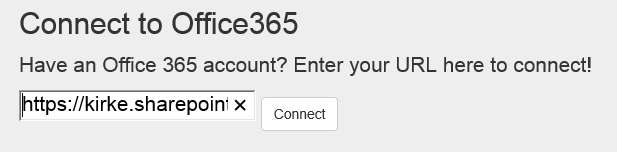
# Office AMS: App Permissions

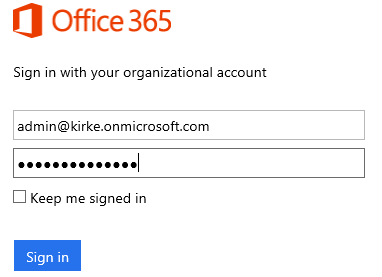
|  |  |
| --- | --- |
| Summary: | Applies to: |
| This sample shows how to dynamically request permissions for an app from any web site using an OAuth code. | * Office 365 Multi Tenant (MT) * Office 365 Dedicated (D) * SharePoint 2013 on-premises |
| Solution: | Core.DynamicPermissions, version 1.0 |
| Author: | Kirk Evans (Microsoft) |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS CODE IS PROVIDED \*AS IS\* WITHOUT WARRANTY OF  // ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING ANY  // IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR  // PURPOSE, MERCHANTABILITY, OR NON-INFRINGEMENT.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

# Scenario: Dynamically request app permissions

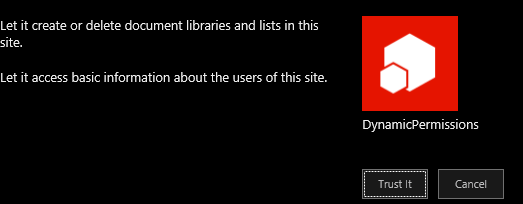
This scenario shows how an app can be used to dynamically request permissions to access SharePoint resources from any web site.



After entering the URL for the SharePoint site and clicking the Connect button, the user is redirected to sign into Office 365.



Once signed in, the user is prompted to trust the app.



Note: This type of app can only be run by users who have Manage permissions to the resources the app wants to access, because only they have sufficient rights to grant the app the permissions that it requests. For example, if an app requests only Read permission to a website, a user who has Read, but not Manage, rights to the website cannot run the app. For more information, see <http://msdn.microsoft.com/en-us/library/office/jj687470.aspx>.

This happens because the app redirects the user to a page in SharePoint, OAuthAuthorize,aspx, passing a client ID, requested permission scope, and the response type as “code”.

https://kirke.sharepoint.com/sites/dev/\_layouts/15/OAuthAuthorize.aspx?IsDlg=1&client\_id=2bb3c34a-b043-4c6c-adda-2e9634f24c3d&scope=Web.Manage&response\_type=code

This URL is formed by using TokenHelper to determine the appropriate redirect URL simply by passing the URL of the SharePoint site with the desired permissions.

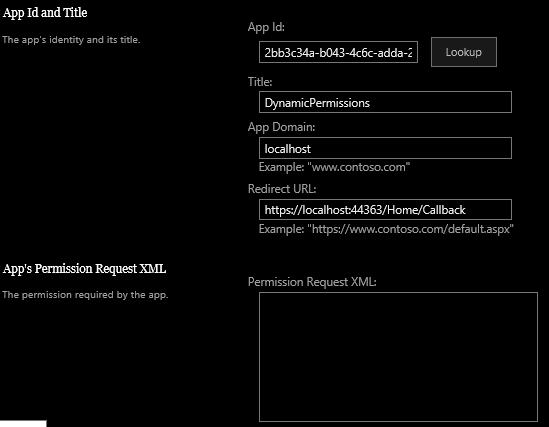
\_response.Redirect(TokenHelper.GetAuthorizationUrl(hostUrl, "Web.Manage"));

Notice the client ID in the above URL example. This indicates the client ID must be known in advance. If the app is registered through the Seller Dashboard, then any SharePoint online site can be used. If the app is not registered through the Seller Dashboard, then you first must register the client ID using appregnew.aspx in the SharePoint site prior to calling. The resulting App Principal includes the callback URL. The client ID must exist in the web.config for the app.

To be able to call into SharePoint, this type of app must first be registered through the Seller Dashboard or the appregnew.aspx page. For more information about registering apps via the Seller Dashboard or appregnew.aspx, see Guidelines for registering apps for SharePoint 2013. For more information, see <http://msdn.microsoft.com/en-us/library/office/jj687470.aspx>.

TokenHelper will read the client ID from web.config and append it to the URL.

As a demonstration, I register the app using AppRegNew.aspx. The client ID identifies the app, and the Redirect URL provides the URL to redirect the browser to once the permissions are granted.



Once the app has been granted permissions, it redirects to the registered Redirect URL (<https://localhost:44363/Home/Callback>) and passes an authorization code. This code is handed in the Home controller in the Callback action:

public ActionResult Callback(string code)

{

TokenRepository repository = new TokenRepository(Request, Response);

repository.Callback(code);

return RedirectToAction("Index");

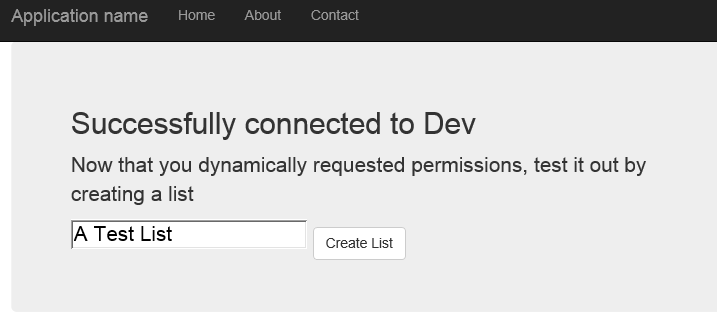
}

Notice the code that is passed. Once this method receives the code, we use TokenHelper.GetAccessToken to obtain an OAuth access token based on the returned code.

string refreshToken = TokenHelper.GetAccessToken(request.QueryString["code"], "00000003-0000-0ff1-ce00-000000000000", targetUri.Authority, TokenHelper.GetRealmFromTargetUrl(targetUri), new Uri(request.Url.GetLeftPart(UriPartial.Path))).RefreshToken;

Use the refresh token and access token in order to create a ClientContext with the client side object model. The app principal was registered in AppRegNew.aspx without any permissions (note the empty permissions XML block above), but was able to successfully request Manage permissions.

The app can now obtain the SharePoint site’s title (notice the “Successfully connected to” in the screen shot below shows the site’s title).



To test that our provider-hosted app actually was granted Manage permissions for the web, we can create a new list by providing the list title. Click Create List and we see the new list is created.

