ADAL and Azure AD Graph

This guide provides setup requirements and steps to demonstrate how to use the Active Directory Authentication Library (ADAL) to obtain an Access Token from Azure AD. It also shows how to use the Azure AD Graph Client Library to programmatically create a user in Azure AD.

Contents

[Pre-Requisites 1](#_Toc430178321)

[Setup 1](#_Toc430178322)

[Demo Steps 1](#_Toc430178323)

[Clean Up 3](#_Toc430178324)

## Pre-Requisites

Complete part A of this demo first.

## Setup

None

## Demo Steps

Estimated time: 10 minutes

|  |  |
| --- | --- |
| 1. Switch over to Visual Studio. 2. Insert snippet **DAL\_Demo\_4\_Add\_AcquireTokenAsyncForUser** under the designated comment. (**Ctrl-K,X**). 3. Insert snippet **DAL\_Demo\_5\_Call\_AcquireTokenAsyncForUser** under the designated comment. (**Ctrl-K,X**).   Explain: To access the Azure AD Graph using the Azure AD Graph Client Library, we need to instantiate an ActiveDirectoryClient. The constructor needs to know the resource (Azure AD Graph) and the tenant the resource belongs to. It also, expects a function delegate it can call to get a token. For this demo, we’re just re-using the function shown in part A of this demo to get a token. The reason we wrap the call to GetTokenForUser in a new method is because the signature for the function delegate want’s an async Task<string> signature.  After we’ve acquired a token to access the Azure AD Graph we will attempt to add a user to the directory, which is explained in the following steps. |  |
| 1. Insert snippet **DAL\_Demo\_6\_Add\_CreateUser** under the designated comment. (**Ctrl-K,X**). 2. Explain the code in the **CreateUser** method. 3. Press **Ctrl-F5** to build and run the solution. 4. When prompted, sign-in as **john@azurecoe01.onmicrosoft.com** with password **P@ssword1**. 5. Point out the exception message that occurred due to insufficient privileges. Recall that earlier you demonstrated adding a ***delegated*** permission to read and write directory data. This means that in this scenario, the operation will be performed under the context of the authenticated user which in this case is John Doe. Even though you configured the app in Azure AD to add a delegated permission to do this, it doesn’t mean that the ***user*** has permission to do this operation. Recall that John Doe is just a user in the directory. Only Global Administrators have sufficient permissions to manipulate the directory. The Clark Griswold user is a Global Administrator though. 6. Press **Ctrl-F5** to build and run the solution. 7. When prompted, sign-in as **clark@azurecoe01.onmicrosoft.com** with password **P@ssword1**. 8. Note that this time the operation was successful. |  |
| 1. Switch over to the Azure Management Portal. 2. Go to the **USERS** page for the **Azure CoE** directory. 3. Show that user **Jay Hamlin** has been added to the directory.   Note: When the user write-back feature is added back to AAD Connect, you can show this user is written back to the on-premises directory as well since you’re using the same directory as was used in the AAD Connect demos. Currently this is not supported though but you could mention this will soon be a capability added back to AAD Connect.  Optional to continue. If time is available and you want to show the TokenCache in ADAL then continue. Otherwise, stop here. |  |
| 1. Switch over to Visual Studio. 2. Revisit the code in the Main method and point out that the code calls **AuthenticationContext.AcquireToken()** twice in its code path. However, as demonstrated, you were only getting prompted for credentials once. The reason for this behavior is that the ADAL library caches tokens for you and will retrieve the token from the cache if it is still valid. This is one of the benefits to using ADAL as compared to using the REST API’s. 3. Set a breakpoint on the call to AquireToken in the GetTokenForUser method. |  |
| 1. Press **F5** to run the application under the debugger. 2. When the breakpoint is hit, expand the **authnContext > TokenCache** property to show that there are no tokens in the cache. 3. Press **F5** to continue. |  |
| 1. When prompted, sign-in as **john@azurecoe01.onmicrosoft.com** with password **P@ssword1**. 2. When the breakpoint is hit again, expand the **authnContext > TokenCache** property to show that there is a token in the cache. |  |
| 1. To show what is in the cache, open the Immediate window (Ctrl+D,I) and type in **authnContext.TokenCache.ReadItems().ToList()[0]** |  |
|  |  |

## Clean Up

To clean up after this demo perform the following steps:

1. In the Azure Management Portal, delete the application that was registered under the name **ADAL Graph Demo**.
2. Close Visual Studio.
3. Delete the folder **c:\azurecoe\demos\identity\ 4\_ADAL\_Graph**.