# Microsoft Graph API

```
USE ROLE ACCOUNTADMIN;
USE DATABASE SANDBOX;
USE SCHEMA SOURCING;

CREATE OR REPLACE NETWORK RULE  MICROSOFT_GRAPH_API_NETWORK_RULE
  MODE = EGRESS
  TYPE = HOST_PORT
  VALUE_LIST = ('graph.microsoft.com');


CREATE OR REPLACE SECURITY INTEGRATION MICROSOFT_GRAPH_API_OAUTH_SECURITY_INTEGRATION
  TYPE = API_AUTHENTICATION
  AUTH_TYPE = OAUTH2
  ENABLED = TRUE
  OAUTH_AUTHORIZATION_ENDPOINT = 'https://login.microsoftonline.com/e4e1abd9-eac7-4a71-ab52-da5c998aa7ba/oauth2
/token'
  OAUTH_TOKEN_ENDPOINT = 'https://login.microsoftonline.com/e4e1abd9-eac7-4a71-ab52-da5c998aa7ba/oauth2/v2.0
/token'
  OAUTH_CLIENT_ID = '<client-id>'
  OAUTH_CLIENT_SECRET =  '<client-secret>'
  OAUTH_GRANT = 'CLIENT_CREDENTIALS'
  OAUTH_ALLOWED_SCOPES = ('https://graph.microsoft.com/.default')
  ;

CREATE OR REPLACE SECRET MICROSOFT_GRAPH_API_TOKEN_SECRET
    TYPE = oauth2
    API_AUTHENTICATION = MICROSOFT_GRAPH_API_OAUTH_SECURITY_INTEGRATION
    ;


CREATE OR REPLACE EXTERNAL ACCESS INTEGRATION MICROSOFT_GRAPH_API_OAUTH_SECURITY_INTEGRATION_EXTERNAL_ACCESS
  ALLOWED_NETWORK_RULES = (MICROSOFT_GRAPH_API_NETWORK_RULE)
  ALLOWED_AUTHENTICATION_SECRETS = (MICROSOFT_GRAPH_API_TOKEN_SECRET)
  ENABLED = TRUE
  ;


CREATE OR REPLACE procedure PS_GET_FROM_MICROSOFT_GRAPH (URL STRING, table_name STRING)
  RETURNS STRING
  LANGUAGE PYTHON
  RUNTIME_VERSION = 3.10
  HANDLER = 'get_data'
  EXTERNAL_ACCESS_INTEGRATIONS = (MICROSOFT_GRAPH_API_OAUTH_SECURITY_INTEGRATION_EXTERNAL_ACCESS)
  PACKAGES = ('snowflake-snowpark-python','requests')
  SECRETS = ('cred' = MICROSOFT_GRAPH_API_TOKEN_SECRET)
  EXECUTE AS CALLER
  AS
  $$
import _snowflake
import requests
import json

def get_token():
    return _snowflake.get_oauth_access_token("cred")

def get_headers():
    headers = {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
        'ConsistencyLevel': 'eventual',
        'Authorization': 'Bearer {}'.format(get_token())
    }

    return headers

def get_nextPageURL(last_result):
```

```
    return last_result.get("@odata.nextLink", None)

def get_data(session, URL, table_name):

    headers = get_headers()
    current_url = URL
    session.sql(f"CREATE OR REPLACE TABLE {table_name} (page_id NUMBER, json_data VARIANT);").collect()
    last_result = None;
    page = 0
    while current_url:
        try:
            page += 1

            last_result = requests.get(url=current_url, headers=headers).json()

            if not last_result:
                break

            # Use a parameterized query with placeholders
            query = "INSERT INTO {0} (page_id, json_data) SELECT :1, PARSE_JSON(:2)".format(table_name)
            session.sql(query, (page, json.dumps(last_result))).collect()

            current_url = get_nextPageURL(last_result)

        except Exception as e:
            return f"TABLE NAME:\t{table_name}\nURL:\t{current_url}\nCURRENT PAGE:\t{page}\nERROR:\t{e}\nLAST
RESULT:\t{last_result}"

    return f"TABLE NAME:\t{table_name}\nURL:\t{URL}\nTOTAL PAGES:\t{page}\nLATEST RESULT:\t{last_result}"

  $$;



GRANT USAGE ON PROCEDURE SANDBOX.SOURCING.PS_GET_FROM_MICROSOFT_GRAPH(VARCHAR, VARCHAR) TO ROLE
HRDP_NP_DOMAIN_ADMIN;

--- EXAMPLE
USE ROLE HRDP_NP_DOMAIN_ADMIN;
CALL SANDBOX.SOURCING.PS_GET_FROM_MICROSOFT_GRAPH('https://graph.microsoft.com/v1.0/groups/84ef5297-4f7a-44b9-
8e05-a4f3a50e1a92/members','SANDBOX.TESTING.GROUP_DGRH_PAPLATFORM_ACKNOLEDGEMENT_PRD');

SELECT JD.VALUE:displayName::STRING AS displayName, JD.VALUE:userPrincipalName::STRING AS userPrincipalName, JD.
VALUE
FROM SANDBOX.TESTING.GROUP_DGRH_PAPLATFORM_ACKNOLEDGEMENT_PRD,
LATERAL FLATTEN (JSON_DATA, 'value' ) AS JD
;
```