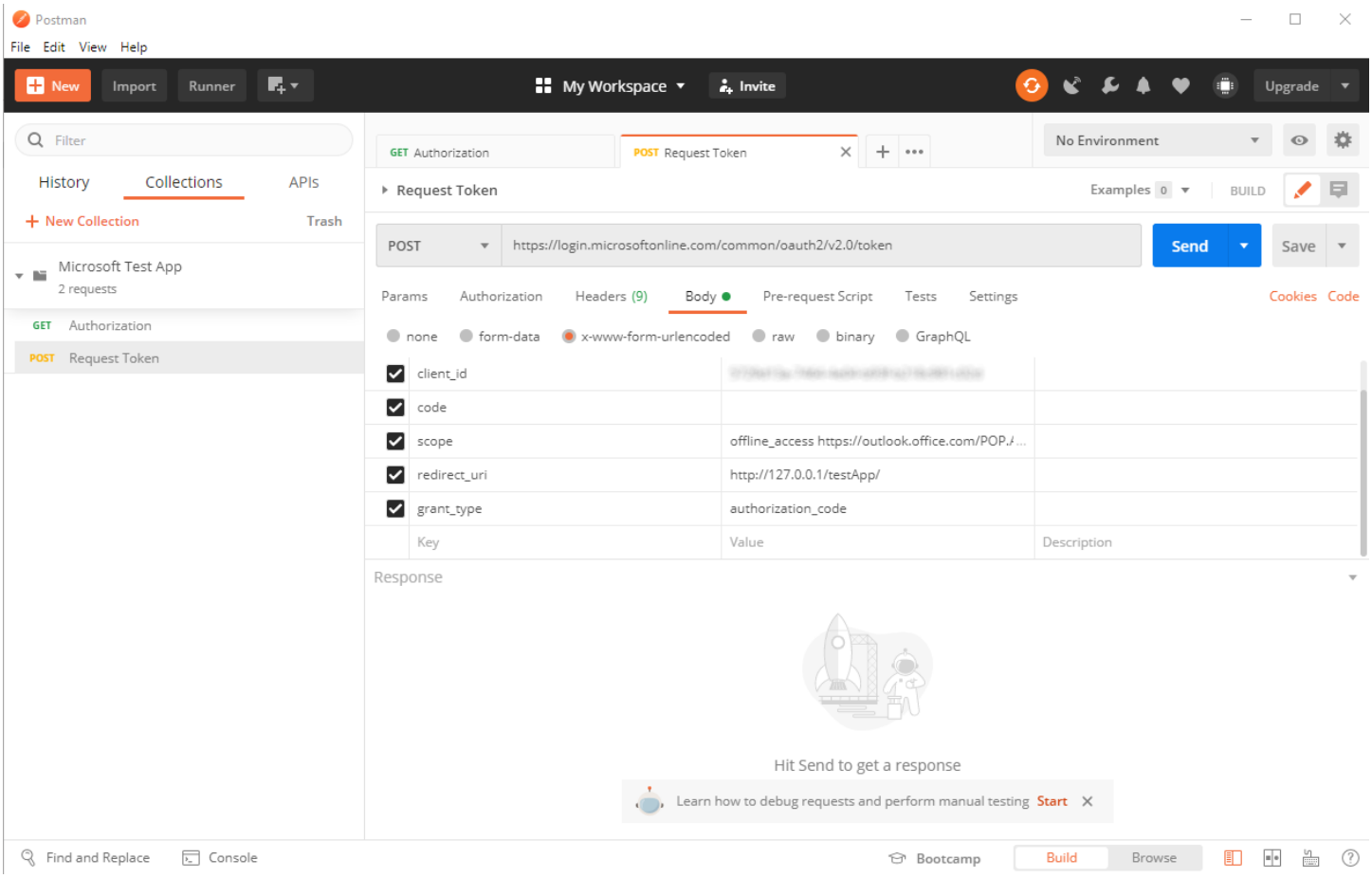
Name the request **Authorization** and it will default to using GET and enter the URL to Microsoft's authorization server `https://login.microsoftonline.com/common/oauth2/v2.0/authorize`

Populate the query parameters with five values which comprise the request.

`client_id`	The client ID you obtained when you registered your application with Azure.

`scope`	This informs Microsoft what kind of access you want to the user's mail account. In this guide, we'll request access for POP3, IMAP and SMTP. The scopes are:
	
	`offline_access`
	`https://outlook.office.com/POP.AccessAsUser.All`
	`https://outlook.office.com/IMAP.AccessAsUser.All`
	`https://outlook.office.com/SMTP.Send`
	
	Each scope should be separated by a space. We'll include **offline_access** so that Microsoft will provide us with a refresh token.

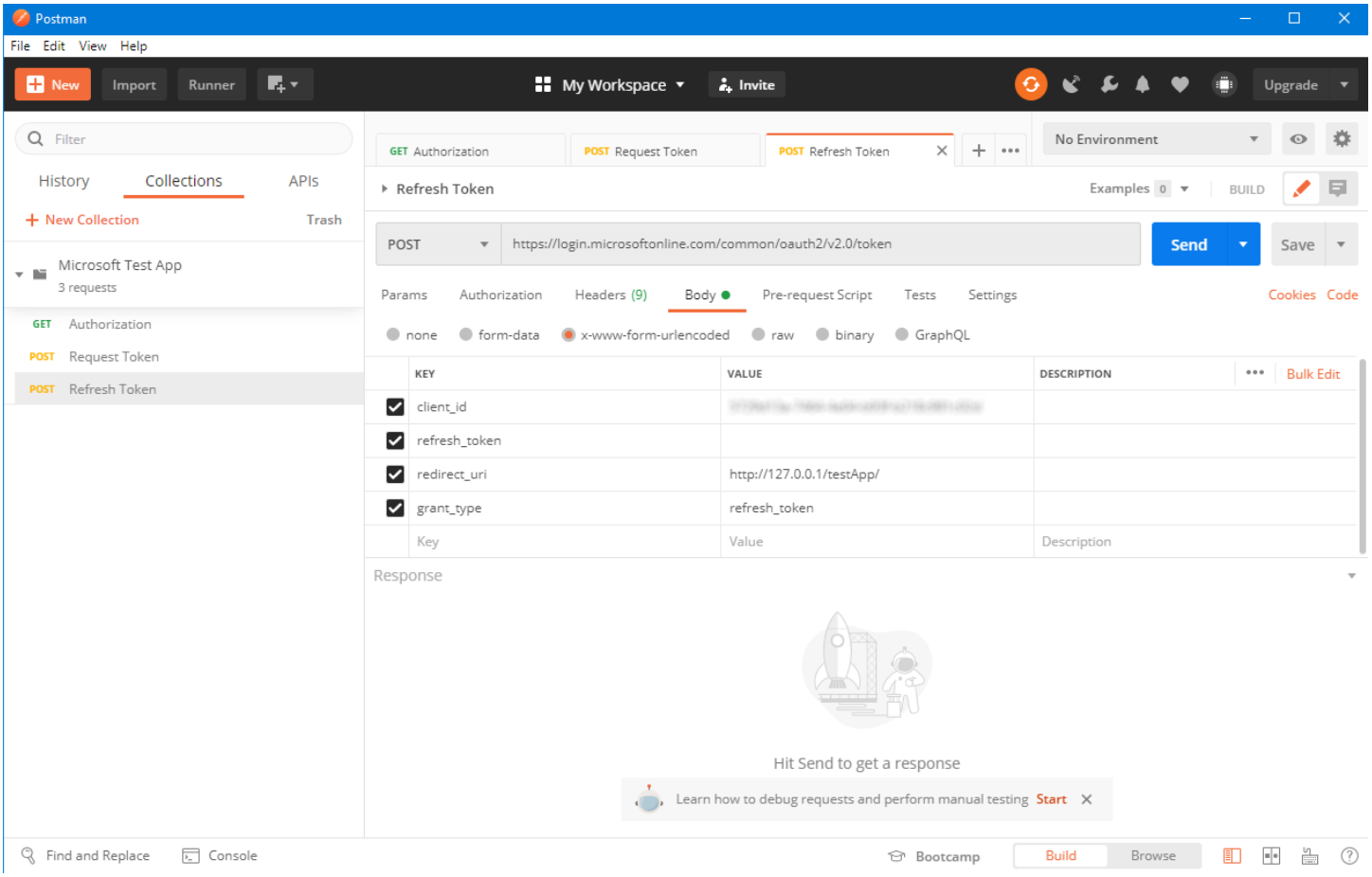
`response_type`	The response type should always be **code** for our purposes. This tells Microsoft we want an authorization code.
`response_mode`	The response mode should always be **query** which indicates we want the authorization code returned as a query in our redirect URI.
`redirect_uri`	This is the redirect URI we provided when we registered the application. This should be `http://127.0.0.1/testApp/`



Next, create the **Request Token** request which uses POST. Select **Body** from the menu, and the encoding type **x-www-form-urlencoded** as shown above. The URL is `https://login.microsoftonline.com/common/oauth2/v2.0/token`

Populate the parameters with five values which comprise the request.

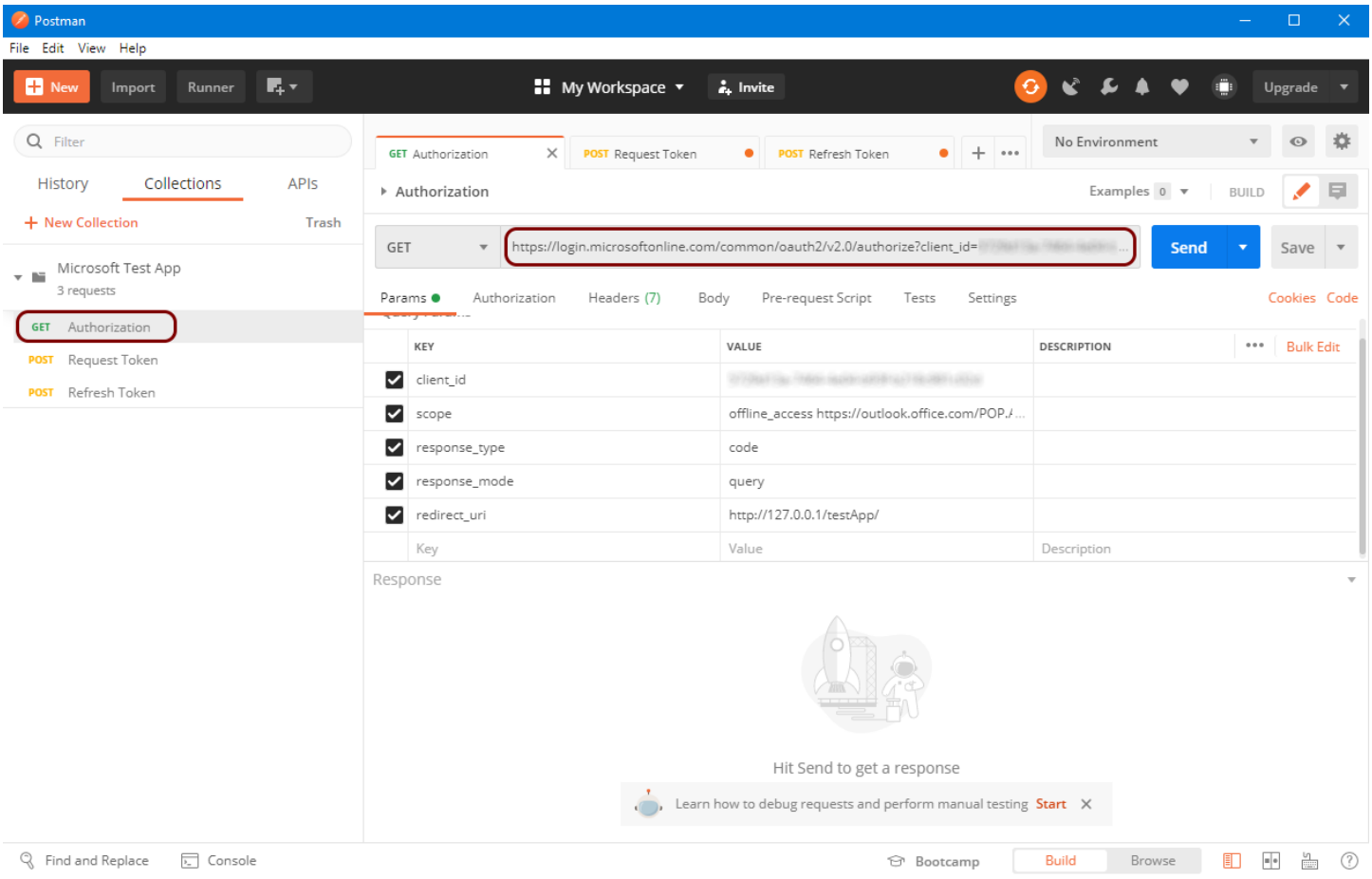
`client_id`	The client ID you obtained when you registered your application with Azure.
`code`	This is the authorization code which will be obtained using the previous request. We don't have the code yet, so leave this blank.
`scope`	This is the same list of scopes as provided with the authorization request.
`redirect_uri`	This is the redirect URI we provided when we registered the application. This should be `http://127.0.0.1/testApp/`
`grant_type`	This value should be **authorization_code** which tells Microsoft we want the bearer and refresh tokens.



The final request type will be **Refresh Token** which is used to refresh an expired bearer token. This will also use a POST to `https://login.microsoftonline.com/common/oauth2/v2.0/token`

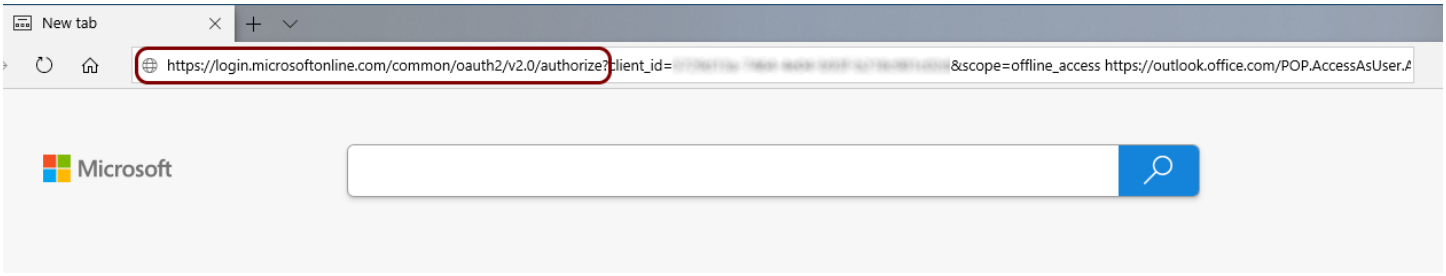
There are four values which comprise the request.

`client_id`	The client ID you obtained when you registered your application with Azure.
`refresh_token`	The refresh token you obtained when the initial authorization was granted. This token has a long validity period and is required to get a new bearer token after it has expired. We don't have the refresh token yet, so leave this blank.
`redirect_uri`	This is the redirect URI we provided when we registered the application. This should be `http://127.0.0.1/testApp/`
`grant_type`	This value should be **refresh_token** which tells Microsoft we want a new bearer token.



Now that we have created all three request types and saved them in Postman, we can use them to obtain our bearer token. Copy the URL from the **Authorization** request and paste it into a browser address bar.

You should not click **Send** in Postman, because this step requires you to login with your Microsoft 365 account and this must be done interactively in a browser.

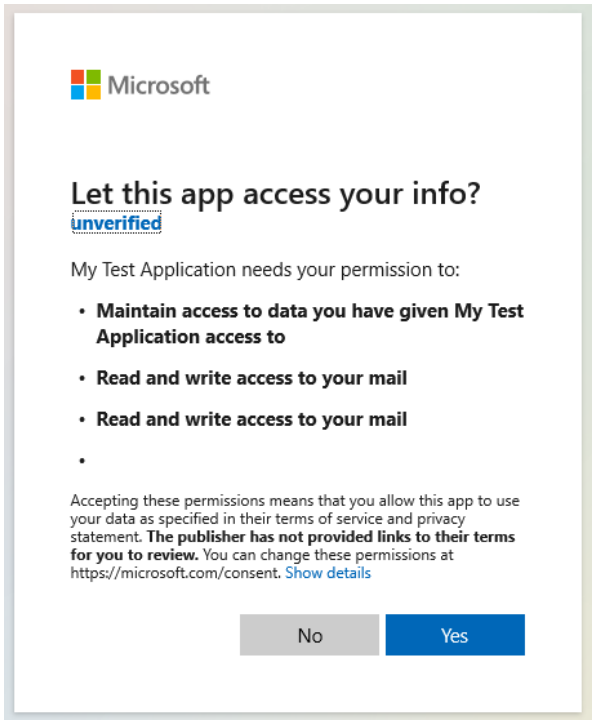


When you hit enter, you will be taken to Microsoft's site and prompted to login with your account. If you have two-factor authentication enabled, you will need to confirm the login attempt.

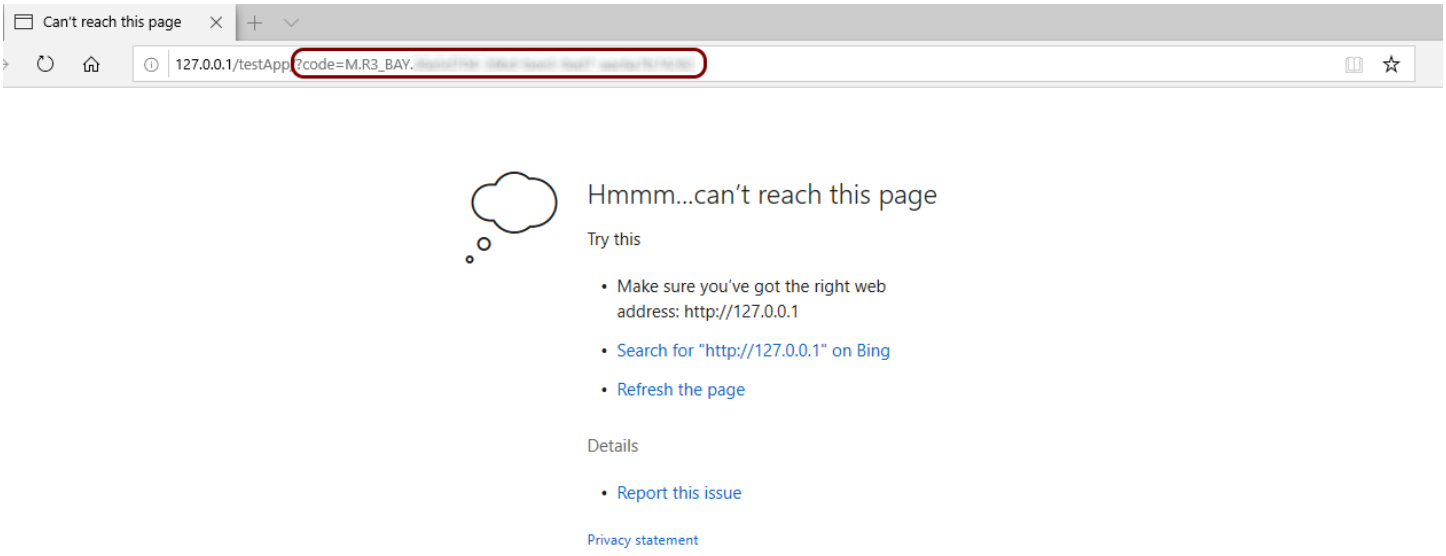
Because your application hasn't been verified, you will get a warning from Microsoft about your test application accessing your account. This is normal during the development and testing period for your application. However, when it's time for you to release your application to the public, you will need to go through the verification process.

You can learn more about publisher verification on Microsoft's website.

After you've logged in, you will be asked to confirm that you want **My Test Application** to have access to your account.



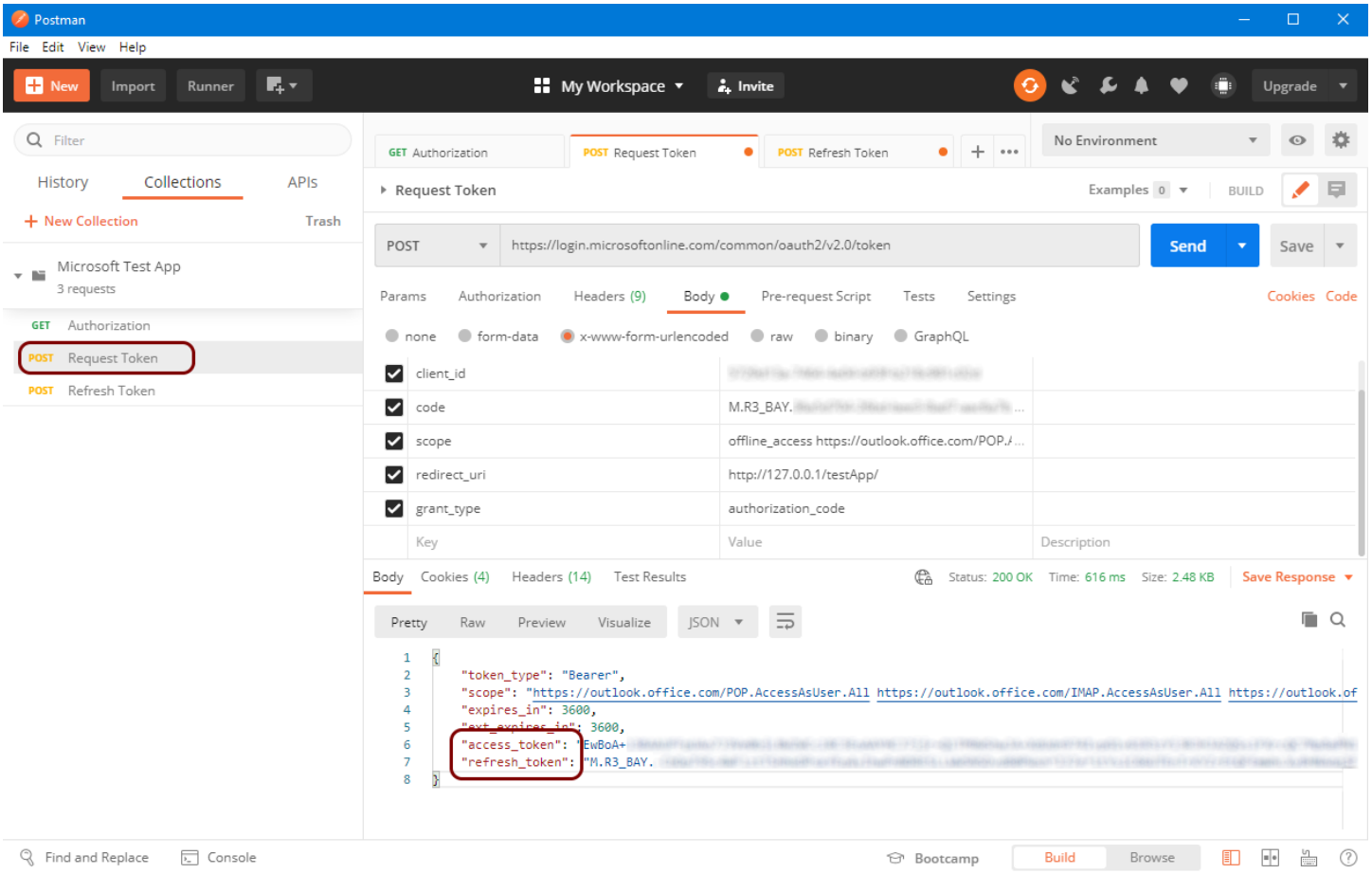
Once you've completed the login process and confirmed you want your test application to have access to the account, Microsoft will redirect back to the redirect URI which you specified when the application was registered with Azure.



Because there won't be any web server able to respond, the browser will return an error. But this isn't a problem because the information we need is returned in the address bar. You will notice the address bar now has a value which looks like your return URI:

`http://127.0.0.1/testApp/?code=[your_authorization_code]`

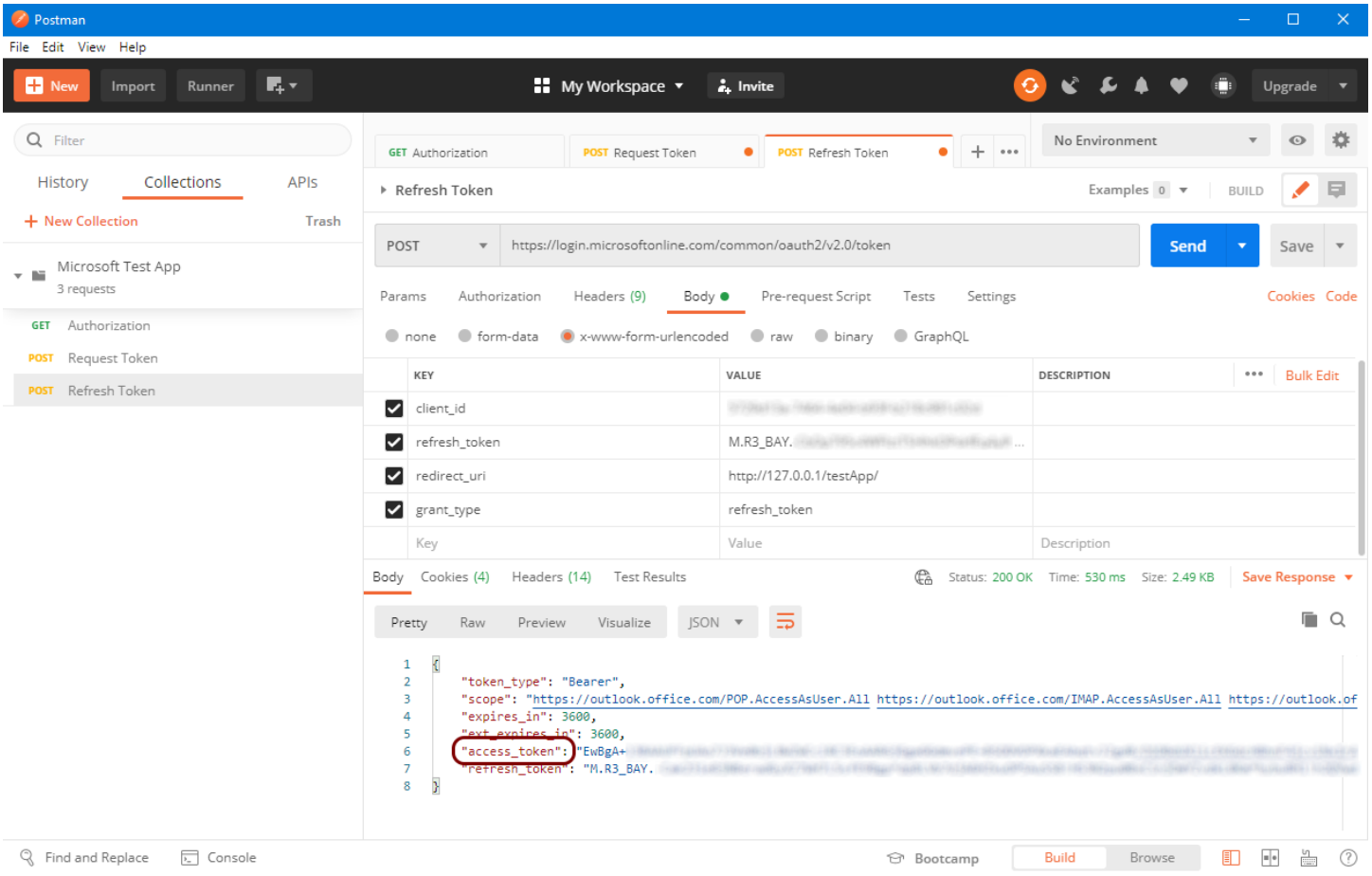
What follows the request URI after the **`?code=`** portion is the authorization code we're interested in. This is going to be a string of letters, numbers, and symbols which you should copy and paste into **Request Token** request we've created in Postman.



With the authorization code pasted into the **code** field, press Send to send the request.

If all has gone well, the server will return a JSON response which contains your bearer token (named `access_token`) and the refresh token. You will note the `expires_in` value is 3600 seconds, which is one hour. This is how long the token can be used to access the account you logged in with. After the token expires, you'll need to refresh it.

To obtain a new, fresh bearer token you will need to send another request to the server. Copy the `refresh_token` value into the **Refresh Token** request we created earlier and click **Send**.



By providing the refresh token, the server will return a new bearer token which is valid for another hour. Microsoft will also send a new refresh token as well, but you can continue to use the original refresh token which was obtained with the initial authorization request.

This completes the guide of how to register your application with Azure Active Directory and use that to obtain the bearer and refresh tokens for a Microsoft 365 account. By performing this process manually using Postman, it should help clarify the process of how the tokens are obtained and how an expired bearer token is refreshed.

If you are using SocketTools .NET or the SocketTools ActiveX controls, the mail components have a new property called **BearerToken** which you can use to specify the OAuth 2.0 bearer token in lieu of a password. For the SocketTools Library Edition, the various APIs have been extended to support bearer tokens as a valid authentication method. Refer to the technical reference documentation for more specific information about how to update your application to use OAuth 2.0 authorization.