Data Brothers

Let’s create your custom connector

25.06.2025 | Power BI Summer School

ŠTĚPÁN REŠL (Steve)

Lead technical consultant

Data Brothers

Microsoft® Most Valuable Professional

Power BI kafíčko

Data Meerkat

LINKEDIN

X

BLUESKY

Agenda

What do we need to cover today?

Function summary

Scenarios for Custom Connectors

Power Query Extensions

Prebuild parts of Custom connectors

Data Brothers

Function summary

What we need to know about functions before we start creating our custom connectors.

Data Brothers

Example of two functions

Can you spot a difference?

Number.Power(**number** as nullable number, **power** as nullable number) as nullable number

```
Number.Power(5) = error
Number.Power(5,null) = null
Number.Power(5,2) = 25
```

List.Transform(**list** as list, **transform** as function) as list

```
List.Transform({1..3}, each _ + 1)
List.Transform({1..3}, (x)=> x +1)
List.Transform({1..3}, fctAddOne)
```

A few useful insights

Little things that can have a big impact!

The return of functions can be **anything**...

You can reference functions without "()"

fctName()() can be **valid** function

Some native functions also ask custom!!!

each _ is just sugar syntax of **(x) => x**

Basic defining

Creating a custom function is not hard. You just need ()=> and that's it!

```
(<parameters>) => <function-body>
```

```
(x, y) => x * y
```

Type supporting

Is great to support users in understanding and preventing the usage of wrong data types!

```
(<parameters> as <type-definition>)
<return-data-type> =>
<function-body>
```

```
(x as number, y as number) as number => x * y
```

Optional parameters

The designation of custom functions is directly subordinate to its author in the given model

```
([<required-parameters>],[<optional-parameters>])
<return-data-type> =>
<function-body>
```

```
(x as number, y as number, optional z as number) as
number =>
        x * y * ( if z <> null then z else 1 )
```

More complex function body

Not every time you will need just simple expression. More often, you will also need [let .. in]

```
(x as number, y as number, optional z as number) as
number =>
    let
        additional = ( if z <> null then z else 1 ),
        vl = x * y * additional
    in
        vl
```

Or maybe… where they can help?

In short, so many complexities have to pay off somehow.

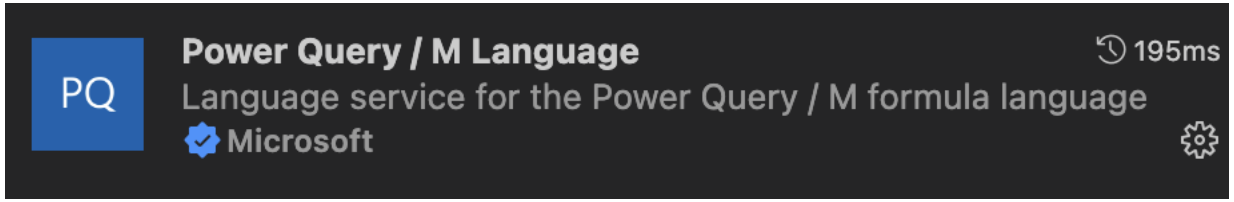
REUSABILITY

READABILITY

SCOPED

Reusability

You can use them whenever you need them. You only need to create once!



Readability

Larger blocks of code are more complicated to read and cause complications to maintain the solution.

```
= Table.AddColumn(
        filteredRowsOnlyToActive,
        “DurationInYears”,
        Number.Round( List.Sum(
                List.Generate(
                        ()=> [initDate = [Date], counter = 1],
                        each _[initDate] <> today,
                        each [initDate = Date.AddMonths([initDate],1), counter = 1],
                        each [counter]
        ) ) /12, 2 )
)
```

Readability

Now, you and other developers can easily understand the code. Go to detail only if it is needed.

```
= Table.AddColumn(
        filteredRowsOnlyToActive,
        “DurationInYears”,
        each #”get-DurationInYears”(_[Date])
)
```

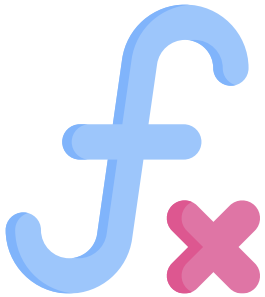
Scenarios for Custom Connector

When we should be thinking about using custom connectors?

Data Brothers

Scenarios

There is a scenario in every solution, yeah? Nope! Let's start with what connectors can provide for you.



Functions Library

Whole query library

Authentication handler

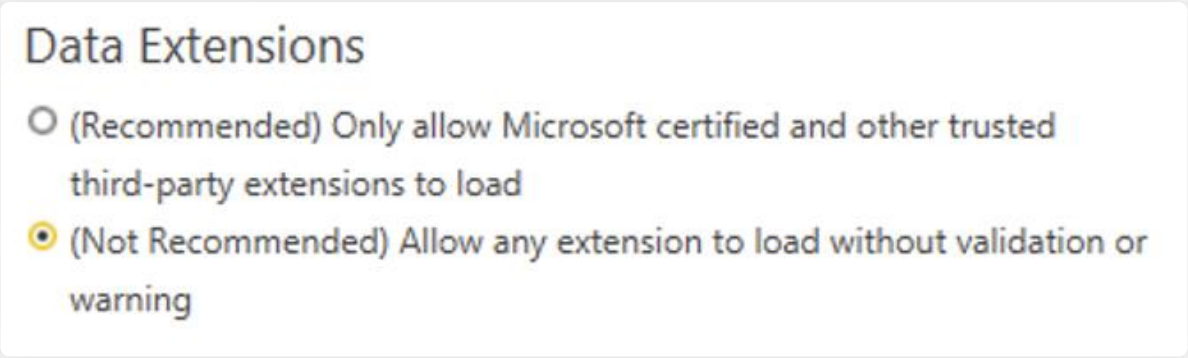
Proxy handler

But there is a catch…

and it is not a small one…

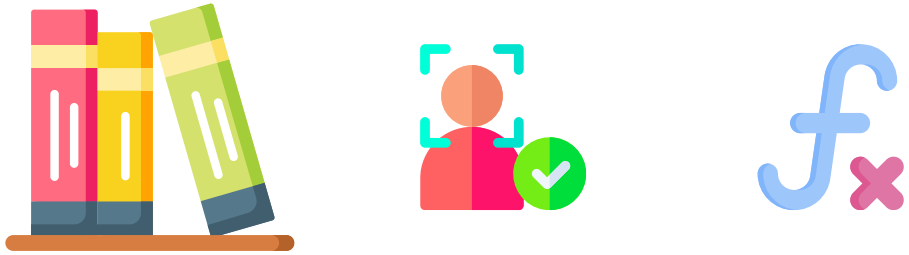
Executing custom connectors in Power BI / Microsoft Fabric Service requires an on-premises Data Gateway that needs to have support for custom connectors enabled. The custom connector needs to be accessible for the gateway on the selected system path.

In the case of the usage of Power BI Desktop, users need to place the custom connector into the system path and also needs to allow usage of Data extensions:



Relevant scenarios

So... in these scenarious we really want to use it.



The most common scenario

Power Query extensions

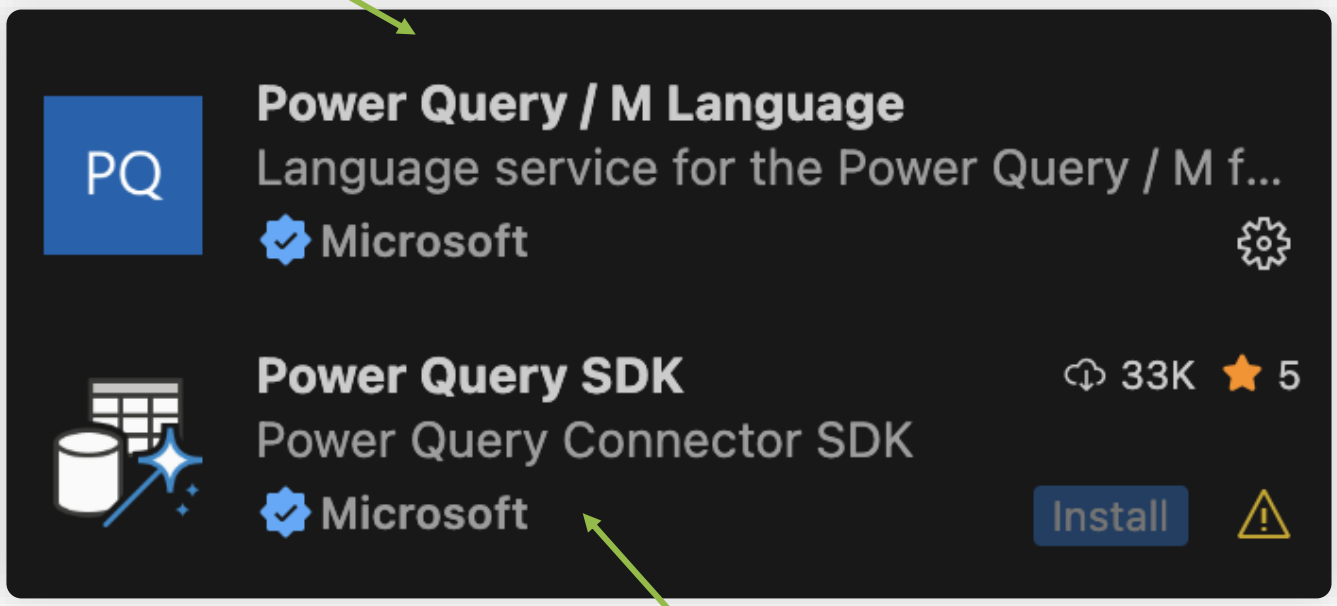
Visual Studio with the ability to work with the M language.

Data Brothers

Extensions

Created and maintained by Microsoft

Language support



Custom Connector Development Support

Power Query / M Language

Not just lexicon but also formatter

```
let
    Source = #table(
        type table [id = number, name = text, age = number, city = text],
        {{1, "Alice", 30, "New York"},{2, "Bob", 25, "Los Angeles"},{3, "Charlie", 35, "Chicago"},{4, "David", 28, "Houston"}
        }
    ),
    filteredRows = Table.SelectRows(Source, each [age] > 30),
    selectedColumns = Table.SelectColumns(filteredRows, {"id", "name", "city"})
in
    selectedColumns
```

```
let
    Source = #table(
        type table [id = number, name = text, age = number, city = text],
        {
            {1, "Alice", 30, "New York"},
            {2, "Bob", 25, "Los Angeles"},
            {3, "Charlie", 35, "Chicago"},
            {4, "David", 28, "Houston"}
        }
    ),
    filteredRows = Table.SelectRows(
        Source,
        each [age] > 30
    ),
    selectedColumns = Table.SelectColumns(
        filteredRows,
        {"id", "name", "city"}
    )
in
    selectedColumns
```

Power Query SDK

Created and maintained by Microsoft

- Create a new extension project using a custom connector template
- Build a connector file (.mez)
- Set and manage credentials
- Test queries
- Test your TestConnection function for refresh on the cloud

Prebuild parts of Custom connectors

All what is already created for us.

Data Brothers

Pre-build set-up for Custom Connector

What will all be created for you

- customConnectorDevelopment.pq — **PQ file, which contains behaviour of the connector**
- customConnectorDevelopment.proj — Based on this file .mez file will be created
- customConnectorDevelopment.query.pq — A file that you can execute to validate the working of connector
- customConnectorDevelopment16.png
- customConnectorDevelopment20.png
- customConnectorDevelopment24.png
- customConnectorDevelopment32.png
- customConnectorDevelopment40.png
- customConnectorDevelopment48.png
- customConnectorDevelopment64.png
- customConnectorDevelopment80.png

  Images that will be used for the connector
- resources.resx — XML file with parameters for names and texts

Pre-build M(ain) code

What we definitely need is already created

```
// This file contains your Data Connector logic
[Version = "1.0.0"]
section customConnectorDevelopment;

[DataSource.Kind = "customConnectorDevelopment", Publish = "customConnectorDevelopment.Publish"]
shared customConnectorDevelopment.Contents = (optional message as text) =>
    let
        _message = if (message <> null) then message else "(no message)",
        a = "Hello from customConnectorDevelopment: " & _message
    in
        a;

// Data Source Kind description
customConnectorDevelopment = [
    Authentication = [
        // Key = [],
        // UsernamePassword = [],
        // Windows = [],
        Anonymous = []
    ]
];

// Data Source UI publishing description
customConnectorDevelopment.Publish = [
    Beta = true,
    Category = "Other",
    ButtonText = {Extension.LoadString("ButtonTitle"), Extension.LoadString("ButtonHelp")},
    LearnMoreUrl = "https://powerbi.microsoft.com/",
    SourceImage = customConnectorDevelopment.Icons,
    SourceTypeImage = customConnectorDevelopment.Icons
];

customConnectorDevelopment.Icons = [
    Icon16 = {
        Extension.Contents("customConnectorDevelopment16.png"),
        Extension.Contents("customConnectorDevelopment20.png"),
        Extension.Contents("customConnectorDevelopment24.png"),
        Extension.Contents("customConnectorDevelopment32.png")
    },
    Icon32 = {
        Extension.Contents("customConnectorDevelopment32.png"),
        Extension.Contents("customConnectorDevelopment40.png"),
        Extension.Contents("customConnectorDevelopment48.png"),
        Extension.Contents("customConnectorDevelopment64.png")
    }
];
```

Section declaration

Shared functions (executable)

Implementing Authentication

Required connector metadata

Implementing Icons

Now it is just up to us!

So... let's create our own connector

Data Brothers

ŠTĚPÁN REŠL (Steve)

Lead technical consultant

Data Brothers

Microsoft Most Valuable Professional

Power BI kafíčko

Data Meerkat

LINKEDIN

X

BLUESKY

MAY THE DATA BE WITH YOU