GraphQL API + PowerBI Integration

Technical Brief

Last Updated: 2/16/2021
Verified using PowerBI Desktop 2.88.1144.0

Table of Content

Introduction

Pluralsight GraphQL APIs provide a robust set of learner data to help enterprises better understand employee engagement and skill attainment on the Pluralsight platform. This data can be leveraged in PowerBI workbooks via Microsoft's Power Query M language. GraphQL is a query language for APIs--similar to SQL. This white paper will walk you through examples of how to connect to Pluralsight's GraphQL APIs from within PowerBI. For a quick start, review the Resources section.

The process provides the customer a sample workbook and documentation on how to query the APIs and pull the raw data into PowerBI. If the API query is correct but the API is returning bad data, naturally ProServ will forward that to PaaS as part of our default API support.

What the PowerBI Integration does not include:

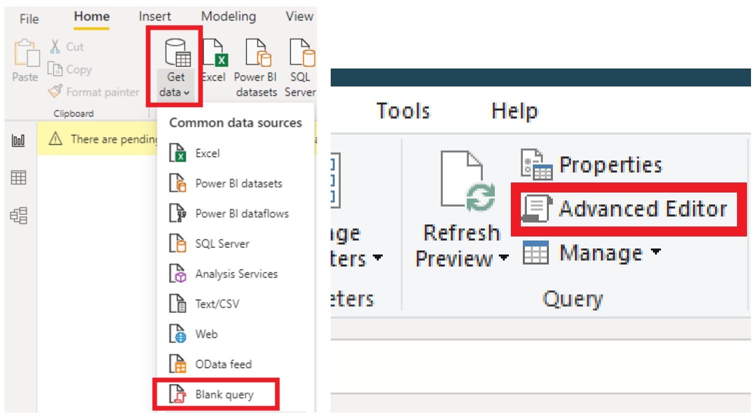
- Building or providing a dashboard, providing a series of complete reports within PowerBI, training the customer (beyond the initial 1 hour Q&A call) on how to use the APIs or PowerBI, creation of a custom sample workbook for the customer, advisement on where to learn about GraphQL (aside from references made in the documentation), upskilling a Business Analyst in scripting, etc.

Prerequisites

- Experience using Microsoft PowerBI,
- Power BI Desktop installed on your machine, and a Pluralsight API Key
    - **NOTE:** *You must have Pluralsight Administrator privileges to access, create and manage keys. If you're not already an administrator, please contact your Pluralsight Administrator to discuss getting administrative access.*

PowerBI Advanced Query Editor

The first step is to make a connection to Pluralsight's APIs and create a GraphQL query to retrieve the data you need. In this example, we'll show you how to pull Course catalog metadata about video courses on the platform. To begin, open PowerBI and go to Get data > Blank Query > Advanced Editor. See screenshots below.



Copy the complete script below into the newly opened Advanced Editor Window.

Complete Script

```
let
    Source = (apiToken as text, optional endCursor as text, optional data as list) =>
let
    endCursor = if endCursor is null then "" else endCursor,

    Source = Json.Document(Web.Contents("https://paas-api.pluralsight.com/graphql",
        [
            Headers=[
                #"Method"="POST",
                            #"Content-Type"="application/json",
                            #"Authorization"="Bearer " & apiToken
            ],
            Content = Text.ToBinary("{ ""query"": ""query{ courseCatalog(first: 5000, after:\""" &
endCursor & "\"" ) { pageInfo {endCursor, hasNextPage}, nodes { title description authors tags {topics}}
} }"" }")
        ]
    )),
    pageInfo = Source[data][courseCatalog][pageInfo],
    nodes = Source[data][courseCatalog][nodes],

    appendedData =
        if pageInfo[hasNextPage] = true and data is null then
            List.Combine({{}, nodes})
        else List.Combine({data, nodes}),

    output =
        if pageInfo[hasNextPage] = true then
            @#"GET courseCatalog"(apiToken, pageInfo[endCursor], appendedData)
        else
            Table.FromList(appendedData,  Record.FieldValues, {"title", "description","authors","tags"})
in
    output
in
    Source
```

Script Explanation

This first section is boilerplate code that you can reuse for any other Pluralsight GraphQL queries. It initializes a placeholder for your API Key, the cursor for paginating through query results, and the HTTP request to the GraphQL endpoint.

```
let
    Source = (apiToken as text, optional endCursor as text, optional data as
list) =>
let
    endCursor = if endCursor is null then "" else endCursor,

    Source =
Json.Document(Web.Contents("https://paas-api.pluralsight.com/graphql",
        [
            Headers=[
                #"Method"="POST",
                        #"Content-Type"="application/json",
                        #"Authorization"="Bearer " & apiToken
              ],
```

This next section contains the GraphQL query that defines what data will be retrieved from the API. The Pluralsight developer portal contains schema documentation, GraphQL examples, and a GraphQL Playground to validate your GraphQL queries before you run them in PowerBI. Do keep in mind that the query used in the GraphQL Playground will have to be modified slightly to properly escape characters and include variables when it is placed into PowerBI's Advanced Editor. The code block below contains the GraphQL query. "courseCatalog" identifies the API we're querying. "first" indicates how many records to return on the first page (more on pagination and cursors later). You must specify the fields you want returned (in this case title, description, authors, and tags) inside the nodes brackets. For more information on GraphQL queries, please see Using GraphQL and the Additional Resources at the bottom of that page.

You can only query one API Object directly per query, however you can list as few or as many Fields from the object as you would like. Please note that the "courseCatalog" name on the "pageInfo" and "nodes" lines at the bottom must match the query name on the first line in the code block below. You will need to replace the values used in the middle bracket (green highlights in code block) with the same value that represents the object you are querying (see highlighted fields for identification).

```
Content = Text.ToBinary("{ ""query"": ""query{ courseCatalog(first: 5000,
after:\""" & endCursor & "\"" ) { pageInfo {endCursor, hasNextPage}, nodes { title
description authors tags{topics}} } }"" }")
        ]
    )),
    pageInfo = Source[data][courseCatalog][pageInfo],
    nodes = Source[data][courseCatalog][nodes],
```

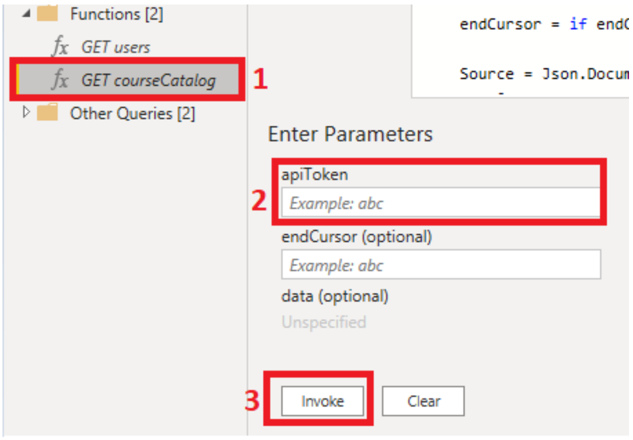
GraphQL queries return a JSON response. In order to use the data in PowerBI, you must map the JSON attributes to table columns. The Table.FromList line below maps the "title" field from the query above to the "title" field in the table, and "description" to "description", etc. Note, that in this example we do not include the "topics" nested field in this section. You do not need to state the name of the nested fields, just the parent field.

```
appendedData =
        if pageInfo[hasNextPage] = true and data is null then
            List.Combine({{}, nodes})
        else List.Combine({data, nodes}),

    output =
        if pageInfo[hasNextPage] = true then
            @#"GET courseCatalog"(apiToken, pageInfo[endCursor], appendedData)
```

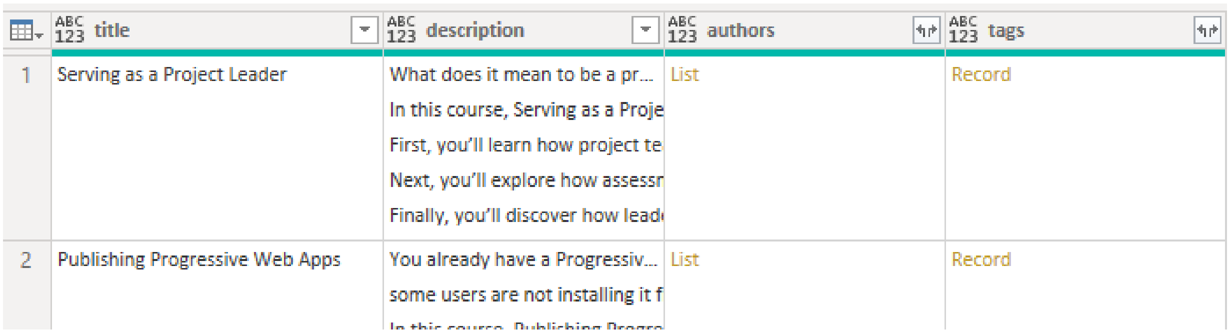
```
         else
             Table.FromList(appendedData,  Record.FieldValues, {"title",
"description","authors","tags"})
in
    output
in
    source
```

You can now close the editor. By default PowerBI will try to load the query and execute it, but it will fail because it is missing your API Key. To correct this, you will need to convert it to a function. Right click on the query entry on the left hand nav item bar and select *Rename*. Update the name for the function using the format "GET <object name>" (eg. GET courseCatalog) so that the function name matches the GraphQL object you are querying.



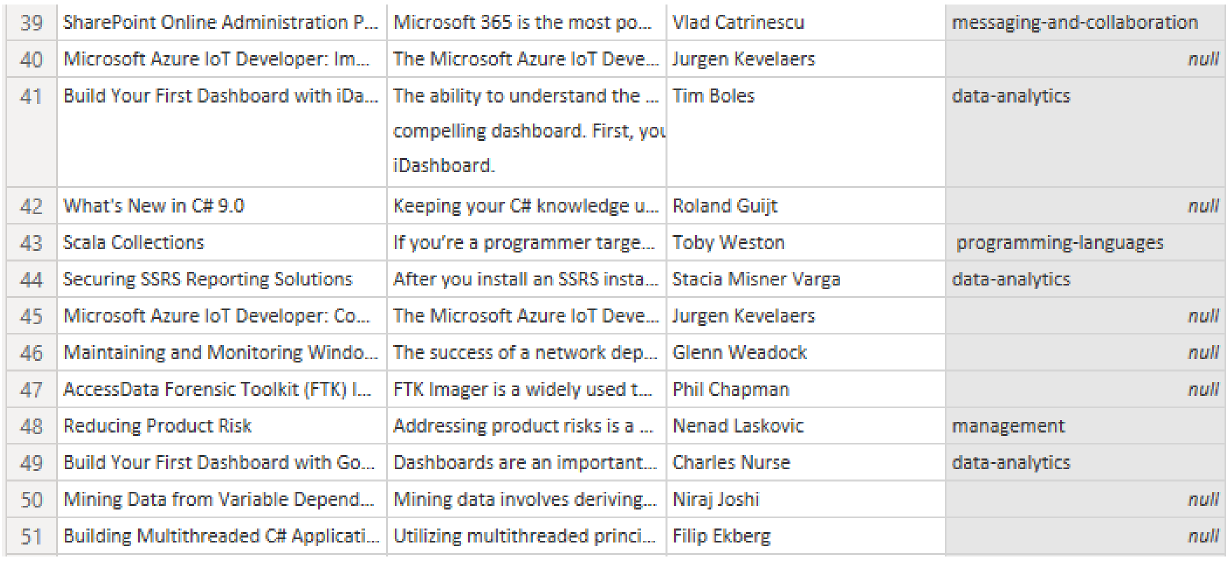
To run the function, click on the function name under the functions folder you created. Note: you may need to click "**Invoke**" before the parameters options appear. After they appear, enter your Pluralsight API key into the apiToken field. This will only need to be entered into the function once. After you have entered the token, click on the button "**Invoke**". This will then create a new query under the Other Queries folder, typically just called Invoked Function. It is recommended you rename it to the same object name used in the naming of the function (ie Get courseCatalog).

To pull the data down you will now use the Invoked Function query under the Other Queries folder. For certain queries further data type mapping may be required (e.g. for nested fields). The query in the example above will result in the table below:



	title	description	authors	tags
1	Serving as a Project Leader	What does it mean to be a pr... In this course, Serving as a Proje First, you'll learn how project te Next, you'll explore how assessn Finally, you'll discover how lead	List	Record
2	Publishing Progressive Web Apps	You already have a Progressiv... some users are not installing it f	List	Record

To expand the columns that display List or Record you will need to click on the icon that is next to the column header. If it is a Record then by default it will select all of the nested fields you specified in the GraphQL query. You will need to click "OK" for it to separate the nested fields into their own columns. It will then return the List placeholder. Click on the icon again next to the header, select the option “Expand to New Rows”, and PowerBI will automatically populate the values into the column. This only has to be done on the first setup with the query. All subsequent runs of the query will automatically know to expand the data. See example result on next page.



39	SharePoint Online Administration P...	Microsoft 365 is the most po...	Vlad Catrinescu	messaging-and-collaboration
40	Microsoft Azure IoT Developer: Im...	The Microsoft Azure IoT Deve...	Jurgen Kevelaers	null
41	Build Your First Dashboard with iDa...	The ability to understand the ... compelling dashboard. First, you iDashboard.	Tim Boles	data-analytics
42	What's New in C# 9.0	Keeping your C# knowledge u...	Roland Guijt	null
43	Scala Collections	If you're a programmer targe...	Toby Weston	programming-languages
44	Securing SSRS Reporting Solutions	After you install an SSRS insta...	Stacia Misner Varga	data-analytics
45	Microsoft Azure IoT Developer: Co...	The Microsoft Azure IoT Deve...	Jurgen Kevelaers	null
46	Maintaining and Monitoring Windo...	The success of a network dep...	Glenn Weadock	null
47	AccessData Forensic Toolkit (FTK) I...	FTK Imager is a widely used t...	Phil Chapman	null
48	Reducing Product Risk	Addressing product risks is a ...	Nenad Laskovic	management
49	Build Your First Dashboard with Go...	Dashboards are an important...	Charles Nurse	data-analytics
50	Mining Data from Variable Depend...	Mining data involves deriving...	Niraj Joshi	null
51	Building Multithreaded C# Applicati...	Utilizing multithreaded princi...	Filip Ekberg	null

You can now reference this data in the PowerBI workbook to build your visualizations and reports. Now that you’ve created the courseCatalog data source you can replicate this same process for other APIs by changing the GraphQL query and the mapping of query fields to table columns.

Conclusion

The Pluralsight GraphQL APIs can be utilized within PowerBI by leveraging Microsoft’s Power Query M language. PowerBI allows developers to combine data from multiple data sources which can enable customers to combine proprietary data with Pluralsight data. By combining both Pluralsight’s GraphQL data with internal customer data, customers can create unique analytical views that are tailored to their needs.

Resources

- Pluralsight Developer Portal
  - Be sure to check out the GraphQL Playground to validate your queries
- PowerBI Query Overview
- Sample PowerBI with Pagination.pbix

FAQ

Q: TLDR. Got anything to quickly get started?
A:. Download example pbix. Invoke functions after adding your API token. Profit.

Q: Does the GraphQL API documentation cover data on SKILLS and FLOW plans?
A: Currently Skills data is available via the Pluralsight GraphQL endpoints and developer portal. See also Flow REST API Intro.

Q: Is there a limitation to the number of queries we create?
A: Best practice is to create 1 query per API unless you have multiple use cases for an API that could benefit from different query filters. We strongly encourage you to use filters to scope the dataset to your needs and improve your reporting performance. For example, you could specify a filter on courseProgress that would return data for the last 30 or 90 days. This would dramatically improve your report performance, reduce load on our servers and provide a better experience to all customers who are using the API endpoint. For more information see the Filters section on the Using GraphQL page.

Q: How do I request an API key?
A: A plan admin on your account can create an API Key on the Manage Keys section of the Developer Portal. If you do not know who your Plan Admin(s) is/are, please reach out to your dedicated CSM or support resource to identify them.

Q: How do I request a multi-plan API key?
A: If your company has multiple Pluralsight plans, you may want a multi-plan API key to query cross-plan data. To request an API key to handle multiple plans, please reach out to us at professionalservices@pluralsight.com. Note: each plan must have the Integrations or ProServ SKU in order to create a single master API token for you.

Q: We keep getting *Expression.Error: We cannot convert the value null to type List.* when we invoke the PowerBI function.
A: This is due to the “first” parameter having a value greater than the total number of records behind the API. You can resolve this by lowering the number and it will then display results. An example would be that you state “**first:500”** while querying the users API, but there are only 200 records there. PowerBI does not know how to handle the response from our API. This can be resolved by reducing the number of records being returned per page to a value of <= 200.

Q: Only x records are being returned by a GraphQL query. I know I have more data than that, how do I pull it?
A: If you use the Complete Script in the example above, PowerBI will paginate through all records and you should have a complete set. If you’re still having issues, review your GraphQL query and any filters you may have applied.

Q: How long does it take to generate an API token?
A: API keys can be generated in a matter of minutes on the developer portal and are active immediately. If there is a delay for any reason or it does not function, please contact support@pluralsight.com.

Q: Will Pluralsight Professional Services create and maintain the PowerBI Dashboards for our organization?

A: No. Professional Services may support a templated dashboard example in the future.

Q: Does this cost extra?
A: If you have purchased the Integration's SKU, or have an Enterprise SKU that was purchased before July 1st 2020 and it has not yet expired and been renewed, then there is no additional charge. If you have not purchased our Integrations SKU and have a subscription starting after July 1st 2020, whether a new subscription or renewal, then you will have to speak to your CSM and Sales rep to get started.

Q: What data can I not see?
A: If it is not listed in the Developer Portal documentation schema, then it is not available. Keep an eye on the change log section of the webpage for updates.

Q: I have resources to maintain the .pbix file, but I don't have resources to dedicate to development from scratch due to time-constraints/business priorities? Do you have any options to assist?
A: We do. The Sample PowerBI.pbix file has multiple preloaded queries already set and ready for analytic use. All you need to do is replace the API token that is present in each function in the file with your own token. It will then pull data for your plan and should remove the bulk of the needed query and data collection setup work. All you would need to do then is create the reports you want.

Q: I would prefer to use the Pluralsight REST APIs. Can I continue to use your REST APIs?
A: The REST APIs will be deprecated in 2021. The GraphQL APIs contain much richer datasets for multiple content types (paths, channels, interactive courses, labs, etc) as well as Skills, Priorities, and Roles that are not available via REST.

Appendix

For a deeper understanding of pagination and other components of the PowerBI script, read on.

Deep Dive: Explaining Pagination Scripts

When dealing with large datasets, pagination can significantly improve performance and system load. By default, Pluralsight GraphQL queries will return the first 100 records unless you explicitly set a different page size by using the "first" parameter. Pagination allows you to specify how many "rows" of data you want to read and the "page" on which you want to start. Each record has a cursor that identifies the record's place in the greater dataset. Queries that offer pagination will have the following two parameters:

- first: Specifies how many records to fetch per query (max of 10,000 per query).
- after: Used to request the next result set from the query. Your query will return the value PageInfo.endCursor, and you should put that value here for the next query.

Recursive Pagination Method

The recursive pagination method leverages both parameters **first** and **after** as well as it requires a new object called **pageInfo**. The **first** parameter allows to fetch a fixed set of records per query and the **after** parameter allows the query to request the next result set. Your query will return the value PageInfo.endCursor, and you should put that value here for the next query.

As an example, let's use this basic query:

```
"{""query"":""query{ courseCatalog { nodes { title } } }"" }"
```

By default every query sent to GraphQL has additional properties that can be added with the query. For this method, it requires the following properties **first, after** and a new object called **pageInfo** that contains two keys: **endCursor** and **hasNextPage** to be added. Using the property and object the query needs to be modified as follows:

```
{""query"":""query { courseCatalog (first: 5000 after: "") { pageInfo {
endCursor, hasNextPage } nodes { title} } }"" }"
```

In addition, since a new object is being introduced, the M statement needs to be updated to reflect this new table. Therefore, the new M statement will need to include a new object to query the endCursor and hasNextPage value. Therefore the M statement will be updated with the following addition:

```
 Content = Text.ToBinary("{ ""query"": ""query{ courseCatalog(first: 500 0,
after:\""" & endCursor & "\"" ) { pageInfo {endCursor, hasNextPage}, nodes
{ title } } }"" }")
      ]
    )),
```

As stated earlier, GraphQL has a query limit of 10,000 records and many datasets exceed this record count. Therefore, the M Script needs to be converted to a function to be able to re-run until it's pulled all the data. In order to solve this problem, three variables need to be introduced above the "let":

```
(apiToken as text, optional endCursor as text, optional data as list) =>

let
```

These three variables will be used to iterate the query over and over until all the records are retrieved. The variables are defined as:

- apiToken: inputbox to allow the function to retrieve any API key
- endCursor: text field to store the last record per query called
- data: list object to store all the records

Next the query returns two objects (pageInfo and nodes) which will need to be parse out as shown below

```
pageInfo = Source[data][courseCatalog][pageInfo],
nodes = Source[data][courseCatalog][nodes],
```

Finally the query needs to be looped over and over until pageInfo[endCursor] is false which is shown on the next page.

```
 appendedData =
        if pageInfo[hasNextPage] = true and data is null then
            List.Combine({{}, nodes})
        else List.Combine({data, nodes}),

    output =
        if pageInfo[hasNextPage] = true then
            @#"GET courseCatalog"(apiToken, pageInfo[endCursor],
appendedData)
        else
            Table.FromList(appendedData,  Record.FieldValues, {"title"})
```

Here is the full query again with comments:

```
(apiToken as text, optional endCursor as text, optional data as list) =>
let
     //endCursor var. capture endCursor value during loop
     endCursor = if endCursor is null then "" else endCursor,

    Source =
Json.Document(Web.Contents("https://paas-api.pluralsight.com/graphql",
        [
            Headers=[
                #"Method"="POST",
                      #"Content-Type"="application/json",
                      #"Authorization"="Bearer " & apiToken
            ],
            //endCursor is being passed inside the query string
            //first: 5000 can be adjusted as needed
            Content = Text.ToBinary("{ ""query"": ""query{
courseCatalog(first: 5000, after:\""" & endCursor & "\"" ) { pageInfo
{endCursor, hasNextPage}, nodes { title } } }"" }")
        ]
    )),
    //Extract pageInfo and the nodes list from results
    pageInfo = Source[data][courseCatalog][pageInfo],
    nodes = Source[data][courseCatalog][nodes],

    //Pagination merging logic. Will append every record to appendedData
    appendedData =
        if pageInfo[hasNextPage] = true and data is null then
            List.Combine({{}, nodes})
        else List.Combine({data, nodes}),

    //Looping logic. If no more records, stop, if not, repeat.
    output =
        if pageInfo[hasNextPage] = true then
            @#"GET courseCatalog"(apiToken, pageInfo[endCursor],
appendedData)
        else
            Table.FromList(appendedData,  Record.FieldValues, {"title"})

in
    output
```