

How to work with Gas operational data in Excel using PowerQuery

The following example will show how to set up and use PowerQuery to fetch Gas operational data from parameter values in an Excel Worksheet.

What you will need

- The URL generated by the website
- A copy of Excel 2016 or later
- A text editor (optional)

Part A will describe how to create a connection and retrieve the data from the API and place it in a table in Excel

Part B will describe how to update the connection to read data from Excel making it more dynamic

Contents

Part A: Fetching data from the API	1
Step 1) Generate the Query URL on the Website	1
Step 2) Create a new Connection in Power Query.....	1
Part B: Updating the Connection with data from Excel	2
Step 3) Create a named range in Excel.....	2
Step 4) (optional) Use a formula to make the list of Publication Ids	3
Step 5) Turn the Connection into a Query and tell it where to look for the publication ids	4
Step 6) Turn the list of parameters into a Query statement.....	5
Step 7) (optional) Adding the remaining parameters to the Query.....	8
Customer Download Parameter descriptions.....	9
Appendix 1 Working with the earlier version of the Custom Download Tool	10

Part A: Fetching data from the API

Step 1) Generate the Query URL on the Website

To retrieve the query string:

- 1) Navigate to the 'FIND GAS TRANSMISSION DATA' page
- 2) Select your required query parameters and date/time range
- 3) Select the data item or items you require
- 4) Click on the URL icon bottom right of the page

The system will automatically copy the URL to your computers clipboard as long string of text that looks something like the following depending on what you selected in the previous steps

```
https://data.nationalgas.com/api/find-gas-data-download?applicableFor=Y&dateFrom=2022-07-07T00:00:00&dateTo=2022-07-12T23:59:59&dateType=GASDAY&latestFlag=Y&ids=PUBOB39,PUBOB42&type=CSV
```

Figure 1 Query provided by the web site

Step 2) Create a new Connection in Power Query

To create a connection in Excel

- 1) Select a cell for the results in a Worksheet (e.g. A1)
- 2) On the menu bar select Data -> From Web and Excel will open up the From Web dialogue box

- 3) If a message appears telling your web browser has restricted this file from showing active content you will need to click on it and select the 'Allow blocked content' option followed by 'Yes' to confirm
- 4) Paste your newly retrieved URL into the entry boxed marked URL and the click OK
- 5) Excel will ask you if you want to use anonymous access for this Web content
- 6) Keep this option and click on the Connect button
- 7) PowerQuery will fetch the data and present it to you in a new dialogue window
- 8) Click on the Load button and PowerQuery will enter the data into the worksheet as a new Table

Part B: Updating the Connection with data from Excel

In the following example we will show how to alter the query to fetch different data items from a list in the spreadsheet. This example will work for all the parameters except the date parameters which need an additional line and this will be demonstrated later in the guide.

Step 3) Create a named range in Excel

To start using values from our spreadsheet in place of the fixed values in our current URL string, we need to make a few simple changes. We'll start by setting a cell in our spreadsheet as a named range e.g. 'newItems' and put our list of Publication Ids in the cell as a comma separated list

Note | A full list of all the available data items and their associated publication ids is available to download from the About section of the website. A link can be found on the Transmission operational data page on the National Gas Transmission website here [Data API Service](#).

Method 1 is to use the Name box (see Figure 2 Making a cell a named range below)

- 1) Select a cell outside of the table our data is in that you will use to hold your list of Publication Ids (Cell I2 in Figure 2)
- 2) Enter the name into the Name Box to the right of the formula bar at the top of the Worksheet (this is also highlighted in Figure 2) and press enter

Method 2 is to use the Name Manager:

- 1) Select Formulas -> Name Manger -> New
- 2) Enter your chosen name in Name box (e.g. 'newItems'), leave the Scope as Workbook, and use the Refers to range selector to select a cell outside of the table our data is in that you will use to hold your list of Publication Ids
- 3) Click on OK and Excel will name the cell and display this in the Name Box as shown in Figure 2 Making a cell a named range below

Note | You can of course call your named range and variable anything you choose that is more meaningful to you. You should however avoid using the same name for both and any of the key words that Power Query relies on to identify elements of the query such as let, Query, Source etc.

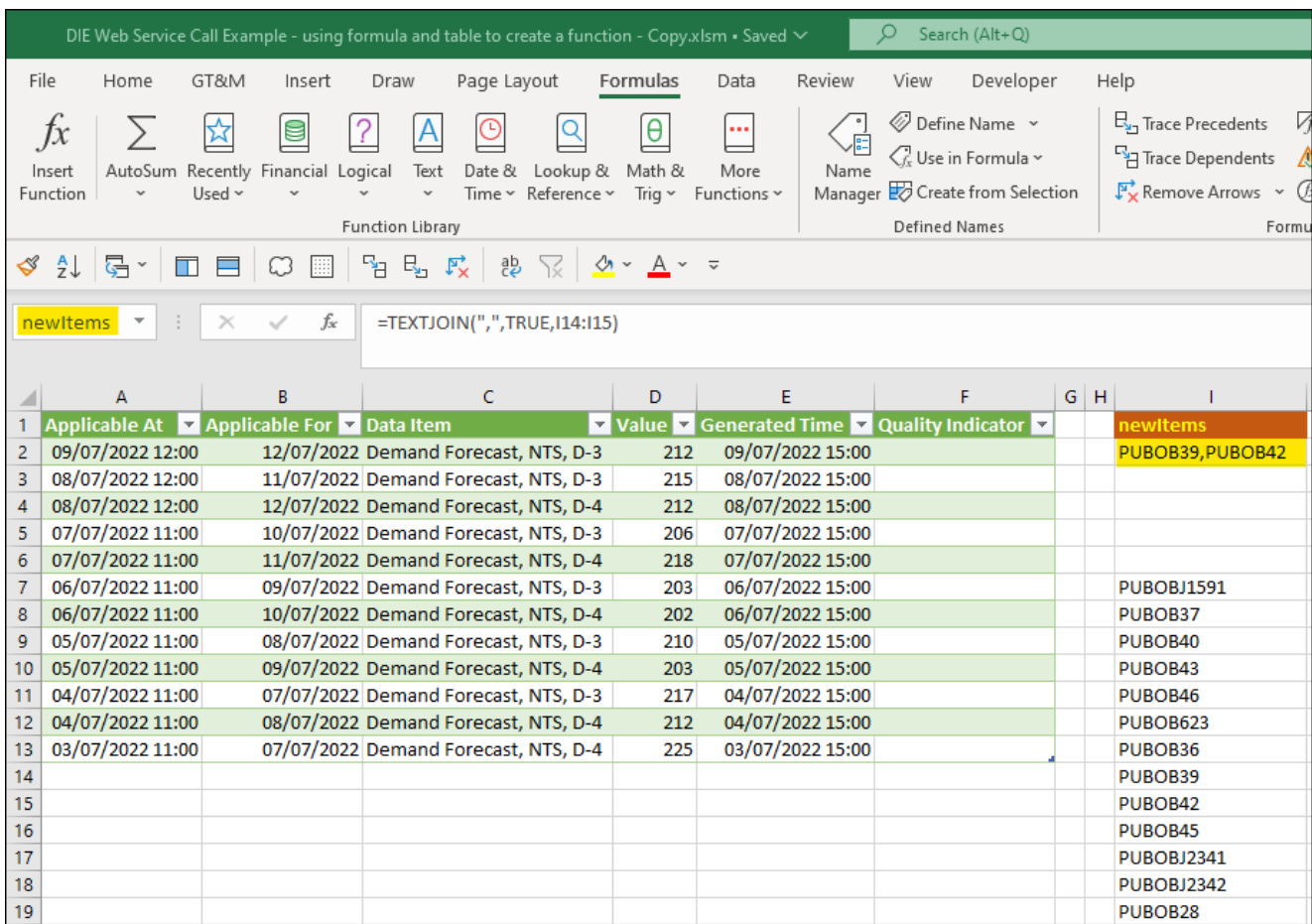


Figure 2 Making a cell a named range

Step 4) (optional) Use a formula to make the list of Publication Ids

You can manually enter a single publication id or comma separated list of publication ids into the new named cell, however, for greater flexibility you can have Excel create this from a list of publication ids or even another query. In the example in Figure 3 (below) we have used the in-built Excel function TEXTJOIN to create the comma separated list in our named cell in Figure 4 from a selection of publication ids.

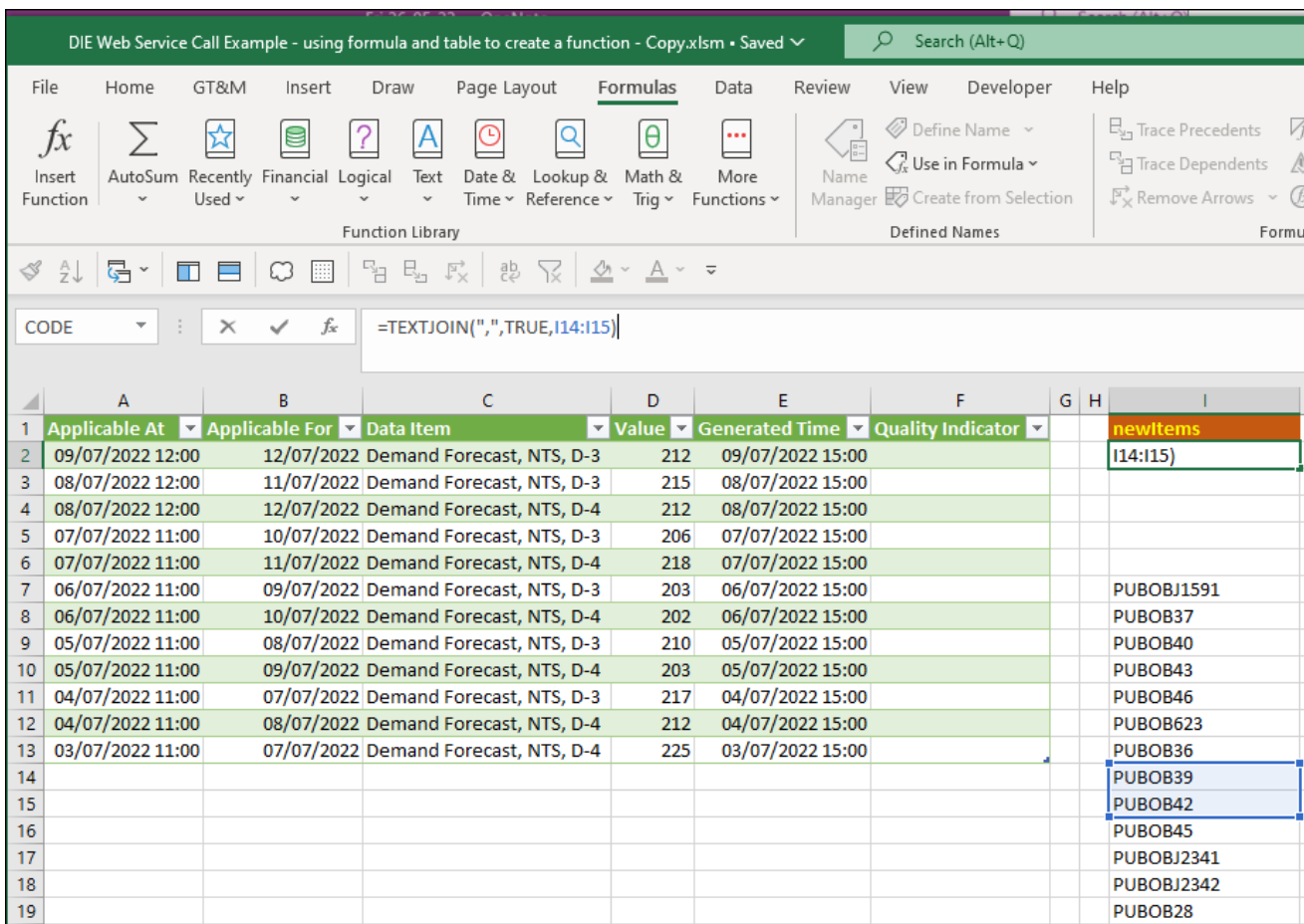


Figure 3 Using the TEXTJOIN function to create a list of data items

We will see later that ids=" PUBOB39,PUBOB42" will become ids=myDataItems in the new query we will create in the next section.

Step 5) Turn the Connection into a Query and tell it where to look for the publication ids

Now that you have created a Connection (see Create a new Connection in Power Query above) you can open the Advance Editor as follows:

- 1) Select any cell in the new Table created in Part A and Excel will display two new menu items on the right of the menu bar next to Help called Table Design and Query
- 2) Select Query -> Edit -> Advanced Editor
- 3) Excel will display the Advanced Query Editor and the query will look like this.

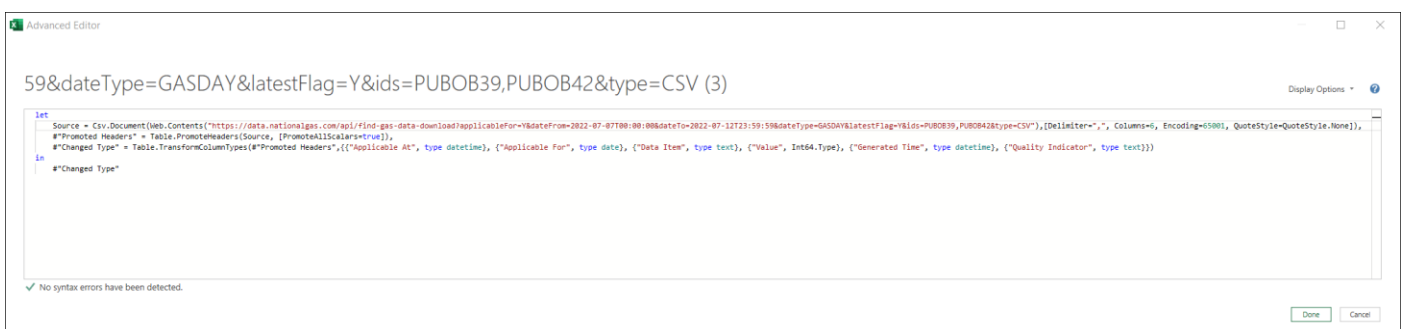
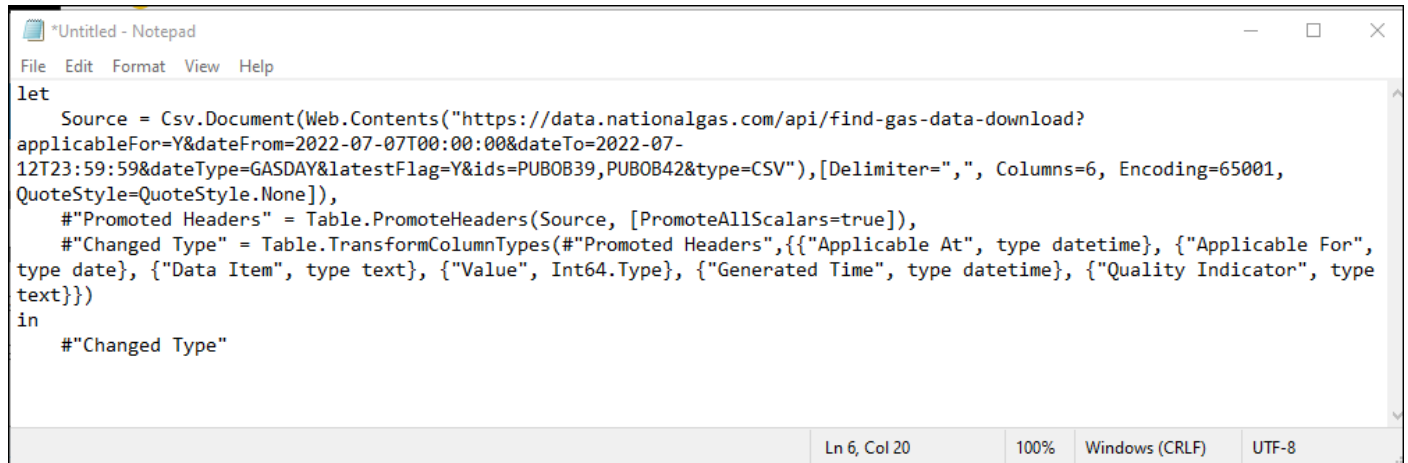


Figure 4 The API Connection before making any changes

Note | The text in the Advanced Editor doesn't wrap so you will need to drag the window to make it larger or use the horizontal scroll control. Alternatively you can copy it and paste it into any other text editor which allows word wrap if you find that more manageable. Once complete simply copy and paste it back into the Advanced Editor overwriting the current contents.

For example, in Notepad it will look like the following



```
let
    Source = Csv.Document(Web.Contents("https://data.nationalgas.com/api/find-gas-data-download?
applicableFor=Y&dateFrom=2022-07-07T00:00:00&dateTo=2022-07-
12T23:59:59&dateType=GASDAY&latestFlag=Y&ids=PUB0B39,PUB0B42&type=CSV"),[Delimiter=",", Columns=6, Encoding=65001,
QuoteStyle=QuoteStyle.None]),
    #"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Applicable At", type datetime}, {"Applicable For",
type date}, {"Data Item", type text}, {"Value", Int64.Type}, {"Generated Time", type datetime}, {"Quality Indicator", type
text}}})
in
    #"Changed Type"
```

Figure 5 What the API Connection looks like in Notepad

Our next step is to add a new line under the 'let' statement to tell PowerQuery where to look for the value(s).

```
let
myDataItems = Excel.CurrentWorkbook(){[Name="newItems"]}[Content]{0}[Column1],
```

Figure 6 Declaring a variable to be used in the query

We put the named range we created containing the list of publication ids as the value for Name (Name="newItems") and we declare this as a variable to be used in the query itself called 'myDataItems'. We will use this in a new query to replace the previous fixed list for ids which we will do next.

First, we split the URL up into the domain name and the relative path and ensure that each is enclosed in double quotes (""). The new items we add are the two sets of square brackets ([]), the term RelativePath= and Query =

```
Source = Csv.Document(Web.Contents("https://data.nationalgas.com/api/find-gas-data-
download?
```

Figure 7 The original line from the Connection with the full path to the API

becomes

```
Source = Csv.Document(
    Web.Contents("https://data.nationalgas.com",
    [
        RelativePath="/api/find-gas-data-download",
```

Figure 8 The full path to the API split into the domain and the relative path

Step 6) Turn the list of parameters into a Query statement

Then we can make the remaining text more legible by putting all the parameters on their own line. We need to replace all the & with a comma, create a new line and then enclose all the fixed values in double quotes.

So

```
applicableFor=Y&dateFrom=2022-07-07T00:00:00&dateTo=2022-07-12T23:59:59&dateType=GASDAY&latestFlag=Y&ids=PUBOB39,PUBOB42&type=CSV"
```

Figure 9 The list of Parameters passed in the original Connection

becomes

```
Query = [  
  applicableFor="Y",  
  dateFrom="2022-07-07T00:00:00",  
  dateTo="2022-07-12T23:59:59",  
  dateType="GASDAY",  
  latestFlag="Y",  
  ids="PUBOB39,PUBOB42",  
  type="CSV"  
]
```

Figure 10 The Parameters when changed into a Query

Finally we replace our old publication id (or list of publication ids) with the new variable 'myDataItems' as described earlier

```
ids=" PUBOB39,PUBOB42",
```

Figure 11 The Publication Id as a hard coded value

becomes

```
ids=myDataItems,
```

Figure 12 The hard coded Publication Id replaced with our reference to the Excel Workbook (see Step 1)

Note | This does not get enclosed in double quotes

So the whole query now looks like

```
Query = [  
  applicableFor="N",  
  dateFrom="2022-07-07T00:00:00",  
  dateTo="2022-07-12T23:59:59",  
  dateType="GASDAY",  
  latestFlag="Y",  
  ids=myDataItems,  
  type="CSV"  
]
```

Figure 13 The finished Query statement with the reference to Excel added

The final query should now look like the following example. Here we have tidied up the "#Promoted Headers" and "#Changed Type" sections to make them more legible but we have not made any changes to these, so it isn't necessary.

```
let  
  myDataItems = Excel.CurrentWorkbook(){[Name="newItems"]}[Content]{0}[Column1],  
  Source = Csv.Document(  
    Web.Contents("https://data.nationalgas.com",  
      [  
        RelativePath="/api/find-gas-data-download",  
        Query = [  
          applicableFor="N",  
          dateFrom="2022-07-07T00:00:00",  
          dateTo="2022-07-12T23:59:59",  
          dateType="GASDAY",  
          latestFlag="Y",  
          ids=myDataItems,  
          type="CSV"  
        ]  
      ]  
    )
```

```

    applicableFor="Y",
    dateFrom="2022-07-07T00:00:00",
    dateTo="2022-07-12T23:59:59",
    dateType="GASDAY",
    latestFlag="Y",
    ids=myDataItems,
    type="CSV"
  ]
]),

[
  Delimiter=",",
  Columns=6,
  Encoding=65001,
  QuoteStyle=QuoteStyle.None
]
),
#"Promoted Headers" = Table.PromoteHeaders(Source, [PromoteAllScalars=true]),
#"Changed Type" = Table.TransformColumnTypes(
  #"Promoted Headers",{
    {"Applicable At", type datetime},
    {"Applicable For", type date},
    {"Data Item", type text},
    {"Value", Int64.Type},
    {"Generated Time", type datetime},
    {"Quality Indicator", type text}}
  )
in
  #"Changed Type"

```

Figure 14 The finished Query in full

In the Advanced Editor in Power Query it will look like this.

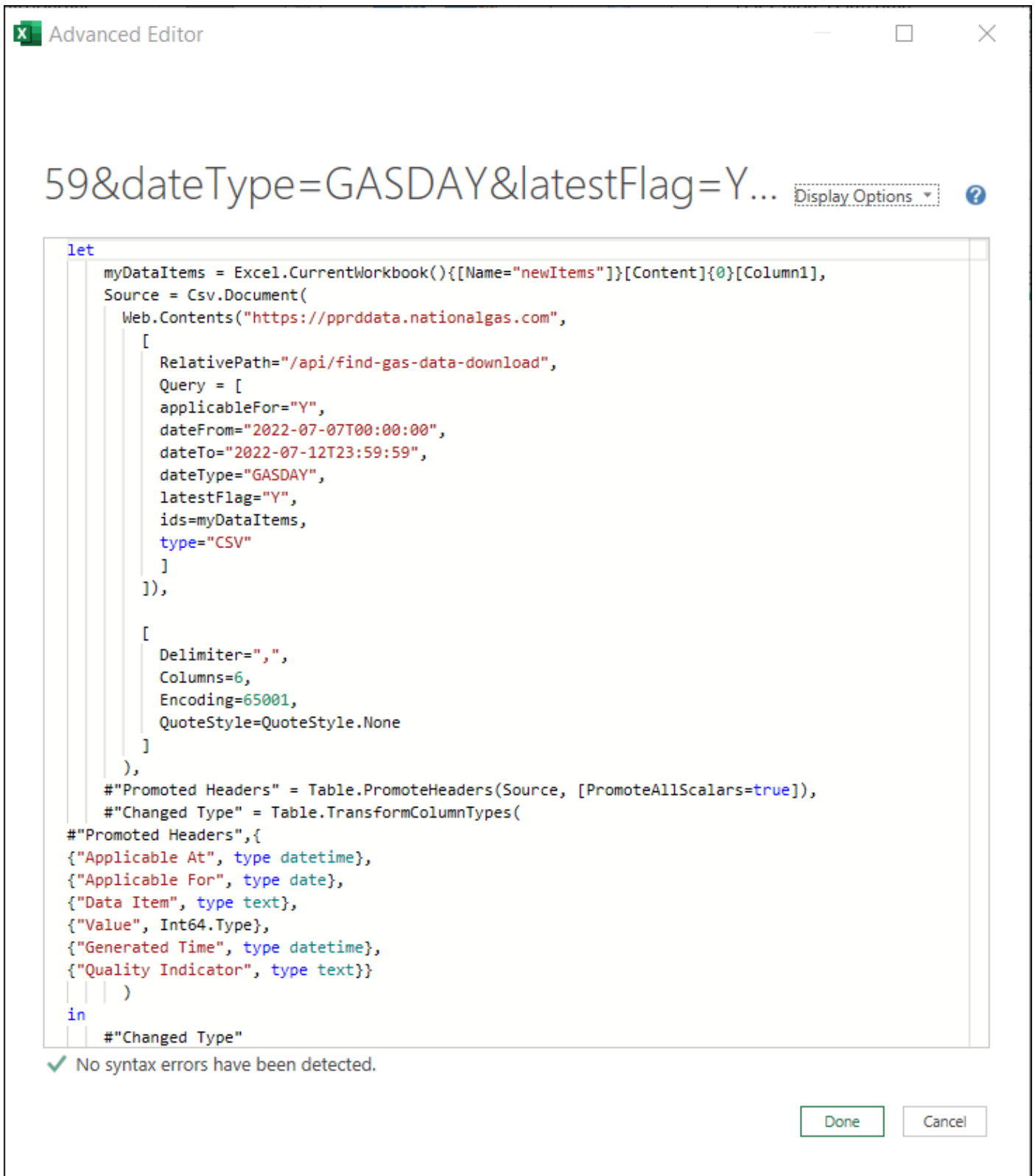


Figure 15 The finished Query in the Advanced Editor

Notes | You can use any text editor such as Notepad, Wordpad, Word, Visual Studio Code etc.). If you have used your own text editor, then you can simply copy from there and paste over the contents in the Advanced Editor in Power Query. Note too that the Advanced editor will show you when all the syntax is correct in the bottom left of the window.

To finish we only need to click on Done then Close & Load.

Step 7) (optional) Adding the remaining parameters to the Query

The following parameters can be added in using the same technique demonstrated above

- applicableFor
- dateType
- latestFlag

The two date parameters require an extra step to convert them to a format that PowerQuery understands

- dateFrom
- dateTo

The following example shows how to convert the value in the named range 'startDate' in our spreadsheet to a variable called 'from' which we can use in our Query (see Figure 16 Converting a date in the spreadsheet to the right format for PowerQuery). Use the following custom format for the cell (yyyy-mm-dd hh:mm:ss) in your worksheet.

```
from = DateTime.ToText(Excel.CurrentWorkbook(){[Name="startDate"]}[Content]{0}[Column1], "yyyy-MM-dd HH:mm:ss"),
```

Figure 16 Converting a date in the spreadsheet to the right format for PowerQuery

Add this to the Query as follows

replace

```
dateFrom="2022-07-07T00:00:00",
```

Figure 17 A date as a hard coded value

with

```
dateFrom=from,
```

Figure 18 The hard coded date replaced with our reference to the Excel Workbook

Customer Download Parameter descriptions

The following table describes each parameter and the allowable values

Parameter	Values	Description	Format	Length	Validation	Example
applicableFor	Y N	Use Y to return data applicable for the date range. Choose N for data created during the date range	Enumerated value	n/a	Mandatory	"Y"
dateFrom	UTZ datetime values	Start date	String	19	Mandatory	"2022-03-09T00:00:00"
dateTo	UTZ datetime values	End date	String	19	Mandatory	"2022-03-09T00:00:00"
dateType	GASDAY NORMALDAY	Use Y if applicable for is Y and N if applicable for is N.	Enumerated value	n/a	Mandatory	"GASDAY"
latestFlag	Y N	Use Y to only return the latest data sets	Enumerated value	n/a	Mandatory	"Y"
ids	Comma separated list of Publication Ids	The unique publication data id(s) for each data item you want to return	Comma separated list of strings	n/a	Mandatory	"PUBOBJ1591" or "PUBOBJ36, PUBOBJ39, PUBOBJ42"
type	CSV	This is a fixed value	String	3	Mandatory	"CSV"

Appendix 1 Working with the earlier version of the Custom Download Tool

The National Gas Transmission site will continue to support the previous version of the URL query string for the API which returned data in an HTML format. Any customers who have already have solutions that use this can continue to do so and the documentation for using this is included here.

Note | The path to the original URL is "https://mip-prod-web.azurewebsites.net/CustomDataDownload" and the list of parameters also remains the same as they were before e.g.

```
LatestValue="false",
Applicable="applicableAt",
FromUtcDatetime="2022-12-19T00:00:00.000Z",
ToUtcDateTime="2022-12-25T23:30:00.000Z",
PublicationObjectStagingIds="PUBOB36
```

Example Code:

Sample Code to return values across defined months from excel. The three highlighted items can be altered/linked to defined names from your spreadsheet to select the right data. Note that ids can use a cell with multiple publication objects listed, and one query has a limit of 3600 line items (although you can add more than one query to a spreadsheet by using multiple tabs). Delete notes in italic before adding to your spreadsheet.

let

```
ids = Excel.CurrentWorkbook(){[Name="EA"]}[Content]{0}[Column1],
```

EA = the data item(s) that you want to display in your query. 'EA' here is a defined name, assigned to a cell in the workbook.

```
startDate = DateTime.Date(Excel.CurrentWorkbook(){[Name="StartDate"]}[Content]{0}[Column1]),
```

StartDate = the date from which you want to collect data from. 'StartDate' is a defined name, assigned to a cell in the workbook. In this example this is looking at after the day data and working backwards, so start date is effectively the last value

```
monthCount = Excel.CurrentWorkbook(){[Name="MonthCount"]}[Content]{0}[Column1],
```

MonthCount = the number of months you want to collect data for, again from a defined cell name in the excel sheet

```
from = Date.ToText(Date.AddMonths(DateTime.Date(startDate), monthCount * -1), "yyyy-MM-dd"),
```

Setting the start range for the data as 1 month (or defined amount) before the defined start date

```
to = Date.ToText(DateTime.Date(startDate), "yyyy-MM-dd"),
```

Setting end date as the date defined by the user

```
Source = Web.Page(
  Web.Contents("https://mip-prod-web.azurewebsites.net",
    [
      RelativePath="/CustomDataDownload",
      Query =
        [
          Applicable = "ApplicableFor",
          PublicationObjectStagingIds=ids,
          FromUTCDateTime=from,
          ToUTCDateTime=to
        ]
    ]
  )
)
```

```

    ])
),
Data0 = Source{0}[Data],
#"Changed Type" = Table.TransformColumnTypes(Data0,{{"Applicable At (UK Local Time)", type datetime},
{"Applicable For (Gas Day)", type date}, {"Data Item", type text}, {"Value", type number}, {"Generated Time (UK
Local Time)", type datetime}, {"Quality Indicator", type text}})
in
#"Changed Type"

```

Other Potential Changes:

1. To change the range from number of months to number of days:
 - 1.1. Change 3rd line to use a difference named cell **DayCount** and change the variable name

```

dayCount = Excel.CurrentWorkbook()[{Name="DayCount"}][Content]{0}[Column1],

```
 - 1.2. Amend calculation of the from date to use the **dayCount** and add number of days (line 4)

```

from = Date.ToText(Date.AddDays(DateTime.Date(startDate), dayCount * -1), "yyyy-MM-dd"),

```
2. To make the data work forwards from the defined date rather than backwards:
 - 2.1. Line 4 change to:

```

from = Date.ToText(DateTime.Date(startDate), "yyyy-MM-dd"),

```
 - 2.2. Line 5 change to:

```

to = Date.ToText(Date.AddMonths(DateTime.Date(startDate), monthCount ), "yyyy-MM-dd"),

```

Sample 1

<https://mip-prd-web.azurewebsites.net/CustomDataDownload?LatestValue=false&Applicable=applicableAt&FromUtcDatetime=2022-12-19T00:00:00.000Z&ToUtcDateTime=2022-12-25T23:30:00.000Z&PublicationObjectStagingIds=PUBOB36,PUBOB39>

```

let
#"Table 0 (2)" = let
Source = Web.Page(
Web.Contents("https://mip-prd-web.azurewebsites.net",
[
RelativePath="/CustomDataDownload",
Query = [
LatestValue="false",
Applicable="applicableAt",
FromUtcDatetime="2022-12-19T00:00:00.000Z",
ToUtcDateTime="2022-12-25T23:30:00.000Z",
PublicationObjectStagingIds="PUBOB36"
]
])
),
Data0 = Source{0}[Data],

```

```
#"Changed Type" = Table.TransformColumnTypes(Data0,{{"Applicable At (UK Local Time)", type datetime},
{"Applicable For (Gas Day)", type date}, {"Data Item", type text}, {"Value", Int64.Type}, {"Generated Time (UK Local
Time)", type datetime}, {"Quality Indicator", type text}})
```

```
in
```

```
#"Changed Type",
Custom1 = #"Table 0 (2)"
```

```
in
```

```
Custom1
```