Getting Started with the Power BI API – Querying Power BI with PowerShell

blog.jpries.com/2019/09/27/getting-started-with-the-power-bi-api

jpries

September 28, 2019

Recently, I was using the Power BI Service at powerbi.com and I had a few questions that I wanted to answer for myself about the environment. What I was wondering about included:

- How many reports are deployed?
- How many / which reports are deployed in multiple workspaces? Which workspaces?
- Which datasets are live ssas queries, which are direct queries, and which are embedded models?
- Which datasources use which enterprise gateways?
- Who are the users for each workspace?

While it is possible to find the answer to all of these questions by just clicking around in the portal on powerbi.com, some of these questions are difficult to answer or would be very time consuming to come up with an answer for. There has to be a better way, right? Enter the Power BI API.

The Power BI API is a way of essentially bypassing the web interface of powerbi.com and asking questions directly to the back-end of the service (without bypassing security). Using this allows you to issue a command, such as “Give me a list of all of the workspaces” and receive just the result data in a bulk data format. While outside the scope of this exercise, the API also allows reading actual business data from Power BI assets and even manipulating assets.

For my first attempt at interacting with the API, I decided to try the Microsoft developed Powershell Cmdlets. These are a set of free Powershell commands which can be installed on your workstation and then will query the Power BI API and return results in a standard way without the need to write any code (C# or otherwise) or understand how to read a JSON response.

Getting Started with the Powershell Cmdlets

The greatest part of using the Power BI Powershell Cmdlets to access the API is how quickly you can get started. With the following command, you’re now ready to use the API:

Install-Module -Name MicrosoftPowerBIMgmt

```
PS C:\WINDOWS\system32> Install-Module -Name MicrosoftPowerBIMgmt

NuGet provider is required to continue
PowerShellGet requires NuGet provider version '2.8.5.201' or newer to interact with NuGet-based repositories. The NuGet
 provider must be available in 'C:\Program Files\PackageManagement\ProviderAssemblies' or
'C:\Users\jpries\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet provider by
running 'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force'. Do you want PowerShellGet to install
and import the NuGet provider now?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): y

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from
'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): a
PS C:\WINDOWS\system32>
```

Installing the Power BI Powershell modules

Now that the modules are installed, you can authenticate your Powershell session to the API via the following command:

Connect-PowerBIServiceAccount

```
PS C:\> Connect-PowerBIServiceAccount


Environment : Public
TenantId    : 438c3c79-[redacted]
UserName    : JPries@[redacted]
```

After successfully authenticating to the service, you can now launch commands to start answering questions. Workspaces are one of the most base types of objects in the Power BI API, so getting a list of workspaces that you have access to is a great place to start. You can do so with the following command:

Get-PowerBIWorkspace -All

```
PS C:\> Get-PowerBIWorkspace -All


Id                    : 41da4ce6-[redacted]
Name                  : IT -Testing
IsReadOnly            : False
IsOrphaned            : False
IsOnDedicatedCapacity : True
CapacityId            : 698975DA-[redacted]
```

It's really that easy. A full list of currently supported Powershell commands and their syntax can be found here.

Taking The Powershell Cmdlets to the Next Level

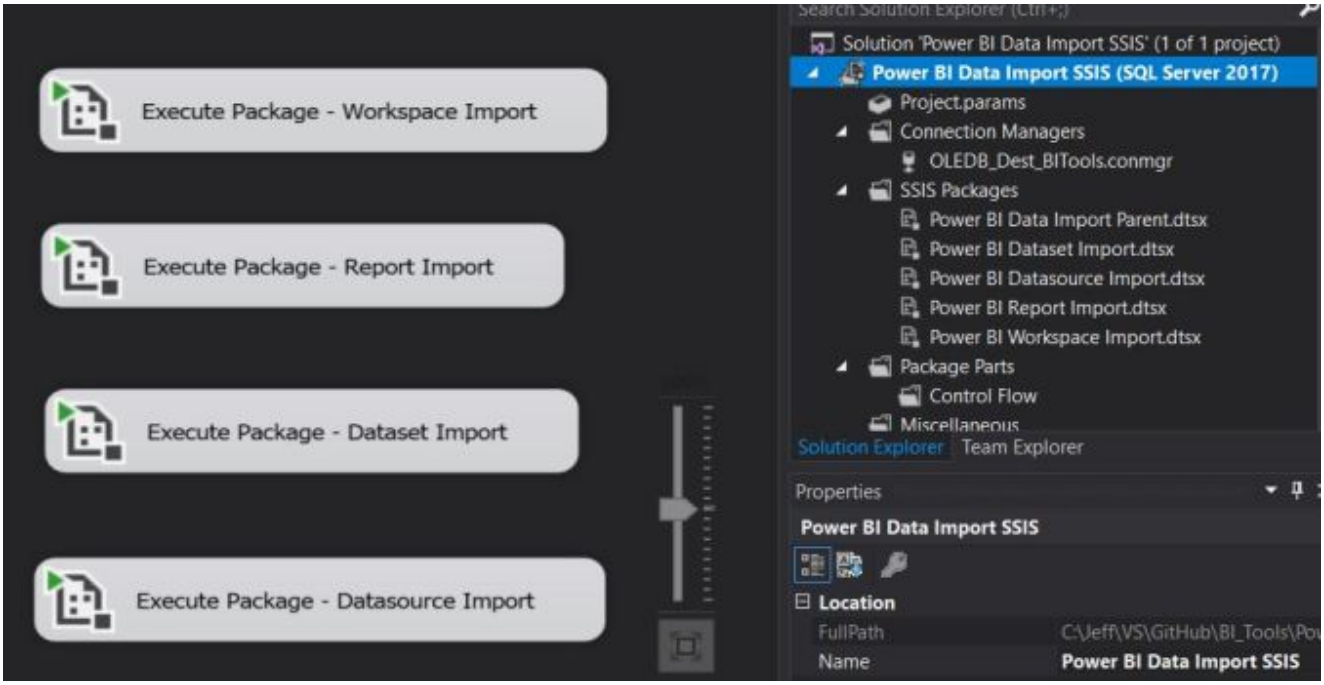
If you're able to successfully authenticate and perform a query, you can now start to take things to the next level. To answer the above questions, I decided to try to use the Powershell Cmdlets to get a list of all Workspaces, then for each of those, get a list of all Reports in each Workspace, then for each of those, get a list of all Datasets for each Report (in each Workspace) and finally a list of all Datasources for each Dataset.

For each one of these 4 sets of data, I had PoweShell output the data to CSV files (via the Export-CSV capability).

```
# Get the report and dataset data for each workspace
foreach ($workspace in $workspaces) {
    $reports = Get-PowerBIReport -WorkspaceId  $workspace.Id
    $datasets = Get-PowerBIDataset -WorkspaceId  $workspace.Id

    $reports | Add-Member -NotePropertyName WorkspaceId -NotePropertyValue $workspace.Id
    $datasets | Add-Member -NotePropertyName WorkspaceId -NotePropertyValue $workspace.Id

    $workspaceReports += $reports
    $workspaceDatasets += $datasets
}

# Get the datasource data for each workspace dataset
foreach ($dataset in $workspaceDatasets) {
    $datasources = Get-PowerBIDatasource -DatasetId $dataset.Id -WorkspaceId $dataset.WorkspaceId

    $datasources | Add-Member -NotePropertyName DatasetId -NotePropertyValue $dataset.Id
    $datasources | Add-Member -NotePropertyName WorkspaceId -NotePropertyValue $dataset.WorkspaceId

    $datasetDatasources += $datasources
}

# Output the results for each type to CSV files
$workspaces | Export-CSV -Force -Path .\PowerBI_Workspaces.csv
$workspaceReports | Export-CSV -Force -Path .\PowerBI_Reports.csv
$workspaceDatasets | Export-CSV -Force -Path .\PowerBI_Datasets.csv
$datasetDatasources | Export-CSV -Force -Path .\PowerBI_Datasources.csv
```

Sample of the PowerShell script to query data from the Power BI API

Finally, I created a basic set of SSIS packages to read the 4 CSV files that were created and import them into tables in a SQL Server database that I could then query to get answers to my questions.



SSIS package to read 4 CSV files of data exported from the Power BI API into SQL database tables

With the data exported, I was now able to query the data in the SQL tables by joining the tables which held the items I was interested in based on the ID values received from the service.

While this information was great (and the above is just a sample of some of the objects available), I wanted more. After doing some digging into the Power BI API itself (which the PowerShell cmdlets pull their data from), I learned that there was even more available through the API directly that the PowerShell cmdlets didn't yet support (such as Apps).

In the next part, I'll outline the process I went through on my journey to query data directly from the Power BI REST API via C#.

SQL database tables for storing the Power BI API data exported via PowerShell

Resources

Posts in Series

```
USE BITools

-- What are the workspaces?
SELECT
    w.[Name] AS WorkspaceName
FROM dbo.PowerShell_PBIWorkspace w
ORDER BY WorkspaceName
```

Querying the SQL database tables for Power BI data