Graph Reports API Module Tutorial

OEA Curated

Date Updated: May, 2022

Table of Contents

1. Initial Module Setup

i. Update/Install the latest OEA framework

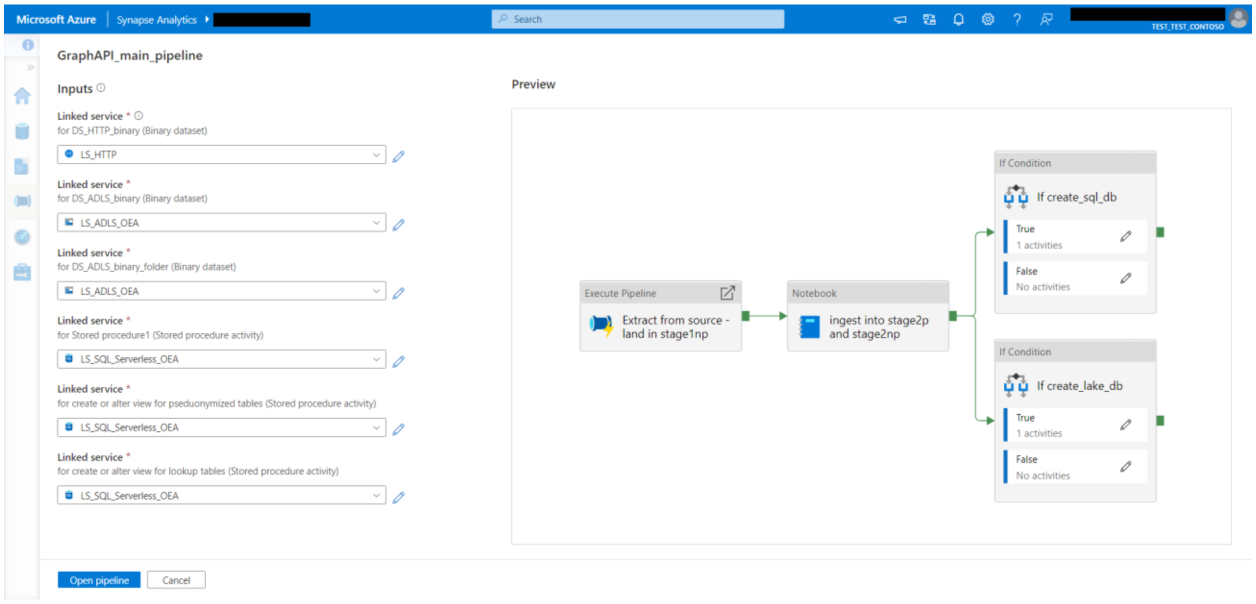
- Make sure that the Azure Synapse environment you're using is up to date. At the date of this tutorial creation, you should be using OEA framework v0.6. This can be done by following the steps outlined on the main OEA GitHub repository landing-page readme.
- You should expect to see many sample OEA pipelines and notebooks in your synapse workspace (ensure that you've followed the steps also outlined/assets provided in the OpenEduAnalytics GitHub under framework).

ii. Cloning OEA GitHub Repository via GitHub Desktop

- To first familiarize yourself with the Graph Reports API module and what the data that Microsoft Graph Reports API has to offer: start by cloning the entire OEA GitHub repository through your own GitHub Desktop.
    - If you've already done this from previous modules, make sure you have the latest updates to the repository by clicking "Pull origin" or "Fetch origin".

iii. Uploading and Using Graph Reports API Module assets into Azure Synapse

- Start by importing the two notebooks for this module: "GraphAPI_py" (the class notebook defining the functions for the Graph module), and "GraphAPI_module_ingestion" (the OEA connector notebook for this module).
    - And then connect your GraphAPI_module_ingestion notebook to a spark pool you've already created. You may need to also import "OEA_py" and "OEA_connector" if not automatically/already imported. Connect the OEA_connector notebook to your spark pool as well.
- Next, import the "GraphAPI_main_pipeline" template, under the Integrate section. The "GraphAPI_copy_test_data" pipeline should automatically be imported.



[redacted] [redacted]

GraphAPI_main_pipeline

Inputs

Linked service *
for DS_HTTP_binary (Binary dataset)
LS_HTTP

Linked service *
for DS_ADLS_binary (Binary dataset)
LS_ADLS_OEA

Linked service *
for DS_ADLS_binary_folder (Binary dataset)
LS_ADLS_OEA

Linked service *
for Stored procedure1 (Stored procedure activity)
LS_SQL_Serverless_OEA

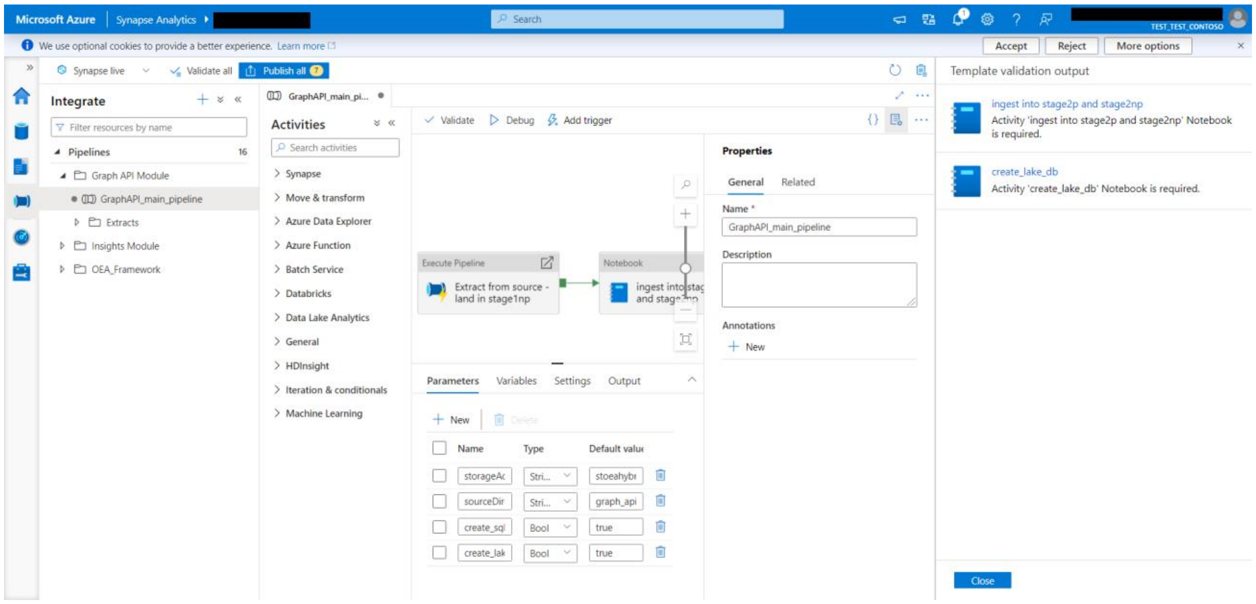
Linked service *
for create or alter view for pseduonymized tables (Stored procedure activity)
LS_SQL_Serverless_OEA

Linked service *
for create or alter view for lookup tables (Stored procedure activity)
LS_SQL_Serverless_OEA

Preview

Execute Pipeline: Extract from source - land in stage1np
Notebook: ingest into stage2p and stage2np
If Condition: If create_sql_db
True: 1 activities
False: No activities
If Condition: If create_lake_db
True: 1 activities
False: No activities

- From here, you will need to properly connect your linked services to the pipeline (note: make sure you've run the shell setup script under framework to use the following linked services); the first one should be "LS_HTTP", then the next two should be "LS_ADLS_OEA", and the last three should be "LS_SQL_Serverless_OEA".
    - You will then be prompted to connect a notebook for the activities "ingest into stage2p and stage2np" and "create_lake_lb". For "ingest into stage2p and stage2np", connect the GraphAPI_module_ingestion.ipynb notebook. For "create_lake_db", open up the pipeline that was imported from the template (it will probably be renamed something like "create_lake_db1" if you already have one a pipeline here) and connect the "OEA_connector" notebook from the OEA framework.



	Name	Type	Default valu
☐	storageAc	Stri...	stoeahybı
☐	sourceDir	Stri...	graph_api
☐	create_sql	Bool	true
☐	create_lak	Bool	true

ingest into stage2p and stage2np
Activity 'ingest into stage2p and stage2np' Notebook is required.

create_lake_db
Activity 'create_lake_db' Notebook is required.

- Make sure the parameter "storageAccount" in the GraphAPI_main_pipeline is changed to match your Synapse workspace's storage account name, and the parameter "timezone" is changed to match your current timezone (for the sake of folder naming)
- Now you're ready to trigger the "GraphAPI_main_pipeline". This should take several minutes, and will automatically pull in the Intune test data, run the Intune module notebook, ingest and transform the data, and then create two databases with the Intune devices tables: s2_graph_api (the lake database), and sqls2_graph_api (the SQL database).

iv. Using the Intune PowerBI Dashboard Template

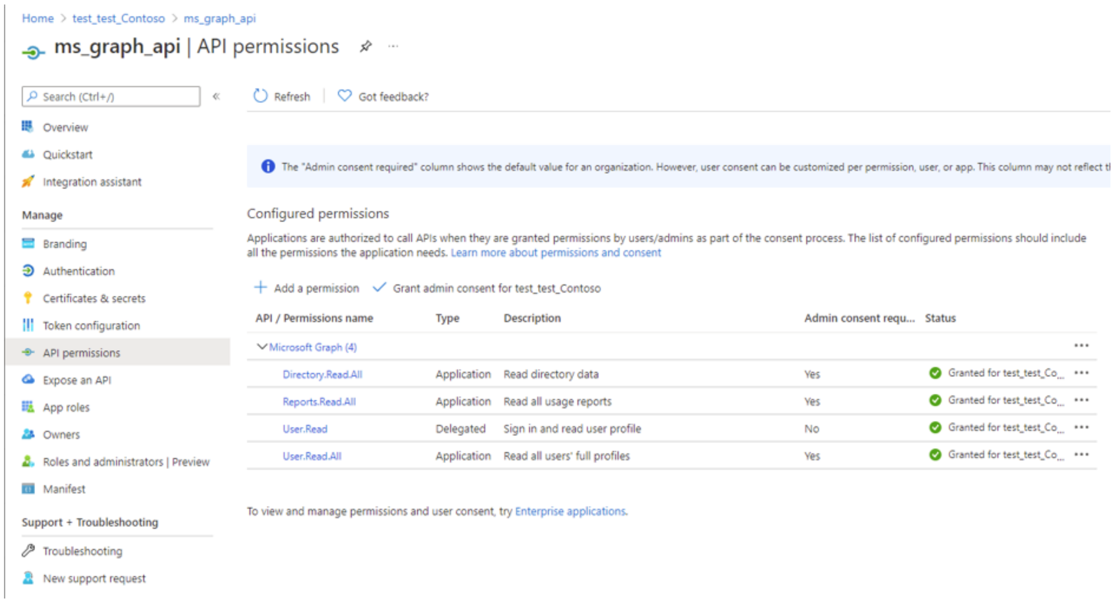
- After the pipeline has completed running and created the Spark databases, you can open your locally downloaded Intune PowerBI Dashboard Template.
    - The dashboard initially is using Import. You'll want to use a DirectQuery of the SQL database sqls2_graph_api; click here for information on how to do this.
- You can interact with this dashboard to gain understanding of what this template can provide within the scope of this module.

2. Graph Reports API Production-Level Data Pipeline Setup

i. Create a linked service for Microsoft Graph Reports API

If you've already walked through the Intune module tutorial, you will not need to create another. Instead, skip to the step "Now in your Synapse Studio…" and follow the steps from here.

- In your Azure environment, first go to AAD -> New registration and create a new app registration specifically for accessing the Graph API for Intune from Synapse.
- Then click "API Permission" -> "Add Permission" and select Microsoft Graph -> select Application Permissions -> choose the "User.Read.All" permission, and click on the "Add permissions" button.
- Follow the same steps above, except now add the "Directory.Read.All", "AuditLog.Read.All", and "Reports.Read.All" permissions for Graph, as application permissions.
- After adding these permissions, you should see something similar to the following screenshot in your own Azure Portal:



Home > test_test_Contoso > ms_graph_api

ms_graph_api | API permissions

The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect t

Configured permissions

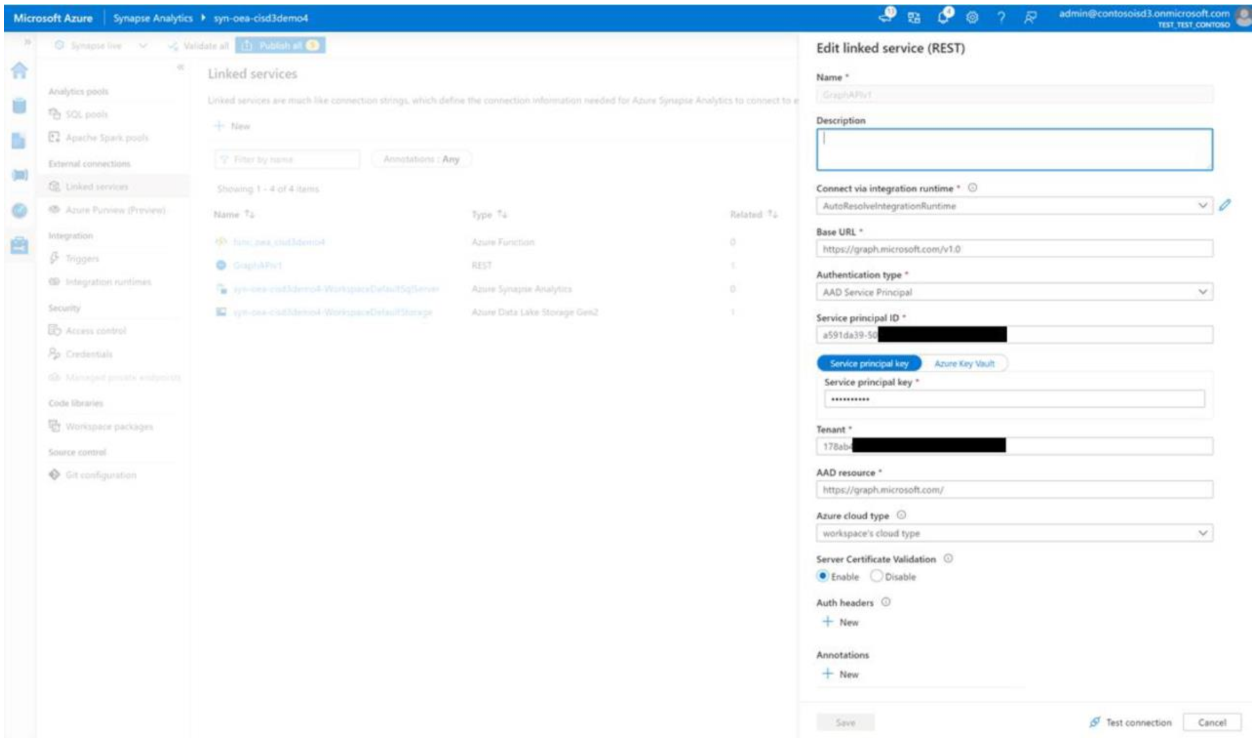
Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. Learn more about permissions and consent

API / Permissions name	Type	Description	Admin consent requ...	Status
Microsoft Graph (4)				
Directory.Read.All	Application	Read directory data	Yes	✓ Granted for test_test_Co...
Reports.Read.All	Application	Read all usage reports	Yes	✓ Granted for test_test_Co...
User.Read	Delegated	Sign in and read user profile	No	✓ Granted for test_test_Co...
User.Read.All	Application	Read all users' full profiles	Yes	✓ Granted for test_test_Co...

To view and manage permissions and user consent, try Enterprise applications.

  - Click on the "Grant admin consent" link once these permissions have been successfully added, so that you can verify all permissions show a status of "Granted" under the status column.

- Now go to "certificates & secrets" and add a new client secret -> copy that value.
- Now in your Synapse Studio, go to "linked services" under "Manage", and add a REST service for Graph Reports API -> select AAD Service Principal as the Authentication type.
- Under the base URL and AAD resource fields, type in: https://graph.microsoft.com/
- Under "Service principal key", paste the value of the secret you copied previously.
- Under "Service principal ID", enter the "Application (client) ID" of the app registration created in the previous step (go to the Overview section of the app registration). After doing so, your Synapse environment should look something similar to the following screenshot:



Edit linked service (REST)

Name *: GraphAPIv1

Description:

Connect via integration runtime *: AutoResolveIntegrationRuntime

Base URL *: https://graph.microsoft.com/v1.0

Authentication type *: AAD Service Principal

Service principal ID *: a591da39-50[redacted]

Service principal key | Azure Key Vault

Service principal key *: ••••••••••

Tenant *: 178ab[redacted]

AAD resource *: https://graph.microsoft.com/

Azure cloud type: workspace's cloud type

Server Certificate Validation: [✓ Enable] Disable

Auth headers

Annotations

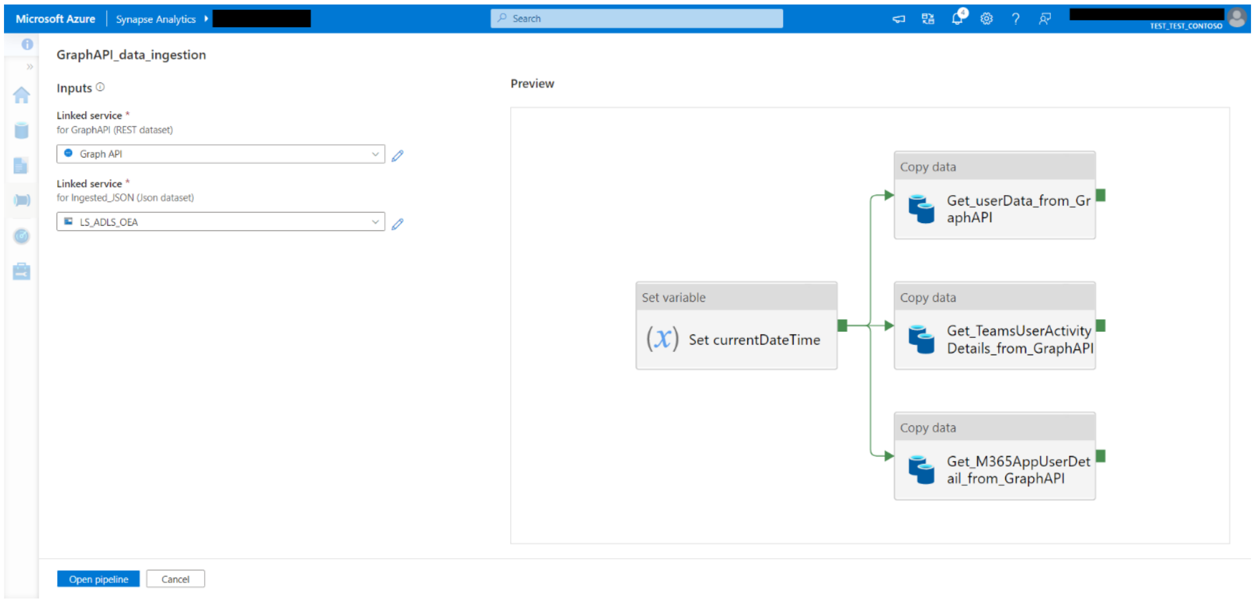
ii. Notes

- As of September 1st, 2021, you will also have to sign in to **https://admin.microsoft.com/AdminPortal** to use your own data (due to the default hashing of userPrincipleNames); go to the admin center and login -> go to Settings -> Org Settings -> “Services” page. Select “Reports”. Uncheck the statement “Display concealed user, group, and site names in all reports,” and then save your changes. Activity Reports in the Microsoft 365 admin center - Microsoft 365 admin | Microsoft Docs.

iii. Using the pipeline template in Synapse for actual Graph Reports API data ingestion

- First, start by deleting all test data previously used. The quickest way to do this is deleting the folder “graph_api” from stage1np, stage2p, and stage2np.
    - Also make sure to drop the test data databases created (s2_graph_api and sqls2_graph_api).

- Next, you will be importing the other pipeline template under “Extracts” on GitHub, titled “GraphAPI_data_ingestion”.


Microsoft Azure | Synapse Analytics ▸ [redacted]
Search
[redacted] TEST_TEST_CONTOSO

GraphAPI_data_ingestion

Inputs

Linked service *
for GraphAPI (REST dataset)
Graph API

Linked service *
for Ingested_JSON (Json dataset)
LS_ADLS_OEA

Preview

Set variable
Set currentDateTime
Copy data
Get_userData_from_GraphAPI
Copy data
Get_TeamsUserActivityDetails_from_GraphAPI
Copy data
Get_M365AppUserDetail_from_GraphAPI
Open pipeline
Cancel

  - From here, you will be prompted to connect two linked services: one for “GraphAPI” (a REST linked service ingesting the Graph Reports API data), and one for “Ingested_JSON” (a JSON integrated dataset that lands the JSON report from Graph in stage1np of the data lake). You should connect your newly created REST linked service (here, named Graph API) “Graph API” to GraphAPI and “LS_ADLS_OEA” to Ingested_JSON, as shown above.
- After successfully importing the GraphAPI_data_ingestion pipeline template, navigate to the pipeline parameters (this can be done clicking on any of the whitespace in the pipeline GUI activities). There is only one pipeline parameter “timezone”, make sure to change this to match your own timezone (for the sake of folder naming).
- Make sure to publish/commit this change.

iv. Running the GraphAPI_main_pipeline

- Now that the GraphAPI_data_ingestion pipeline is ready for use, go back to the GraphAPI_main_pipeline, and change the connection of the pipeline trigger-activity “Extract from source – land in stage1np” to “GraphAPI_data_ingestion” from “GraphAPI_copy_test_data” in “Settings”.
  - Note: you may need to copy the default parameter value (timezone) shown from the GraphAPI_data_ingestion pipeline, into the values for the pipeline trigger-activity. Just copy and paste this value into their corresponding timezone field seen from the pipeline-trigger activity.
- After doing so, you are good to trigger the main pipeline, but now using your own data for ingestion. The run may take a few minutes. If you run into any errors, make sure to go back through the steps and make sure you filled out the information properly. Also make sure all necessary notebooks are connected to spark pools (i.e. Intune_module_ingestion and OEA_connector).

    - If you're running into an error with the lake database creation, this module references the OEA_connector notebook, which calls the ContosoSIS_py notebook (but doesn't use any of its functions). Make sure the ContosoSIS_py is imported to your Synapse workspace.
- You can then connect the PowerBI dashboard to the SQL database using the Serverless SQL database ID (you will probably want to use the devices_pseudo_refined table since this contains the additional columns and data analysis).