LEAD WITH CURIOSITY

REST Connector in the world of IoT

Christof Schwarz
Principal Solution Architect
15-May-2019

Qonnections
QLIK GLOBAL CONFERENCE

Agenda

Don´t rest `till you REST

- Introduction to REST
- Understand an API
  - test with 3rd party tool Postman

LIVE!

- Working with Qlik REST Connector
- Some Qlik Script tricks
  - Request and use bearer token
  - ISO-Date handling
  - Transposing Data
  - Paging Techniques

What`s out there in the jungle?

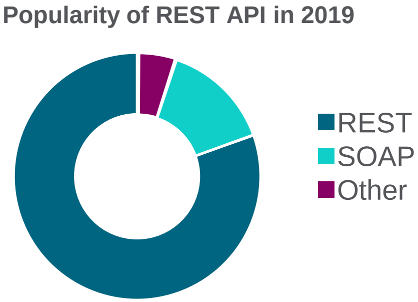
Examples of REST

Qonnections
QLIK GLOBAL CONFERENCE

Before we start ...

REST vs SOAP

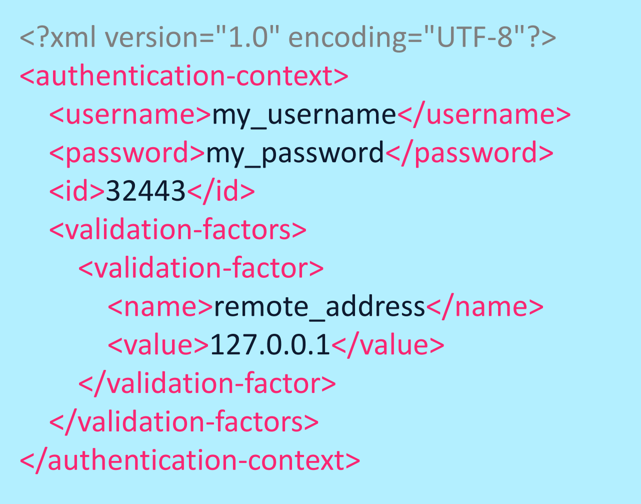
- REST
  - Representational State Transfer
  - started to spread in 2005
  - is a design style
  - Uses an URI to access information
  - Many different data formats
- SOAP
  - Simple Object Access Protocol
  - Is a protocol (“envelope”)
  - A service interface (calling functions)
  - More overhead
  - XML data format



Before we start ...

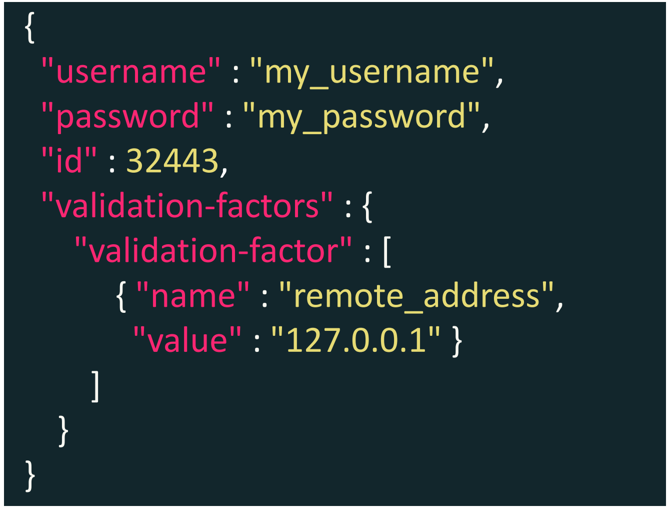
JSON vs XML

XML
Extensible Markup Language



```
<?xml version="1.0" encoding="UTF-8"?>
<authentication-context>
   <username>my_username</username>
   <password>my_password</password>
   <id>32443</id>
   <validation-factors>
      <validation-factor>
         <name>remote_address</name>
         <value>127.0.0.1</value>
      </validation-factor>
   </validation-factors>
</authentication-context>
```

JSON
JavaScript Object Notation



```
{
 "username" : "my_username",
 "password" : "my_password",
 "id" : 32443,
 "validation-factors" : {
    "validation-factor" : [
        { "name" : "remote_address",
          "value" : "127.0.0.1" }
      ]
   }
}
```

Strict notation!

Before we start ...

JSON vs XML

Language

```
oding="UTF-8"?>
>
ame</username>
ord</password>



ddress</name>
</value>


:>
```

JSON

JavaScript Object Notation

```
{
 "username" : "my_username",
 "password" : "my_password",
 "id" : 32443,
 "validation-factors" : {
   "validation-factor" : [
       { "name" : "remote_address",
        "value" : "127.0.0.1" }
     ]
   }
}
```

Strict notation!

```
{
 username : "my_username",
 password : "my_password",
 id : 32443,
 "validation-factors" : {
   "validation-factor" : [
     { name : 'remote_address',
       value : '127.0.0.1' }
     ]
   }
}
```

Relaxed notation ...

Understand and test the API

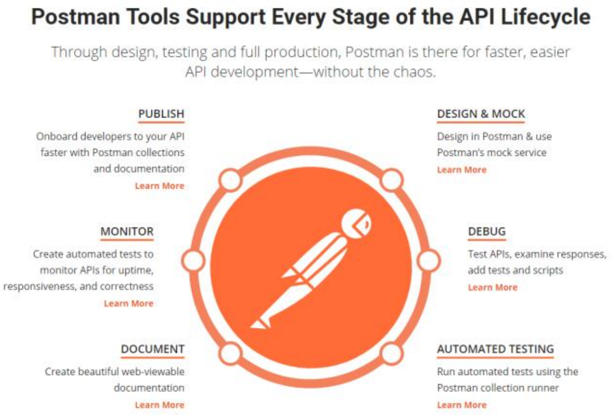
Using the tool Postman

Qonnections
QLIK GLOBAL CONFERENCE

Test with a tool

For example: Postman

www.getpostman.com



Postman Tools Support Every Stage of the API Lifecycle

Through design, testing and full production, Postman is there for faster, easier API development—without the chaos.

PUBLISH
Onboard developers to your API faster with Postman collections and documentation
Learn More

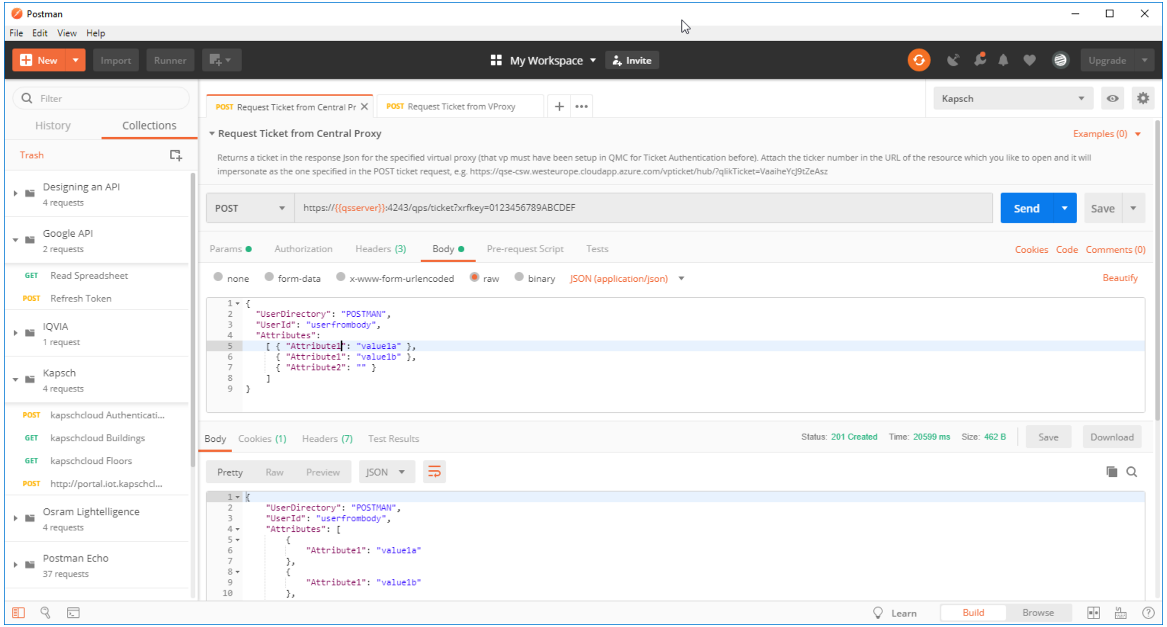
DESIGN & MOCK
Design in Postman & use Postman's mock service
Learn More

MONITOR
Create automated tests to monitor APIs for uptime, responsiveness, and correctness
Learn More

DEBUG
Test APIs, examine responses, add tests and scripts
Learn More

DOCUMENT
Create beautiful web-viewable documentation
Learn More

AUTOMATED TESTING
Run automated tests using the Postman collection runner
Learn More



Request Ticket from Central Proxy

Returns a ticket in the response Json for the specified virtual proxy (that vp must have been setup in QMC for Ticket Authentication before). Attach the ticker number in the URL of the resource which you like to open and it will impersonate as the one specified in the POST ticket request, e.g. https://qse-csw.westeurope.cloudapp.azure.com/vpticket/hub/?qlikTicket=VaaiheYcJ9tZeAsz

POST https://{{qsserver}}:4243/qps/ticket?xrfkey=0123456789ABCDEF

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary JSON (application/json)

```
{
  "UserDirectory": "POSTMAN",
  "UserId": "userfrombody",
  "Attributes":
    [ { "Attribute1": "value1a" },
      { "Attribute1": "value1b" },
      { "Attribute2": "" }
    ]
}
```

Status: 201 Created Time: 20599 ms Size: 462 B

```
{
    "UserDirectory": "POSTMAN",
    "UserId": "userfrombody",
    "Attributes": [
        {
            "Attribute1": "value1a"
        },
        {
            "Attribute1": "value1b"
        },
```

Play with API

Echo Server

Mock Server

Copy Code

Ask for the documentation, test with a tool

Documention

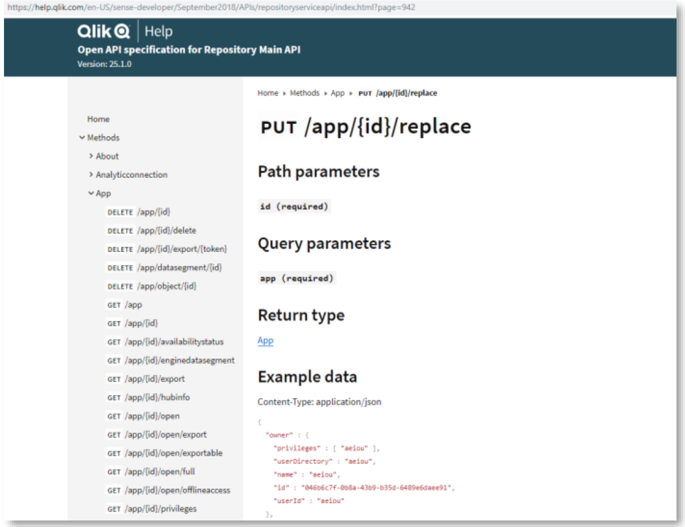
- Describe what methods (endpoints) are doing and which parameters they need
- Often created with Swagger

Sending Request

- Method (GET, POST, PUT ...)
- Path parameters
- QueryString parameters
- http-headers
- Body

Receiving Answer

- Response code
- Body



https://help.qlik.com/en-US/sense-developer/September2018/APIs/repositoryserviceapi/index.html?page=942

Qlik Help

Open API specification for Repository Main API

Version: 25.1.0

- Home
- Methods
  - About
  - Analyticconnection
  - App
    - DELETE /app/{id}
    - DELETE /app/{id}/delete
    - DELETE /app/{id}/export/{token}
    - DELETE /app/datasegment/{id}
    - DELETE /app/object/{id}
    - GET /app
    - GET /app/{id}
    - GET /app/{id}/availabilitystatus
    - GET /app/{id}/enginedatasegment
    - GET /app/{id}/export
    - GET /app/{id}/hubinfo
    - GET /app/{id}/open
    - GET /app/{id}/open/export
    - GET /app/{id}/open/exportable
    - GET /app/{id}/open/full
    - GET /app/{id}/open/offlineaccess
    - GET /app/{id}/privileges

Home ▸ Methods ▸ App ▸ PUT /app/{id}/replace

PUT /app/{id}/replace

Path parameters

`id (required)`

Query parameters

`app (required)`

Return type

App

Example data

Content-Type: application/json

```
{
  "owner" : {
    "privileges" : [ "aeiou" ],
    "userDirectory" : "aeiou",
    "name" : "aeiou",
    "id" : "046b6c7f-0b8a-43b9-b35d-6489e6daee91",
    "userId" : "aeiou"
  },
```

Example https://help.qlik.com/en-US/sense-developer/April2019/APIs/repositoryserviceapi

Working with REST APIs

Qonnections
QLIK GLOBAL CONFERENCE

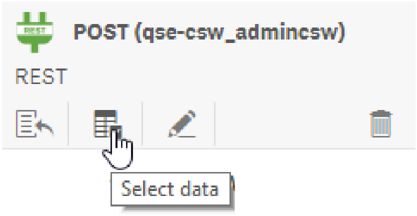
Embrace Qlik Scripting

It takes a program logic to interact with REST APIs

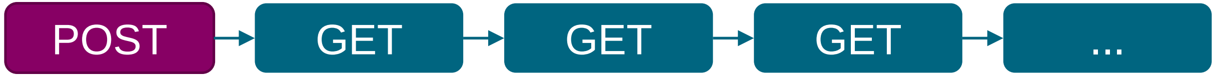
- In most cases, it is not a static, single call of a REST URI
  - The „Select Data“ wizard alone won‘t do the job
  - A series of calls are needed, which build on each other
  - Embrace the capabilities of Qlik Scripting

- For example
  - First of all, get a token for the next calls
  - Make multiple calls since one reply would be too big (paging)

Select Data wizard



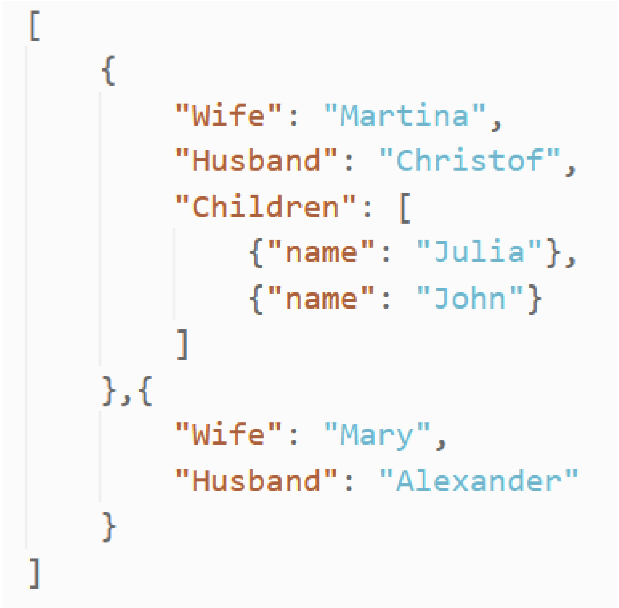
+ Scripting



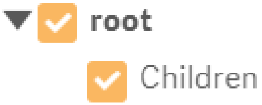
Qlik REST Connector under the hood

De-mystify the generated script

Json Response

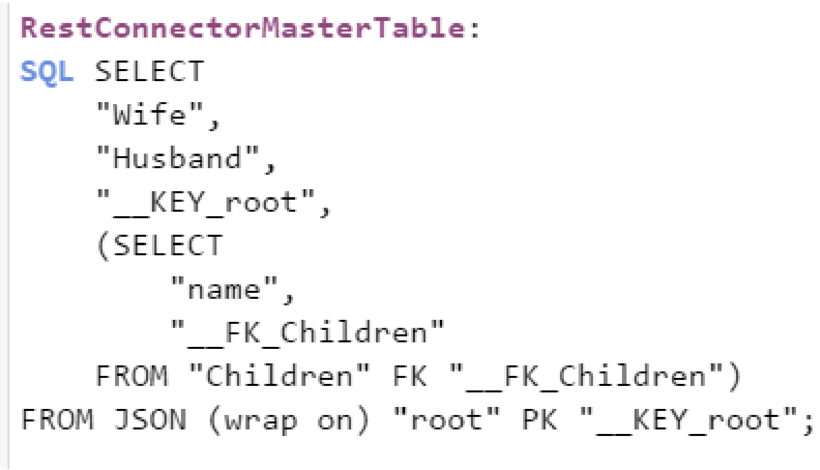


```
[
    {
        "Wife": "Martina",
        "Husband": "Christof",
        "Children": [
            {"name": "Julia"},
            {"name": "John"}
        ]
    },{
        "Wife": "Mary",
        "Husband": "Alexander"
    }
]
```



- ☑ root
  - ☑ Children

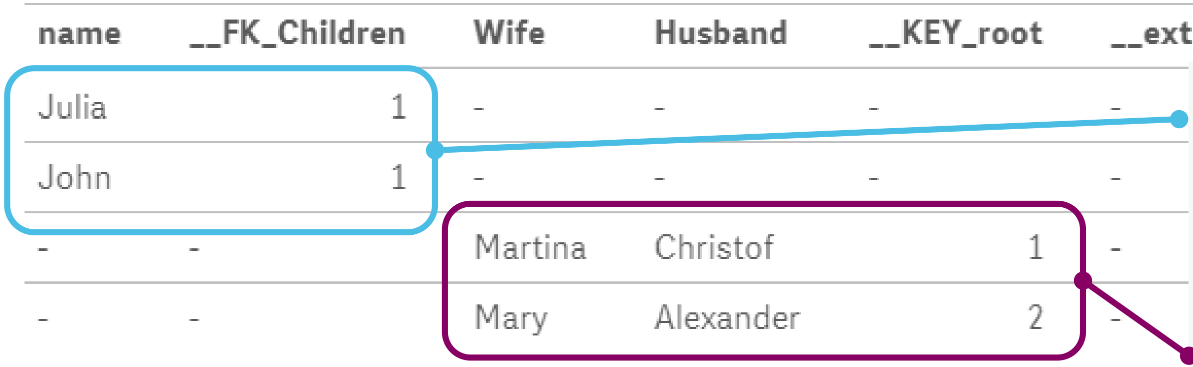
REST Connector Wizard



```
RestConnectorMasterTable:
SQL SELECT
    "Wife",
    "Husband",
    "__KEY_root",
    (SELECT
        "name",
        "__FK_Children"
    FROM "Children" FK "__FK_Children")
FROM JSON (wrap on) "root" PK "__KEY_root";
```

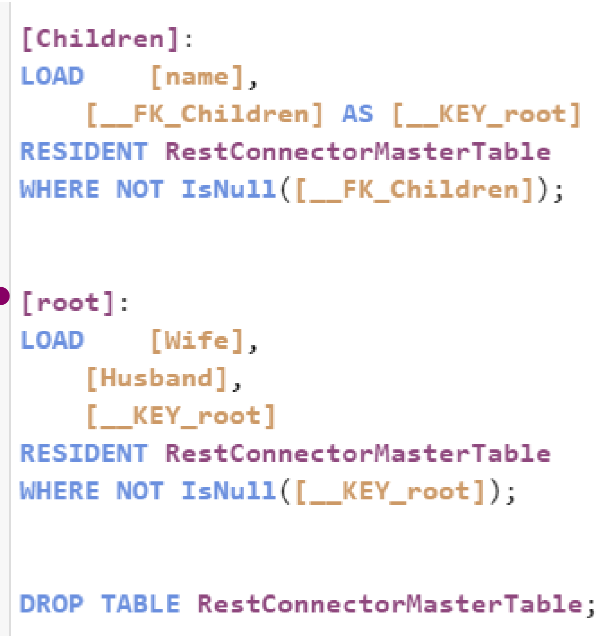
Field names are tolerant. The LOAD doesn‘t break if you attempt to load a non-existing key.

Qlik REST Connector under the hood

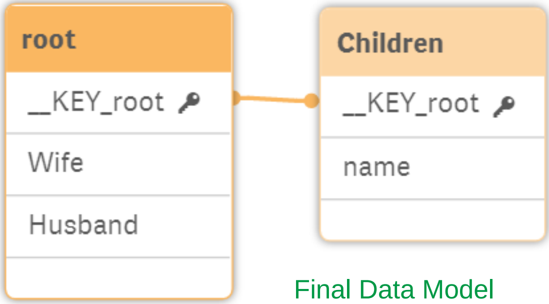


name	__FK_Children	Wife	Husband	__KEY_root	__extra_
Julia	1	-	-	-	-
John	1	-	-	-	-
-	-	Martina	Christof	1	-
-	-	Mary	Alexander	2	-

RestConnectorMasterTable (temporary)

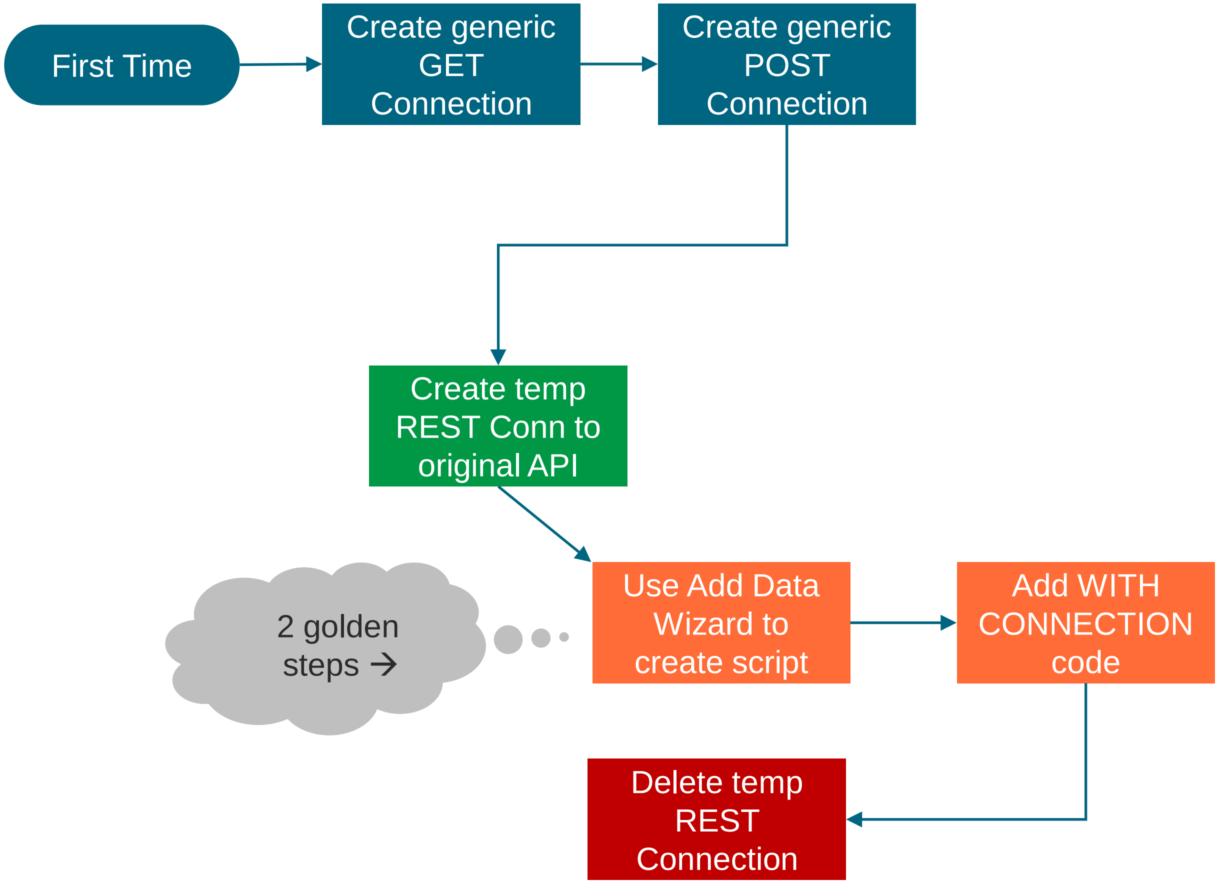


```
[Children]:
LOAD    [name],
    [__FK_Children] AS [__KEY_root]
RESIDENT RestConnectorMasterTable
WHERE NOT IsNull([__FK_Children]);


[root]:
LOAD    [Wife],
    [Husband],
    [__KEY_root]
RESIDENT RestConnectorMasterTable
WHERE NOT IsNull([__KEY_root]);


DROP TABLE RestConnectorMasterTable;
```



Final Data Model

Workflow for working with REST Connector


First Time
Create generic GET Connection
Create generic POST Connection
Create temp REST Conn to original API
Use Add Data Wizard to create script
Add WITH CONNECTION code
Delete temp REST Connection
2 golden steps →

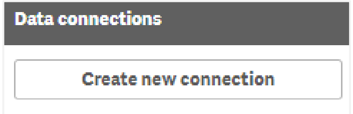
Set up 2 placeholder REST-connections

Create New Connections in Qlik Sense

- Create one placeholder **POST** request
  (e.g. https://postman-echo.com/post)
- Create one placeholder **GET** request
  (e.g. https://postman-echo.com/get)
- Leave all params emtpy, you will later parameterize the call with script
  - Dynamically provide: URL, Query-strings, Http-Header settings, Body
- Why two requests?
  - Because the only thing you cannot parameterize in the call itself is the http-method.
  - The http-method will come from this script line just before the SELECT ...

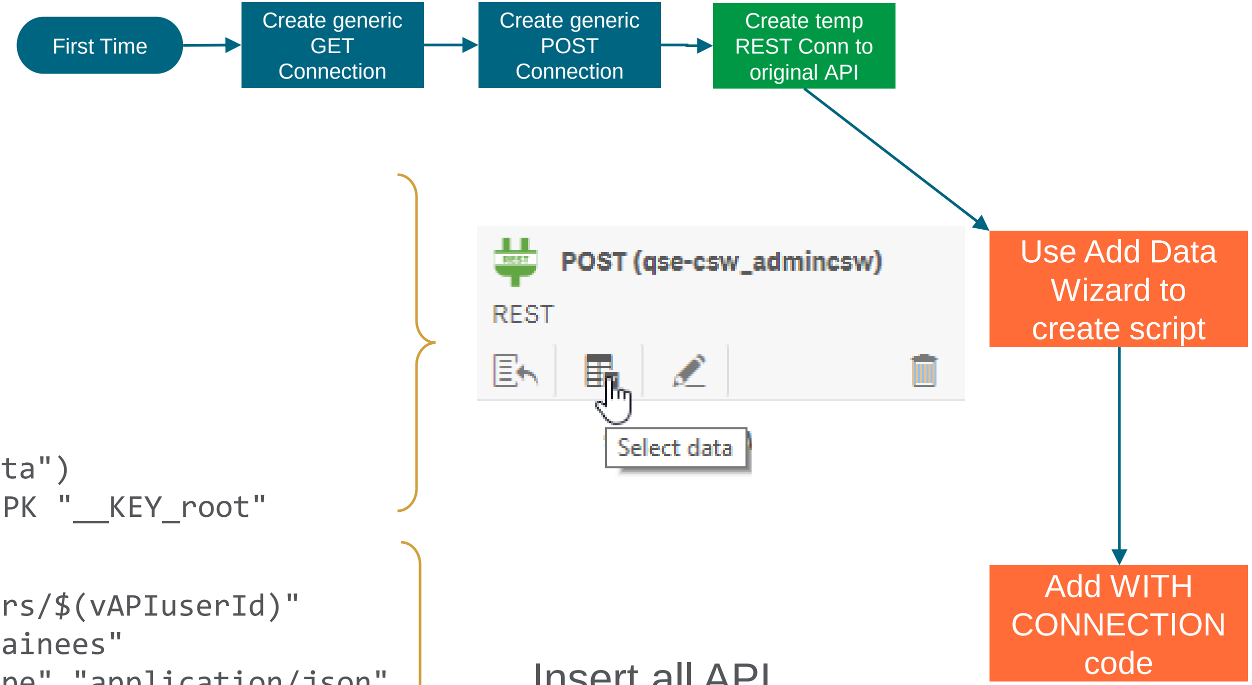
```
LIB CONNECT TO 'get_connection';
LIB CONNECT TO 'post_connection';
```

LIVE!



Note:

The „Create New Connection“ dialog can only be safed when there was a proper REST response


First Time
Create generic GET Connection
Create generic POST Connection
Create temp REST Conn to original API
POST (qse-csw_admincsw)
REST
Select data
Use Add Data Wizard to create script
Add WITH CONNECTION code

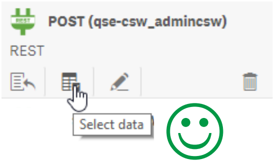
```
SQL SELECT
    "__KEY_root",
    "rootfield",
    (SELECT
        "id",
        "name",
        "__FK_data"
    FROM "data" FK "__FK_data")
FROM JSON (wrap on) "root" PK "__KEY_root"

WITH CONNECTION (
    URL "$(vBaseAPIurl)/users/$(vAPIuserId)"
    ,QUERY "tenent" "qliktrainees"
    ,HTTPHEADER "Content-Type" "application/json"
    ,HTTPHEADER "Authorization" "Bearer $(vToken)"
    //,HTTPHEADER "X-HTTP-Method-Override" "PUT",
    ,HTTPHEADER "cookie" "$(vCookie)"
    ,BODY "{""path"":""$(vAttribute)""}"
);
```

Insert all API parameters

2 golden steps

Part 1/2) Select Syntax

The „Select Data“ Wizard is your friend



☺

```
SQL SELECT
    "__KEY_root",
    "rootfield",
    (SELECT
        "id",
        "name",
        "__FK_data"
    FROM "data" FK "__FK_data")
FROM JSON (wrap on) "root" PK "__KEY_root"
```

- Temporarily create a <u>third</u> REST connection (POST or GET) and make your way to the response.
- If it is a complex request, use Postman‘s **mock server** instead
  - https://xxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.mock.pstmn.io/<apiendpoint>
  - No authentication needed
  - No query-strings, http-header etc. needed
  - Consistent sample response
- Target is to get a working load script that converts the response (Json, XML, CSV) into Qlik tables, not necessarily from the original API

Part 2/2) Endpoint parameters

Endpoint documentation is your friend

```
WITH CONNECTION (
    URL "$(vBaseAPIurl)/users/$(vAPIuserId)"
    ,QUERY "tenent" "qliktrainees"
    ,HTTPHEADER "Content-Type" "application/json"
    ,HTTPHEADER "Authorization" "Bearer $(vToken)"
    //,HTTPHEADER "X-HTTP-Method-Override" "PUT",
    ,HTTPHEADER "cookie" "$(vCookie)"
    ,BODY "{""path"":""$(vAttribute)""}"
);
```

- Set WITH CONNECTION (...) of to the SELECT command to work with the original API

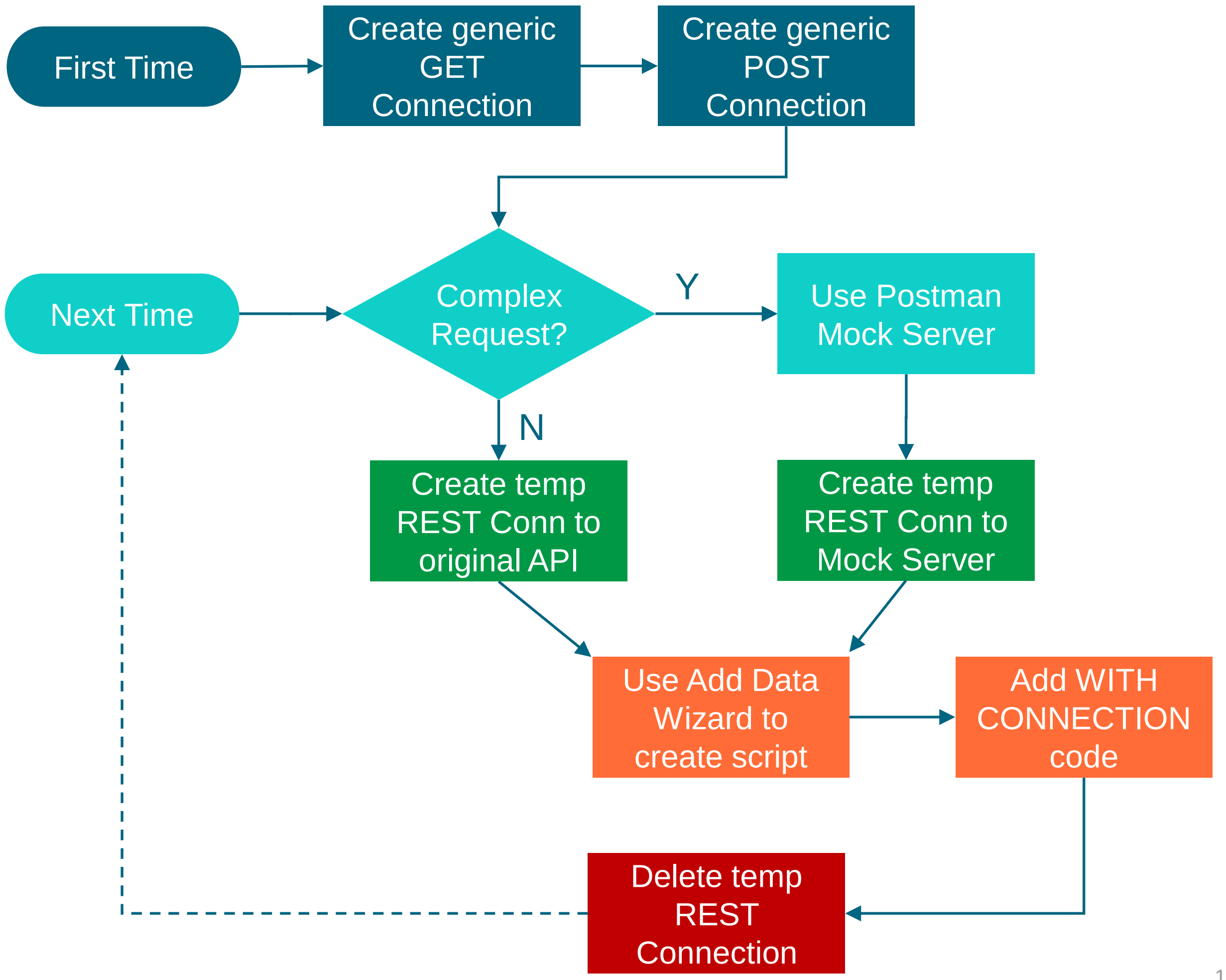
code snippet → https://github.com/ChristofSchwarz/qs_script_rest_api

- Provide all necessary params to satisfy the API
  - URL
  - Query-string(s)
  - Http-Header(s)
  - Body

Note: A Json Body has to use two double-quotes for keys and values, because it is already inside a double-quoted string

Qonnections
QLIK GLOBAL CONFERENCE

Workflow for working with REST Connector

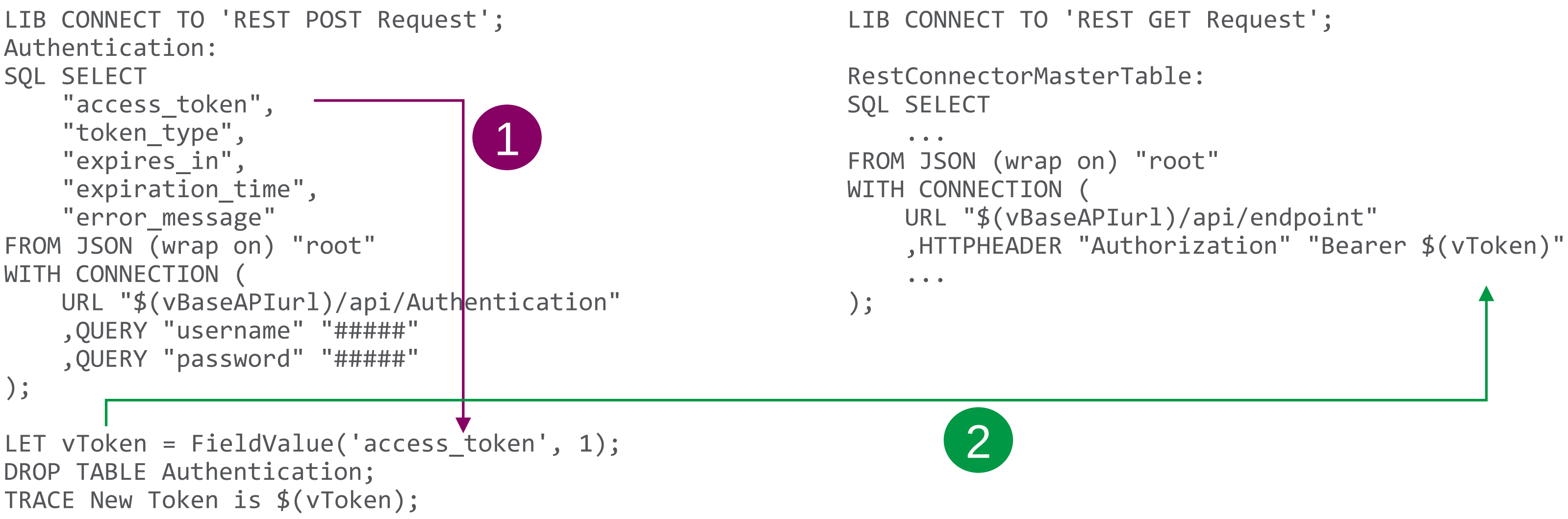

First Time
Create generic GET Connection
Create generic POST Connection
Next Time
Complex Request?
Y
Use Postman Mock Server
N
Create temp REST Conn to original API
Create temp REST Conn to Mock Server
Use Add Data Wizard to create script
Add WITH CONNECTION code
Delete temp REST Connection

Some Qlik Script Tricks

Request and use bearer token,
ISO-Date handling, Transposing Data, Paging

Receiving an access token

And use it as bearer authentication in subsequent calls



1) Get the token

```
LIB CONNECT TO 'REST POST Request';
Authentication:
SQL SELECT
    "access_token",
    "token_type",
    "expires_in",
    "expiration_time",
    "error_message"
FROM JSON (wrap on) "root"
WITH CONNECTION (
    URL "$(vBaseAPIurl)/api/Authentication"
    ,QUERY "username" "#####"
    ,QUERY "password" "#####"
);

LET vToken = FieldValue('access_token', 1);
DROP TABLE Authentication;
TRACE New Token is $(vToken);
```

2) Use the token

```
LIB CONNECT TO 'REST GET Request';

RestConnectorMasterTable:
SQL SELECT
    ...
FROM JSON (wrap on) "root"
WITH CONNECTION (
    URL "$(vBaseAPIurl)/api/endpoint"
    ,HTTPHEADER "Authorization" "Bearer $(vToken)"
    ...
);
```

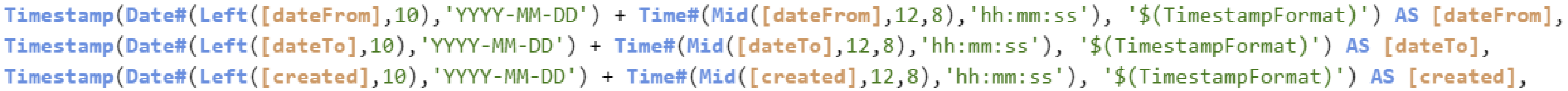
Even better: Try the old token first, get a new only if the old doesn‘t work anymore
→ https://github.com/ChristofSchwarz/qs_script_rest_api/blob/master/sub_try_request.md

Date handling

Reading ISO dates

```
                        1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20
      Field →           2  0  1  8  -  1  1  -  0  1  T  1  4  :  3  2  :  5  1  Z
Date#( Left(field,10) ,'Y  Y  Y  Y  -  M  M  -  D  D  '  )
Time#( Mid(field,12,6)                          ,  '  h  h  :  m  m  :  s  s  '  )
```

All together:



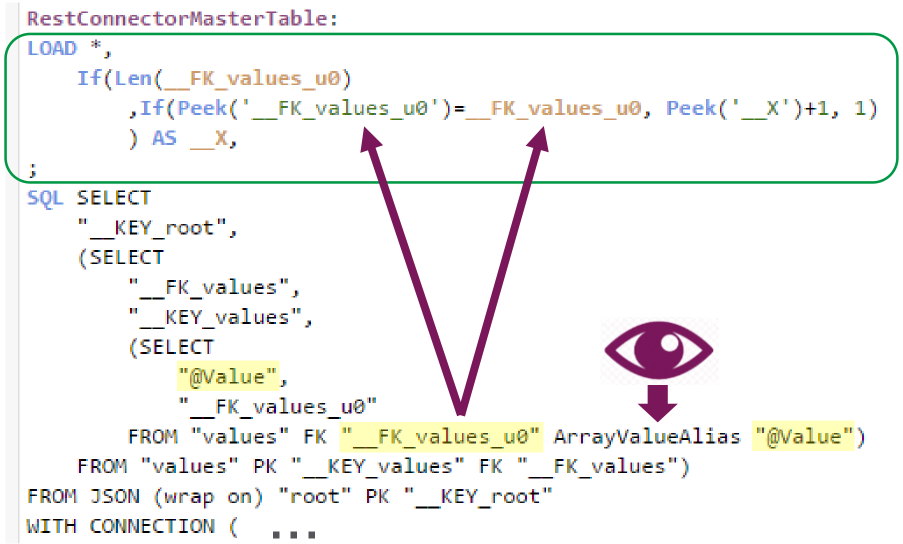
```
Timestamp(Date#(Left([dateFrom],10),'YYYY-MM-DD') + Time#(Mid([dateFrom],12,8),'hh:mm:ss'), '$(TimestampFormat)') AS [dateFrom],
Timestamp(Date#(Left([dateTo],10),'YYYY-MM-DD') + Time#(Mid([dateTo],12,8),'hh:mm:ss'), '$(TimestampFormat)') AS [dateTo],
Timestamp(Date#(Left([created],10),'YYYY-MM-DD') + Time#(Mid([created],12,8),'hh:mm:ss'), '$(TimestampFormat)') AS [created],
```

Code Snippet → https://github.com/ChristofSchwarz/qs_script_rest_api/blob/master/date_field_processing.md

Transposing Arrays

1	1
1	2
1	3
1	4
1	5
2	1
2	2
2	3
2	4
3	1
3	2
3	3
3	4
3	5

- Introduce a row autoid __X which
  - restarts at 1 and
  - increments when the main key is the same as above



```
RestConnectorMasterTable:
LOAD *,
    If(Len(__FK_values_u0)
        ,If(Peek('__FK_values_u0')=__FK_values_u0, Peek('__X')+1, 1)
        ) AS __X,
;
SQL SELECT
    "__KEY_root",
    (SELECT
        "__FK_values",
        "__KEY_values",
        (SELECT
            "@Value",
            "__FK_values_u0"
        FROM "values" FK "__FK_values_u0" ArrayValueAlias "@Value")
    FROM "values" PK "__KEY_values" FK "__FK_values")
FROM JSON (wrap on) "root" PK "__KEY_root"
WITH CONNECTION (  ...
```

Transposing Arrays

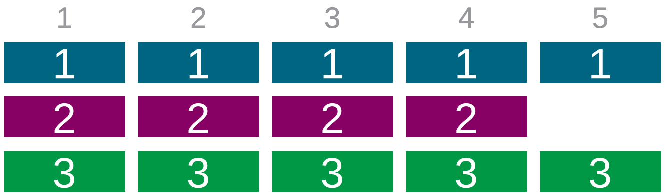
1	2	3	4	5
1	1	1	1	1
2	2	2	2	
3	3	3	3	3

- Use Generic Load to achieve this transpose

```
GENERIC LOAD
    __FK_values_u0, __X, @Value
RESIDENT RestConnectorMasterTable
WHERE __FK_values_u0 > 0;

DROP TABLE RestConnectorMasterTable;
```

Script Snippets → https://github.com/ChristofSchwarz/qs_script_rest_api/blob/master/transposing.md

Transposing Arrays



1	2	3	4	5
1	1	1	1	1
2	2	2	2	
3	3	3	3	3

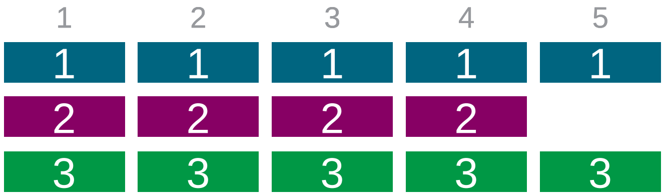
- Use Generic Load to achieve this transpose

If the field names are not part of the response ...

```
__FieldNames:
MAPPING LOAD * INLINE [
    1, Timestamp
    2, Passenger
    3, From Airport
    4, To Airport
    5, Date
    6, Operator
    7, Aircraft Type
] (no labels);

GENERIC LOAD
      __FK_values_u0, ApplyMap('__FieldNames', __X), @Value
RESIDENT RestConnectorMasterTable
WHERE __FK_values_u0 > 0;

DROP TABLE RestConnectorMasterTable;
```

Script Snippets →
https://github.com/ChristofSchwarz/qs_script_rest_api/blob/master/transposing.md

Transposing Arrays



1	2	3	4	5
1	1	1	1	1
2	2	2	2	
3	3	3	3	3

- Use Generic Load to achieve this transpose

If the field names are in block 1 of the response

```
__FieldNames:
MAPPING LOAD __X, @Value
RESIDENT RestConnectorMasterTable
WHERE __FK_values_u0 = 1 AND Len(@Value);
```

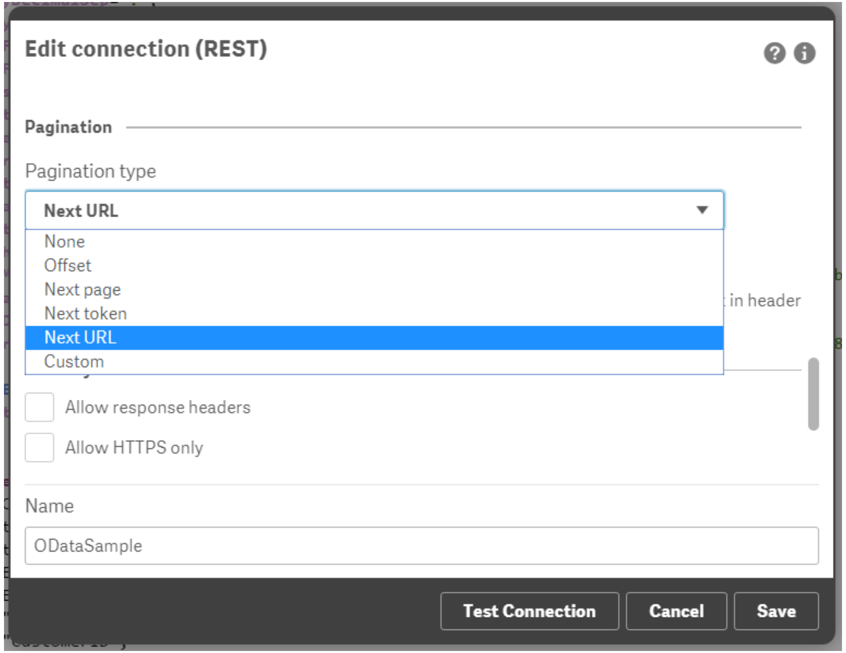
Script Snippets →
https://github.com/ChristofSchwarz/qs_script_rest_api/blob/master/transposing.md

```
GENERIC LOAD
      __FK_values_u0, ApplyMap('__FieldNames', __X), @Value
RESIDENT RestConnectorMasterTable
WHERE __FK_values_u0 > 1;

DROP TABLE RestConnectorMasterTable;
```

Paging with REST APIs

Built-in paging types

- There are some paging strategies supported with no coding, e.g.
  - BestBuy
  - Facebook
  - Google Analytics
- REST Connector and Pagination Video (M. Tarallo) https://youtu.be/QICT55_712I



Edit connection (REST)

Pagination

Pagination type: Next URL

□ Allow response headers

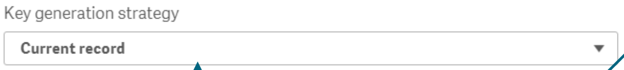
□ Allow HTTPS only

Name: ODataSample

→ https://help.qlik.com/en-US/connectors/Subsystems/REST_connector_help/Content/Connectors_REST/Create-REST-connection/Pagination-scenarios.htm

Tricks for Paging

OData



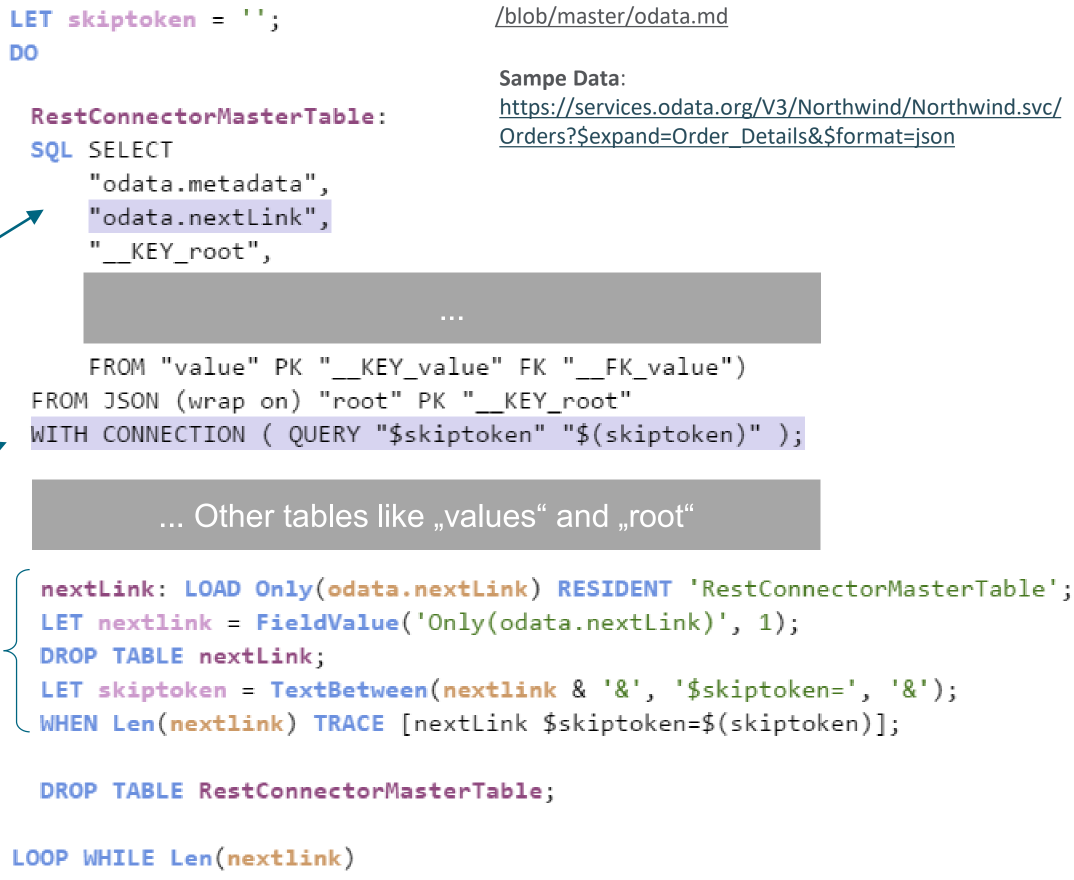
Key generation strategy

Current record

1. Set Key Generation strategy to „Current Record“
2. Get the first data page with the REST Connector Wizard.
3. Before first LOAD block
   - Create variable and start „DO“ loop
4. In the first LOAD block
   - (If missing add "odata.nextLink“)
   - Add „WITH CONNECTION“ to RestConnectorMasterTable
5. Before „DROP RestConnectorMasterTable“
   - Parse the „$skiptoken“ argument from odata.nextLink field
6. After „DROP RestConnectorMasterTable“
   - Close „LOOP WHILE“

Code Snippets: https://github.com/ChristofSchwarz/qs_script_rest_api/blob/master/odata.md

Sampe Data: https://services.odata.org/V3/Northwind/Northwind.svc/Orders?$expand=Order_Details&$format=json



```
LET skiptoken = '';
DO

  RestConnectorMasterTable:
  SQL SELECT
      "odata.metadata",
      "odata.nextLink",
      "__KEY_root",
                          ...
      FROM "value" PK "__KEY_value" FK "__FK_value")
  FROM JSON (wrap on) "root" PK "__KEY_root"
  WITH CONNECTION ( QUERY "$skiptoken" "$(skiptoken)" );

          ... Other tables like „values“ and „root“

  nextLink: LOAD Only(odata.nextLink) RESIDENT 'RestConnectorMasterTable';
  LET nextlink = FieldValue('Only(odata.nextLink)', 1);
  DROP TABLE nextLink;
  LET skiptoken = TextBetween(nextlink & '&', '$skiptoken=', '&');
  WHEN Len(nextlink) TRACE [nextLink $skiptoken=$(skiptoken)];

  DROP TABLE RestConnectorMasterTable;

LOOP WHILE Len(nextlink)
```

Qonnections QLIK GLOBAL CONFERENCE

More resources

Help

- https://help.qlik.com/en-US/connectors/Subsystems/REST_connector_help/Content/Connectors_REST/Create-REST-connection/Create-REST-connection.htm

Code Snippets

- https://github.com/ChristofSchwarz/qs_script_rest_api
- https://github.com/ChristofSchwarz/qs_script_rest_api/blob/master/transposing.md

Videos

- REST Connector Deluxe (C. Schwarz) https://youtu.be/7m9ZejlzkkY
- Qlik and REST (M. Tarallo) https://youtu.be/ibCACdF_tPo
- REST Connector and Pagination (M. Tarallo) https://youtu.be/QlCT55_712I
- Google Sheet API with Qlik Script (C. Schwarz) https://youtu.be/l9sk-v_PTf8

Qonnections

QLIK GLOBAL CONFERENCE

Thank You