Get an Azure Active Directory token using Azure Active Directory Authentication Library

07/15/2020 • 9 minutes to read •

In this article

Configure an app in Azure portal
Get an Azure Active Directory access token
Use an Azure AD access token to access the Databricks REST API
Refresh an access token

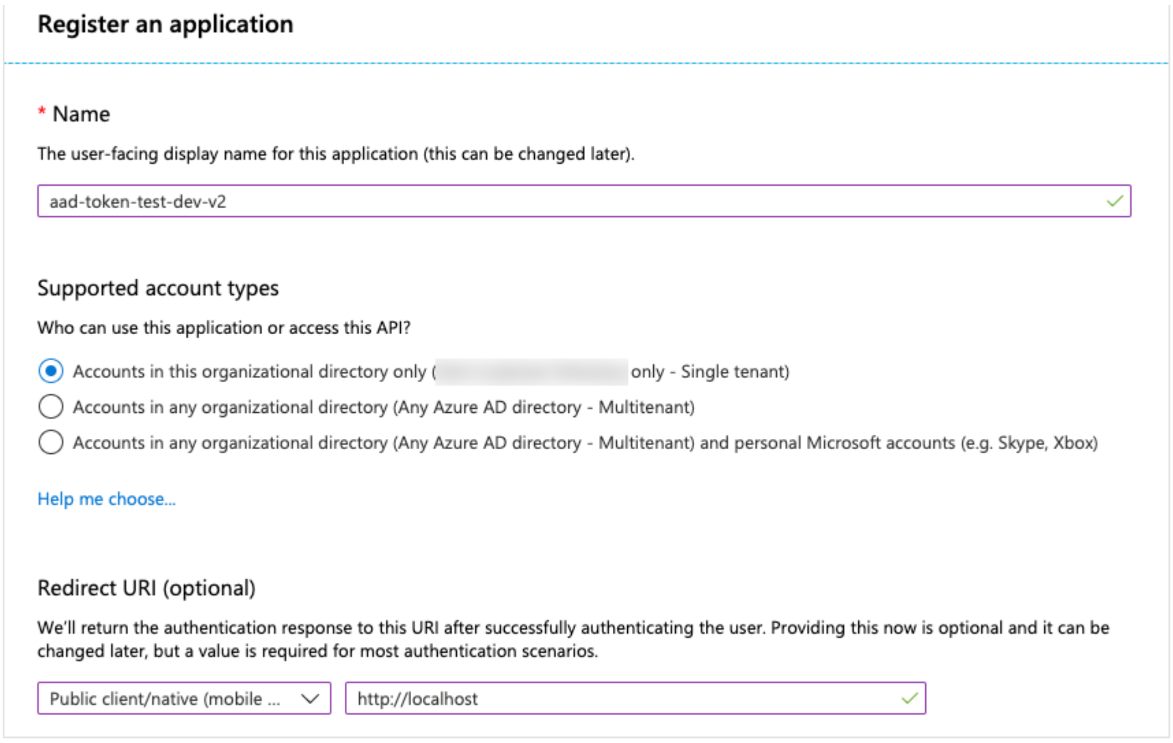
You can use the Azure Active Directory Authentication Library (ADAL) to acquire Azure Active Directory (Azure AD) access tokens programatically. This article describes basic usage of the ADAL library and required user inputs, with Python examples.

You can also define a service principal in Azure Active Directory and get an Azure AD access token for the service principal rather than a user. See Get an Azure Active Directory token using a service principal.

Configure an app in Azure portal

1. Register an application with the Azure AD endpoint in Azure portal. Follow the instructions in Quickstart: Register an app with the Azure Active Directory v1.0 endpoint. Alternatively, you can use an app that is already registered.

   In the Redirect URI field, select **Public client/native (mobile & desktop)** and enter a redirect URI. In the following example, the redirect URI value is `http://localhost.`



Register an application

* Name

The user-facing display name for this application (this can be changed later).

aad-token-test-dev-v2

Supported account types

Who can use this application or access this API?

[✓ Accounts in this organizational directory only ([redacted] only - Single tenant)]

Accounts in any organizational directory (Any Azure AD directory - Multitenant)

Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

Help me choose...

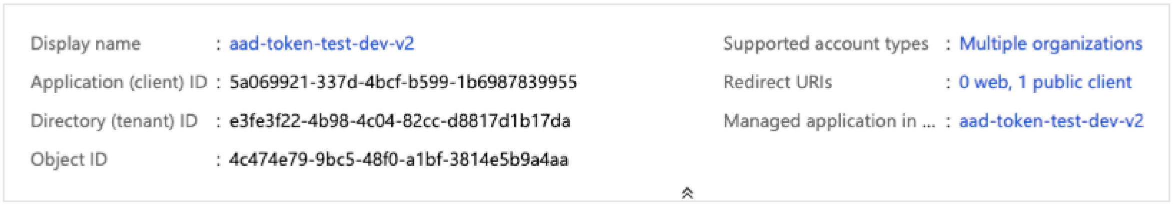
Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Public client/native (mobile ... | http://localhost

2. Click **Register**.

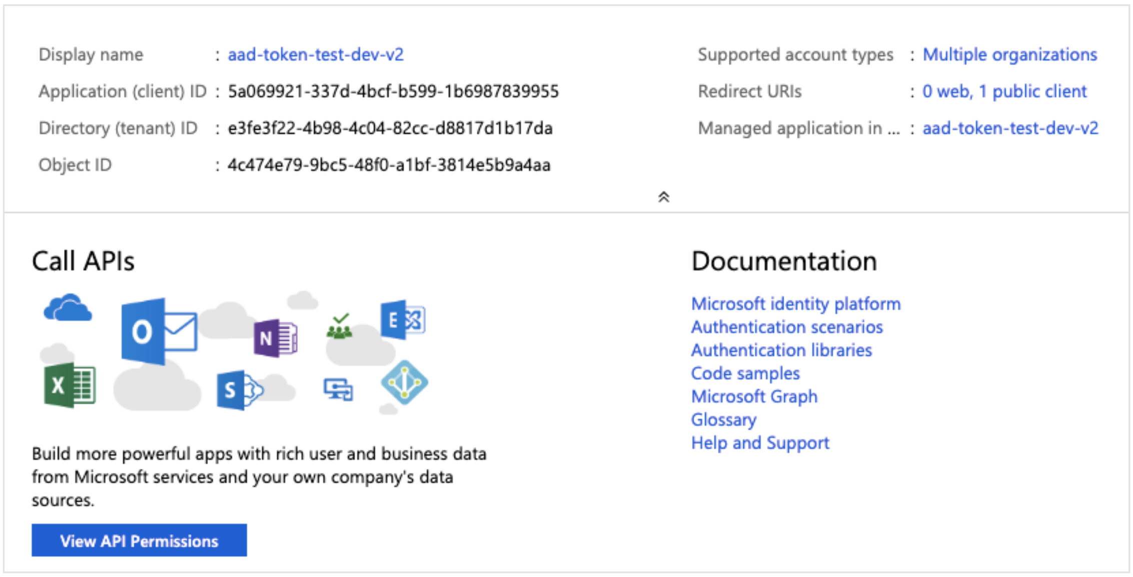
3. Go to **App registrations > View all applications** and select the app. Copy the **Application (client) ID**.



Display name	: aad-token-test-dev-v2	Supported account types	: Multiple organizations
Application (client) ID	: 5a069921-337d-4bcf-b599-1b6987839955	Redirect URIs	: 0 web, 1 public client
Directory (tenant) ID	: e3fe3f22-4b98-4c04-82cc-d8817d1b17da	Managed application in ...	: aad-token-test-dev-v2
Object ID	: 4c474e79-9bc5-48f0-a1bf-3814e5b9a4aa		

4. Add **AzureDatabricks** to the **Required permissions** of the registered application. You must be an admin user to perform this step. If you encounter a "permission" problem performing this step, contact your administrator for help.

   a. On the application page, click **View API Permissions**.



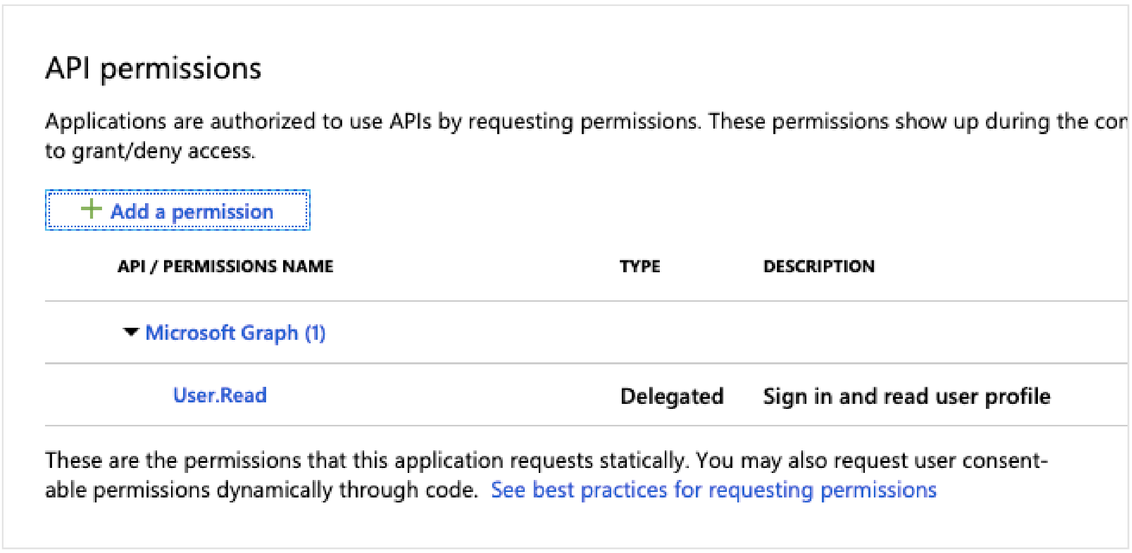
Display name	: aad-token-test-dev-v2	Supported account types	: Multiple organizations
Application (client) ID	: 5a069921-337d-4bcf-b599-1b6987839955	Redirect URIs	: 0 web, 1 public client
Directory (tenant) ID	: e3fe3f22-4b98-4c04-82cc-d8817d1b17da	Managed application in ...	: aad-token-test-dev-v2
Object ID	: 4c474e79-9bc5-48f0-a1bf-3814e5b9a4aa		

Call APIs

Build more powerful apps with rich user and business data from Microsoft services and your own company's data sources.

Documentation

b. Click **Add a Permission**.



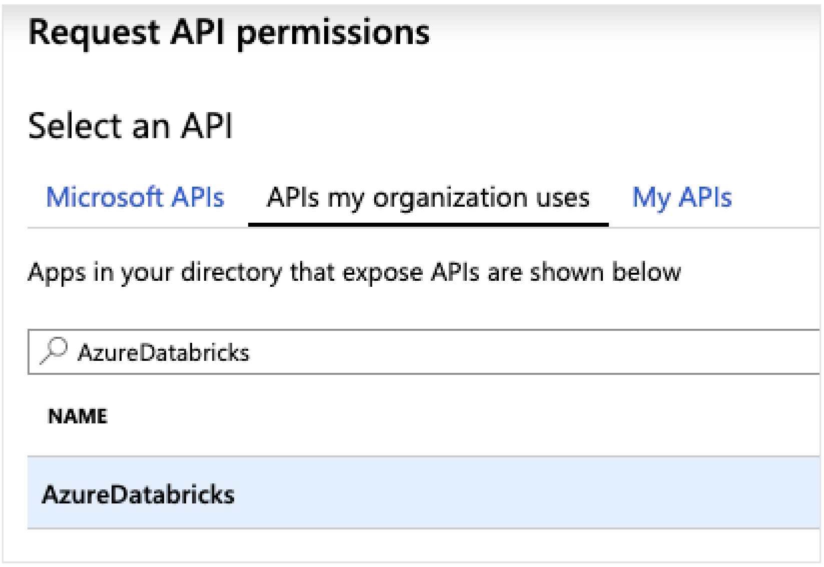
API permissions

Applications are authorized to use APIs by requesting permissions. These permissions show up during the cor
to grant/deny access.

API / PERMISSIONS NAME	TYPE	DESCRIPTION
▾ Microsoft Graph (1)		
User.Read	Delegated	Sign in and read user profile

These are the permissions that this application requests statically. You may also request user consent-able permissions dynamically through code. See best practices for requesting permissions

c. Select the tab **APIs my organization uses**, search for **AzureDatabricks** and select it.



Request API permissions

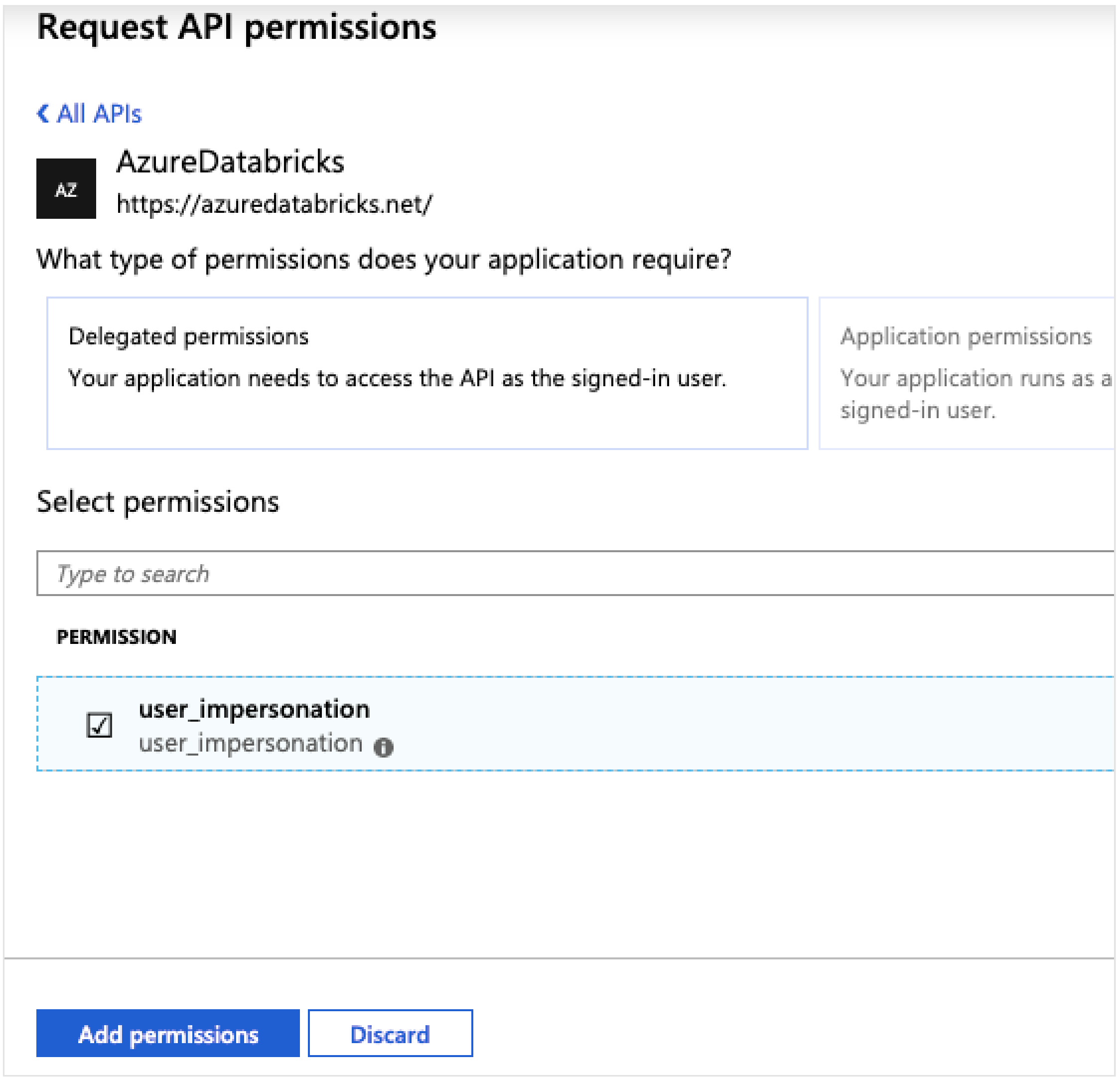
Select an API

Apps in your directory that expose APIs are shown below

AzureDatabricks

NAME
AzureDatabricks

d. Select **user_impersonation**, then click **Add permissions**.



Request API permissions

< All APIs

AZ AzureDatabricks
https://azuredatabricks.net/

What type of permissions does your application require?

Delegated permissions
Your application needs to access the API as the signed-in user.

Application permissions
Your application runs as a
signed-in user.

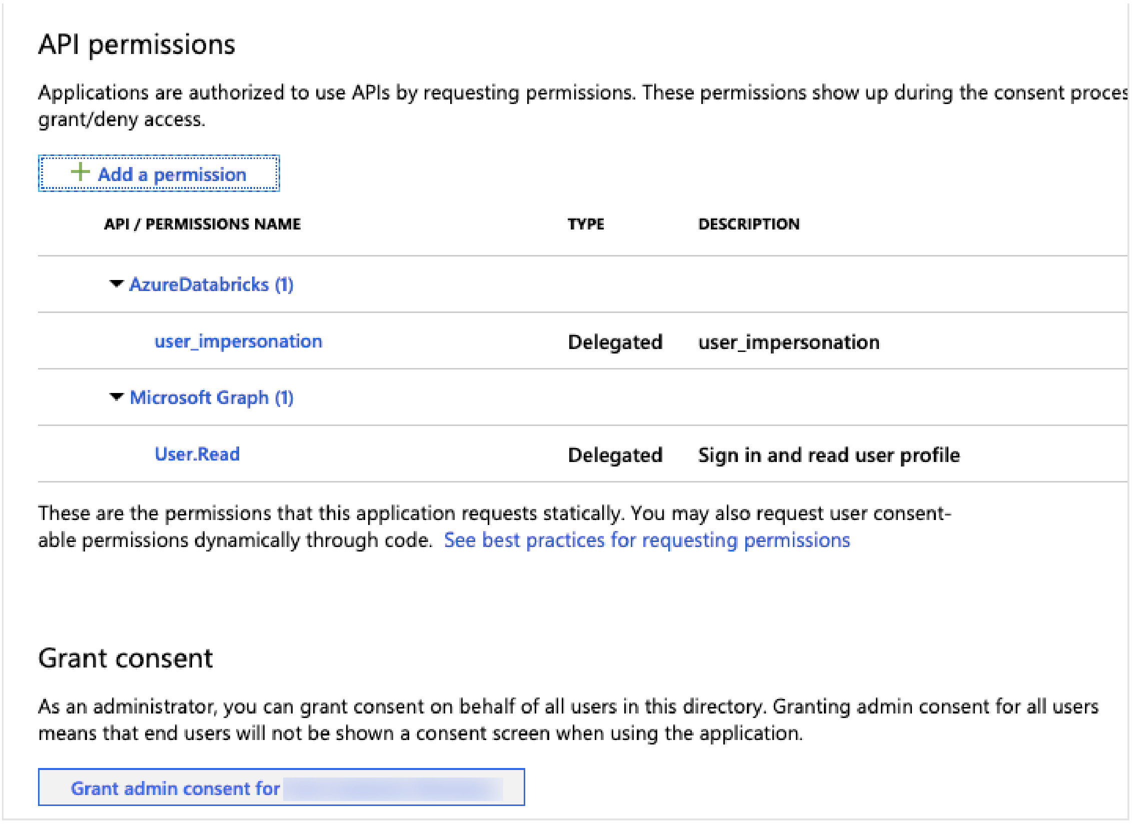
Select permissions

Type to search

PERMISSION

☑ user_impersonation
user_impersonation

e. Click **Grant admin consent for ###** and then **Yes**. To perform this step you must be an admin user or have the privilege to grant consent to the application. If you skip this step, you must use the Authorization code flow (interactive) the first time you use the application to provide consent. After that, you can use the Username-password flow (programmatic) method.



API permissions

Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent proces grant/deny access.

API / PERMISSIONS NAME	TYPE	DESCRIPTION
▼ AzureDatabricks (1)		
user_impersonation	Delegated	user_impersonation
▼ Microsoft Graph (1)		
User.Read	Delegated	Sign in and read user profile

These are the permissions that this application requests statically. You may also request user consent-able permissions dynamically through code. See best practices for requesting permissions

Grant consent

As an administrator, you can grant consent on behalf of all users in this directory. Granting admin consent for all users means that end users will not be shown a consent screen when using the application.

You can add additional users to the application. For more information, see Assign users and groups to an application in Azure Active Directory. A user will not be able to obtain a token without required permissions.

Get an Azure Active Directory access token

To get an access token, you can use either:

- Authorization code flow (interactive)
- Username-password flow (programmatic)

You must use the authorization code flow to get the Azure AD access token if:

- Two factor authentication is enabled in Azure AD.
- Federated authentication is enabled in Azure AD.
- You are not granted consent to the registered application during application registration.

If you have the authority to sign in with a username and password, you can use the username-password flow to obtain an Azure AD access token.

Authorization code flow (interactive)

There are two steps to acquire an Azure AD access token using the authorization code flow.

1. Obtain the authorization code, which launches a browser window and ask for user login. The authorization code is returned after the user successfully logs in.
2. Use the authorization code to acquire the access token. A refresh token will be returned at the same time and can be used to refresh the access token.

Get the authorization code

Note

You must successfully pass this step before moving forward. If you encounter a “permission” problem, contact your administrator for help.

Parameter	Description
Tenant ID	Tenant ID in Azure AD.
Client ID	The ID of the application registered in Configure an app in Azure portal.
Redirect URI	One of the redirect URIs in your registered application (for example, `http://localhost`). The authentication responses are sent to this URI with the authorization code piggybacked.

Get the authorization code using a browser

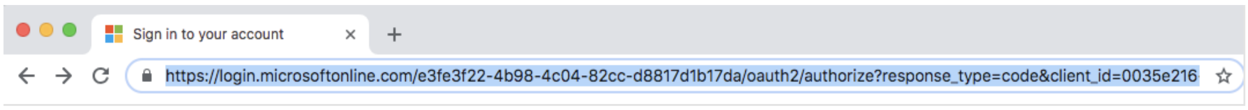
This is the interactive method to obtain an Azure AD access token.

You can request the authorization code by sending an HTTP request in the browser. Reference Request an authorization code for more information. Replace the fields in the following URL example accordingly:

Copy

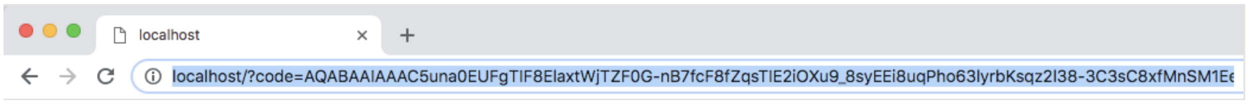
```
// Line breaks for legibility

https://login.microsoftonline.com/<tenant>/oauth2/authorize?client_id=
<client-id>
&response_type=code
&redirect_uri=<redirect URI in encoded format: e.g.,
http%3A%2F%2Flocalhost>
&response_mode=query
&resource=2ff814a6-3304-4ab8-85cb-cd0e6f879c1d
&state=<a random number or some encoded info>
```

Paste the URL into your browser and sign in to Azure when you are prompted.



https://login.microsoftonline.com/e3fe3f22-4b98-4c04-82cc-d8817d1b17da/oauth2/authorize?response_type=code&client_id=0035e216

After successful login, the authorization code is attached to the code field in the returned URL. Save the code for later use.



localhost/?code=AQABAAIAAAC5una0EUFgTIF8ElaxtWjTZF0G-nB7fcF8fZqsTIE2iOXu9_8syEEi8uqPho63lyrbKsqz2I38-3C3sC8xfMnSM1E

Get the authorization code programmatically

You can alternatively use the semi-programmatic way to obtain the authorization code. The following code snippet will open a browser for user login. After successful login, the code will be returned.

1. Install the ADAL Python SDK using `pip install adal`.
2. Use the Selenium library to open the browser:

Bash

```bash
pip install selenium
```

3. Download the browser driver and extract the executable file into your `PATH`. In this example, the Chrome driver is used. Download the Chrome driver.
4. Run the following code snippet to obtain the authorization code:

Python

```python
from adal import AuthenticationContext
from selenium import webdriver
from urlparse import urlparse, parse_qs
import time

authority_host_url = "https://login.microsoftonline.com/"
azure_databricks_resource_id = "2ff814a6-3304-4ab8-85cb-
cd0e6f879c1d"

# Required user input
user_parameters = {
     "tenant" : "<tenant-id>",
     "client_id" : "<application-id>",
     "redirect_uri" : "<redirect-uri>"
}

TEMPLATE_AUTHZ_URL =
('https://login.windows.net/{}/oauth2/authorize?'+
     'response_type=code&client_id={}&redirect_uri={}&'+
     'state={}&resource={}')
# the auth_state can be a random number or can encoded some info
# about the user. It is used for preventing cross-site request
# forgery attacks
auth_state = 12345
# build the URL to request the authorization code
authorization_url = TEMPLATE_AUTHZ_URL.format(
            user_parameters['tenant'],
            user_parameters['client_id'],
            user_parameters['redirect_uri'],
            auth_state,
            azure_databricks_resource_id)

def get_authorization_code():
  # open a browser, here assume we use Chrome
  dr = webdriver.Chrome()
  # load the user login page
  dr.get(authorization_url)
  # wait until the user login or kill the process
  code_received = False
  code = ''
  while(not code_received):
      cur_url = dr.current_url
      if cur_url.startswith(user_parameters['redirect_uri']):
          parsed = urlparse(cur_url)
          query = parse_qs(parsed.query)
          code = query['code'][0]
          state = query['state'][0]
          # throw exception if the state does not match
          if state != str(auth_state):
              raise ValueError('state does not match')
          code_received = True
          dr.close()

  if not code_received:
      print 'Error in requesting authorization code'
      dr.close()
  # authorization code is returned. If not successful,
  # then an empty code is returned
  return code
```

Use the authorization code to obtain the access and refresh tokens

Python Copy

```
def get_refresh_and_access_token():
  # configure AuthenticationContext
  # authority URL and tenant ID are used
  authority_url = authority_host_url + user_parameters['tenant']
  context = AuthenticationContext(authority_url)

  # Obtain the authorization code in by a HTTP request in the browser
  # then copy it here or, call the function above to get the
authorization code
  authz_code = get_authorization_code()

  # API call to get the token, the response is a
  # key-value dict
  token_response = context.acquire_token_with_authorization_code(
    authz_code,
    user_parameters['redirect_uri'],
    azure_databricks_resource_id,
    user_parameters['clientId'])

  # you can print all the fields in the token_response
  for key in token_response.keys():
    print str(key) + ': ' + str(token_response[key])

  # the tokens can be returned as a pair (or you can return the full
  # token_response)
  return (token_response['accessToken'], token_response['refreshToken'])
```

Username-password flow (programmatic)

If you have the authority to sign in with a username and password you can use this programmatic method to obtain an Azure AD access token.

Parameter	Description
Tenant ID	Tenant ID in Azure AD.
Client ID	The application ID of the application registered in Configure an app in Azure portal.
Username and Password	The username (i.e., the email address when logging into Azure portal) and password of the user in the tenant.

You can use the example code below to acquire an Azure AD access token with a username and password. Error handling is omitted. For a list of possible errors when getting the token, see the get_token function definition in the ADAL GitHub repository.

Python Copy

```
from adal import AuthenticationContext

authority_host_url = "https://login.microsoftonline.com/"
# the Application ID of  AzureDatabricks
azure_databricks_resource_id = "2ff814a6-3304-4ab8-85cb-cd0e6f879c1d"
```

```
# Required user input
user_parameters = {
   "tenant" : "<tenant-id>",
   "client_id" : "<application-id>",
   "username" : "<username>",
   "password" : "<password>"
}

# configure AuthenticationContext
# authority URL and tenant ID are used
authority_url = authority_host_url + user_parameters['tenant']
context = AuthenticationContext(authority_url)

# API call to get the token
token_response = context.acquire_token_with_username_password(
  azure_databricks_resource_id,
  user_parameters['username'],
  user_parameters['password'],
  user_parameters['client_id']
)

access_token = token_response['accessToken']
refresh_token = token_response['refreshToken']
```

Use an Azure AD access token to access the Databricks REST API

You can use an Azure AD token to call the Databricks REST API. In addition to the Azure AD token, also provide the following information in the request header:

- If you are a non-admin user and do not require admin privilege: the `X-Databricks-Org-Id` header.
- If you are a non-admin user and you want to log in to Azure Databricks as an admin user: the `X-Databricks-Azure-Workspace-Resource-Id` header. You must be in a Contributor or Owner role on the workspace resource in Azure.

An admin user will always be logged in as admin regardless of the header.

Parameter	Description
Databricks workspace org ID	Header: `X-Databricks-Org-Id`. To find the workspace org ID, see Get workspace, cluster, notebook, model, and job identifiers.

Parameter	Description
Databricks workspace resource ID	Header: `X-Databricks-Azure-Workspace-Resource-Id`. The ID of the workspace resource in Azure. You construct it using the Azure subscription ID, resource group name, and workspace resource name.

Python example

This example shows how to list the clusters in an Azure Databricks workspace. It gets the tokens using the `get_refresh_and_access_token` method defined in Use the authorization code to obtain the access and refresh tokens and shows how to construct and use both types of request headers.

Python Copy

```python
import requests

(refresh_token, access_token) =  get_refresh_and_access_token()

def list_cluster_with_aad_token():

  DOMAIN = '<databricks-instance>'
  TOKEN = access_token
  BASE_URL = 'https://%s/api/2.0/clusters/list' % (DOMAIN)

  # set the ORG_ID if it is available.
  # otherwise, include DB_RESOURCE_ID in the header
  ORG_ID = '<workspace-org-id>'

  # information required to build the DB_RESOURCE_ID
  SUBSCRIPTION = '<azure-subscription-id>'
  RESOURCE_GROUP = '<azure-resource-group-name>'
  WORKSPACE = '<databricks-workspace-name-in-azure>'

  DB_RESOURCE_ID =
'/subscriptions/%s/resourceGroups/%s/providers/Microsoft.Databricks/works
paces/%s' % (
    SUBSCRIPTION,
    RESOURCE_GROUP,
    WORKSPACE
  )

  # request header with org_id if org_id is known
  headers_with_org_id = {
    'Authorization' : 'Bearer ' + TOKEN,
    'X-Databricks-Org-Id' : ORG_ID,
  }

  # request header with resource ID if org_id is not available
  # (e.g., the workspace has not been created yet when you call the REST
API)
  headers_with_resource_id = {
    'Authorization' : 'Bearer ' + TOKEN,
    'X-Databricks-Azure-Workspace-Resource-Id' : DB_RESOURCE_ID
  }

  # call the API with org_id if it is known
  response = requests.get(
    BASE_URL,
    headers=headers_with_org_id
  )

  # OR, call the API with resource_id if org_id is not known
  response = requests.get(
    BASE_URL,
    headers=headers_with_resource_id
  )

  print 'response header: ' + str(response.headers)
  print 'the response is: ' + str(response.content)

  try:
    print 'Decoding response as JSON... '
    res_json = response.json()

    for cluster in res_json['clusters']:
        print str(cluster)
  except Exception as e:
    print 'Response cannot be parsed as JSON:'
    print '\t: ' + str(response)
    print 'The exception is: %s' % str(e)
```

`curl` example

This `curl` request uses the `X-Databricks-Org-Id` header. See the Python example for how to construct the `X-Databricks-Azure-Workspace-Resource-Id` header.

Bash Copy

```bash
curl -X GET \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer <aad-access-token>' \
-H 'X-Databricks-Org-Id: <workspace-org-id>' \
https://<databricks-instance>/api/2.0/clusters/list
```

Refresh an access token

If you get a refresh token along with your access token, you can use the refresh token to obtain a new token. By default, the lifetime of access tokens is one hour. You can configure the lifetime of access tokens using the methods in Configurable token lifetimes in Azure Active Directory.

Python Copy

```python
# supply the refresh_token (whose default lifetime is 90 days or longer
[token lifetime])
def refresh_access_token(refresh_token):
  context = AuthenticationContext(authority_url)
  # function link
  token_response = context.acquire_token_with_refresh_token(
                  refresh_token,
                  user_parameters['client_id'],
                  azure_databricks_resource_id)
  # print all the fields in the token_response
  for key in token_response.keys():
      print str(key) + ': ' + str(token_response[key])

  # the new 'refreshToken' and  'accessToken' will be returned
  return (token_response['refreshToken'], token_response['accessToken'])
```

Is this page helpful?

Yes No