ABBYY Vantage

Microsoft Outlook integration to send Email using Graph API

Developed by: Rahul Kapoor

Table of Contents

About ABBYY Vantage Integration to Send Email

ABBYY Vantage provides content IQ skills that allow you to turn unstructured content into structured, actionable information by means of classification and extraction, as well as processing specific document types like invoices, orders, W2 forms, and more.

ABBYY Vantage integration with Graph API is a module that sends email from ABBYY Vantage using MS Outlook email account. It can send a formatted email, adding source or output file of Vantage as attachment. You can also send fields extracted by Vantage in the body of email.

System Requirements and Limitations

You will require an ABBYY Vantage account, a valid subscription for ABBYY Vantage, and a Vantage user that is assigned the Skill Designer role to configure and to run your workflow.

You will also need an Azure AD account that has the permissions required to access Microsoft Outlook.

ABBYY Vantage Integration to Send Email works with:

- ABBYY Vantage 2.5 or later,
- Outlook 365.

Installing the Connector

ABBYY Vantage Integration with Graph API is a script that runs in an Output or Custom Activity of a Process Skill.

The current version of ABBYY Vantage Integration with Graph API is configured by modifying the script (see Configuring the Connector below).

Using ABBYY Vantage to Send Email

Registering the Application in Microsoft Azure

In order to establish a connection between Outlook API and Vantage, you need to register an application in Azure AD and specify the credentials for this app when modifying the Connector script.

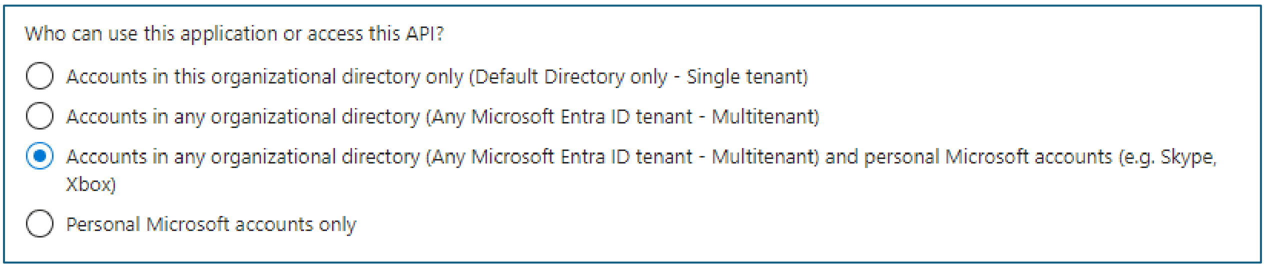
To create an application, an Azure Active Directory tenant with application registration and editing permissions is required.

You can switch to the correct directory on the Portal settings | Directories + subscriptions page.

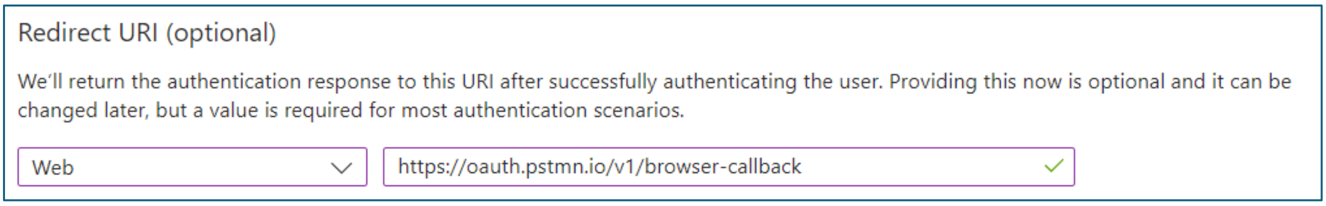
Registering the application and creating a client secret

1. Navigate to the App registrations page.
2. Click **New registration**.

3. Specify a name for your application and select option “Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)”.



4. Also set the redirect URL to https://oauth.pstmn.io/v1/browser-callback
   Set the platform as Web.



5. Click **Register** at the bottom of the page.

6. Copy the **Application (client) ID** and **Directory (tenant) ID** to a text file. Then click **Add a certificate or secret**.

6. In the dialog box that will open, specify a name for the client secret and an expiration date.

Note: The maximum expiration date is 24 months. After the secret has expired, you have to update it following the instructions in Appendix: Updating Client secret.

7. Click **Add**. This will close the dialog and display information about your new client secret. Copy **Value** to a text file. You will need it together with the Client Id you saved to configure the Connector.

Important! You must copy and save the **Value** at this step, since you will not be able to access it again once you close the page.

Setting up application permissions

1. Navigate to the **API permissions** tab.

2. Click **Add permission**.

3. In the dialog that will open, select the **Microsoft Graph** section.

4. Select **Application permissions**.

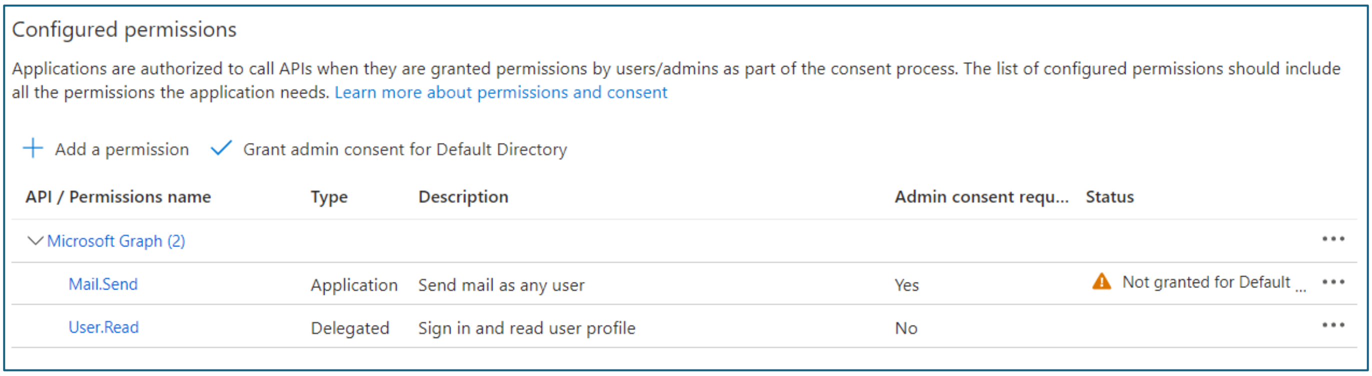
5. Add the following permissions:

    - Mail.Send

Note: You can search for a permission either by browsing the list or by typing its name in the search box.

6. Click **Add permissions**. This will close the dialog and display the selected permissions.

    Some permissions may require admin consent as follows:



If you don't have sufficient rights to grant this consent, you should contact your system administrator.

Getting Refresh Token

Delegated permissions are used in the delegated access scenario. They're permissions that allow the application to act on a user's behalf. To *access* a protected resource like email, your application needs the resource owner's *authorization*. The resource owner can *consent* to or deny your app's request. This section details the involved for an app to get access on behalf of a user using a popular flow called the [OAuth 2.0 authorization code grant flow](). The first step in the authorization code flow is for the user to authorize the app to act on their behalf.

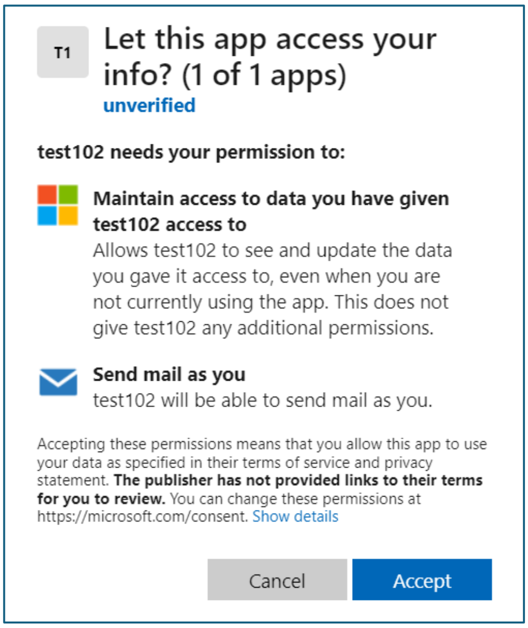
In the flow, the app redirects the user to the Microsoft identity platform /authorize endpoint. Through this endpoint, Microsoft Entra ID signs the user in and requests their consent for the permissions that the app requests. After consent is obtained, Microsoft Entra ID will return an authorization **code** to the app. The app can then redeem this code at the Microsoft identity platform /token endpoint for an access token.

1. To get the code, copy paste this URL in your browser and login using your outlook account:

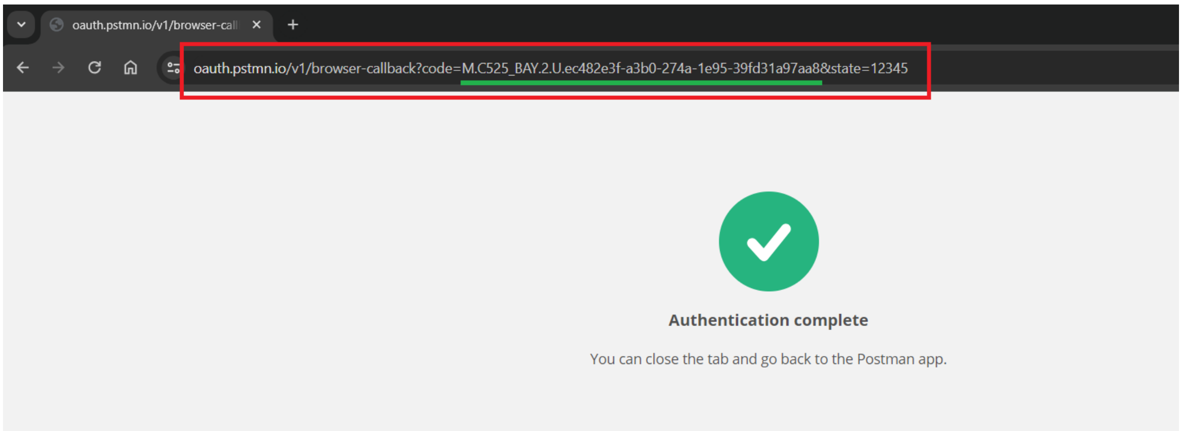
https://login.microsoftonline.com/common/oauth2/v2.0/authorize?client_id=b6baba66-666e-6666-6ca6-6dd6b666d666&response_type=code&redirect_uri=https://oauth.pstmn.io/v1/browser-callback&response_mode=query&scope=offline_access%20Mail.Send&state=12345

Note: Replace highlighted value with your Client Id.

2. Grant the permission



3. Once the Authentication is complete, copy the browser URL.



4. This URL contains the authorization **code**:

   https://oauth.pstmn.io/v1/browser-callback?code=M.C525_BAY.2.U.ec482e3f-a3b0-274a-1e95-39fd31a97aa8&state=12345

5. Next step is to request an Access & Refresh Token. The app uses the authorization code received in the previous step to request an access token by sending a POST request to the /token endpoint.

   You need to make a POST request to get the token. You can use Postman Application and below cURL to make request:

```
curl --location --request POST 'https://login.microsoftonline.com/common/oauth2/v2.0/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'client_id=11111111-1111-1111-1111-111111111111' \
--data-urlencode 'scope=Mail.Send' \
--data-urlencode 'code=M0ab92efe-b6fd-df08-87dc-2c6500a7f84d' \
--data-urlencode 'redirect_uri=https://oauth.pstmn.io/v1/browser-callback' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode 'client_secret=zHF8Q~Krjqh4r...''
```

   Note: You need to modify the highlighted parameters.

6. In the response, you'll get the Access token & Refresh Token.


```
{
    "token_type": "Bearer",
    "scope": "Mail.Send",
    "expires_in": 3600,
    "ext_expires_in": 3600,
    "access_token": "EwBYA8l6BAAUbDba3x2OMJElkF7gJ4z/VbCPEz0AAUFtrtWOd0
        +NvwGwsiz07kZck1FBUMYvzSfoFnrvgefn+wOipHApAohfGh9ZYkFh4rG8nFctr
        +2SWFZ4vJZUbVsroEC3mfVBV8vyIXvp7dxGBmAQgsP4gmPhBBBTIE5JK68lvJmY
        cykFbZBsLFEdctXPBaqLkk31+0bokDZgAACDd2QJrmdhMAKAKbO9+bftEMkqi/F
        +Bj95oDjJNer61VB560FsRg1sDBVhEWyFQJ85MODikgSv6tM8v05MVb2s/V8L+r
        +iTlR0qjU7KsaocPggKrNkdVDKjsfjqk3oKVIKXSOxQNipqNRUlwDnwz3ND9Y0R
        eYNRRv4OfRhUFEermRTGuFrtmoqlti6FuvIfpBNBrkgPY33T4ReYVBzzf1td15f
        +y2bg07tUxIhntl413x8IoXiiky6NJJ+OccYRxjtDz0wKzG8zP3HqQvpT6RjXYn
        +bEUoKpnArsPoPLrWb5nsta6Wpb5FwxDf8aVyuQVtE2sYaNOekAqT1x9hYwJz1l
        +gJdiPtjaoELVwZO5EKFp18Qit65YzuFkz63tY+8hlwC7jFtJ3SVJcEN43qXMoI
        +AfXt98vvuIMsI3tH6fs7d2+QFfZxsAg==",
    "refresh_token": "M.C525_BAY.0.U.
        -CgPJNN8nENZ45cb3ynMUCVVXPrr78ARoVO0YBKIhks5HYQUo8anPBWoJ8TR*qG
        AQMquXBvcX8vrCxQHnm5!QjFm3IqIwwW*RRYC1dwjJu5NN09w4m2TV27CUbpzyV
        YieLPTgxw*CevT9H*Aw4!*HcRI0TCuvjebif3bkO1dbGCupDq9m0Nv8EoqBBSEu
}
```


Configuring the Connector

To configure ABBYY Vantage Integration with Graph API, you should do the following by modifying below variables in the script in the Custom Activity:

Configure custom activity

Create an algorithm detailing how the custom activity will work. It may transform the transaction's data, send and receive dat until specific actions have been completed in the external system. See Documentation for more info

Functions ▾ IF = > < AND OR NOT FOR WHILE

```
// Define the client ID, client secret, and refresh token for your Azure AD application
const clientId = "Enter-Client-Id";
const clientSecret = "Enter-Client-Secret";
const refreshToken = "Enter-Refresh-Token";
```

When you set up Integration to Send Email, you also need to configure the email content:

Below you will find a sample **email content** variable:

```
// Create the email content with the extracted data
var raw = JSON.stringify({
    "message": {
        "subject": "Invoice Processed in Vantage",
        "body": {
            "contentType": "HTML",
            "content": "<p><b><i>This is an email sent from Vantage.</i></b></p><p>A
new invoice of <Vendor-Name> with Invoice Number <Invoice-Number> was processed by
Abbyy Vantage.</p>"
        },
        "toRecipients": [{
            "emailAddress": {
                "address": "rahul.kapoor@abbyy.com"
```

```
            }
        }],
        "ccRecipients": [{
            "emailAddress": {
                "address": "rahul.kapoor@abbyy.com"
            }
        }],
        "attachments": [
        {
            "@odata.type": "#microsoft.graph.fileAttachment",
            "name": sourceFileName,
            "contentType": "text/plain",
            "contentBytes": sourceFileContent
        }
        ]
    },
    "saveToSentItems": "false"
});
```

Appendix: Updating Client secret.

The Client secret value is used for serverside client identification and constitutes confidential information. For security purposes, data like this should periodically be updated. Some services like Azure Active Directory limit the validity period for such data.

Updating Client secret in Microsoft Azure

1. Navigate to the App registrations page and select the application used for data export to Microsoft Outlook API.

2. Navigate to the **Certificates & secrets** tab and click **New client secret**.

3. In the dialog box that will open, specify a name for the client secret and its expiration date.

4. Click **Add**. This will close the dialog and display information about the new client secret. It is important that you copy and save the **Value**, since you will not be able to access it again once you close the page.

5. If the current client secret has not expired yet, you can delete it in order to only be able to use the new client secret to identify the client.

6. Update the **client_secret** value in the script used in the Custom or Output Activity of a Process skill.

7. Publish the skill.