

Connecting Services to PowerApps and Power Automate



Reiner Ganser

Cloud Productivity Consultant & Trainer

- More than 30 years in the IT space in the areas of software development | collaboration | migrations | intranet solutions | cloud services
- As a Cloud Productivity Consultant I use the potential of Microsoft Cloud technologies, applications and services to build new solutions that would not have been possible a few years ago.

reiner@ganser-it-consulting.ch

<https://ganser-it-consulting.ch>



Agenda



Extend Power Automate for
SharePoint access



Access to external Web
services from Power Apps and
Power Automate



Use Azure Functions to
support Power Apps and
Power Automate

Extend Power Automate for SharePoint access



Scenario

Team members should have read-only access to documents



Possible solutions

- Do it manually -> it's a bit tricky
- PowerShell -> requires appropriate rights
- Create flow and adjust permissions for members group -> should work :-)

Add action for setting permissions for a SharePoint group

No action available in Flow OOTB 😞

Alternatives

- Third party extensions:
e.g. <https://docs.microsoft.com/en-us/connectors/plumsailsp/>
- Use SharePoint REST API

Understanding the basics

SharePoint REST/OData APIs

Good resources

- <https://docs.microsoft.com/en-us/sharepoint/dev/sp-add-ins/get-to-know-the-sharepoint-rest-service>
- <https://docs.microsoft.com/en-us/sharepoint/dev/sp-add-ins/complete-basic-operations-using-sharepoint-rest-endpoints>

Necessary API calls

Setting permissions

- **Set permissions for the desired SharePoint groups**

REST API (POST): `_api/web//getByTitle(,Name der Liste,)/items(ID)/roleassignments/addroleassignment(principalid=<Group-ID>,roleDefId=<Role-ID>)`

Supporting SharePoint REST API calls

- **Determine the ID of a certain authorization level (e.g. for read permissions)**

REST API (GET): `_api/web/roledefinitions/GetByName('Name of permission level')/Id`

- **Determine the ID of a specific SharePoint group**

REST API (GET): `_api/web/sitegroups/GetByName('Name of the group')/Id`

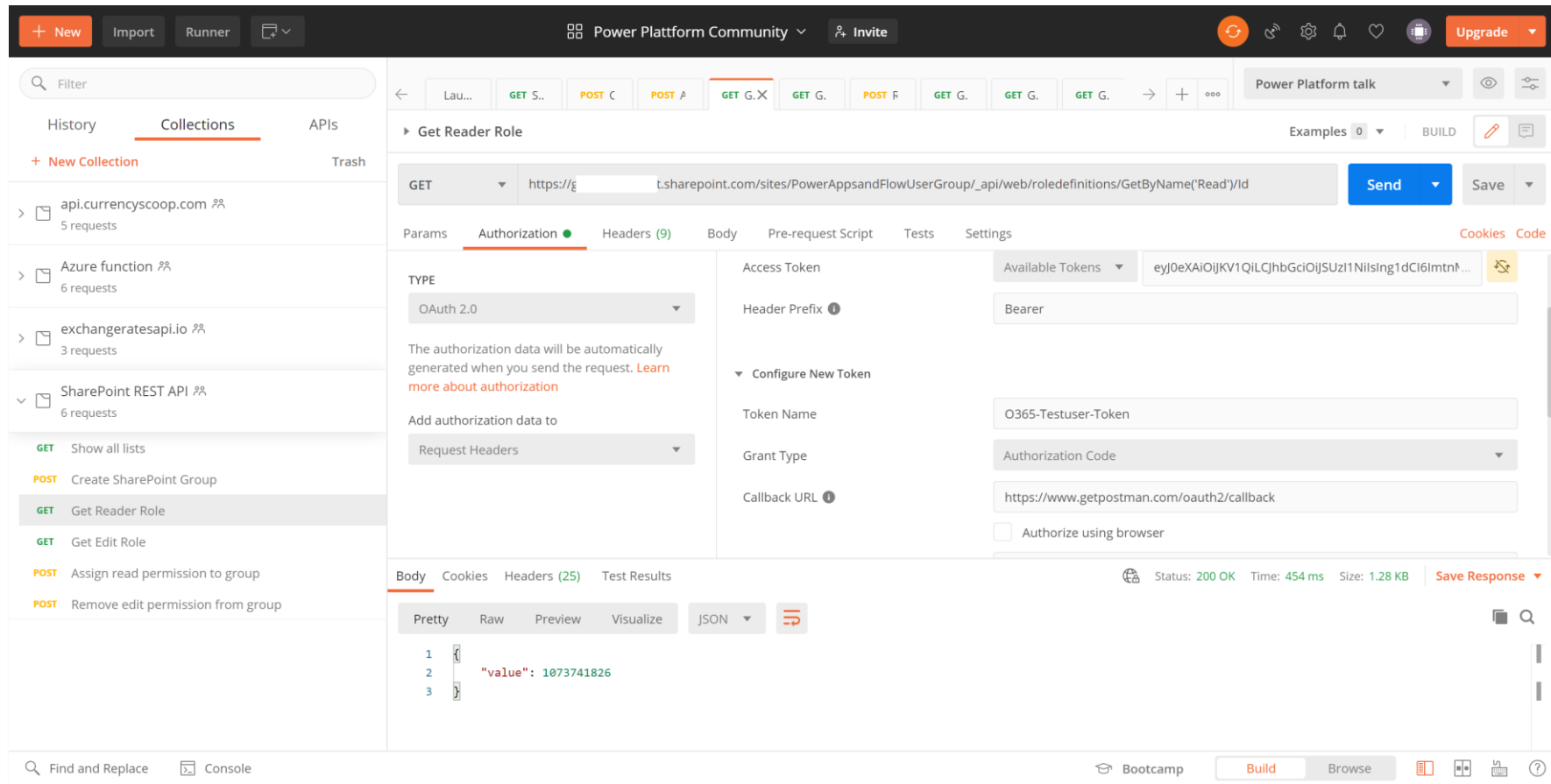
Testing the API calls

GET methods could be tested in the browser

```
<?xml version="1.0" encoding="utf-8" type="text/xml" ><entry xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices" xmlns:gml="http://www.opengis.net/gml"><id>e6a29d91-9b16-47d3-b128-3432e0ac6901</id><title /><updated>2018-12-12T21:06:13Z</updated><entry m:etag="&quot;1&quot;"><id>https://rganser1.sharepoint.com/_api/Web/Lists(guid'a1c0b543-be51-4500-a2cd-eff58677373f')</id><category term="SP.List" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/related/FirstUniqueAncestorSecurat
```

And what about POST?

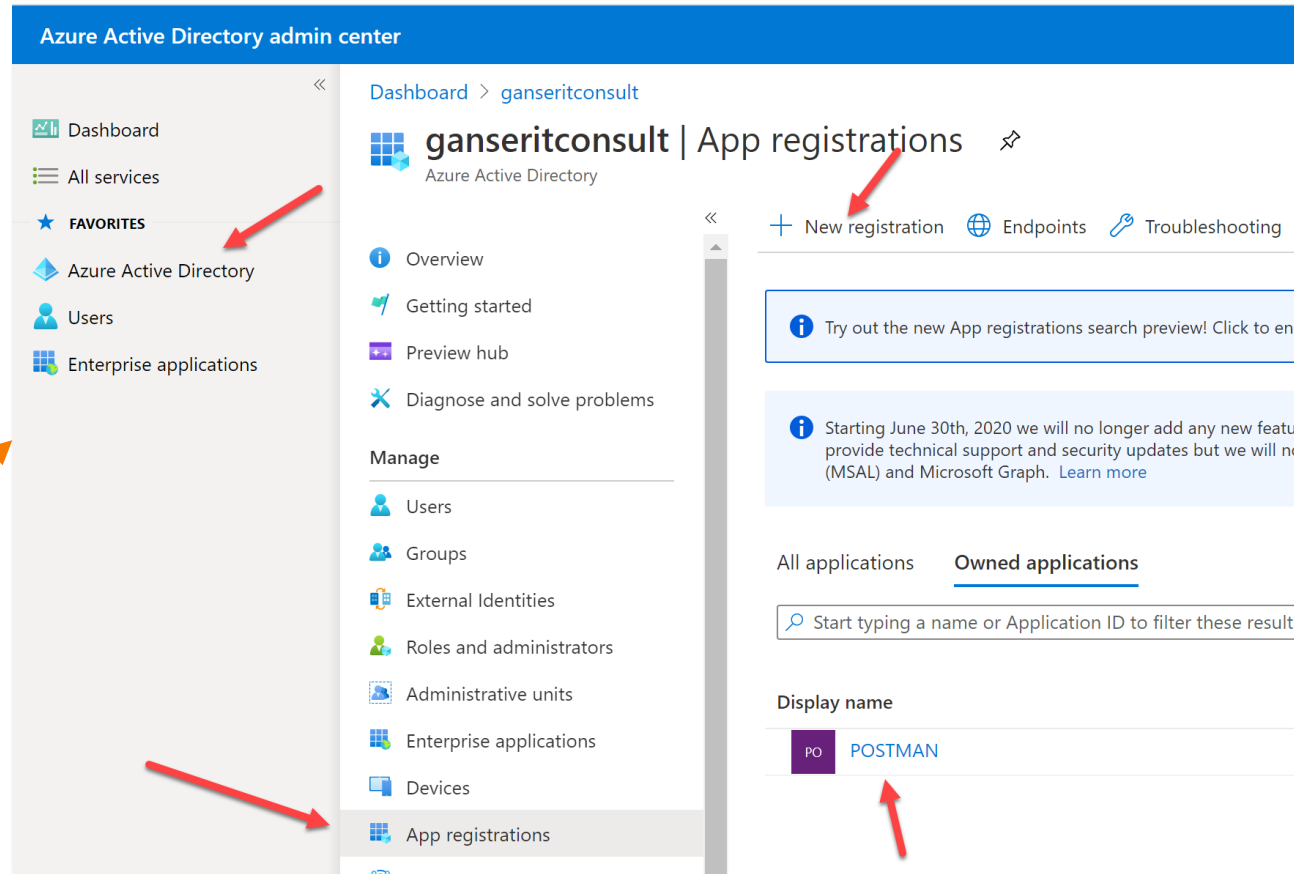
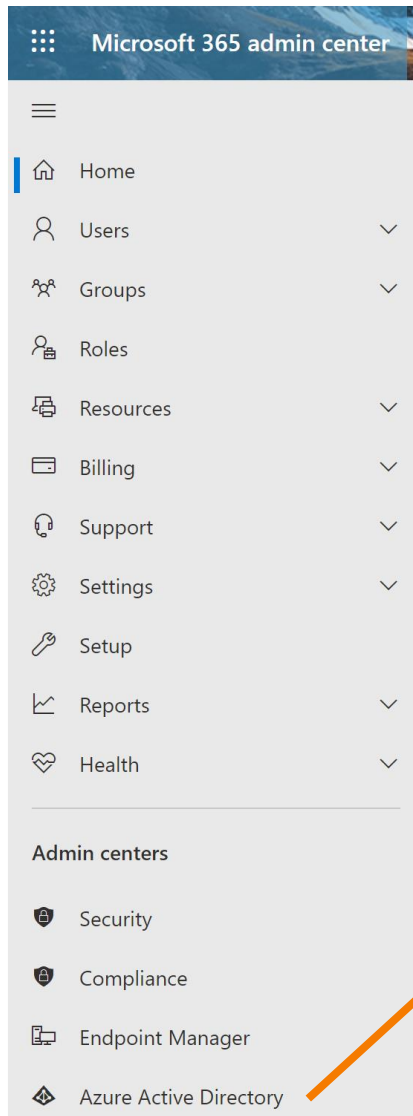
Use postman to evaluate the REST API interface of SharePoint



Postman
<https://www.getpostman.com/downloads/>

Register Postman in Azure Active Directory

To access SharePoint from Postman, Postman must first be registered in Azure Active Directory.



Define Authorization in Postman

The screenshot shows the Postman interface for defining an OAuth 2.0 authorization. The URL bar at the top shows a GET request to a SharePoint API endpoint. The 'Authorization' tab is selected, and the 'TYPE' dropdown is set to 'OAuth 2.0'. The 'Grant Type' is 'Authorization Code'. The 'Client ID' and 'Client Secret' fields are populated with Postman variables: `{{Azure Client ID}}` and `{{Azure Client Secret}}`. A callout box points to these variables with the text: "Data from Azure Active Directory (defined in Postman variables)".

GET https://[redacted].sharepoint.com/sites/PowerAppsandFlowUserGroup/_api/web/roledefinitions/GetByName('Read')/Id Send Save

Params **Authorization** Headers (9) Body Pre-request Script Tests Settings Cookies Cod

TYPE
OAuth 2.0

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add authorization data to
Request Headers

Grant Type: Authorization Code

Callback URL:

Authorize using browser

Auth URL:

Access Token URL:

Client ID: **Data from Azure Active Directory (defined in Postman variables)**

Client Secret: **Data from Azure Active Directory (defined in Postman variables)**

Scope:

State:

Client Authentication:

Demo

Call the SharePoint API with
Postman



Building the flow



Create new instant flow



Define triggers



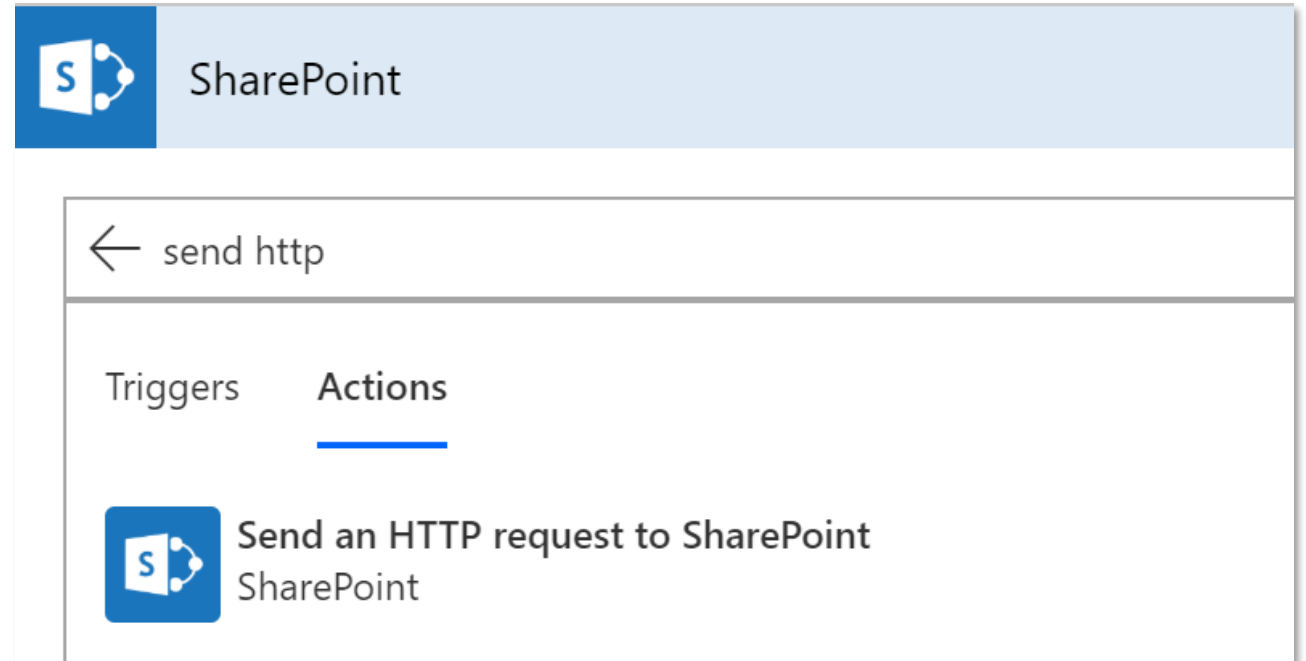
Define and set variables



Insert REST API calls

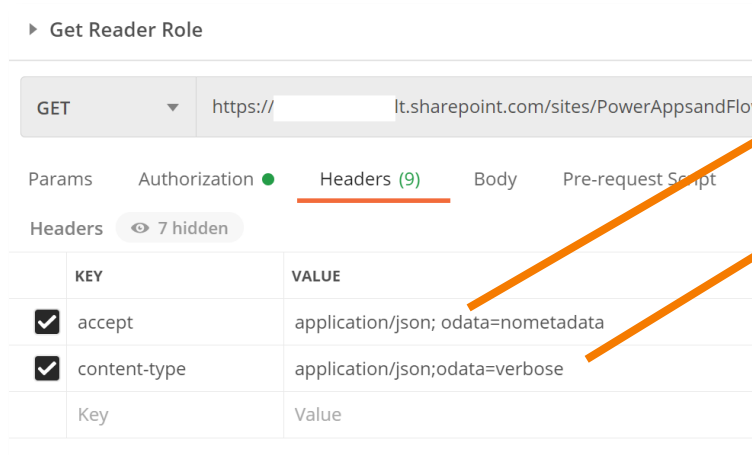
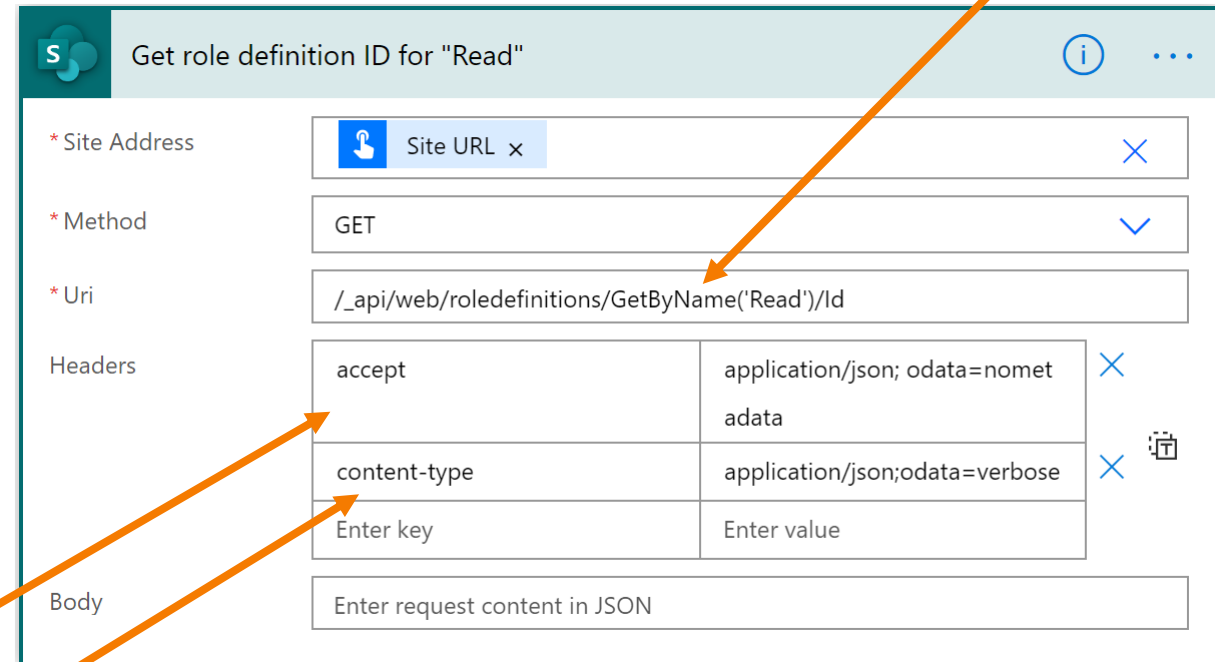
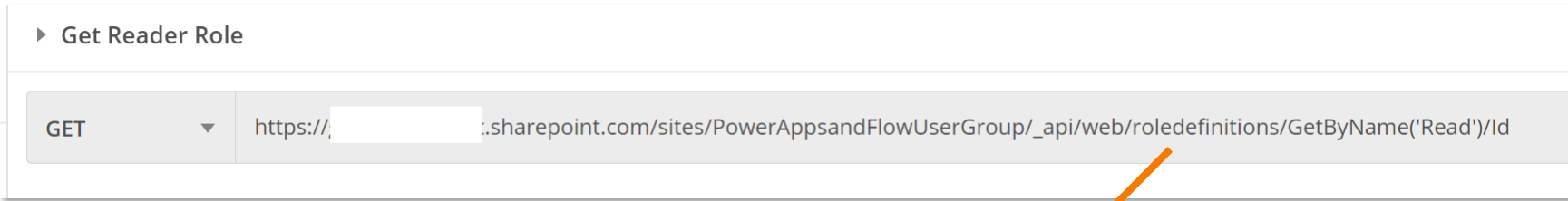
Example: Get
Role Definition
ID for "read
only

Send an HTTP
request to SharePoint



Set parameter

=> Take over from Postman



Further steps



Parse JSON result



Set Variable with Role ID



Remove edit permissions



Set read permissions

Overview of the entire flow



Access to external Web services from Power Apps and Power Automate



Scenario 2

- Daily updated currency data is needed in a Power App
- Access to historical data should also be possible

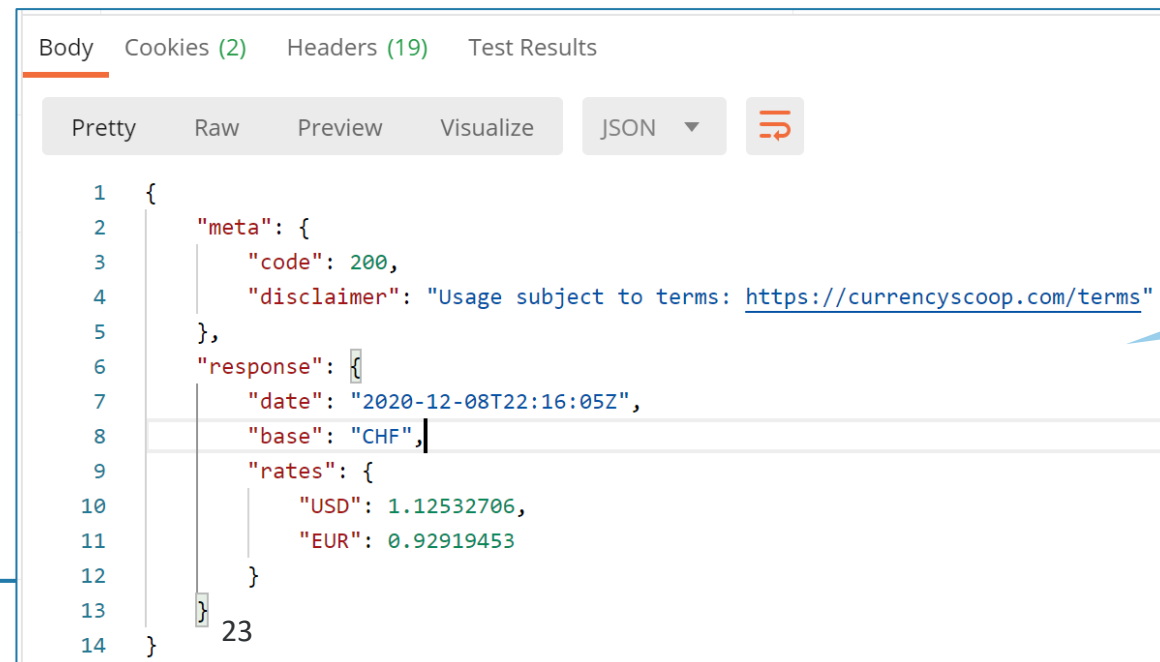
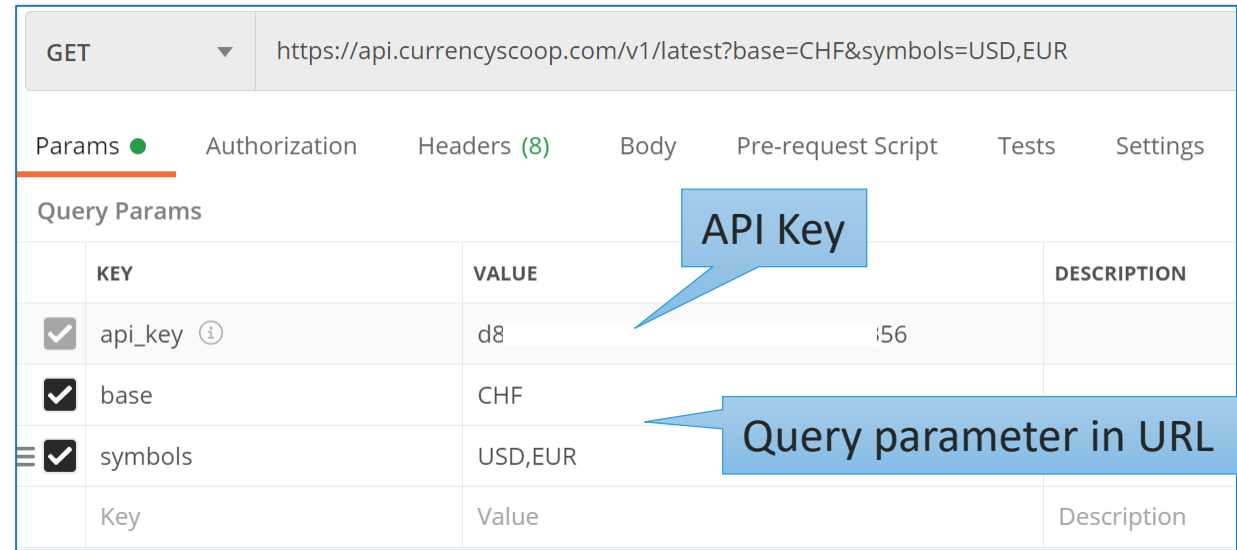
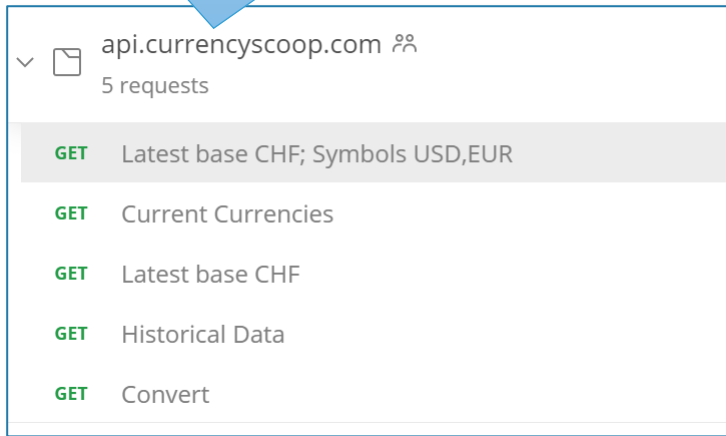


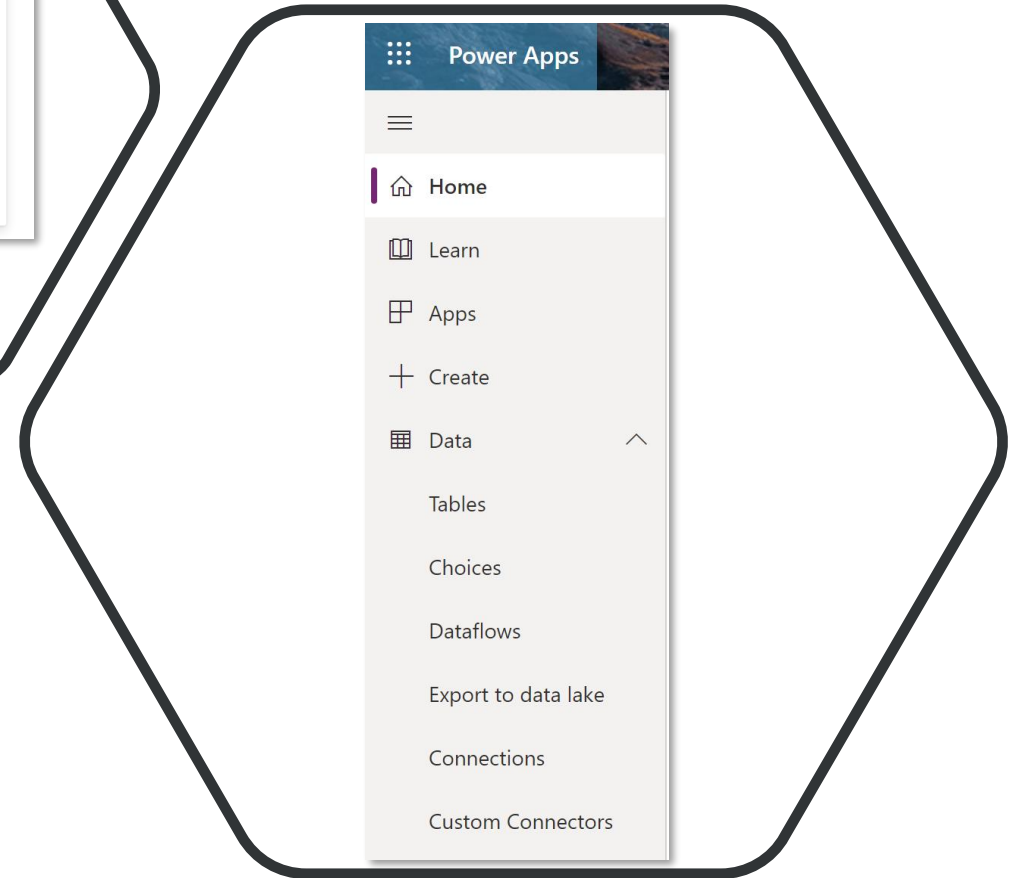
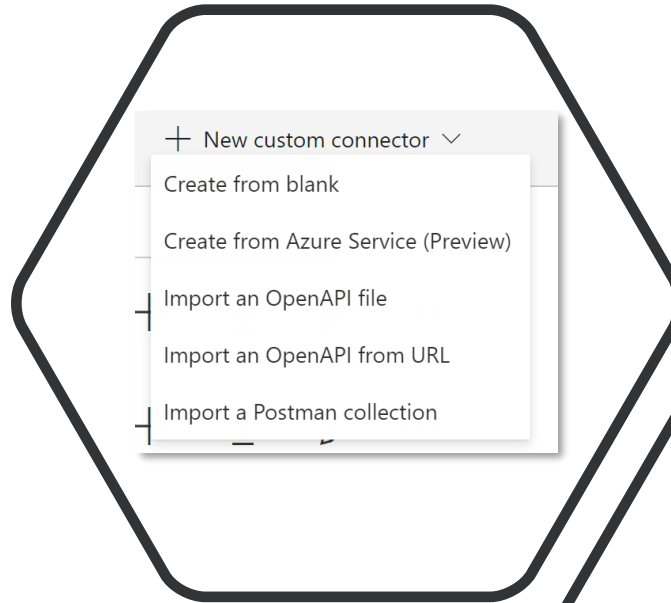
Evaluate external services

- Examples
 - <http://exchangeratesapi.io/>
 - Open source, no key necessary
 - <https://currencyscoop.com/>
 - Several pricings plans -> API key necessary

Test API with Postman

Postman collection -> can be used to define a custom connector





Access to the external service via custom connector

Define in Power Apps or Power Automate

Create connector from blank

General information and Security


Connector Name: Currencyscoop

1. General > 2. Security > 3. Definition > 4. Test

Swagger Editor: Update connector: Close:

General information

Add an icon and short description to your custom connector. Your host and base URL will be automatically generated from the swagger file.



Upload connector icon
Supported file formats are PNG and JPG. (< 1MB)

Icon background color: #007ee5

Description: Currencyscoop V7

Connect via on-premises data gateway [Learn more](#)

Scheme *
 HTTPS HTTP

Host *: api.currencyscoop.com

Base URL: /

Connector Name: Currencyscoop

1. General > 2. Security > 3. Definition > 4. Test

Swagger Editor: Update connector: Close:

Security

Choose the authentication type and fill in the required fields to set the security for your custom connector.
[Learn more](#)

Authentication type

Choose what authentication is implemented by your API *

API Key

API Key

Users will be required to provide the API Key when creating a connection

Parameter label *
API Key

Parameter name *
api_key

Parameter location *
Query

Authentication via API Key

Create connector from blank

Define action and Request

1. General > 2. Security > **3. Definition** > 4. Test Swagger Editor Update connector Close

Actions (4)

Actions determine the operations that users can perform. Actions can be used to read, create, update or delete resources in the underlying connector.

- 1 Historical ...
- 2 Current... ...
- 3 Latest** ...
- 4 Convert ...

General

Summary [Learn more](#)

Description [Learn more](#)

Operation ID * [Learn more](#)

This is the unique string used to identify the operation.

Visibility [Learn more](#)

none advanced internal important

References (0)

References are reusable

Request

It defines the pre-requirements needed in order to make a request. Describes a single operation parameter. A unique parameter is defined by a combination of a name and location.

Request

Verb *

The verb describes the operations available on a single path.

GET

URL *

This is the request URL.

Path

Path is used together with Path Templating, where the parameter value is actually part of the operation's URL.

Query

Query parameters are appended to the URL. For example, in /items?id=####, the query parameter is id.

base ... symbols ...

Headers

These are custom headers that are part of the request.

Create connector from blank

Import request data from Postman

Request

It defines the pre-requirements needed in order to make a request. Describes a single operation parameter. A unique parameter is defined by a combination of a name and location.

Request + Import from sample

Verb *
The verb describes the operations available on a single path.

GET

URL *
This is the request URL.

`https://api.currencyscoop.com/v1/latest`

Path
Path is used together with Path Templating, where the parameter value is actually part of the operation's URL.

Query
Query parameters are appended to the URL. For example, in /items?id=####, the query parameter is id.

base ... symbols ...

Headers
These are custom headers that are part of the request.

Import from sample ×

Define verb

Verb *

GET DELETE POST PUT HEAD OPTIONS

PATCH

URL *

Paste request url

`https://api.currencyscoop.com/v1/latest?base=CHF&symbols=USD,EUR`

This is the request URL.

Headers

Headers separated by a new line, e.g.:
Content-Type application/json
Accept application/json

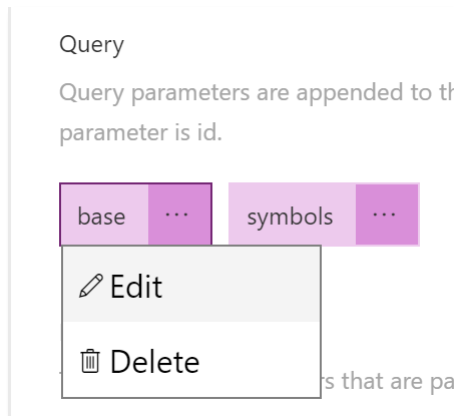
Import data

These are custom headers that are part of the request.

Import **Close**

Create connector from blank

Configure query parameter



Parameter

Name *
base

Description [Learn more](#)

Summary [Learn more](#)

Default value

Is required?
 Yes No

Visibility [Learn more](#)
 none advanced internal important

Location *
 Path Query Header Body

Type Format

Set default value if optional

Define whether required (different access notation in Power Apps)

Define data type

Create connector from blank

Define response structure

Response

← Back

+ Import from sample

Name *

default

Response

It defines the shape of response returned by the connector.

default default

+ Add default response

Import from sample

Headers

Headers separated by a new line, e.g.:
Content-Type application/json
Accept application/json

Body

```
https://currencyscoop.com/terms"
},
"response": {
  "date": "2020-12-08T22:16:05Z",
  "base": "CHF",
  "rates": {
    "USD": 1.12532706,
    "EUR": 0.92919453
  }
}
}
```

Paste data from Postman

Response

+ Import from sample

Name *

default

Headers

These are custom headers that are part of the response.

References Used

The following are the references used by this entity

Body

The payload that is available on the response. These are the tokens that will show up as the outputs in designer.

code ... disclaimer ... base ... date ... AED ... AFN ... ALL ...

AMD ... ANG ... AOA ... ARS ... AUD ... AWG ... AZN ...

BAM ... BBD ... BDT ... BGN ... BHD ... BIF ... BMD ...

BND ... BOB ... BRL ... BSD ... BTN ... BWP ... BYN ...

BZD ... CAD ... CDF ... CHF ... CLP ... CNY ... COP ...

CRC ... CUC ... CLP ... CVE ... CZK ... DJF ... DKK ... DOP ...

Response is now defined

Create connector from blank

Test custom connector

1. General > 2. Security > 3. Definition > 4. Test

Swagger Editor ✓ Update connector ✕

Test operation

Test a specified operation of this custom connector using the selected connection. You must update the custom connector in order to test recent changes.

Connections

Selected connection *

Currencyscoop (Created at 2020-12-08T21:07:39.0230574Z)

+ New connection

Create a connection first

Operations (4)

These are the operations defined by your custom connector. This includes actions and triggers.

- 1 Historical
- 2 CurrentCurrencies
- 3 Latest
- 4 Convert

Latest

base

CHF

symbols

USD,EUR

Test operation

Enter parameters

Request

Response

Status

OK (200)

Headers

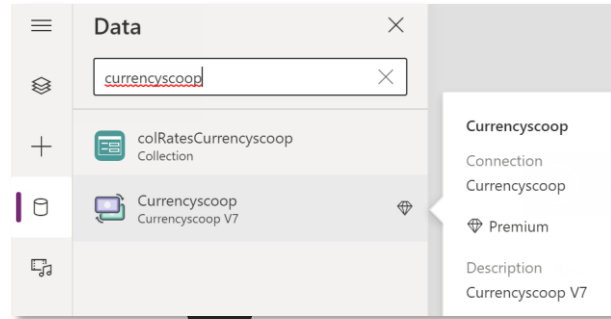
```
{
  "cache-control": "no-cache",
  "cf-cache-status": "DYNAMIC",
  "cf-ray": "5fef997dbd9f9c39-AMS",
  "cf-request-id": "06e99e429300009c39b3883000000001",
}
```

Body

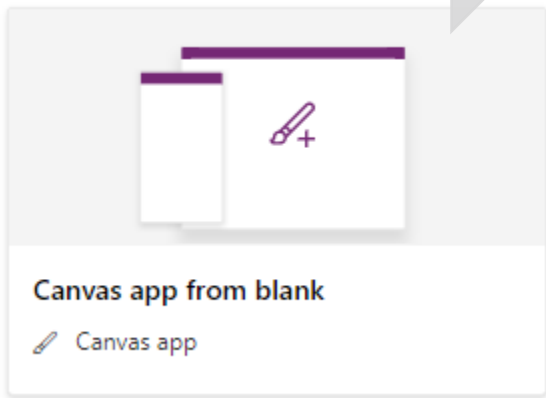
```
{
  "meta": {
    "code": 200,
    "disclaimer": "Usage subject to terms: https://currencyscoop.com/terms"
  },
  "response": {
    "date": "2020-12-09T15:01:07Z",
    "base": "CHF",
    "rates": {
      "USD": 1.12598935,
      "EUR": 0.92834884
    }
  }
}
```

Verify response

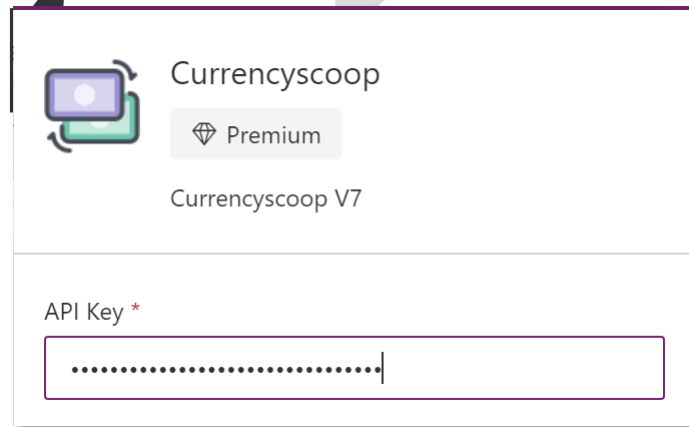
PowerApp erstellen



Create Canvas App



Add Data Source
API Key angeben



Add controls

Define actions

Add logic for accessing services

- Declare and set variables
- Define necessary actions

```
ClearCollect(  
    colRatesCurrencyscoop,  
    Currencyscoop.Latest(  
        {  
            base: txtBase2.Text,  
            symbols: txtSymbols2.Text  
        }  
    ).response.rates  
);
```

Connector name

Function to call

Parameters
(required)

Results


Access to result

Text = fx First(colRatesCurrencyscoop).USD

Tree view


Format text Remove formatting

Sample app

 CURRENCYSCOOP

Base currency:

Symbols:

Date: 

Results

USD EUR

CHF EUR

Troubleshooting

Display returned JSON

- Use JSON Function to get result in JSON format

Transform
collection
to JSON

```
Set(  
  colRatesCurrencyscoopJSON,  
  JSON(  
    colRatesCurrencyscoop,  
    IndentFour  
  )  
);
```

Default = fx colRatesCurrencyscoopJSON

Textbox

```
[  
  {  
    "EUR": 0.92834884,  
    "USD": 1.12598935  
  }  
]
```

Troubleshooting

Use Monitor to evaluate results

The screenshot displays the Power Apps Monitor interface. The top navigation bar includes 'Power Apps | Monitor', 'Environment ganseritconsult (default)', and icons for 'Invite', 'Options', and 'Filter'. Below the navigation, there are buttons for 'Clear data', 'Upload', and 'Download'. The main area is divided into two panes. The left pane, titled 'Studio session', contains a table of operations:

I.	Time	Category	Operation	Result	R...	Status	Duration
1	16:19:46.084	UserAction	Select	Success			
2	16:19:46.863	Network	Latest	Success			200
3	16:19:46.865	Function	ClearCollect	Success	1 row...		
4	16:19:46.869	Function	Text	Success	The n...		

The right pane, titled 'Latest', shows a detailed view of the selected network operation. It includes tabs for 'Details', 'Formula', 'Request', and 'Response'. The 'Response' tab is active, displaying a JSON response structure. A blue callout box points to the 'rates' object in the response, with the text 'Evaluate response'.

```
txtBase2.Text,\n symbols: txtSymbols2.Text\n}\n ).response.rates\n);\nSet(\n colRatesCurrencycoopJSON,\n JSON(\n colRatesCurrencycoop,\n JSONFormat.IndentFour\n ));\n";\n\n27\n28\n29\n30\n31\n32\n33\n34\n35\n36\n37\n38\n39\n40\n41\n42\n43\n44\n45\n46\n47\n48\n49\n50\n51\n52\n53\n54\n55\n56\n57\n58\n59\n60\n61\n62\n63\n64\n65\n66\n67\n68\n69\n70\n71\n72\n73\n74\n75\n76\n77\n78\n79\n80\n81\n82\n83\n84\n85\n86\n87\n88\n89\n90\n91\n92\n93\n94\n95\n96\n97\n98\n99\n100\n101\n102\n103\n104\n105\n\n    "span": {\n      "start": 45,\n      "end": 163\n    }\n  },\n},\n\n"request": {\n},\n\n"response": {\n  "duration": 755.04,\n  "size": 195,\n  "status": 200,\n  "headers": {\n},\n},\n\n"body": {\n  "meta": {\n    "code": 200,\n    "disclaimer": "Usage subject to terms: https://currencycoop.com/terms\n  }\n},\n\n"response": {\n  "date": "2020-12-09T15:19:46Z",\n  "base": "CHF",\n  "rates": {\n    "EUR": 0.92834884,\n    "USD": 1.12598935\n  }\n},\n\n"responseType": "json"\n},\n\n"startTime": 21593928.03,\n"name": "https://europe-002.azure-apim.net/invoke",\n"fetchStart": 21593928.03,\n"responseEnd": 21594673.51\n}\n}
```

Use Azure Functions to support Power Apps and Power Automate

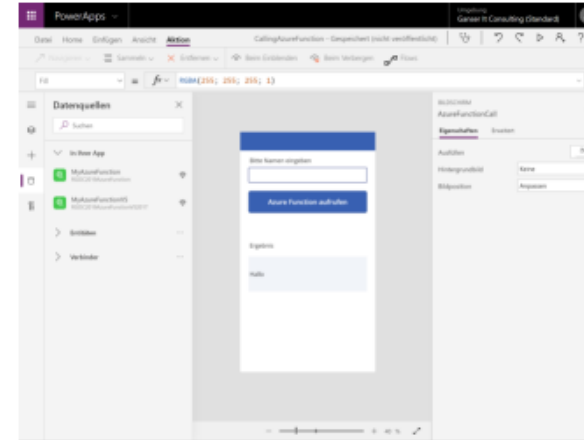


Scenario 3

JSON format returned by a service is not optimal and must be converted

Overview

- Azure Function App contains Code
 - C#, Java, JavaScript, PowerShell, Python
- Creation in Visual Studio or Visual Studio Code
- Create Swagger API Definition
- Creating a Custom Connector in Power Apps or Power Automate
- Call Connector in Power Apps or Power Automate



PowerApps cross-platform app
to create and view inventory list



PowerApps Custom connector
for PowerApps to understand the API Schema



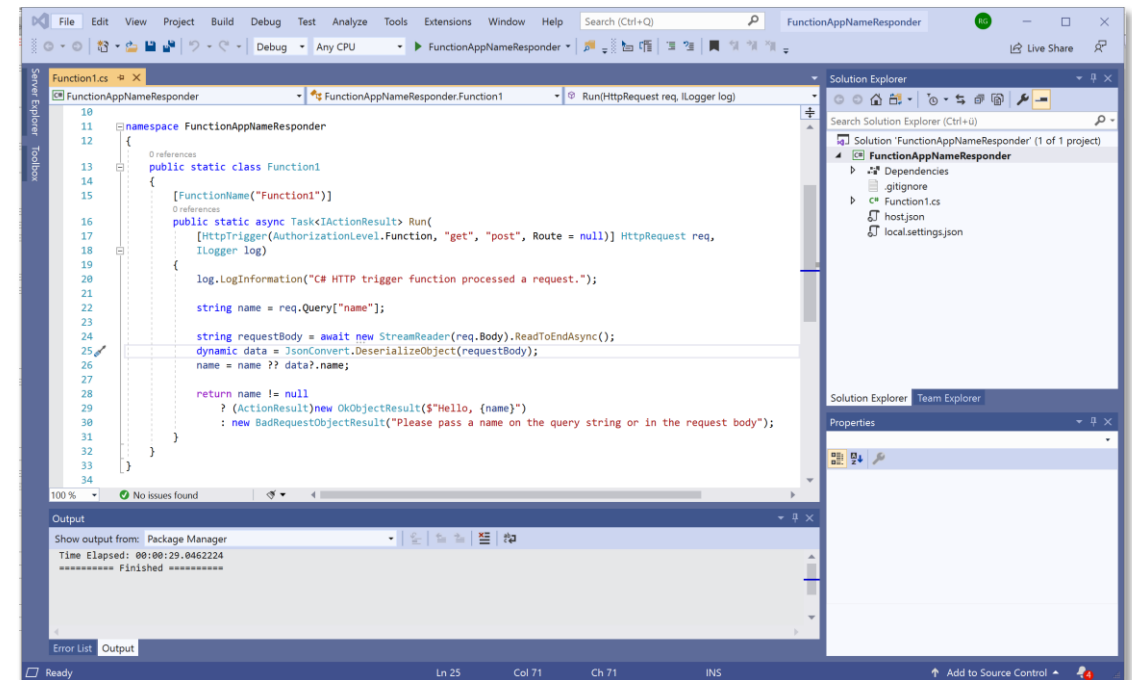
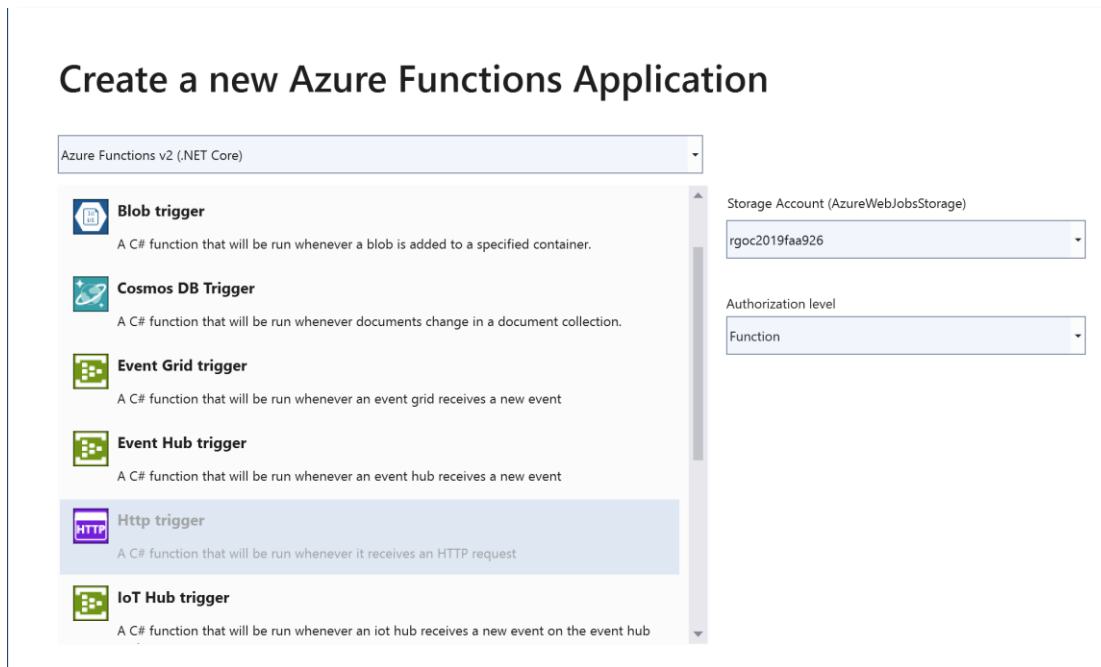
Azure Function app
Serverless code to host REST APIs



Azure Table Storage
To persist the data in the App

Visual Studio zur Erstellung

- Define trigger
- Set Authorization Level
- Testing and debugging in local runtime environment
- Publish to Azure



See also: <https://docs.microsoft.com/de-de/azure/azure-functions/functions-create-your-first-function-visual-studio>

```
"swagger": "2.0",
"info": {
  "version": "1.0.0",
  "title": "MyAzureFunction"
},
"host": "rgoc2019fa.azurewebsites.net",
"basePath": "/",
"schemes": [
  "https"
],
"consumes": [],
"produces": [],
"paths": {
  "/api/NameRespond": {
    "get": {
      "description": "Calls my azure function over",
      "operationId": "NameRespond",
      "parameters": [
        {
          "name": "code",
          "in": "query",
          "description": "code",
          "default": "NaYC4dcgEG7wfcTqmwj7WYeYINbW",
          "type": "string"
        },
        {
          "name": "name",
          "in": "query",
          "required": true,
          "type": "string"
        }
      ],
      "responses": {
```

Define the interface


- Create in Azure Portal
- Manual in custom connector definition

Create Custom Connector

1. General > 2. Security > 3. Definition > 4. Test Swagger Editor Update connector ✕

General information

Add an icon and short description to your custom connector. Your host and base URL will be automatically generated from the swagger file.



Upload connector icon
Supported file formats are PNG and JPG. (< 1MB)

↑ Upload

Icon background color

Description

Connect via on-premises data gateway [Learn more](#)

Scheme *
 HTTPS HTTP

Host *

Base URL

Use it in Power Apps

CURRENCYSLOOP

Base currency: CHF

Symbols: EUR,USD,CLP

Date: December 7, 2020

CLP	836.56384
EUR	0.92692554
USD	1.1225594

```
ClearCollect(
  colRatesCurrencycoop,
  Currencyscoop.Latest(
    {
      base: cmbCurrencies.Selected.Name,
      symbols: txtSymbols2_1.Text
    }
  ).response.rates
);
ClearCollect(
  colRatesTable,
  RGAzureFunction1.GetRatesTable({body: colRatesCurrencycoop}).rates
);
```

Name of the custom connector

Function to call

Results

Input for the function

Questions?