**MS Word implementation imformation**

**Timing – 3 Months**

**Pricing-**

**Azure-Account:-**

**1** **Start free**. Get USD 200 credit to use in 30 days. While you have your credit, get free amounts of popular services and 55+ other services.

**2** After your credit, move to **pay as you go** to keep getting popular services and 55+ other services. Only pay if you use more than the free monthly amounts.

**3** **After 12 months**, you'll continue getting 55+ services free always—and still only pay for what you use beyond the free monthly amounts.

<https://azure.microsoft.com/en-in/free/search/?ef_id=_k_Cj0KCQjwvL-oBhCxARIsAHkOiu3e7nS4WkqFohXkXLfGR1Kvn5zyIo3HtToGQk2rEIb7_2LfbDp9g2oaAlIlEALw_wcB_k_&OCID=AIDcmmf1elj9v5_SEM__k_Cj0KCQjwvL-oBhCxARIsAHkOiu3e7nS4WkqFohXkXLfGR1Kvn5zyIo3HtToGQk2rEIb7_2LfbDp9g2oaAlIlEALw_wcB_k_&gclid=Cj0KCQjwvL-oBhCxARIsAHkOiu3e7nS4WkqFohXkXLfGR1Kvn5zyIo3HtToGQk2rEIb7_2LfbDp9g2oaAlIlEALw_wcB>

**Azure API Management price-**

[**https://azure.microsoft.com/en-in/pricing/details/api-management/**](https://azure.microsoft.com/en-in/pricing/details/api-management/)

**Note-** There all api's present insteadOf MS word - <https://developer.microsoft.com/en-us/graph/graph-explorer/>

**Office 365 Management APIs-**

**Documentation -** [**https://learn.microsoft.com/en-us/office/office-365-management-api/get-started-with-office-365-management-apis**](https://learn.microsoft.com/en-us/office/office-365-management-api/get-started-with-office-365-management-apis)

There are four key steps: **Overview** What need to do-

1. **Register your application in Azure AD**. To allow your application access to the Office 365 Management APIs, you need to register your application in Azure AD. This allows you to establish an identity for your application and specify the permission levels it needs to access the APIs.
2. **Get Office 365 tenant admin consent**. An Office 365 tenant admin must explicitly grant consent to allow your application to access their tenant data by means of the Office 365 Management APIs. The consent process is a browser-based experience that requires the tenant admin to sign in to the **Azure AD consent UI** and review the access permissions that your application is requesting, and then either grant or deny the request.
3. **Request access tokens from Azure AD**. Using your application's credentials as configured in Azure AD, your application requests additional access tokens for a consented tenant on an ongoing basis, without the need for further tenant admin interaction. These access tokens are called app-only tokens because they do not include information about the tenant admin.
4. **Call the Office 365 Management APIs**. The app-only access tokens are passed to the Office 365 Management APIs to authenticate and authorize your application.

**In Detail What is required for implementation-**

1. Sign into the [Azure portal](https://portal.azure.com/), using the credential of your Microsoft tenant that has the subscription to Office 365 you wish to use. You can also access the Azure Portal via a link that appears in the left navigation pane in the [Microsoft 365 admin center](https://admin.microsoft.com/).
2. In the left navigation pane, select **Azure Active Directory** (1).
3. On the **Azure Active Directory** page, select **App registrations** (2), and then select **New registration** (3).

A new page appears for you to start the registration of your app.

1. On the **Register an application** page, do the following things:
2. Name your app.
3. Choose who can use the app and access the API.
4. Provide a redirect URL for user redirect after authentication, if needed and Register.

**Now we will get the application properties in Azure AD.**

1. **Client ID**. This value is automatically generated by Azure AD. Your application will use this value when requesting consent from tenant admins and when requesting app-only tokens from Azure AD.
2. **Application is multi-tenant**. This property must be set to **YES** to allow tenant admins to grant consent to your app to access their data by using the Office 365 Management APIs. If this property is set to **NO**, your application will only be able to access your own tenant's data.
3. **Reply URL**. This is the URL that a tenant admin will be redirected to after granting consent to allow your application to access their data by using the Office 365 Management APIs. You can configure multiple reply URLs as needed. Azure automatically sets the first one to match the sign-on URL you specified when you created the application, but you can change this value as needed.

### **Generate a new key for your application**

Keys, also known as **client secrets**, are used when exchanging an authorization code for an access token.

* 1. On the **Azure Active Directory** page in the Azure portal, select **App registrations**, and then select your application.
  2. After the page for your app is displayed, select **Certificates & secrets** (1) in the left pane. On this page, you can upload certificates and create new client secrets (2).
  3. On the **Certificates & secrets** (1) page, select **New client secret** (2), type a description and select the duration for your key (3), and then select **Add** (4).
  4. fter creating the client secret, the value is displayed under **Client secrets** (2). Click the clipboard icon (3) to copy the client secret value to the clipboard.

### **Specify the permissions your app requires to access the Office 365 Management APIs**

1. In the Azure Portal, go to **App registrations** > **All applications**, select your application, and then select **API Permissions** (1) in the left pane. Click **Add a permission** (2) to display the **Request API permission** (3) flyout page.
2. On the **Microsoft APIs** tab, select **Office 365 Management APIs** (4).
3. On the flyout page, select the following types of permissions (3) that your app requires, and then click **Add permissions**
4. **Delegated Permissions**. Enables your client app to perform operations on behalf of the signed-in user, such as reading email or modifying the user's profile.
5. **Application Permissions**. Permissions that enable the client app to authenticate as itself without user interaction or consent, such as an app used by background services or daemon apps.
6. Office Management APIs now appear in the list of applications that your app requires permissions for. Under both **Application Permissions** and **Delegated Permissions**, if needed, select the permissions your application requires. Refer to the specific API reference for more details about each permission.
7. Select **Grant admin consent for "tenant name"** to consent to the permissions given to your app.

## **Get Office 365 tenant admin consent**

* Now that your application is configured with the permissions it needs to use the Office 365 Management APIs, a tenant admin must explicitly grant your application these permissions in order to access their tenant's data by using the APIs. To grant consent, the tenant admin must sign in to Azure AD by using the following specially constructed URL, where they can review your application's requested permissions. This step is not required when using the APIs to access data from your own tenant.

<https://login.windows.net/common/oauth2/authorize?response_type=code&resource=https%3A%2F%2Fmanage.office.com&client_id=2d4d11a2-f814-46a7-890a-274a72a7309e&redirect_uri=http%3A%2F%2Fwww.mycompany.com%2Fmyapp%2F>

* After choosing **Accept**, you are redirected to the specified page, and there will be a code in the query string.

<http://www.mycompany.com/myapp/?code=AAABAAAAvPM1KaPlrEqdFSB>...

* Your application uses this authorization code to obtain an access token from Azure AD, from which the tenant ID can be extracted. After you have extracted and stored the tenant ID, you can obtain subsequent access tokens without requiring the tenant admin to sign in.

## **-Request access tokens from Azure AD**

There are two methods for requesting access tokens from Azure AD:

* The [Authorization Code Grant Flow](https://learn.microsoft.com/en-us/previous-versions/azure/dn645542(v=azure.100)) involves a tenant admin granting explicit consent, which returns an authorization code to your application. Your application then exchanges the authorization code for an access token. This method is required to obtain the initial consent that your application needs to access the tenant data by using the API, and this first access token is needed in order to obtain and store the tenant ID.
* The [Client Credentials Grant Flow](https://learn.microsoft.com/en-us/previous-versions/azure/dn645543(v=azure.100)) allows your application to request subsequent access tokens as old ones expire, without requiring the tenant admin to sign in and explicitly grant consent. This method must be used for applications that run continuously in the background calling the APIs once the initial tenant admin consent has been granted.

### **Request an access token using the authorization code**

1. After a tenant admin grants consent, your application receives an authorization code as a query string parameter when Azure AD redirects the tenant admin to your designated URL.

<http://www.mycompany.com/myapp/?code=AAABAAAAvPM1KaPlrEqdFSB>...

1. Your application makes an HTTP REST POST to Azure AD to exchange the authorization code for an access token. Because the tenant ID is not yet known, the POST will be to the "common" endpoint, which does not have the tenant ID embedded in the URL:

<https://login.windows.net/common/oauth2/token>

1. The body of the POST contains the following:

resource=https%3A%2F%2Fmanage.office.com&amp;client\_id=a6099727-6b7b-482c-b509-1df309acc563 &amp;redirect\_uri= http%3A%2F%2Fwww.mycompany.com%2Fmyapp%2F &amp;client\_secret={your\_client\_key}&amp;grant\_type=authorization\_code&amp;code= AAABAAAAvPM1KaPlrEqdFSB...

**What we need to request sample request is –**

POST https://login.windows.net/common/oauth2/token HTTP/1.1

Content-Type: application/x-www-form-urlencoded

Host: login.windows.net

Content-Length: 944

resource=https%3A%2F%2Fmanage.office.com&amp;client\_id=a6099727-6b7b-482c-b509-1df309acc563 &amp;redirect\_uri= http%3A%2F%2Fwww.mycompany.com%2Fmyapp%2F &amp;client\_secret={your\_client\_key}&amp;grant\_type=authorization\_code&amp;code=AAABAAAAvPM1KaPlrEqdFSB...

**The body of the response will include several properties, including the access token.**

#### **Sample response that we get-**

HTTP/1.1 200 OK

Content-Type: application/json; charset=utf-8

Content-Length: 3265

{"expires\_in":"3599","token\_type":"Bearer","scope":"ActivityFeed.Read ActivityReports.Read ServiceHealth.Read","expires\_on":"1438290275","not\_before":"1438286375","resource":"https://manage.office.com","access\_token":"eyJ0eX...","refresh\_token":"AAABAAA...","id\_token":"eyJ0eXAi..."}

**NOTE-** The access token that is returned is a JWT token that includes information about both the admin that granted consent and the application requesting access. The following shows an example of an un-encoded token. Your application must extract the tenant ID "tid" from this token and store it so that it can be used to request additional access tokens as they expire, without further admin interaction.

#### **Sample token**

{

"aud": "https://manage.office.com",

"iss": "https://sts.windows.net/41463f53-8812-40f4-890f-865bf6e35190/",

"iat": 1427246416,

"nbf": 1427246416,

"exp": 1427250316,

"ver": "1.0",

"tid": "41463f53-8812-40f4-890f-865bf6e35190",

"amr": [

"pwd"

],

"oid": "1cef1fdb-ff52-48c4-8e4e-dfb5ea83d357",

"upn": "admin@contoso.onmicrosoft.com",

"puid": "1003BFFD8EC47CA6",

"sub": "7XpD5OWAXM1OWmKiVKh1FOkKXV4N3OSRol6mz1pxxhU",

"given\_name": "John",

"family\_name": "Doe",

"name": "Contoso, Inc.",

"unique\_name": "admin@contoso.onmicrosoft.com",

"appid": "a6099727-6b7b-482c-b509-1df309acc563",

"appidacr": "1",

"scp": "ActivityFeed.Read ServiceHealth.Read",

"acr": "1"

}

### **Request an access token by using client credentials**

After the tenant ID is known, your application can make service-to-service calls to Azure AD to request additional access tokens as they expire. These tokens include information only about the requesting application and not about the admin that originally granted consent. Service-to-service calls require that your application use an X.509 certificate to create client assertion in the form of a base64-encoded, SHA256 signed JWT bearer token.

When you are developing your application in .NET, you can use the [Microsoft Authentication Library (MSAL)](https://learn.microsoft.com/en-us/azure/active-directory/develop/msal-overview) to create client assertions. Other development platforms should have similar libraries.

**NOTE-** Service-to-service (S2S) authentication is a type of authentication that allows integrations to run without user interaction. S2S authentication uses the Client Credentials OAuth 2.0 Flow. This flow allows you to access resources using the identity of an application.

An un-encoded JWT token consists of a header and payload that have the following properties.

HEADER:

{

"alg": "RS256",

"x5t": "{thumbprint of your X.509 certificate used to sign the token",

}

PAYLOAD:

{

"aud": "https://login.windows.net/{tenantid}/oauth2/token",

"iss": "{your app client ID}",

"sub": "{your app client ID}",

"jti": "{random GUID}",

"nbf": "{epoch time, before which the token is not valid}",

"exp": "{epoch time, after which the token is not valid}"

}

The client assertion is then passed to Azure AD as part of a service-to-service call to request an access token. When using client credentials to request an access token, use an HTTP POST to a tenant-specific endpoint, where the previously extracted and stored tenant ID is embedded in the URL.

[https://login.windows.net/{tenantid}/oauth2/token](https://login.windows.net/%7btenantid%7d/oauth2/token)

The body of the POST contains the following:

resource=https%3A%2F%2Fmanage.office.com&amp;client\_id={your\_app\_client\_id}&amp;grant\_type=client\_credentials&amp;client\_assertion\_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&amp;client\_assertion={encoded\_signed\_JWT\_token}

## **Build your app**

Now that you've registered your app in Azure AD and configured it with the necessary permissions, you're ready to build your app. The following are some of the key aspects to consider when designing and building your app.

* **The consent experience**. To obtain consent from your customers, you must direct them in a browser to the Azure AD website, using the specially constructed URL described previously, and you must have a website to which Azure AD will redirect the admin once they grant consent. This website must extract the authorization code from the URL and use it to request an access token from which it can obtain the tenant ID.
* **Store the tenant ID in your system**. This will be needed when requesting access tokens from Azure AD and when calling the Office Management APIs.
* **Manage access tokens**. You'll need a component that requests and manages access tokens as needed. If your app calls the APIs periodically, it can request tokens on demand, or if it calls the APIs continuously to retrieve data, it can request tokens at regular intervals (for example, every 45 minutes).
* **Implement a webhook listener**. As needed by the particular API you are using.
* **Data retrieval and storage**. You'll need a component that retrieves data for each tenant, either by using continuous polling or in response to webhook notifications, depending on the particular API you are using.