Microsoft APIs for CSP

Hands-on Lab 2

# Hands-on Lab 2: Assign user to azure subscription with special role and query usage records via Azure Management APIs & Azure Graph APIs & Partner Center APIs

## Exercise 1: Config your application based Hands-on Lab1 demo

In this exercise, you will continue to write the demo based on Hands-on lab 1.

1: Add settings in App.config file and SettingsHelper.cs ,like Azure Graph APIs endpoint,

Azure Management APIs endpoint.

2: Add common HttpUtils

## Exercise 2: Assign user to azure subscription with special role via Azure Graph APIs & Azure Management APIs

In this exercise, you will try to get the user object id via Azure graph APIs and get the role definition id from azure Management APIs. Then you can try to assign the user with special role to the azure subscription.

## Exercise 3: Query usage records via Partner Center SDK & Azure Management APIs

In this exercise, you will try to query the usage of azure subscription via Partner Center SDK. At the same time, you will try to query the resources tags, resource group based on the resource Uri in usage records via Azure management APIs.

## Prerequisites

To ensure the native app you created in Hands-on Lab 1 to own the permission to

manage customer’s azure subscription, you will need to add the app into the Adminagents group via PowerShell

### Log in with Global administrators and add your native app into the Adminagents group via PowerShell

PS C:\windows\system32> Connect-AzureAd

PS C:\windows\system32>$group = Get-AzureADGroup -Filter "displayName eq 'Adminagents'"

PS C:\windows\system32> $sp = Get-AzureADServicePrincipal -Filter "appId eq '<your client id>'"

PS C:\windows\system32> Add-AzureADGroupMember -ObjectId $group.ObjectId -RefObjectId $sp.ObjectId

Now your app will have the permission to invoke the Azure Graph APIs &Azure Management APIs, later you will need to use them to assign user to azure subscription and do other operations.

## Exercise 1: Config your application based Hands-on Lab1 demo

In this exercise, you will add some settings in App.config and SettingsHelper.cs. Due to you will use the Azure Graph APIs, Azure management APIs directly, the common HttpUtils is necessary

### Add Azure Graph APIs endpoint and Azure management APIs endpoint in App.config file

1. Open the App.config file and add below settings

<add key="azure-resoure-management-endpoint" value="https://management.azure.com" />

<add key="azure-resoure-management-resource" value="https://management.core.windows.net/" />

<add key="azure-ad-graph-endpoint" value="https://graph.windows.net" />

<add key="azure-ad-graph-resource" value="https://graph.windows.net/" />

1. Open *SettingHelper.cs* and add below codes

public static string AzureRMApiEndpoint

{

get { return ConfigurationManager.AppSettings["azure-resoure-management-endpoint"]; }

}

public static string AzureADGraphApiEndpoint

{

get { return ConfigurationManager.AppSettings["azure-ad-graph-endpoint"]; }

}

public static string AzureRMApiResourceId

{

get { return ConfigurationManager.AppSettings["azure-resoure-management-resource"]; }

}

public static string AzureADGraphApiResourceId

{

get { return ConfigurationManager.AppSettings["azure-ad-graph-resource"]; }

}

### Create Utils Folder& Add HttpUtils.cs in Utils folder

1. Solution>Right Click>Add>New Folder>>Change the folder name “***Utils***”
2. ***Utils*** folder >Right click > Add>Class, here is the codes:

using Newtonsoft.Json.Linq;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

namespace MyDemos.Utils

{

public class HttpUtils

{

/// <summary>

/// Http Get

/// </summary>

/// <param name="url"></param>

/// <param name="headers">headers</param>

/// <returns></returns>

public static JObject DoGet(string url, Dictionary<string, string> headers)

{

//Console.WriteLine("url:" + url);

var request = WebRequest.Create(url);

if (null != headers)

{

//Console.WriteLine("headers start");

foreach (string key in headers.Keys)

{

//Console.WriteLine(key+":"+ headers[key]);

request.Headers.Add(key, headers[key]);

}

//Console.WriteLine("headers end");

}

request.Method = "GET";

try

{

var response = request.GetResponse();

using (var reader = new StreamReader(response.GetResponseStream()))

{

var responseContent = reader.ReadToEnd();

//Console.WriteLine("responseContent:" + responseContent);

var retJson = Newtonsoft.Json.JsonConvert.DeserializeObject<JObject>(responseContent);

return retJson;

}

}

catch (WebException webException)

{

if (webException.Response != null)

{

using (var reader = new StreamReader(webException.Response.GetResponseStream()))

{

var responseContent = reader.ReadToEnd();

//Console.WriteLine("responseContent:" + responseContent);

}

}

}

return null;

}

/// <summary>

/// Http Put

/// </summary>

/// <param name="url"></param>

/// <param name="headers">headers</param>

/// <param name="content">the json content to put</param>

/// <returns></returns>

public static JObject DoPut(string url, Dictionary<string, string> headers,string content)

{

//Console.WriteLine("url:" + url);

var request = WebRequest.Create(url);

request.ContentType = "application/json";

if (null != headers)

{

//Console.WriteLine("headers start");

foreach (string key in headers.Keys)

{

//Console.WriteLine(key + ":" + headers[key]);

request.Headers.Add(key, headers[key]);

}

//Console.WriteLine("headers end");

}

request.Method = "PUT";

if (!string.IsNullOrEmpty(content))

{

//Console.WriteLine("content:" + content);

using (var writer = new StreamWriter(request.GetRequestStream()))

{

writer.Write(content);

}

}

try

{

var response = request.GetResponse();

using (var reader = new StreamReader(response.GetResponseStream()))

{

var responseContent = reader.ReadToEnd();

//Console.WriteLine("responseContent:" + responseContent);

var retJson =

Newtonsoft.Json.JsonConvert.DeserializeObject<JObject>(responseContent);

return retJson;

}

}

catch (WebException webException)

{

if (webException.Response != null)

{

using (var reader = new StreamReader(webException.Response.GetResponseStream()))

{

var responseContent = reader.ReadToEnd();

//Console.WriteLine("responseContent:" + responseContent);

}

}

}

return null;

}

}

}

## Exercise 2: Assign user to azure subscription with special role via Azure Graph APIs&Azure Management APIs

### Query customer object id from Azure Graph APIs

You will try to query the user object id via Azure Graph APIs, you can use the QueryCusotmer in hands-on Lab 1 to get the customer id and azure subscription id

1. Solutions>Right Click>Add>New Folder >> Change the folder name “Services”
2. Services folder>Right Click >Add>Class>>add UsersService.cs

using MyDemos.Utils;

using Newtonsoft.Json.Linq;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MyDemos.Services

{

public class UsersService

{

/// <summary>

/// Query user information from azure graph

/// </summary>

/// <param name="aadToken">AAD access token for azure graph APIs</param>

/// <param name="filter">the filer expression </param>

/// <returns></returns>

public static JObject List(string tenantId,string aadToken,string filter)

{

string requestUrl = string.Format("{0}/{1}/users?api-version=1.6&$filter={2}", SettingsHelper.AzureADGraphApiEndpoint, tenantId, filter);

Dictionary<string, string> headers = new Dictionary<string, string>();

headers.Add("Authorization", string.Format("{0}{1}", "Bearer ", aadToken));

return HttpUtils.DoGet(requestUrl, headers);

}

}

}

1. Open Program.cs, add below codes.

To invoke the Azure graph APIs, you need to get the access token for azure graph APIs from azure AD first. Then you can invoke the /users with the

AccessToken in http headers.

/// <summary>

/// Use Azure AD Graph API to query user's Object ID

/// </summary>

/// <param name="mail">user's email </param>

/// <returns>user's Object ID</returns>

private static string QueryObjectIdByMail(string tenantId,string mail)

{

string tokenADGraph = Program.LoginToAad(tenantId, SettingsHelper.AzureADGraphApiResourceId).Result.AccessToken;

String filter = HttpUtility.UrlEncode(string.Format("mail eq '{0}' or userPrincipalName eq '{1}'", mail, mail));

JObject retJson = UsersService.List(tenantId,tokenADGraph, filter);

return (string)retJson["value"][0]["objectId"];

}

1. You can use below codes to get user’s object id

string objectId = QueryObjectIdByMail(“tenant id /azure ad directory id/ customer id”,”user’s email address(the email in the same directory),like admin@<domainPrefix>.onmicrosoft.com”);

### Query role definition id from Azure Mangement APIs

1. Services folder>Right Click >Add>Class>>add RoleDefinitionsService.cs

using MyDemos.Utils;

using Newtonsoft.Json.Linq;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MyDemos.Services

{

public class RoleDefinitionsService

{

/// <summary>

/// Query role definition id via role name

/// </summary>

/// <param name="aadToken">AAD access token for azure management apis</param>

/// <param name="subscriptionId">the azure subscription id</param>

/// <param name="filter">the filter expression</param>

/// <returns></returns>

public static JObject List(string aadToken, string subscriptionId, string filter)

{

string requestUrl = string.Format("{0}/subscriptions/{1}/providers/Microsoft.Authorization/roleDefinitions?api-version=2017-05-01&$filter={2}", SettingsHelper.AzureRMApiEndpoint, subscriptionId, filter);

Dictionary<string, string> headers = new Dictionary<string, string>();

headers.Add("Authorization", string.Format("{0}{1}", "Bearer ", aadToken));

return HttpUtils.DoGet(requestUrl, headers);

}

}

}

1. Open Program.cs and add below codes

Before you involve the azure management APIs, you should get the access token from azure ad for the azure management APIs first. Then you can invoke the */subscriptions/{your azure subscription}/providers/Microsoft.Authorization/roleDefinitions*

to get the role definition id.

/// <summary>

/// Use the Azure Management API to query roleDefinationId

/// </summary>

/// <param name="roleName">customer's role name </param>

/// <returns></returns>

private static string QueryRoleDefinitionIdByRoleName(string tenantId, string subscriptionId, string roleName)

{

string tokenAzureRM= Program.LoginToAad(tenantId, SettingsHelper.AzureRMApiResourceId).Result.AccessToken;

string filter = HttpUtility.UrlEncode(string.Format("roleName eq '{0}'", roleName));

JObject retJson = RoleDefinitionsService.List(tokenAzureRM, subscriptionId, filter);

return (string)retJson["value"][0]["id"];

}

1. You can use the below codes to get the role definition id

string roleDefinitionId = QueryRoleDefinitionIdByRoleName(“your tenant id /azure ad directory id/customer id”,”your azure subscription id under your tenant”, “role name you want to assign to user , like Owner”);

### Assign user to azure subscription with special role

1. Services folder>Right Click >Add>Class>>add RoleAssignmentsService.cs

using MyDemos.Utils;

using Newtonsoft.Json.Linq;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MyDemos.Services

{

public class RoleAssignmentsService

{

/// <summary>

/// Assign use to azure subscription with special role

/// </summary>

/// <param name="aadToken">AAD access token for azure management apis</param>

/// <param name="subscriptionId">azure subscription id</param>

/// <param name="roleDefinitionId">the azure role's definition id</param>

/// <param name="objectId">the user's objectid retrieved from azure graph APIs</param>

/// <returns></returns>

static public JObject Create(string aadToken, string subscriptionId, string roleDefinitionId, string objectId)

{

string roleAssignmentId = Guid.NewGuid().ToString();

string requestUrl = string.Format("{0}/subscriptions/{1}/providers/Microsoft.Authorization/roleAssignments/{2}?api-version=2017-10-01-preview", SettingsHelper.AzureRMApiEndpoint, subscriptionId, roleAssignmentId);

Dictionary<string, string> headers = new Dictionary<string, string>();

headers.Add("Authorization", string.Format("{0}{1}", "Bearer ", aadToken));

string jsonParams = "{" +

"\"properties\":{" +

"\"roleDefinitionId\":\"" + roleDefinitionId + "\"," +

"\"principalId\":\"" + objectId + "\"" +

"}" +

"}";

return HttpUtils.DoPut(requestUrl, headers, jsonParams); ;

}

}

}

1. Open Program.cs and add below codes

You will try to get the user object id and role definition id, then create role assignment via azure manage APIs.

/// <summary>

/// Assign user to azure subscription with special role name

/// </summary>

/// <param name="subscriptionId">customer's azure subscription ID</param>

/// <param name="roleName">role name to assign to user </param>

/// <param name="mail">the user will be assign to subscription</param>

private static void AssignUserToSubscriptionWithRole(string tenantId, string subscriptionId, string roleName, string mail)

{

Console.WriteLine("Start to assign user [{0}] with role [{1}] to subscription [{2}]", mail, roleName, subscriptionId);

string objectId = QueryObjectIdByMail(tenantId,mail);

string roleDefinitionId = QueryRoleDefinitionIdByRoleName(tenantId,subscriptionId, roleName);

string tokenAzureRM= Program.LoginToAad(tenantId, SettingsHelper.AzureRMApiResourceId).Result.AccessToken;

JObject retJson = RoleAssignmentsService.Create(tokenAzureRM, subscriptionId, roleDefinitionId, objectId);

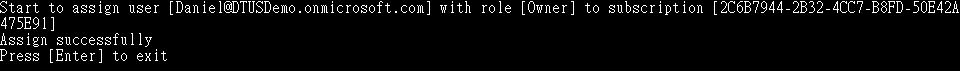
Console.WriteLine("Assign successfully");

}

1. Add below codes in Main method to assign user to azure subscription with special role

AssignUserToSubscriptionWithRole(“tenant id /azure ad directory id/customer id”, “the azure subscription under tenant”,“role name to assign user ,like Owner”, “user’s email in same tenant like admin@<domainPrefix>.onmicrosoft.com”);

1. Save your changes, and press F5 to test the application.



## Exercise 3: Query usage records via Partner Center SDK&Azure Management APIs

1. Services folder>Right Click >Add>Class>>add ResoucesService.cs. Here are the codes:

There are mainly two points:

1. Due to the resources API version are different, you should list the available API version from providers list
2. After you get the API version, you can get tags from the resources details

using MyDemos.Utils;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web;

s

namespace MyDemos.Services

{

public class ResouceService

{

/// <summary>

/// Query resource details and get tags

/// </summary>

/// <param name="aadToken">AAD access token for azure management api</param>

/// <param name="resouceUrl">the azure resource's relative url</param>

/// <param name="apiVersion">the api version of azure management APIs</param>

/// <returns></returns>

public static string ListTags(string aadToken, string resouceUrl,string apiVersion)

{

string requestUrl = string.Format("{0}{1}?api-version={2}", SettingsHelper.AzureRMApiEndpoint, resouceUrl, apiVersion);

Dictionary<string, string> headers = new Dictionary<string, string>();

headers.Add("Authorization", string.Format("{0}{1}", "Bearer ", aadToken));

JObject retJson = HttpUtils.DoGet(requestUrl, headers);

if (retJson == null)

return "not exist or resouces has been removed";

Dictionary<string, string> tags = JsonConvert.DeserializeObject<Dictionary<string, string>>(retJson["tags"].ToString());

StringBuilder builder = new StringBuilder();

foreach(string tag in tags.Keys)

{

builder.AppendFormat("{0}:{1}|", tag, tags[tag]);

}

return builder.ToString();

}

/// <summary>

/// Query providers under subscription (which contains the api versions of resources’ api)

/// </summary>

/// <param name="aadToken"></param>

/// <param name="subscriptionId"></param>

/// <returns></returns>

public static JObject ListProvider(string aadToken,string subscriptionId)

{

string requestUrl = string.Format("{0}/subscriptions/{1}/providers?api-version=2017-05-10", SettingsHelper.AzureRMApiEndpoint, subscriptionId);

Dictionary<string, string> headers = new Dictionary<string, string>();

headers.Add("Authorization", string.Format("{0}{1}", "Bearer ", aadToken));

return HttpUtils.DoGet(requestUrl, headers);

}

}

}

1. Open the Program.cs and add below codes:

In this method, you will query the usage records via partner center SDK.

Based on the resource Uri in record, try to parse the resource group name.

Last query the resources available API version from provider and parse the tags from the resource details.

/// <summary>

/// Query the utilization of customer's azure subscription

/// </summary>

/// <param name="partner"></param>

/// <param name="customerId">the customer's id/tenant id/directory id</param>

/// <param name="subscriptionId">the customer's azure subscription id</param>

private static void QueryUtilization(IPartner partner, string customerId,string subscriptionId)

{

// Retrieve the utilization records for the last year in pages of 100 records.

// If you want to retrieve other records, please modify the parameter by yourself

var utilizationRecords = partner.Customers[customerId].Subscriptions[subscriptionId].Utilization.Azure.Query(

DateTimeOffset.Now.AddYears(-1),

DateTimeOffset.Now,

size: 100);

// Create an Azure utilization enumerator which will aid us in traversing the utilization pages.

var utilizationRecordEnumerator = partner.Enumerators.Utilization.Azure.Create(utilizationRecords);

int pageNumber = 1;

string tokenAzureRM = Program.LoginToAad(customerId, SettingsHelper.AzureRMApiResourceId).Result.AccessToken;

JObject retJson = ResouceService.ListProvider(tokenAzureRM, subscriptionId);

while (utilizationRecordEnumerator.HasValue)

{

foreach (var record in utilizationRecordEnumerator.Current.Items)

{

//Parse resource name from resource uri

string resourceUri = record.InstanceData.ResourceUri.ToString();

int rsgStart = resourceUri.ToLower().IndexOf("/resourcegroups/");

int rsgNameStart = rsgStart + 16;

int rsgNameLength = resourceUri.Substring(rsgNameStart).IndexOf("/");

string rsgName = resourceUri.Substring(rsgNameStart, rsgNameLength);

string tags = "";

//Based on the resourceuri, you can parse the resource group name;

//if the resource still exists, you can try to query the resources's details to

//get the tags. but if the resource has been removed, errors will occur.

try

{

//parse the provider name

int providerStart = resourceUri.ToLower().IndexOf("/providers/");

int providerNameStart = providerStart + 11;

int providerNameLength = resourceUri.Substring(providerNameStart).IndexOf("/");

string providerName = resourceUri.Substring(providerNameStart, providerNameLength);

//parse the resource type name

int resouceStart = resourceUri.IndexOf(string.Format("/{0}/", providerName));

int resourceTypeStart = resouceStart + providerName.Length + 2;

int resourceTypeNameLength = resourceUri.Substring(resourceTypeStart).IndexOf("/");

string resourceTypeName = resourceUri.Substring(resourceTypeStart, resourceTypeNameLength);

//use the provider name and resource type to parse the available api version

var provider = retJson["value"].ToObject<List<Dictionary<string, object>>>().Where(x => x["id"].Equals(string.Format("/subscriptions/{0}/providers/{1}", subscriptionId.ToLower(), providerName))).First();

var resourceTypes = ((JArray)provider["resourceTypes"]).ToObject<List<Dictionary<string, object>>>();

var resourceType = resourceTypes.Where(x => x["resourceType"].Equals(resourceTypeName)).First();

JArray apiVersions = (JArray)resourceType["apiVersions"];

string apiVersion = apiVersions[0].ToString();

//query the resource details and get tags

tags = ResouceService.ListTags(tokenAzureRM, resourceUri, apiVersion);

}

catch (Exception e)

{

// Console.WriteLine(e.Message);

}

Console.WriteLine("Name:{0}|Resource Group Name:{1}|Tags:{2}|Quantity:{3}|Unit:{4}|UsageStartTime:{5}|UsageEndTime:{6}", record.Resource.Name,rsgName,tags, record.Quantity, record.Unit, record.UsageStartTime, record.UsageEndTime);

}

utilizationRecordEnumerator.Next();

pageNumber++;

//break; this is for test

}

}

1. Add below codes in Main method to query the azure subscription usage records

QueryUtilization(partner, “tenant id/azure ad directory id/customer id”, “azure subscription id”);

1. Save your changes, and press F5 to test the application.

