|  |  |
| --- | --- |
|  | Azure Active Directory B2C  Module 4 Lab – Work with Graph and User Objects |
|  |  |
|  |  |

Contents

[Overview 3](#_Toc455400533)

[Learning Objectives 3](#_Toc455400534)

[1. Setup a Service Application 3](#_Toc455400535)

[2. Run local account CRUD operations 4](#_Toc455400536)

[3. Use custom attributes 5](#_Toc455400537)

# Overview

During this lab, you will run several exercises that will help you understand the programmatic access features available via Azure AD Graph API in Azure AD B2C.

**Estimated time to complete this lab: 30 minutes**.

# Learning Objectives

After completing the exercises in this lab, you will be able to:

* Understand how Azure AD Graph API allows you to perform CRUD (Create-Read-Update-Delete) operations on local accounts in your B2C tenant
* Understand how to use a service application to do programmatic, automated tasks in your B2C applications
* Understand the use of custom attributes (directory extensions) in your B2C tenant

## Setup a Service Application

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Download tools | 1. Download and install the Microsoft Online Services Sign-In Assistant from **http://go.microsoft.com/fwlink/?LinkID=286152** 2. Download and install the Azure Active Directory module for Windows PowerShell from **http://go.microsoft.com/fwlink/p/?linkid=236297** |
| Create a service application for use with Azure AD Graph API and give it the right permissions | 1. Run **PowerShell** 2. Connect to your B2C tenant using the following commands using your **admin@<B2CDirName>.onmicrosoft.com** – to ensure that this is created in your B2C directory and not your subscription directory:   $msolcred = Get-Credential  Connect-MsolService -credential $msolcred   1. Create a Client Secret to use with your application   $bytes = New-Object Byte[] 32  $rand = [System.Security.Cryptography.RandomNumberGenerator]::Create()  $rand.GetBytes($bytes)  $rand.Dispose()  $newClientSecret = [System.Convert]::ToBase64String($bytes)  $newClientSecret   1. Copy down the Client Secret that is shown on screen and save it in your <**B2CDirName>.txt** file on your desktop 2. Create your service application as follows:   New-MsolServicePrincipal -DisplayName "My B2C Graph API App" -Type password -Value $newClientSecret   1. Copy both the **ObjectID** and **AppPrincipalID** and save it in your <**B2CDirName>.txt** file on your desktop 2. List all directory roles using the following command:   Get-MsolRole   1. Copy down ObjectIDs of the following 3 directory roles and save it in your <**B2CDirName>.txt** file:    1. **Directory readers** (to read users)    2. **Directory writers** (to create and update users)    3. **User account administrator** (to delete users) 2. Replace the ObjectIDs with the ones you just noted down and replace the Application ID in the three commands below (which adds the service application to these 3 directory roles):   Add-MsolRoleMember -RoleObjectId <ObjectID1> -RoleMemberObjectId <Your-App’s-ObjectId> -RoleMemberType servicePrincipal  Add-MsolRoleMember -RoleObjectId <ObjectID2> -RoleMemberObjectId <Your-App’s-ObjectId> -RoleMemberType servicePrincipal  Add-MsolRoleMember -RoleObjectId <ObjectID3> -RoleMemberObjectId <Your-App’s-ObjectId> -RoleMemberType servicePrincipal   1. You now have a service application ready to use with Azure AD Graph API in your B2C environment |
| Download and build B2C sample command line app. (This will enable you to build your own C# Graph API applications outside of this course) | 1. Download the B2C sample command line app on GitHub at **https://github.com/AzureADQuickStarts/B2C-GraphAPI-DotNet** 2. Extract/copy all the files to the **c:\B2C\** folder 3. Open the **B2CGraphClient.sln** in Visual Studio 4. In the **B2CGraphClient** project, open the file **App.config** 5. Replace the following app settings with your own values saved in the <B2cDirName>.txt file:   <appSettings>      <add key="b2c:Tenant" value="<Enter your tenant name, e.g. contoso.onmicrosoft.com>" />      <add key="b2c:ClientId" value="<Enter the client ID (a.k.a AppPrincipalId) as obtained from the Azure AD Powershell, e.g. 82692da5-a86f-44c9-9d53-2f88d52b478b>" />      <add key="b2c:ClientSecret" value="<Enter the client secret that you generated, e.g. ONHJGaI232VenJIboyg8hmTlyNXh0Ef0brRRRNWBRfc=>" />    </appSettings>   1. Right-click on the **B2CGraphClient** solution and **rebuild** the solution 2. To test the sample app, from a command line enter the following:   cd B2CGraphClient\bin\Debug  B2C Help  **Note:** You can see the commands for CRUD operations that we will use in the next exercise. |

## Run local account CRUD operations

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Get all users in your B2C tenant | 1. Run the following command:   B2C Get-User |
| Create new username-based and email-based local accounts in your B2C tenant | 1. Run the following commands, saving the Object ID’s of each one in your **<B2CDirName>.txt**:   B2C Create-User ..\..\..\usertemplate-email.json  B2C Create-User ..\..\..\usertemplate-username.json   1. **Open** up both the .json files and inspect the contents 2. Run the **B2C Get-User** command to see the new local accounts you just created |
| Search for specific users in your B2C tenant | 1. Run the following commands (use the object ID of one of the local accounts created in the previous step):   B2C Get-User <Object-ID-here>  B2C Get-User $filter=signInNames/any(x:x/value%20eq%20%27joeconsumer@gmail.com%27) |

## Use custom attributes

|  |  |
| --- | --- |
| Task | Detailed Steps |
| Return custom attributes in your B2C application and update them | 1. Run the following command:   B2C Get-B2C-Application   1. Copy the **objectId** that was returned and then run:   B2C Get-Extension-Attribute <insert-object-id-from-above>   1. Copy the name of the Gamertag attribute (extension\_GUID\_Gamertag) 2. Update one of your .json files with (only) the new property and a value for the property to be updated:   {  “extension\_GUID-as-above\_Gamertag”: “Superstar”  }   1. Save it as **updateusertemplate-username.json** 2. Run the command:   B2C Update-User <object-id-of-user> ..\..\..\updateusertemplate-username.json  **Note:** We have programmatically found the Object ID of an application, queried it to find the metadata for any extension attributes and updated one of those extension attributes. |