# Qlik REST Connector Installation and User Guide

*Qlik REST Connector Version 1.0*

*Newton, Massachusetts, November 2015*

*Authored by QlikTech International AB*

# Contents

# 1 Qlik REST Connector

The Qlik REST Connector enables QlikView to efficiently load data into a QlikView application from a REST service that returns data in XML, CSV, or JSON format. Many web-based data sources expose data through a REST API. The Qlik REST Connector is a generic connector, that is, it is not tailored to a specific REST data source.

# 2 Install the Qlik REST Connector

## 2.1 System requirements

Version 1 of the Qlik REST Connector runs only on systems with 64-bit processors. It can be installed on any 64-bit system that runs QlikView. Those systems include:

- Microsoft Windows Server 2008 x64 Edition
- Microsoft Windows Server 2008 R2
- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows 7
- Microsoft Windows 8.1
- Microsoft Windows 10

The Qlik REST Connector runs with QlikView and is designed to work with the same web browsers as QlikView.

Version 1 of the Qlik REST Connector also requires Microsoft .NET Framework 4.5.2 or later version.

## 2.2 Installing the Qlik REST Connector

1. Download the `QvRestConnector_setup.exe` file from the Qlik download site.
2. Run the `QvRestConnector_setup.exe`.

   The Qlik REST Connector InstallShield Wizard sets up the installation environment and completes the installation.

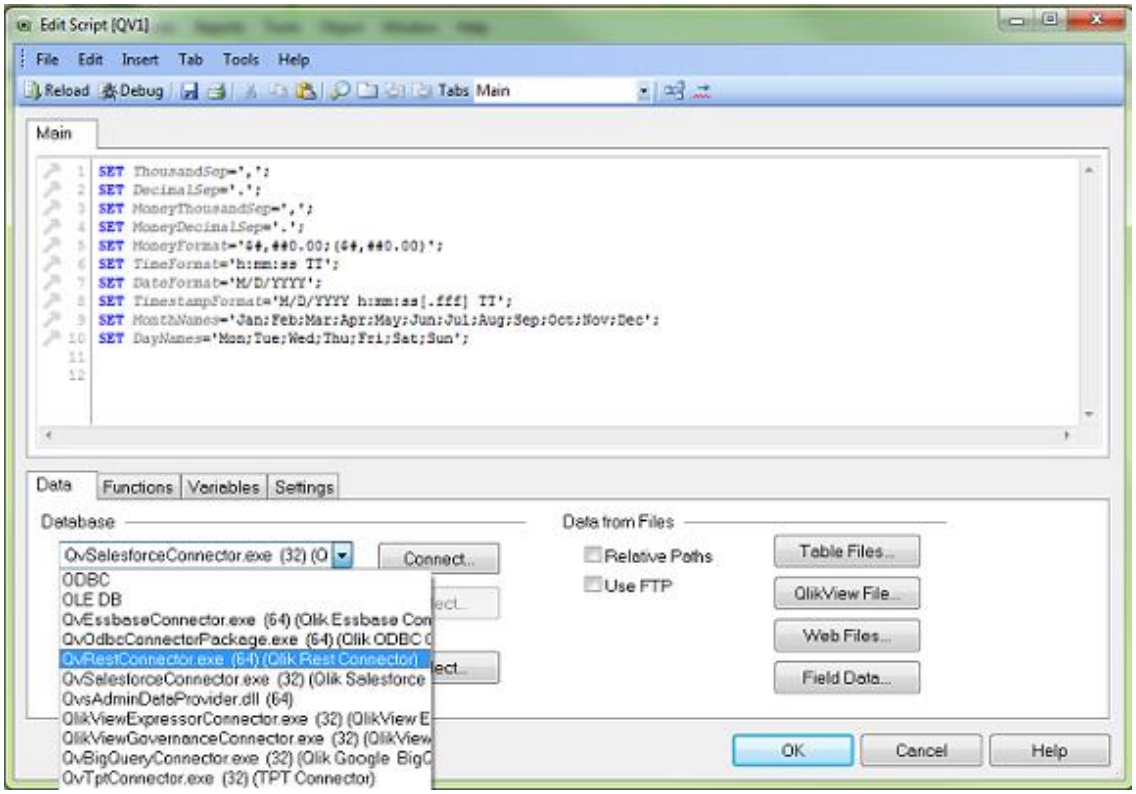
   The Connector is then available from QlikView with the name "QvRestConnector.exe."
3. Open the QlikView application.
   If QlikView Personal Edition is open on your desktop when you install the REST Connector, you must close and reopen it in order for QlikView to recognize the connector.

# 3 Create a REST connection

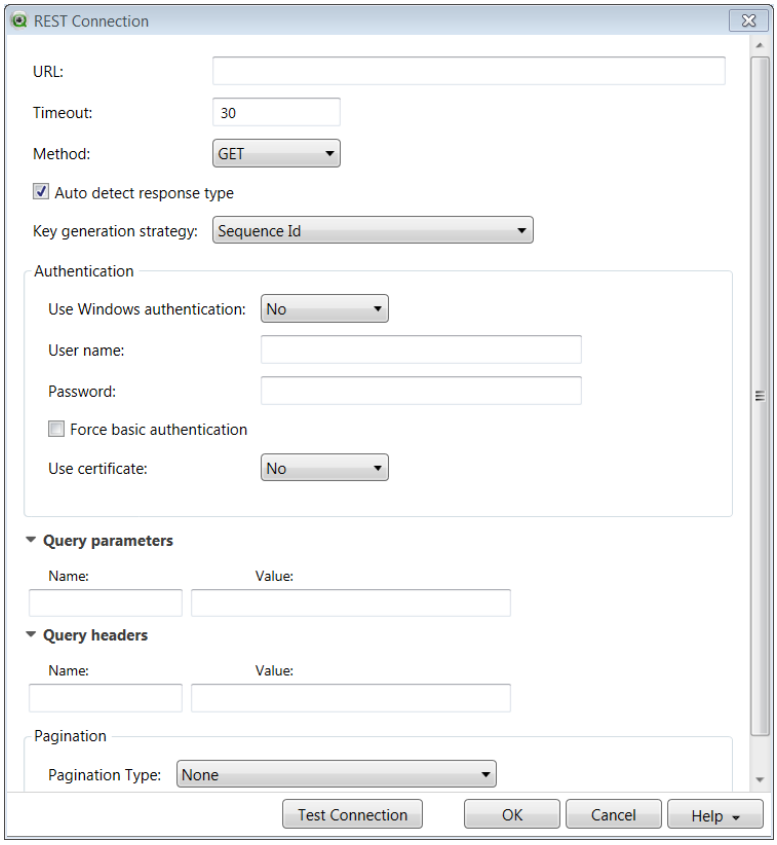
Access to the Qlik REST Connector is provided by the Qlik Visual Analytics Platform. QlikView applications connect to data sources through the Script Editor.

## 3.1 Specifying the REST data source

1. Open a new or existing QlikView application.
2. Click **Edit Script...** on the **File** menu in the main QlikView window.
3. Select QvRestConnector.exe from the drop-down list next to **Connect...** on the **Data** tab in the lower part of the **Edit Script** dialog.



4. Click **Connect...** to open the **REST Connection** dialog.




5. Enter the URL of the resource being requested from the REST service.
   The URL syntax must include http://. In addition to the location, the URL can include query parameters and other information about the data being retrieved. For example, a URL to Google Maps can be:

   ```
   http://maps.googleapis.com/maps/api/geocode/xml?address=1600+
   Amphitheatre+Parkway,+Mountain+View,+CA&amp;sensor=false
   ```

6. Enter a timeout value in seconds.
   This is the time limit for the connection to be completed. If the connection request takes longer than the timeout value set, an error message is displayed.

7. Select **GET** or **POST** from the **Method** drop-down list.
   The GET method retrieves information from the resource and does not change the state of the server. It is considered a *safe* method with regard to the server.

   The POST method creates a new entry in the resource. The entry is specified in the **Request body** field, which appears when the POST method is selected. The request or data sent with the POST method can be, for example, a message to a bulletin board or newsgroup.

8. Enter the data to be posted to the resource with the POST method in the **Request body** field.
   If you use the GET method, this field is not displayed.

9. Select **Auto detect response type** to detect the data format in the REST data source when the connection is made.
   The data formats are CSV, XML, and JSON. If the data format is detected during connection, the **Create Select Statement** dialog displays the field required for selecting data in the source's format when it opens. If **Auto detect response type** is not selected, you can use **Auto detect** in the **Create Select Statement** dialog, or you can select the format manually if you know what it is.
   See Select and load data

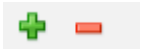
10. Select the **Key generation strategy** to use for creating parent-child relationships in the data fields.
    To associate parent and child fields appropriately, the connector generates ID fields to serve as primary and foreign keys. The key generation strategy determines how the keys are generated.
    - **Sequence ID**: Generates an ID for each child of a parent record. This strategy is simple and efficient, but it requires that the order of field is the same in every record.
    - **Current record**: Generates a hash key based on all the values of the current record and uses that as the primary key. It also generates a hash key based on all the values of the direct parent and uses that as the foreign key.
    - **Fully qualified record**: Generates a hash key based on all the parent values.
    - **No keys**: Keys for parent-child relationships will not be shown in the **Create Select Statement** dialog, but they will be used in the SELECT statements.

## 3.2 Specifying credentials for the REST service

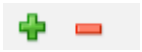
1. Select **Yes** for the **User Windows Authentication** property if you intend to use the Windows credentials on the local machine to connect to the REST service.
2. Enter the user name and password for the REST service.
   These two fields can be blank if the service does not require authentication. The authentication credentials are used only if both **User name** and **Password** fields are not blank.
   If you use Windows authentication, the user name and password fields are hidden because they are not needed.
3. Select **Force basic authentication** if the server uses the OAuth authentication schema.
   The REST Connector does not support the OAuth schema. If you select **Force basic authentication**, the REST Connector forces the server to use the Basic authentication schema.
4. Select **Installed** or **From file** for the **Use certificate** property to use an X509 certificate for authentication.
   If you select **Installed**, additional properties are displayed for selecting the certificate location, the certificate store, and the certificate name. The certificate location can be either the current user or local machine. The certificate store is one of the default Windows stores (Personal, Trusted Root Certification Authorities, Intermediate Certification Authorities, Trusted Publisher, Untrusted Certificates, Third-Party Root Certification Authorities, Trusted People, or Other People). The drop-down list of certificate names is derived from the certificate store.
   If you select **From file**, properties are displayed for specifying the file location and the certificate key.
5. Select the type of paging to use from the **Pagination Type** drop-down menu.
   To select the appropriate type of paging, you must know how the data source to which you are connecting implements the REST API. The REST API allows for Offset, Next token, Next URL, and Custom pagination. You can also choose **None** from the list of **Pagination Type** selections.
   See Loading paged data
6. Click **Test Connection** to verify that a connection can be made with the REST service properties entered.

## 3.3 Specifying additional parameters

1. Enter the name and value for any additional query parameters.
   Query parameters are specific to the REST service to which you are connecting. They can specify information to return. For example, a parameter named “filter” could be used to specify a value such as “salary>10000.” Query parameters can also be used to provide security keys.

**Name** and **Value** fields can be added or deleted with the  icons to the right of the fields.

2. Enter query header field names to be included in the *http* request header and the values for each query header field.

   Query header **Name** and **Value** fields can be added or deleted with the  icon to the right of the fields. `Accept-Language` is an example of a query header. To specify English as the language, you would use the value "`en`."

3. Click **OK** to create the connection string used when sending the load statement to the REST service.

## 3.4 Loading paged data

Many REST API implementations enforce a limit on the maximum number of records that can be returned in a single REST call. This applies mainly to JSON and XML data. When there are more records than can be returned by a single call, additional record pages can be returned by subsequent calls. How those additional pages are returned depends on the REST pagination type implemented by the data source.

**Note: CSV data sources usually do not provide paging values. When no paging is provided, the None paging option should be selected.**

The following paging options are available in the REST API:

- **None** is used when no paging is supported. Some REST API implementations simply return records with a single call and do not allow subsequent calls.
- **Offset** uses a starting value from which to read additional records.
- **Next token** uses a token that is passed to the URL call for the next set of records.
- **Next URL** uses a value that contains the URL for the next set of records.
- **Custom** is a special option that can be used when none of the other paging options are implemented. It allows the user to write a customized query to return a set of records.

The Qlik REST Connector enables the selection of **Pagination Type** in the **Create Select Statement** dialog.

**Note: When specifying field names in any of the pagination fields described below, you must use the name as returned by the data source, not the name displayed by the connector in the Select data to load step. The connector renames some fields in some cases, such as when a field name is duplicated in different levels of a JSON or XML document. For example, the field name "total," used in Total field path, might be renamed "total_u1" by the connector.**

The **None** option is selected when the REST API implemented by the data source does not support paging. The **None** option can also be selected if you do not want to use a paging option. In that case, only the first set of records is returned.

When the **Offset** option is selected, the following fields display in the **Create Select Statement** dialog:

- **Start field name**: The name of the data source's specified start field. When a data source such as LinkedIn uses Offset for pagination, it must specify which field is considered the first field in a record.
- **Is header field**: Specifies how the Start field name value is transmitted by the connector. When **Is header** field is selected, the connector transmits the field name in the header. If **Is header** field is not selected, the field name is transmitted in the URL.
- **Start field value**: The number of the record from which to start reading records.

- **Count field name**: The name of the count field used by the data source. The count field indicates the amount of data to return with each page. For example, JIRA uses the name "maxresults" for the count field name.
- **Is header field**: Specifies how the **Start field value** and **Count field name** are transmitted by the connector. When **Is header field** is selected, the connector transmits the value and field name in the header. If **Is header field** is not selected, those values are transmitted in the URL.
- **Count field value**: The number of records to return with each call. The maximum allowed depends on the data source.
- **Total field path**: The path to where the total field amount is stored. For example, JIRA returns the value in the root element, root/total. The connector can use this value to understand when all the data has been loaded.
- **Is header field**: Specifies how the **Count value** and **Total field path** are transmitted by the connector. When **Is header field** is selected, the connector transmits the value and field name in the header. If **Is header field** is not selected, those values are transmitted in the URL.
- **Data field path**: Contains the path to the field in a response that indicates whether the response contains data or iterations should stop. This field is used when the data source does not return an entire value.

When the **Next token** option is selected, the following fields display:

- **Next token field name**: The name of the parameter in the URL or request header that is used to send the token of the next page with each request.
- **Is header field**: Specifies how the **Next token field name** value is transmitted by the connector. When **Is header field** is selected, the connector transmits the field name in the header. If **Is header field** is not selected, the field name is transmitted in the URL.
- **Next token field path**: The path to where the token field is stored. The token field stores the token value for the next page.
- **Is header field**: Specifies how the Next token path value is transmitted by the connector. When **Is header field** is selected, the connector transmits the path in the header. If **Is header field** is not selected, the path is transmitted in the URL.

When the **Next URL** option is selected, the following fields display:

- **Next URL field path**: The path to the field in the response that contains the URL for the next page. When a data source has more pages, it returns the URL for the next page in the response the previous request.
- **Is header field**: Specifies how the **Next URL field path** value is transmitted by the connector. When **Is header field** is selected, the connector transmits the path in the header. If **Is header field** is not selected, the path is transmitted in the URL.

When the **Custom** option is selected, no additional fields are displayed. The **Custom** option inserts a script template in the **Edit script** dialog. After the script has been inserted in the **Edit script** dialog, you must adapt the script for the source from which you are retrieving data in order to retrieve all of the intended records.

The following script illustrates how the template can be adapted to retrieve the desired number of pages in a JIRA data source. Without the adaptation, the script would retrieve only the first page of data.

```
// Implement the logic to retrieve the total records from the REST source and assign to the 'total'
//Getting the "total" value from the first API URL call
totalTable:
SQL SELECT
            "total" AS "total_u1"
FROM JSON (wrap on) "root";

//This variable is a total of JIRA entries. Will be user later in 'For' loop
Let total = peek('total_u1',0,'totalTable');
DROP TABLE totalTable;

Let totalfetched = 0;
Let startAt = 0;
Let pageSize = 10;

for startAt = 0 to total

TRACE "startAt: " $(startAt);

RestConnectorMasterTable:
SQL SELECT
    "__KEY_root",
    (SELECT
        "self" AS "self_u38",
        "key" AS "key_u8",
        "__FK_issues",
        "__KEY_issues"
    FROM "issues" PK "__KEY_issues" FK "__FK_issues")
FROM JSON (wrap on) "root" PK "__KEY_root"
WITH CONNECTION(
    QUERY "startAt" "$(startAt)"
);
// Change URL included in 'WITH CONNECTION' as needed to support pagination for the REST source.
// Please see the documentation for "Loading paged data."

startAt = startAt + pageSize;

NEXT startAt;

[issues]:
LOAD    [self_u38] AS [self_u38],
    [key_u8] AS [key_u8]
RESIDENT RestConnectorMasterTable
WHERE NOT IsNull([__FK_issues]);

DROP TABLE RestConnectorMasterTable;
```

## 3.5 Setting up selected REST sources for data loading

The following procedures explain how to acquire the connection requirements for selected social networking platforms:

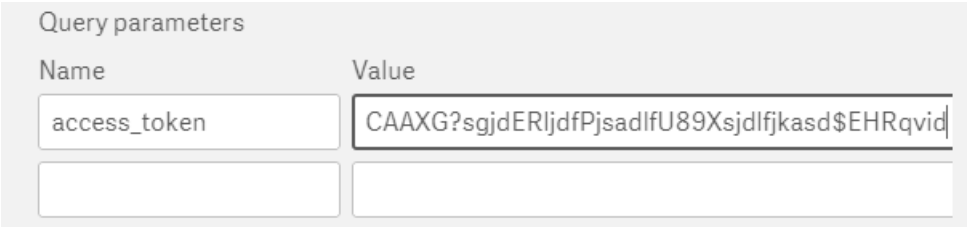
- Facebook
- LinkedIn
- Twitter
- Google Analytics

Connecting to these social networking platforms requires a developer account. You use the developer interface for each platform to get credentials and query parameters for the specific application or service you intend to collect data from.

### 3.5.1 Connecting to Facebook

Do the following:

1. Sign into your Facebook developer account at https://developers.facebook.com/.
2. Create a new application.
   The registration and validation process for the application takes time, so the application may not be available as soon as you add it. If you have an application that was written previously, you can access that immediately, but it must be an application created with a developer account.
3. Navigate to the developer explorer page at https://developers.facebook.com/tools/explorer.
4. Select the application.
5. Select **Get Access Token**.
   The objective is to provide access to data. **Get Access Token** is currently the interface label Facebook uses. Even if that label changes, there must be a means of getting an access token.
6. Generate an API method.
7. Submit.
8. Go to the **Create a connection - REST** step in the REST Connector.
9. In the **URL** field, enter the URL from you Facebook application.
   The URL contains the API method that determines the type of response the data source returns.
   For example:
   `https://graph.facebook.com/+'me?fields=albums.limit(5),posts.limit(5)'`
10. Enter your Facebook developer credentials in the **User name** and **Password** fields.
11. Add a parameter in the **Query parameters** field for the access token you received from the application.
    For example:



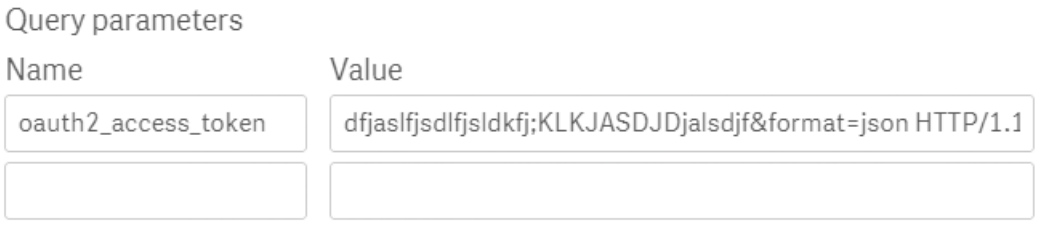

12. Complete the remaining fields and selections in the **Create a connection - REST** step.
    See: Create a REST connection

### 3.5.2 Connecting to LinkedIn

Do the following:

1. Sign into your LinkedIn developer account at https://www.linkedin.com.
2. Navigate to the developer page at https://www.linkedin.com/developer/apps
3. Create a new application.
   The registration and validation process for the application takes time, so the application may not be available as soon as you add it. If you have an application that was written previously, you can access that immediately, but it must be an application created with a developer account.
4. Open the REST console and sign in.
   The REST console is an apigee.com account, https://apigee.com/console/linkedin. You sign in with your LinkedIn credentials.

5. Select **Authentication--OAuth 2**.
6. Select a **Service**.
7. Select an API method.
8. Select Send to generate the JSON URL and query parameters.
   The JSON URL looks like:
   `https://api.linkedin.com/v1/people/~?format=json`
   The request query parameters include "~?oauth2_access_token=" with a token value that is used in the **Query parameters** section of the **Create a connection - REST** step in the REST Connector.
9. Go to the **Create a connection - REST** step in the REST Connector.
10. In the **URL** field, enter the URL generated for your LinkedIn application.
11. Enter your developer credentials for the apigee.com account in the **User name** and **Password** fields.
12. Add a parameter in the **Query parameters** field for the access token you received for the application.
    For example:

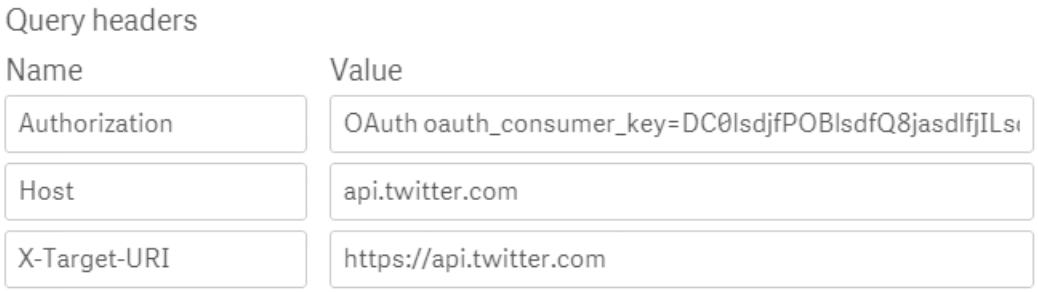



13. Complete the remaining fields and selections in the **Create a connection - REST** step.
    See: Create a REST connection

### 3.5.3 Connecting to Twitter

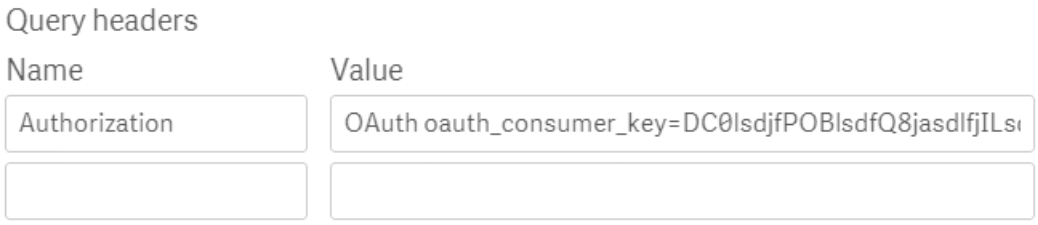
Do the following:

1. Sign into your Twitter account at https://twitter.com/.
2. Enter the URL to the developer page at https://dev.twitter.com/rest/tools/console.
3. Select **Authentication--OAuth 1**.
   You are asked to authorize the developer console to use you Twitter authentication. When you authorize, the **Authentication** field shows your Twitter account user name.
   For example, “twitter-JonSmith.”
4. Select a **Service**.
5. Select an API method.
   For example, you can select GET methods to "/trends/place.json" or "/trends/closest.json."
6. Select **Send** to generate the JSON URL and Headers.
   The JSON URL looks like:
   `https://api.twitter.com/1.1/help/configuration.json`
   The Headers include OAuth authorization with a token value, a host name, and X-Target-URI that are used in the **Query headers** section of the **Create a connection - REST** step in the REST Connector.
7. Go to the **Create a connection - REST** step in the REST Connector.
8. In the **URL** field, enter the URL generated for your Twitter connection.

9. Enter your Twitter credentials in the **User name** and **Password** fields.
10. Add parameters in the **Query headers** field for the authorization token, host name, and X-Target-URI.
    For example:
    
    
11. Complete the remaining fields and selections in the **Create a connection - REST** step.
    See: Create a REST connection

### 3.5.4 Connecting to Google Analytics

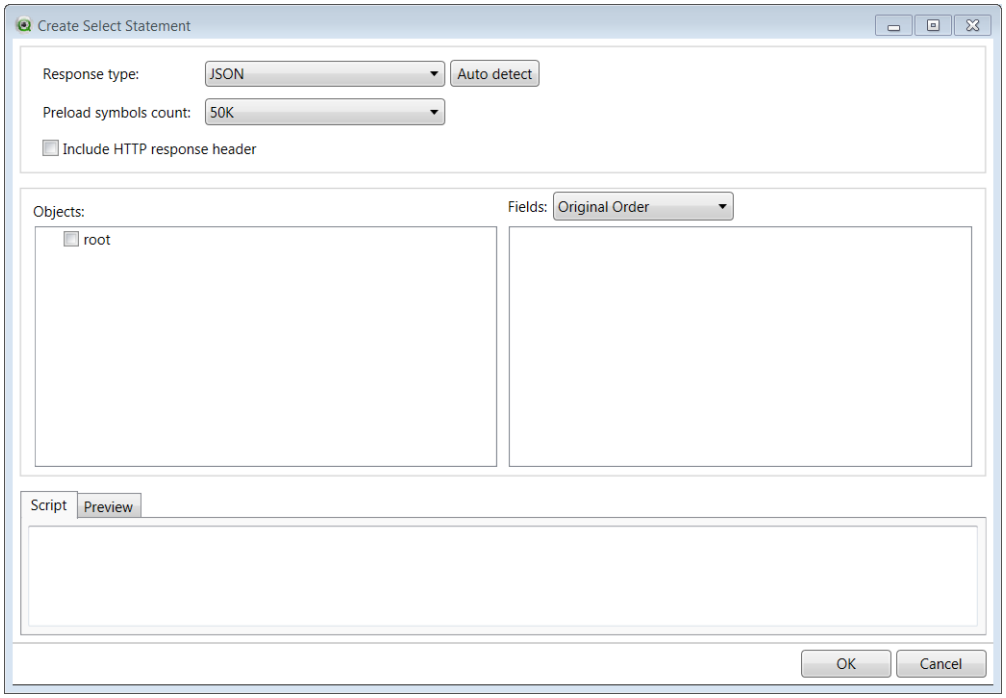
Do the following:

1. Sign into your Google Analytics account at https://www.google.com.
   A Google Analytics account is separate from an account you might have for other Google services, like gmail.
2. Navigate to the developer console at https://developers.google.com/apis-explorer/#p/analytics/v3/.
3. Turn on **Authorize requests using OAuth 2.0**.
   When you turn on the authorization, you are asked to authorize scopes that the Google Analytics API has access to.
4. Select an API method.
   For example, you can select methods such as "analytics.data.realtime.get" or "analytics.management.accountSummaries.list."
5. Enter required fields for the method, if any.
   Each method has fields, such as Unique table ID for retrieving real time data, and the fields marked in red are required.
6. Select **Execute** to generate the GET request.
   The GET looks like:
   ```
   GET https://www.googleapis.com/analytics/v3/management/accountSummarie
   ```
7. Go to the **Create a connection - REST** step in the REST Connector.
8. In the **URL** field, enter the generated URL beginning with "https://".
9. Add a parameter in the **Query headers** field for the authorization token.
   For example:
   
   
10. Complete the remaining fields and selections in the **Create a connection - REST** step.
    See: Create a REST connection

# 4 Select and load data

1. Click the enabled **Select...** on the **Data** tab in the lower part of the **Edit Script** dialog.
   The **Edit Script** dialog has two **Select...** buttons when the REST Connector has been connected but only the bottom button is enabled.

2. Select a Response type from the drop-down list in the **Create Select Statement** dialog.
   Response types are JSON, CSV, and XML. You can also click **Auto detect** to have the connector try to identify the response type in the data source, in case it did not identify it upon connection (see **Auto detect response type** above).




3. If the response type is XML or JSON, select the number of symbols to be preloaded from the **Preload symbols count** drop-down list.
   The symbols count limits the response data used to build metadata. It specifies maximum number of characters, not bytes.

   **Note: When building the metadata, the REST connector parses all characters in the data source, including data as well as metadata. If some fields in the data source contain large amounts of data, the symbols count could be reached before all the metadata has been parsed. As a result, the list of tables and fields in the Select data to load dialog would be incomplete.**

4. If the response type is CSV, select the character used to separate fields in the **Csv delimiter** text box.
   The default delimiter character is comma. Other delimiter characters supported are semicolon (;), pipe (|), caret (^), and tab.

5. If the response type is CSV, select the character to use for the purpose of "quoting" a character or string of characters that are to be interpreted literally.
   Any special significance of the quoted character or characters is ignored. The default quote character is the double quotation mark (“). The other supported “quoting” character is the single quotation mark (’).

6. If the response type is CSV, select **Csv has header** if the CSV data has a header row.

7. Select **Include HTTP response header** to provide data from the HTTP response header.
When this option is selected, the connector inserts a synthetic table, "_response_header," as a root table. This table can be selected like any other table.

   **Note: The Include HTTP response header must be selected when an Is header field has been selected for Pagination in the REST Connection dialog. Otherwise, pagination will not work as expected. See** Loading paged data**.**

8. Click an object name in the **Objects** list.
XML and JSON data is usually hierarchical, and each level in the hierarchy contains both fields and tables. The lowest level contains only fields. When the data in XML or JSON objects is structured hierarchically, the hierarchy is represented in the **Objects** list. Fields from each level of the object's hierarchy are displayed as you click on the names of the object levels.

9. Select the fields to load from the **Fields** list.
When you select an object level by selecting its check box in the **Objects** list, all of its fields are selected.
The SELECT statement that will be used to load data from the REST service displays under the **Script** tab in the bottom portion of the dialog. Each field must be selected explicitly unless you select the object's check box, in which case, as indicated above, all the object's fields are selected.

   The SELECT statement does not include a WHERE clause, but one can be added in the **Edit Script** dialog after the script built automatically in the **Create Select Statement** dialog is inserted there.

   Field names can also be changed in the Edit Script dialog by adding aliases.

10. Click **OK** when you are finished making your selections.
The script built automatically under the **Script** tab is inserted in the **Main** tab in the **Edit Script** dialog.

11. Insert WITH CONNECTION section to override the URL and query parameters.
This step is optional. The WITH CONNECTION section allows you to change or add to the query transmitted by the CONNECT TO statement that has been inserted in the **Edit Script** dialog. Add the WITH CONNECTION section in the **Edit Script** dialog as shown in the following illustration.

```
19  LIB CONNECT TO 'northwind2.xml';
20
21  RestConnectorMasterTable:
22  SQL SELECT
23      (SELECT
24          (SELECT
25              "EmployeeID",
26              "LastName",
27              "FirstName"
28           FROM "Employee")
29       FROM "Employees")
30  FROM XML "northwind"
31  WITH CONNECTION (
32      URL "http://localhost:81/northwind.xml",
33      QUERY "type" "XML",
34      QUERY "size" "full",
35      HTTPHEADER "auth" "token",
36      HTTPHEADER "token" "345213452345");
37
38  [Employee]:
39  LOAD     [EmployeeID],
40      [LastName],
41      [FirstName]
42  RESIDENT RestConnectorMasterTable;
```

# 5 Locate the REST connector log file

The REST Connector's log file is located at:

`{ProgramData}\QlikTech\Custom Data\QvRestConnector\Log`

Where `{ProgramData}` is `%ALLUSERSPROFILE%`.