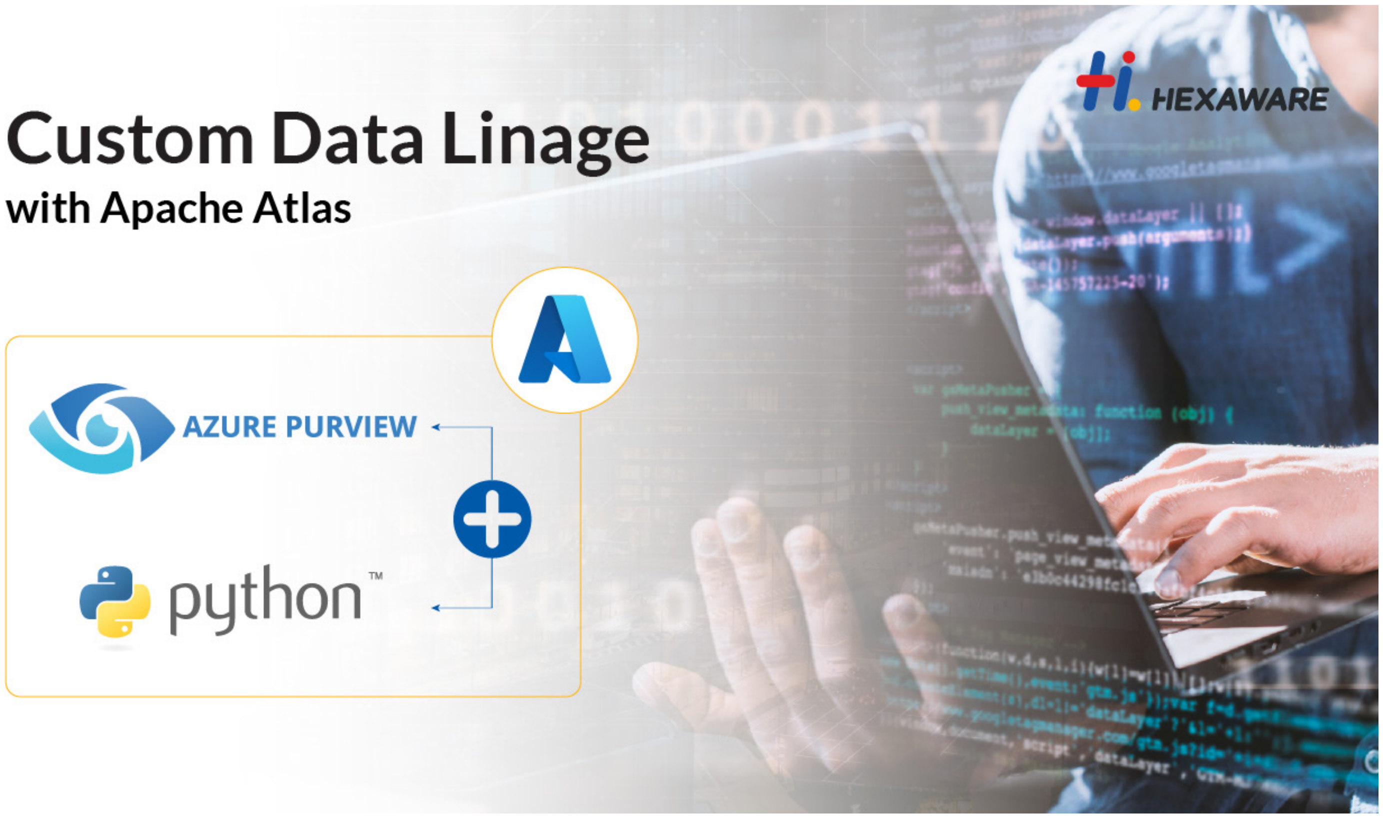
Custom Lineage Mapping between Azure Analysis Services & Microsoft Purview Service | Proof of Concept

Custom Data Linage
with Apache Atlas



Microsoft Purview provides a unified data governance solution to help manage and govern your on-premises, multicloud, and software as a service (SaaS) data. Easily create a holistic, up-to-date map of your data landscape with automated data discovery, sensitive data classification, and end-to-end data lineage. Enable data consumers to access valuable, trustworthy data management.

Some salient features are:
- Automate and manage metadata from hybrid sources.
- Classify data using built-in and custom classifiers and Microsoft Information
- Protection sensitivity labels.
- Label sensitive data consistently across SQL Server, Azure, Microsoft 365, and
- Power BI.
- Easily integrate all your data catalogs and systems using Apache Atlas APIs.

MS confirmed that Azure analytics services is not a connector in Microsoft Purview. Hence the lineage can not be created for objects presents in Azure analytics services.

PyApacheAtlas lets you work with the Azure Purview and Apache Atlas APIs in a Python-ic way. Supporting creation of custom lineage from an SDK and Excel templates.
The script reads the lineage information from the excel. which stores the relation of the entities in Purview.

There are primarily 3 components in a lineage :
1. Source
2. Target
3. Process

The target related the target entity to the source(s).
In this script, the relation in created at entity level and not at column level of an entity.

```
import json
from pyapacheatlas.auth import ServicePrincipalAuthentication
from pyapacheatlas.core.client import PurviewClient
from pyapacheatlas.readers import ExcelConfiguration, ExcelReader
from pyapacheatlas.core import AtlasEntity
from pyapacheatlas.core import AtlasProcess
import pandas as pd
import numpy as np


# information for the azure account connection
auth = ServicePrincipalAuthentication(
    tenant_id = "***************",
    client_id = "***************",
    client_secret = "***************"
)

# create an object for the connection to the azure account
client = PurviewClient(
    account_name= "",
    authentication = auth
)


# open the csv file where the lineage information is stored and
# create an intermittent file by grouping the lineage information by the
target and process name

df1=pd.read_csv(r"< path to the csv file >")
df1['Source qualifiedName']=df1.groupby(['Process name','Target
qualifiedName'])['Source qualifiedName'].transform(lambda x : ', '.join(x))
df1['Source typeName']=df1.groupby(['Process name','Target
qualifiedName'])['Source typeName'].transform(lambda x : ', '.join(x))
df1 = df1.drop_duplicates()
df1.to_csv(r"< path to save the intermittent csv file >")

# iterate over the rows in the intermittent file, this is for each of the
lineage.
for i,rows in df1.iterrows():
    v_i_entity=[]
    processName = rows['Process name']

    # based on sources in each of the lineage get the id of the entities
and create a list of that
    for idx, s in enumerate(rows['Source qualifiedName'].split(",")):
        v_i_qualifiedName = s
        v_i_typeName= rows['Source typeName'].split(",")[idx]
        pre_v_i_entity =
client.get_entity(qualifiedName=v_i_qualifiedName.strip(),typeName=v_i_type
Name.strip())
        v_i_entity.append(pre_v_i_entity["entities"][0])

    # get the id of the target entity
    v_o_qualifiedName = rows['Target qualifiedName']
    v_o_typeName=rows['Target typeName'].split(",")
    v_o_typeName = list(dict.fromkeys(v_o_typeName))
    v_o_entity
=client.get_entity(qualifiedName=v_o_qualifiedName,typeName=v_o_typeName[0]
.strip())

    process_qn = rows['Process qualifiedName']


    process_type_name =rows['Process typeName']

    # create a lineage object using the Atlas library
    newLineage = AtlasProcess(
        name= processName.strip()
        ,typeName= process_type_name.strip()
        ,qualified_name= process_qn
        , inputs= v_i_entity
        , outputs=  [v_o_entity["entities"][0]]
        , guid = -101
    )

    # post request to the azure, for the lineage object created above.
    results = client.upload_entities(batch = [newLineage])
    print(json.dumps(results, indent=2))
```

The metdata of the excel which is required as input for the script is as below :

```
The metadata of the excel is as follow :
Target typeName : The typedefs of the target entity in purview (e.g.
Target qualifiedName : The qualified name of the target entity in purview
Source typeName : The typedefs of the source entity in purview
Source qualifiedName : The qualified name of the source entity in purview
Process name : The name of the process which needs to hold the
Process qualifiedName : The qualified id of the process in the azure account.
Process typeName : free text to describe the process.
```