USING GET GROUPS AS ADMIN API

November 9, 2019 Power BI, Power BI Administration 3 comments

Earlier this year I published a few posts describing approaches for retrieving Power BI artifacts via PowerShell scripts. As a Power BI administrator, the scheduled retrieval of this data could become a key input to an overall admin BI solution to quickly answer common questions.

Over the summer an alternative method for retrieving workspaces and the Power BI artifacts they contain was made available via the Power BI REST API. Specifically, an 'Expand' parameter was added to the GetGroupsAsAdmin REST API which includes support for workspace users, datasets, dashboards, and reports. With this enhancement, a single call to the REST API can quickly retrieve up to 5,000 workspaces along with the user and artifacts associated with

BLOG ARCHIVE

Search .

BLOG TOPICS

Select Category

TAGS

ANALYSIS SERVICES DAX EXCEL M MSBI POWER BI POWER BI DESKTOP POWER PIVOT POWER QUERY POWERSHELL SSAS TABULAR

them thus avoiding the need to loop over the workspaces for each artifact type.

The purpose of this post is to share some components and examples of using this REST API in the context of a Power BI Administration solution.

Prerequisites

This example requires either the Power BI service administrator role or Office 365 Global Admin role), the Power BI Management PowerShell module, and SQL Server 2016 or later database engine.

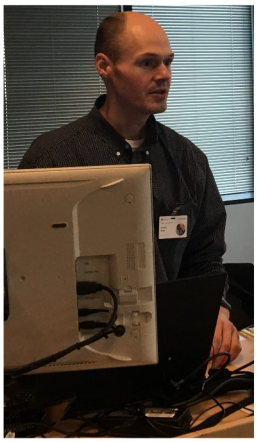
Code Samples

The PowerShell script and three SQL files described in this post are available for download from the following folder in my GitHub repository: Get Groups as Admin API

PowerShell Script

The PowerShell script executes the following five simple steps:

1. Retrieve the user name and password for a Power BI service administrator account from a separate PS1 file.
2. Authenticate to the Power BI service with a



PowerShell credential.

3. Define a file path for the JSON data to be written to.
4. Build a string to define the call to the REST API (Get Request)
    - The Expand parameter references the four supported values: dashboards, reports, datasets, users.
    - Workspace types of 'PersonalGroup' (My Workspace) are excluded and only Active workspaces are retrieved
        - Two alternative request filter definitions are included in the comments below step five for including personal workspaces and workspaces which are not in an active state.
5. Use the Invoke-PowerBIRestMethod PowerShell command along with the string variable defined in step #4 and write this data to the file path defined in step #3.

Given the filters excluding personal workspaces and any workspace which isn't in an active state, a single call supporting 5,000 workspaces will probably cover all the workspaces in most Power BI tenants and within seconds will generate a small (e.g. 1MB) JSON file. Additional calls (and corresponding files) could be made with the Skip parameter if you need to retrieve over 5,000.

November 2019

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

« Sep Dec »

FIVE POPULAR POSTS


Conditional Formatting for Card Visuals


Refresh Power BI Datasets with PowerShell

Analysis Services vs Power BI Premium Model Feature Matrix

T-SQL

You 'could' simply retrieve and parse the JSON file in Power BI Desktop but the three SQL files contain T-SQL statements which allow you to persist and query the REST API data in SQL Server.

1. Create Workspace SQL Table
    - This batch creates a 'PBIWorkspaces' table in a BIMonitoring schema in a SQL Server 2017 database.
    - Notice the four NVARCHAR(MAX) columns corresponding to the values specified in the Expand parameter of the REST API – this is necessary to store this data as JSON
    - A Primary Key constraint is defined based on the Workspace ID
2. Insert JSON to SQL Table
    - This batch truncates the table created in Step #1 and then inserts data from the JSON file output of the PowerShell script
    - The OPENJSON() table-valued function is used to parse the JSON and the four Expand parameter values are defined as JSON types and inserted into the NVARCHAR(MAX) columns
    - You could add these truncate and insert statements to a stored

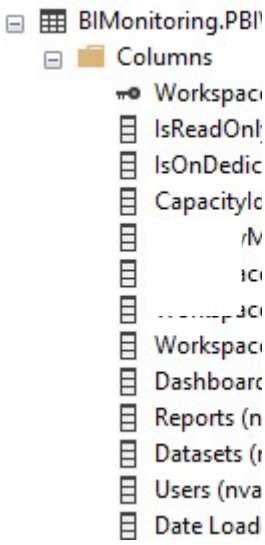
procedure as part of a scheduled process for retrieving and loading this data

3. Create Workspace Views
    - This batch creates five separate SQL view objects in the BIMonitoring schema corresponding to the different artifacts: workspaces, reports, users, dashboards, datasets
    - For the JSON data stored in the NVARCHAR(MAX) columns, OPENJSON() is again used along with the CROSS APPLY operator to associate the expanded array values with their parent row.

At this point, you could point any tool which supports the SQL Server relational database engine at the SQL views for reporting and analysis. As described in past posts, I tend to favor a Power BI dataset (internally an Analysis Services model or 'cube') as a central semantic layer supporting all or most Power BI admin analysis built on top of a relational database and it's supporting ETL processes.

Note: As of this writing only the V2 workspaces are supported for the Workspace users, not the legacy Office 365-based workspaces. Hopefully there will be an optional and then mandatory migration path to the V2 workspaces soon (or the API will add

support for V1). Here are a few images of using the examples:



Workspace Table

Wrapping Up

In summary, with a bit of setup and some fairly lightweight PowerShell and T-SQL, you can retrieve and store some of the most important Power BI metadata for administrative

purposes. I hope this blog post was helpful and feel welcome to share feedback in the comments.

Share this:

Tweet Share Share

Related

Power BI Admin Datasets
In "Power BI"

Parallel Power BI Scripts
PowerShell 7.0 was made generally available last month and one
In "Power BI"

Power BI Artifact to Workspace Relationships
In "Power BI"

« Analysis Services DMVs in SSMS

Analysis Services vs Power BI Premium Model Feature Matrix »

3 COMMENTS

Pingback: Find V1 Workspace Owners – Insight Quest

Pingback: Parallel Power BI Scripts – Insight Quest

Don Aldridge says:

April 6, 2020 at 1:33 am

Hi Brett. Thanks for your work on this topic. I've used your examples for a tenant migration project and they have really helped. My question relates to filtering personal groups that include at least one report. Can you describe the syntax for filtering on an expanded property? eg Groups where Report id is not null. Out of 5700 personal groups, only 370 have reports. I would like to exclude those at the REST API call.

★ Like

Reply

LEAVE A REPLY

Enter your comment here...

Blog at WordPress.com.