Azure Active Directory Authentication (using adal.js) and Application registration

Why do we need Azure AD Authentication Library (ADAL) to authenticate our Dynamics CRM application ?

Start from Dynamics CRM 2016, Microsoft introduced new big feature called Web API endpoint. That means we can build our own Web Application, SPA or Mobile Application and play with Dynamics CRM data. In that case we need a library for Azure AD Authentication to get access token before making request to Web API and that is Azure AD Authentication Library (ADAL)

List of platforms that ADAL support. View detail here

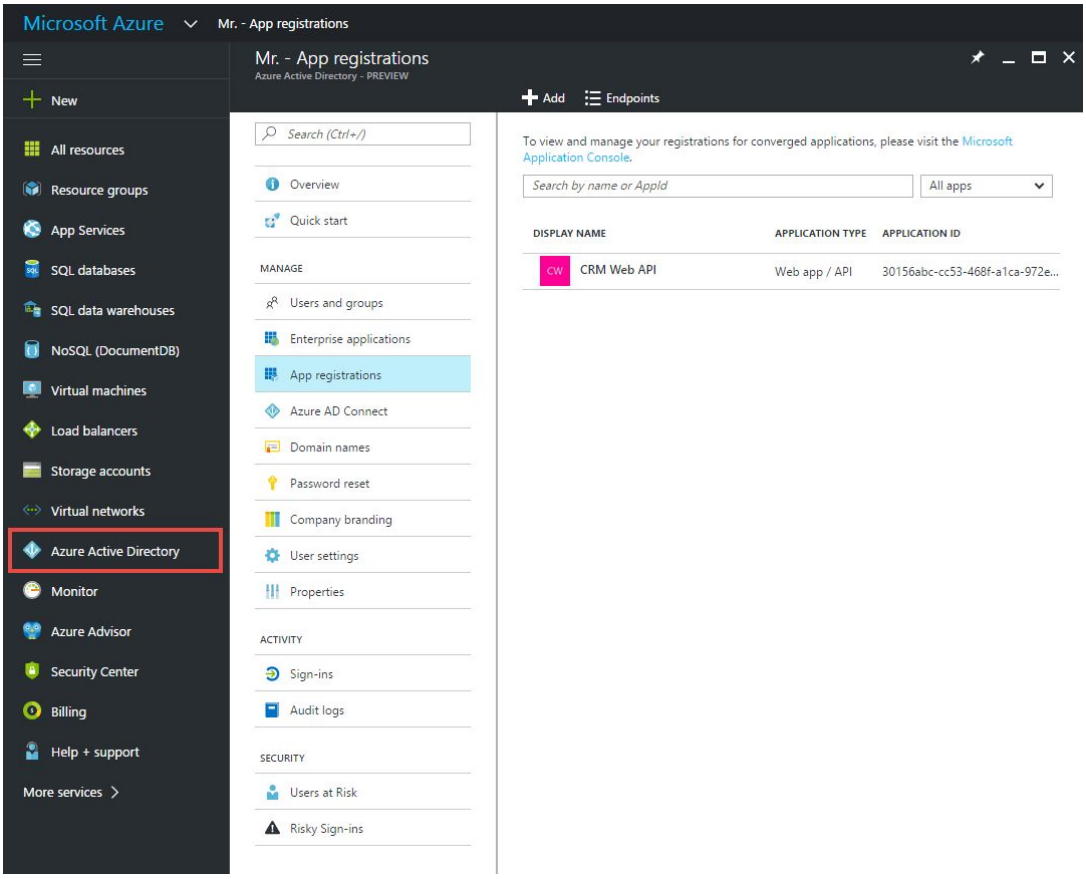
- Note that not only Dynamics CRM Web API can be authenticated via ADAL but also Microsoft Graph applications (view detail here)

1. Azure Active Directory Authentication using adal.js

1.1. Register Application in Azure Active Directory

Any application that wants to use the capabilities of Azure AD must first be registered in an Azure AD tenant. During registration process, Azure AD needs some information about your application like your application URL, callback URL after a user is authenticated

First, access to Azure portal (https://portal.azure.com) and go to Azure Active Directory

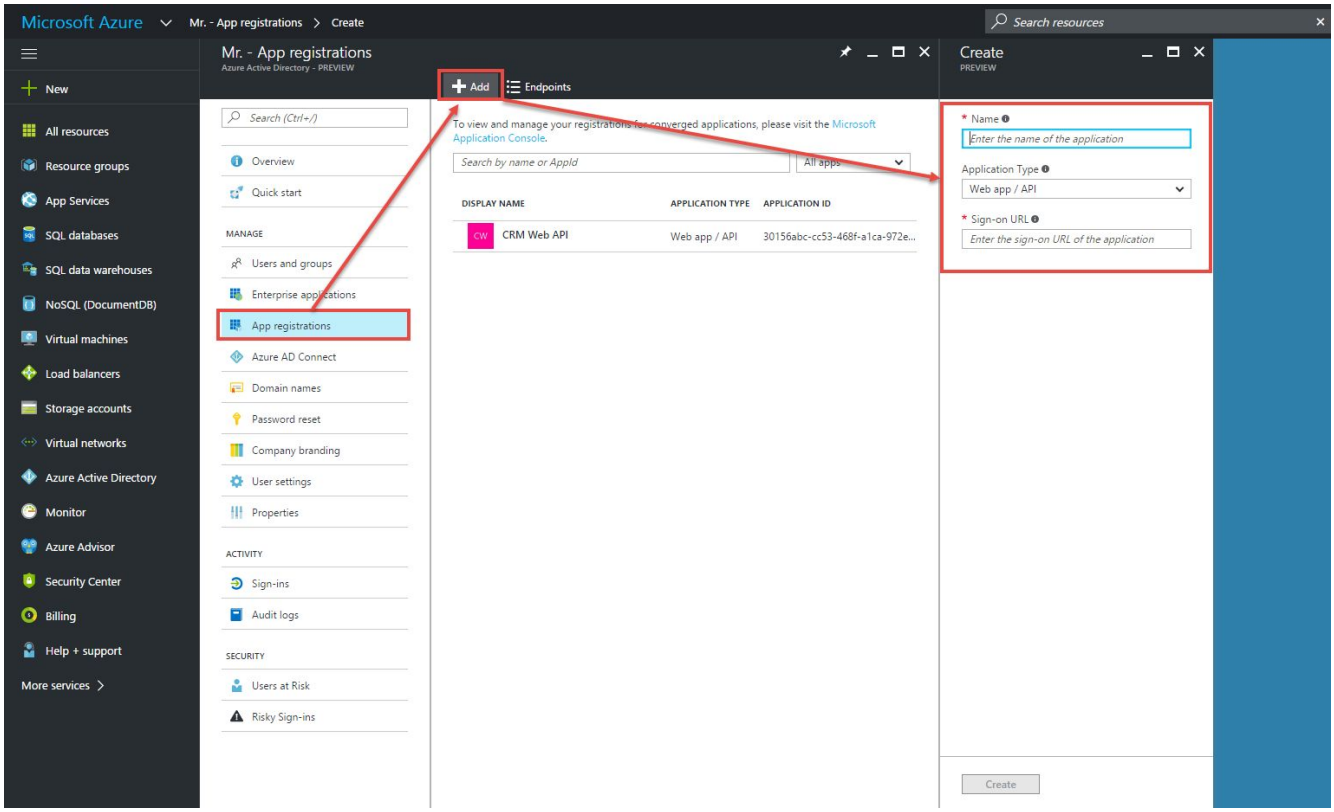


From App Registrations, Add new application

You need to specify a name for your application and select Application Type also. There are 2 applications type that you can select: Web app/API or a Native client application. In this sample we will choose Web app/API

Depend on your application is a web or native application, you will see different options on steps to add your application

In this sample, we will put Sign-on URL as our home page (e.g http://localhost:61950/Portal/Sample.html)



1.2. Enabling OAuth 2.0 implicit grant for Single Page Applications

1.2.1. What is OAuth2 implicit grant ?

The OAuth2 implicit grant is a variant of other authorization grants. It allows a client to obtain an access token directly from the authorization endpoint, without contacting the token endpoint nor authenticating the client. It's designed for JavaScript based applications, especially for Single Page Applications (SPAs). By that way, our application can invoke APIs (like Microsoft Graph API, Office API, Azure API) outside the domain where the application is hosted

1.2.2. How to enable OAuth 2.0 implicit grant

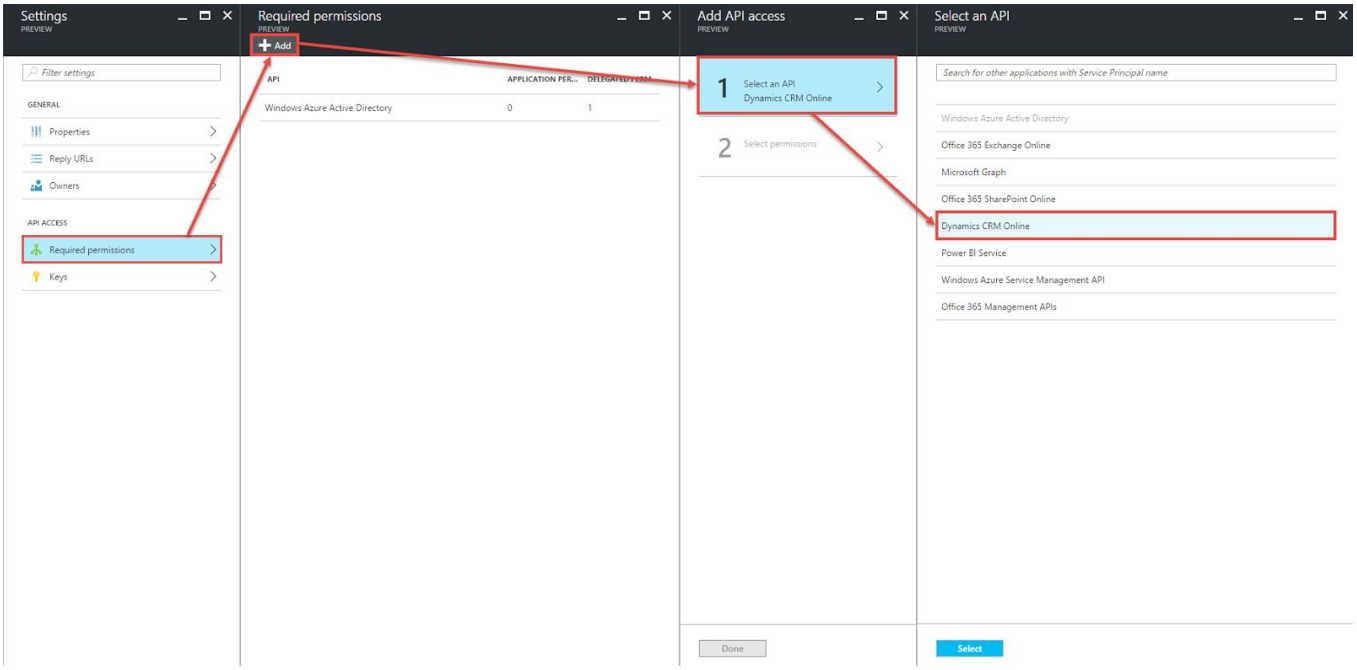
Once our application is registered, select on that app, select Manifest file and download this file

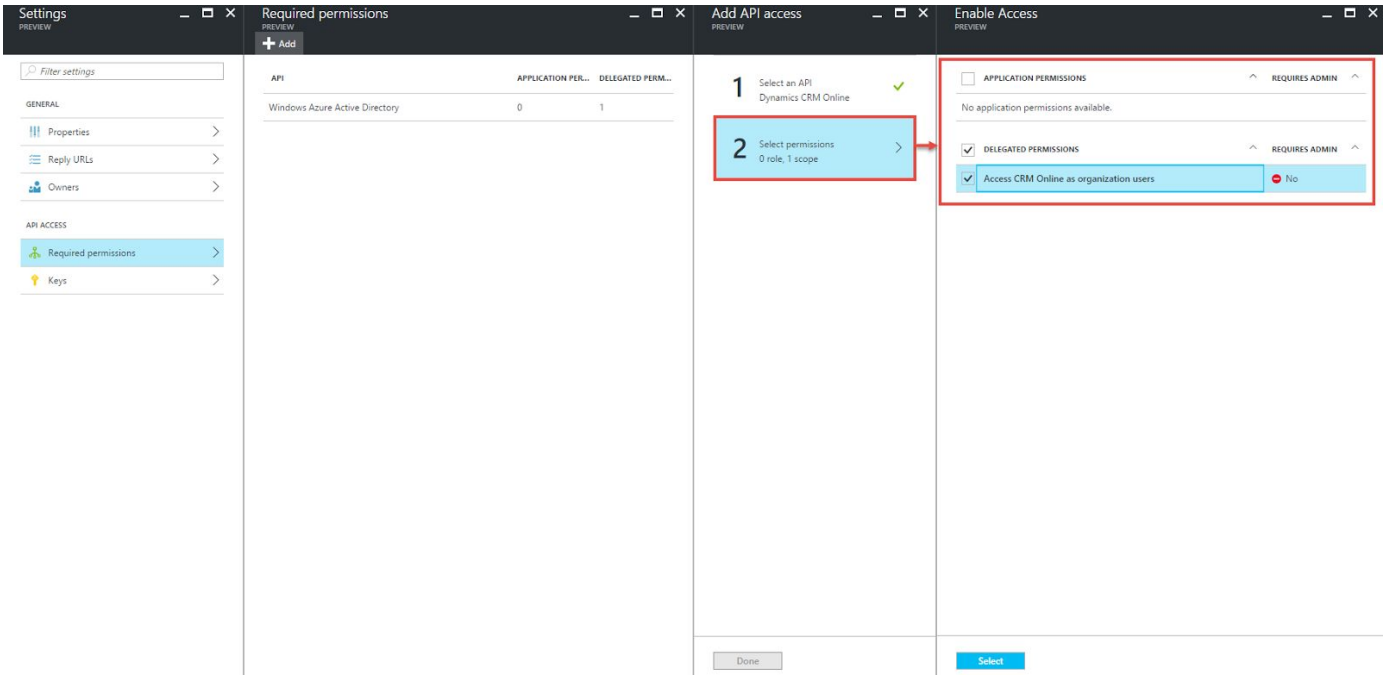
Open Manifest file and update **"oauth2AllowImplicitFlow"** value to **true**

Upload Manifest file to your application

1.2.3. Add permission for your application

From application, add permission for user to access your Dynamics CRM App


Settings
PREVIEW
Filter settings
GENERAL
Properties
Reply URLs
Owners
API ACCESS
Required permissions
Keys
Required permissions
PREVIEW
Add
API
APPLICATION PER...
DELEGATED PERM...
Windows Azure Active Directory
0
1
Add API access
PREVIEW
1
Select an API
Dynamics CRM Online
2
Select permissions
Done
Select an API
PREVIEW
Search for other applications with Service Principal name
Windows Azure Active Directory
Office 365 Exchange Online
Microsoft Graph
Office 365 SharePoint Online
Dynamics CRM Online
Power BI Service
Windows Azure Service Management API
Office 365 Management APIs
Select


Settings
PREVIEW
Filter settings
GENERAL
Properties
Reply URLs
Owners
API ACCESS
Required permissions
Keys
Required permissions
PREVIEW
Add
API
APPLICATION PER...
DELEGATED PERM...
Windows Azure Active Directory
0
1
Add API access
PREVIEW
1
Select an API
Dynamics CRM Online
2
Select permissions
0 role, 1 scope
Done
Enable Access
PREVIEW
APPLICATION PERMISSIONS
REQUIRES ADMIN
No application permissions available.
DELEGATED PERMISSIONS
REQUIRES ADMIN
Access CRM Online as organization users
No
Select

1.3. Authentication for web application using adal.js

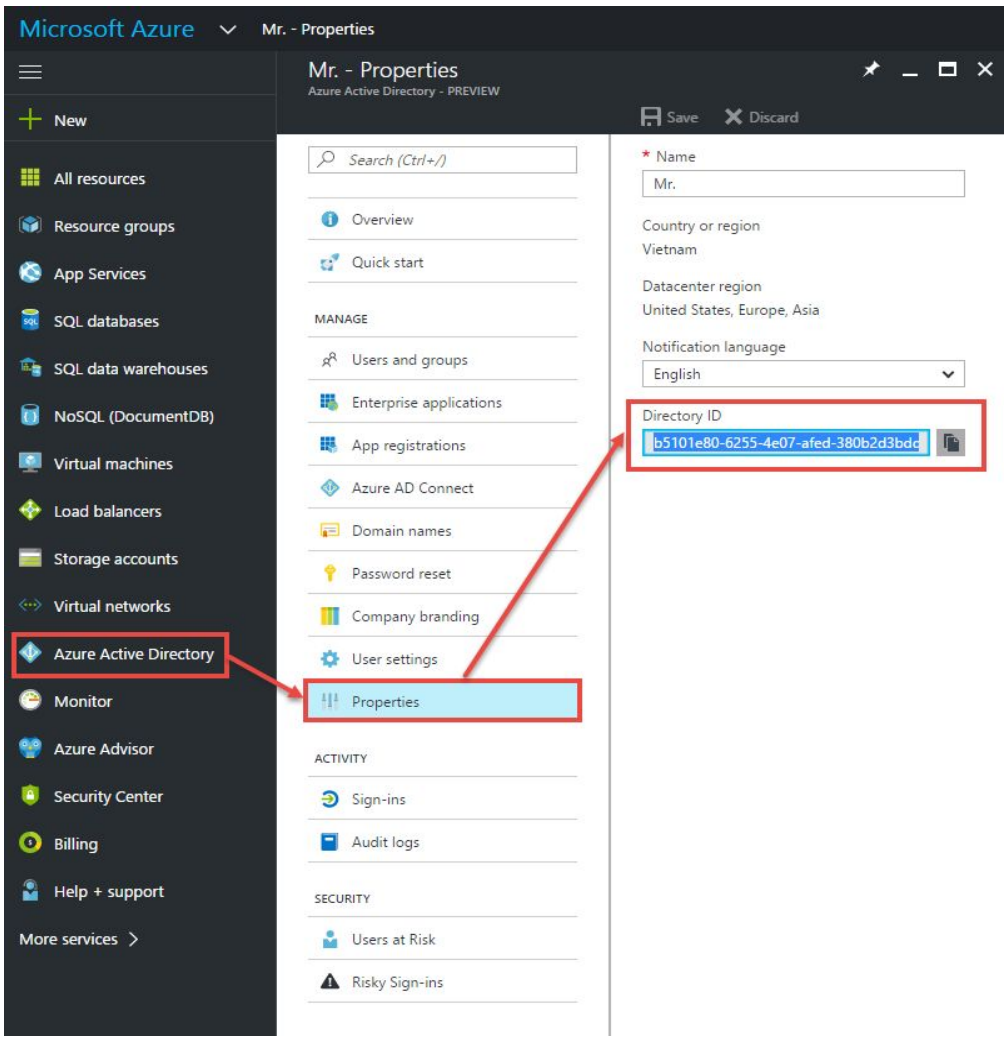
Base on library from https://github.com/Scaleable-solutions/WebAPIAuthFromJavaScript, I fixed some issued that happened when losing cookie or token expired and updated into **util.js**

We need to update configuration for our application before using Dynamics CRM 365 Web API

```
var organizationURI = "https://<ORG_NAME>.api.crm5.dynamics.com"; // TODO: Add your
organizationURI
...
var tenant = "b5101e80-6255-4e07-afed-380b2d3bdd51"; // TODO: add your tenant
var clientId = "30156abc-cc53-468f-a1ca-972e1c3b02d6"; // TODO: Add your Client Id
var pageUrl = "http://localhost:61950/Portal/Sample.html";// TODO: Add your Reply URL
```

ClientId: Your Application ID when you register your app into Azure AD

TenanID: Your Directory ID (in Azure Active Directory) – see image below



Once web page is loaded, we will run **authenticate()** function for checking whether token is available or not

```
function authenticate() {
    var isCallback = authContext.isCallback(window.location.hash);
    if (isCallback) {
        authContext.handleWindowCallback();
    }
    var loginError = authContext.getLoginError();

    if (isCallback && !loginError) {
        window.location = authContext._getItem(authContext.CONSTANTS.STORAGE.LOGIN_REQUEST);
    }
    else {
        //errorMessage.textContent = loginError;
        //alert(loginError);
    }
    if (authContext._loginInProgress == true) {
        return;
    }
    user = authContext.getCachedUser();
    var hasToken = true;
    if (authContext._getItem(authContext.CONSTANTS.STORAGE.EXPIRATION_KEY + organizationURI) ==
e ||
        authContext._getItem(authContext.CONSTANTS.STORAGE.RENEW_STATUS +
window.config.clientId) == authContext.CONSTANTS.TOKEN_RENEW_STATUS_COMPLETED) {
        authContext.acquireToken(organizationURI,
        function (error, token) {
            if (!token) {
                console.warn("Cannot find token");
                hasToken = false;
                return false;
            }
            if (!isCallback) {
```

For login button, we will add login function from authentication context (adaj.js) to redirect user to Azure AD authorization endpoint

```
document.getElementById('login').addEventListener('click', function () {
        login();
})

function login() {
    authContext.login();
}
```

1.4. Problems during implementation

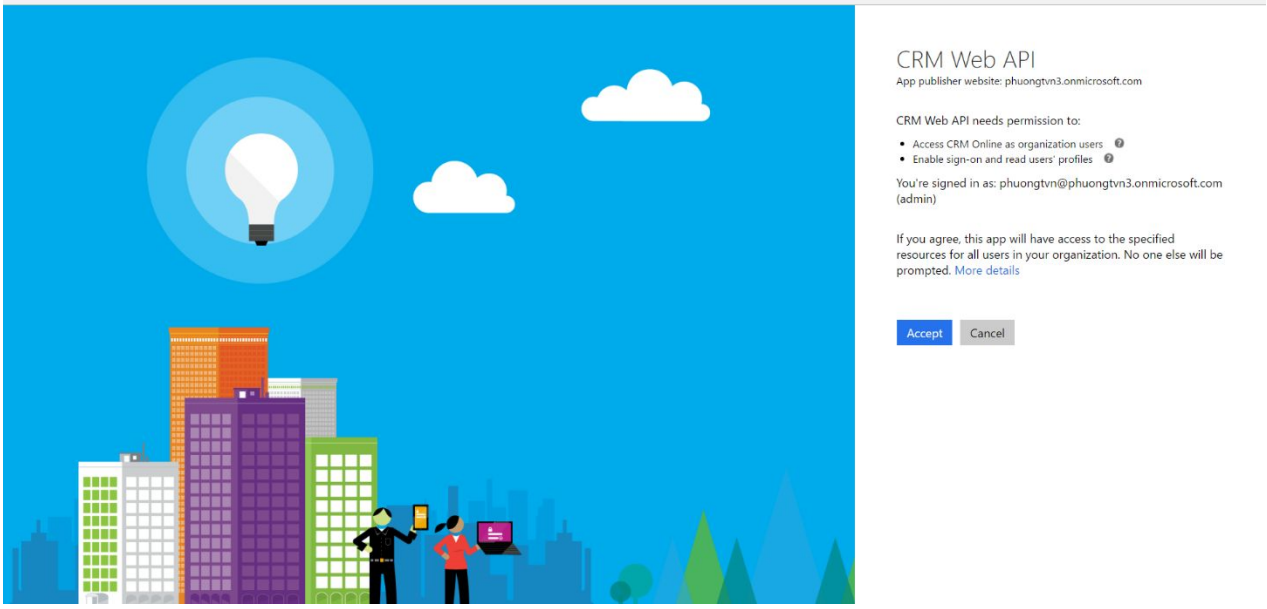
During implementation, I encountered one issue in authorization. Let's assume that we've already finished setting up authorization for our application. In this case you may see this error when you're trying to click login to authorization endpoint

```
AADSTS65001: The user or administrator has not consented to use the
application with ID 'xxxx-xxxxx-xxxx-xxx'.
```

In this case we need to add one more parameter **prompt** before make the authorization request:

```
prompt=admin_consent
```

After that we will login with Administrator account and then accept our application requests (to allow organization user access CRM Online and enable sign-on and read user's profile



After accept these request, then we will remove parameter **prompt** from authorization request. Then all organization users can authorize via authorization endpoint. Otherwise you will see this error on the next time login:

```
AADSTS90093: Does not have access to consent
```