Retrieving Power BI Dataset Information and Refresh Details

To gather information about your Power BI datasets, their data sources, and refresh details, you can use a combination of Power BI APIs and source system queries. Here's a comprehensive approach:

Option 1: Power BI REST API

The Power BI REST API provides endpoints to retrieve dataset and datasource information:

powershell

*# PowerShell example using Power BI REST API*

*# First, authenticate (install MicrosoftPowerBIMgmt module if needed)*

Install-Module -Name MicrosoftPowerBIMgmt -Force

Connect-PowerBIServiceAccount

*# Get all dashboards*

$dashboards = Get-PowerBIDashboard

foreach ($dashboard in $dashboards) {

*# Get tiles for each dashboard*

$tiles = Get-PowerBITile -DashboardId $dashboard.Id

foreach ($tile in $tiles) {

*# Get dataset for each tile*

$datasetId = $tile.datasetId

if ($datasetId) {

$dataset = Get-PowerBIDataset -Id $datasetId

*# Get datasources for the dataset*

$datasources = Invoke-PowerBIRestMethod -Url "datasets/$datasetId/datasources" -Method Get | ConvertFrom-Json

Write-Output "Dashboard: $($dashboard.Name)"

Write-Output "Dataset: $($dataset.Name)"

Write-Output "Last Refresh: $($dataset.ConfiguredBy)"

Write-Output "Refresh Details: $($dataset.RefreshSchedule)"

foreach ($datasource in $datasources.value) {

Write-Output "Datasource Type: $($datasource.datasourceType)"

Write-Output "Connection Details: $($datasource.connectionDetails)"

}

}

}

}

Option 2: Power BI Admin API

For more comprehensive metadata across your entire tenant:

powershell

*# Get all datasets in the tenant*

$datasets = Get-PowerBIDataset -Scope Organization

foreach ($dataset in $datasets) {

*# Get datasources*

$datasources = Invoke-PowerBIRestMethod -Url "datasets/$($dataset.Id)/datasources" -Method Get | ConvertFrom-Json

*# Get refresh history*

$refreshHistory = Invoke-PowerBIRestMethod -Url "datasets/$($dataset.Id)/refreshes" -Method Get | ConvertFrom-Json

Write-Output "Dataset: $($dataset.Name)"

Write-Output "Last Refresh: $($refreshHistory.value[0].endTime)"

foreach ($datasource in $datasources.value) {

Write-Output "Server: $($datasource.connectionDetails.server)"

Write-Output "Database: $($datasource.connectionDetails.database)"

}

}

Option 3: Power BI Dataset XMLA Endpoint

For more detailed technical metadata, connect via XMLA endpoint:

1. Enable XMLA endpoint in Power BI tenant settings
2. Connect using SQL Server Management Studio or similar tool
3. Query the TMSCHEMA tables

sql

*-- Example query via XMLA endpoint*

SELECT

D.[Name] AS DatasetName,

T.[Name] AS TableName,

C.[Name] AS ColumnName,

C.[DataType] AS DataType

FROM

TMSCHEMA\_TABLES AS T

INNER JOIN TMSCHEMA\_COLUMNS AS C ON T.[ID] = C.[TableID]

INNER JOIN TMSCHEMA\_DATASOURCES AS DS ON T.[DataSourceID] = DS.[ID]

INNER JOIN TMSCHEMA\_DATABASES AS D ON T.[DatabaseID] = D.[ID]

Retrieving Source System Refresh Information

For each data source type, you'll need to query the source systems:

SQL Server:

sql

SELECT

t.name AS TableName,

STATS\_DATE(p.object\_id, p.index\_id) AS LastStatsUpdate

FROM

sys.tables t

INNER JOIN

sys.indexes p ON t.object\_id = p.object\_id

WHERE

p.index\_id < 2 *-- clustered index or heap*

PostgreSQL:

sql

SELECT

schemaname,

relname,

last\_analyze,

last\_autoanalyze,

last\_vacuum,

last\_autovacuum

FROM

pg\_stat\_all\_tables

Databricks:

python

*# Using Databricks Python API*

from databricks import sql

import pandas as pd

connection = sql.connect(server\_hostname="your-hostname",

http\_path="your-http-path",

access\_token="your-token")

*# Get table metadata*

tables = pd.read\_sql("SHOW TABLES", connection)

*# For each table, get detailed info*

for index, row in tables.iterrows():

table\_info = pd.read\_sql(f"DESCRIBE DETAIL {row['database']}.{row['tableName']}", connection)

print(f"Table: {row['tableName']}, Last Modified: {table\_info['lastModified'][0]}")

Consolidating the Information

You can create a PowerShell script or Python application that:

1. Uses Power BI API to get all datasets and their datasources
2. For each datasource, connects to the source system to get refresh information
3. Stores all metadata in a central database or Excel file
4. Sets up a refresh schedule for this metadata collection

Would you like me to elaborate on any specific part of this approach or provide more detailed examples for a particular data source type?

New chat

Using Power BI REST API to Retrieve Dataset Connections

Here's a comprehensive guide to retrieve dataset connection information using the Power BI REST API:

Prerequisites

1. **Azure AD App Registration** (for service principal authentication) or user authentication
2. **Required permissions**: Dataset.Read.All or Dataset.ReadWrite.All
3. **Power BI module for PowerShell** or ability to make REST API calls

Method 1: Using PowerShell with MicrosoftPowerBIMgmt Module

powershell

*# Install and import the module*

Install-Module -Name MicrosoftPowerBIMgmt -Force -Scope CurrentUser

Import-Module MicrosoftPowerBIMgmt

*# Connect to Power BI service*

Connect-PowerBIServiceAccount

*# Get all datasets in the workspace (or organization)*

$workspaceId = "your-workspace-id" *# Use "All" for all workspaces*

$datasets = Get-PowerBIDataset -WorkspaceId $workspaceId

foreach ($dataset in $datasets) {

Write-Host "`nDataset: $($dataset.Name)" -ForegroundColor Green

Write-Host "ID: $($dataset.Id)"

*# Get datasources for this dataset*

$datasources = Invoke-PowerBIRestMethod -Method Get -Url "datasets/$($dataset.Id)/datasources"

$datasourcesObj = $datasources | ConvertFrom-Json

foreach ($ds in $datasourcesObj.value) {

Write-Host "`nDatasource Type: $($ds.datasourceType)" -ForegroundColor Yellow

if ($ds.datasourceType -eq "Sql") {

Write-Host "Server: $($ds.connectionDetails.server)"

Write-Host "Database: $($ds.connectionDetails.database)"

}

elseif ($ds.datasourceType -eq "Postgres") {

Write-Host "Server: $($ds.connectionDetails.server)"

Write-Host "Database: $($ds.connectionDetails.database)"

}

elseif ($ds.datasourceType -eq "OData") {

Write-Host "URL: $($ds.connectionDetails.url)"

}

*# Add more datasource types as needed*

Write-Host "Connection Details: $(($ds.connectionDetails | ConvertTo-Json -Compress))"

}

*# Get refresh history*

$refreshes = Invoke-PowerBIRestMethod -Method Get -Url "datasets/$($dataset.Id)/refreshes"

$refreshesObj = $refreshes | ConvertFrom-Json

if ($refreshesObj.value.Count -gt 0) {

Write-Host "`nRefresh History:" -ForegroundColor Cyan

$latestRefresh = $refreshesObj.value[0]

Write-Host "Last Refresh: $($latestRefresh.endTime)"

Write-Host "Status: $($latestRefresh.status)"

Write-Host "Refresh Type: $($latestRefresh.refreshType)"

}

}

Method 2: Using Direct REST API Calls (Python Example)

python

import requests

import json

from msal import ConfidentialClientApplication

*# Authentication setup*

tenant\_id = "your-tenant-id"

client\_id = "your-client-id"

client\_secret = "your-client-secret"

authority = f"https://login.microsoftonline.com/{tenant\_id}"

scope = ["https://analysis.windows.net/powerbi/api/.default"]

*# Get access token*

app = ConfidentialClientApplication(

client\_id=client\_id,

client\_credential=client\_secret,

authority=authority

)

result = app.acquire\_token\_for\_client(scopes=scope)

access\_token = result['access\_token']

*# API headers*

headers = {

'Authorization': f'Bearer {access\_token}',

'Content-Type': 'application/json'

}

*# Get all datasets*

base\_url = "https://api.powerbi.com/v1.0/myorg"

datasets\_url = f"{base\_url}/datasets"

response = requests.get(datasets\_url, headers=headers)

datasets = response.json()['value']

for dataset in datasets:

print(f"\nDataset: {dataset['name']}")

print(f"ID: {dataset['id']}")

*# Get datasources*

datasources\_url = f"{base\_url}/datasets/{dataset['id']}/datasources"

ds\_response = requests.get(datasources\_url, headers=headers)

if ds\_response.status\_code == 200:

datasources = ds\_response.json()['value']

for ds in datasources:

print(f"\nDatasource Type: {ds['datasourceType']}")

connection\_details = ds.get('connectionDetails', {})

if ds['datasourceType'] == 'Sql':

print(f"Server: {connection\_details.get('server', 'N/A')}")

print(f"Database: {connection\_details.get('database', 'N/A')}")

elif ds['datasourceType'] == 'Postgres':

print(f"Server: {connection\_details.get('server', 'N/A')}")

print(f"Database: {connection\_details.get('database', 'N/A')}")

elif ds['datasourceType'] == 'Databricks':

print(f"Server: {connection\_details.get('server', 'N/A')}")

print(f"Database: {connection\_details.get('database', 'N/A')}")

print(f"HTTP Path: {connection\_details.get('httpPath', 'N/A')}")

print(f"Connection Details: {json.dumps(connection\_details, indent=2)}")

*# Get refresh history*

refreshes\_url = f"{base\_url}/datasets/{dataset['id']}/refreshes"

refresh\_response = requests.get(refreshes\_url, headers=headers)

if refresh\_response.status\_code == 200:

refreshes = refresh\_response.json()['value']

if refreshes:

latest\_refresh = refreshes[0]

print(f"\nLast Refresh: {latest\_refresh.get('endTime', 'N/A')}")

print(f"Status: {latest\_refresh.get('status', 'N/A')}")

Method 3: Get Detailed Gateway and Connection Information

powershell

*# Get gateway information for datasets*

$datasets = Get-PowerBIDataset -WorkspaceId "your-workspace-id"

foreach ($dataset in $datasets) {

*# Check if dataset uses an on-premises data gateway*

$gateways = Invoke-PowerBIRestMethod -Method Get -Url "datasets/$($dataset.Id)/gateways"

$gatewaysObj = $gateways | ConvertFrom-Json

foreach ($gateway in $gatewaysObj.value) {

Write-Host "Gateway: $($gateway.name)" -ForegroundColor Magenta

Write-Host "Gateway ID: $($gateway.id)"

Write-Host "Gateway Type: $($gateway.gatewayType)"

Write-Host "Gateway Status: $($gateway.gatewayStatus)"

*# Get datasource details from gateway*

$gatewayDatasources = Invoke-PowerBIRestMethod -Method Get -Url "gateways/$($gateway.id)/datasources"

$gatewayDatasourcesObj = $gatewayDatasources | ConvertFrom-Json

foreach ($gds in $gatewayDatasourcesObj.value) {

Write-Host "Gateway Datasource: $($gds.datasourceName)"

Write-Host "Connection Details: $($gds.connectionDetails)"

}

}

}

Method 4: Export to CSV

powershell

*# Create CSV output of all dataset connections*

$output = @()

$datasets = Get-PowerBIDataset -Scope Organization

foreach ($dataset in $datasets) {

$datasources = Invoke-PowerBIRestMethod -Method Get -Url "datasets/$($dataset.Id)/datasources"

$datasourcesObj = $datasources | ConvertFrom-Json

foreach ($ds in $datasourcesObj.value) {

$output += [PSCustomObject]@{

DatasetName = $dataset.Name

DatasetId = $dataset.Id

DatasourceType = $ds.datasourceType

Server = $ds.connectionDetails.server

Database = $ds.connectionDetails.database

Url = $ds.connectionDetails.url

ConnectionDetails = ($ds.connectionDetails | ConvertTo-Json -Compress)

}

}

}

*# Export to CSV*

$output | Export-Csv -Path "PowerBIDatasetConnections.csv" -NoTypeInformation

Write-Host "Exported $($output.Count) dataset connections to PowerBIDatasetConnections.csv"

Important Notes:

1. **Permissions**: Ensure your service principal or user account has appropriate permissions
2. **Rate Limiting**: The API has rate limits, so implement proper error handling
3. **Authentication**: Choose between user authentication (interactive) or service principal (automated)
4. **Workspace Scope**: Use -Scope Organization to access all workspaces (requires admin rights)
5. **Sensitive Data**: Connection details may contain sensitive information - handle appropriately

This approach will give you comprehensive information about all dataset connections, including server names, databases, and connection details for each data source type.

**🔹 Part 1: Power BI Dataset Refresh Dates**

Power BI datasets maintain refresh history (manual, scheduled, API-triggered). You can retrieve refresh info via **Power BI REST APIs**.

**1. Authenticate to Power BI Service**

* Register an **Azure AD app** and grant permissions for Power BI.
* Use OAuth2.0 to get an access token.
* Required scopes:  
  Dataset.Read.All, Report.Read.All, Dataflow.Read.All, Workspace.Read.All.

**2. Get Dataset Refresh Info**

REST API endpoint:

GET https://api.powerbi.com/v1.0/myorg/groups/{groupId}/datasets/{datasetId}/refreshes

Response includes:

{

"refreshes": [

{

"refreshType": "Scheduled",

"startTime": "2025-08-30T08:23:00Z",

"endTime": "2025-08-30T08:25:00Z",

"status": "Completed"

}

]

}

👉 This gives you the **dataset-level refresh history** (not the source system).

**🔹 Part 2: Data Source Connections in Power BI**

To know what sources a dataset is connected to:

GET https://api.powerbi.com/v1.0/myorg/groups/{groupId}/datasets/{datasetId}/datasources

Sample response:

[

{

"datasourceType": "Sql",

"connectionDetails": {

"server": "sql-prod01",

"database": "claimsdb"

}

}

]

👉 You can parse this to identify **SQL / MySQL / Postgres / Databricks** sources.

**🔹 Part 3: Getting Last Updated Timestamp from Databases**

Once you know the source, you’ll need to **query the source system metadata**:

**1. SQL Server**

SELECT name, create\_date, modify\_date

FROM sys.tables

ORDER BY modify\_date DESC;

Or for last ETL/load:

* If using staging tables → check ETL log tables (best practice).
* If CDC → query change tracking tables.

**2. MySQL**

SELECT TABLE\_NAME, UPDATE\_TIME

FROM information\_schema.tables

WHERE TABLE\_SCHEMA = 'your\_db'

ORDER BY UPDATE\_TIME DESC;

**3. PostgreSQL**

SELECT relname AS table\_name, pg\_stat\_all\_tables.last\_vacuum, pg\_stat\_all\_tables.last\_autovacuum

FROM pg\_stat\_all\_tables

WHERE schemaname = 'public';

(For actual ETL loads, track via audit/log tables.)

**4. Databricks (Delta Lake)**

* Delta tables maintain transaction logs in \_delta\_log.
* Use:

DESCRIBE HISTORY your\_table

This gives you **operation timestamps** for loads.

**🔹 Part 4: Unified Mechanism**

You can build an orchestration layer (say in **Python, Node.js, or Databricks Notebook**) to:

1. **Call Power BI REST API** → fetch dataset refresh metadata.
2. **Get dataset → data source mapping**.
3. For each source:
   * Connect using respective DB drivers.
   * Run metadata query (information\_schema, sys.tables, DESCRIBE HISTORY, etc.).
   * Get **last modified timestamp**.
4. Store results in a monitoring table (say in SQL or Databricks).
5. Expose via:
   * Power BI dashboard (meta-monitoring dashboard).
   * Alerts if Power BI dataset refresh is later than source update (possible stale data).

**🔹 Example Workflow**

* **Dataset "Claims Analysis"** refreshed at 2025-08-30 08:25 UTC (Power BI API).
* Its source is SQL Server → claimsdb.claims.
* From SQL metadata → claims table last ETL load = 2025-08-30 07:50 UTC.
* ✅ Dataset is in sync.
* If dataset refresh < table update → stale.

**🔹 Tools You Can Use**

* **Power BI REST API + MSAL** for authentication.
* **pyodbc / psycopg2 / mysql-connector / databricks-sql-connector** for source queries.
* **Airflow / ADF / custom Python scheduler** to automate checks.
* **Push results into Power BI dataset** → build a **Data Refresh Monitoring Dashboard**.

👉 Navnit, would you like me to draft a **Python script framework** that does this end-to-end (Power BI API + DB source query + consolidated output)? That way you’ll have a working skeleton to plug in your datasources.