# API Access

# Table of Contents

1. Enable API Access ..... 3
1.1 Set up API Access ..... 4
2. Test API Access ..... 8
2.1 Test API Access / Get API Token via MI API Test Tool ..... 9
2.2 API Test Tool ..... 14
2.3 Get Token API via curl ..... 17
2.4 API via curl ..... 19
2.5 Chrome browser REST client ..... 20
2.6 Python example of using API ..... 22
3. API calls ..... 25
3.1 User and Group API ..... 26
3.2 Report Data API ..... 35
3.3 Metric API ..... 42
3.4 Multi-Metric API ..... 46
3.5 Dimension API ..... 50
3.6 Calendar Event API ..... 56
3.7 Alerts API ..... 61
3.8 Target API ..... 74
3.9 Bursts API ..... 79
3.10 Dataset API ..... 82
3.11 Data Source API ..... 94
4. API Calls in JavaScript ..... 102
4.1 Displaying Metric Insights content in web page using API via JavaScript ..... 103
4.2 Looking in Metric Insights to see what items to pull via API ..... 110
4.3 Qlik Sense mashup using Metric Insights API ..... 115

# 1. Enable API Access

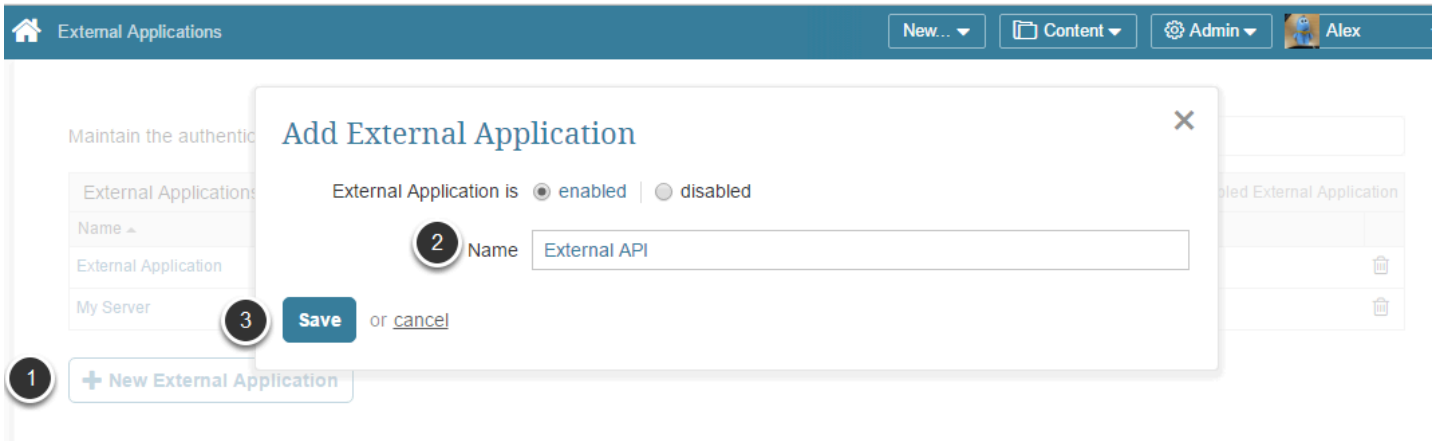
# 1.1 Set up API Access

To configure your Metric Insights instance for API Access, you should create a secret key by the means of which your API client can access the data.

Once you set up API Access you can use most of the API calls. A few calls are unavailable until you set up specific access for them.

Steps 4 and 5 describe what they are and how to enable access for them. In general, this includes access to the user API as well as access to all API calls from external web sites (typically via JavaScript).

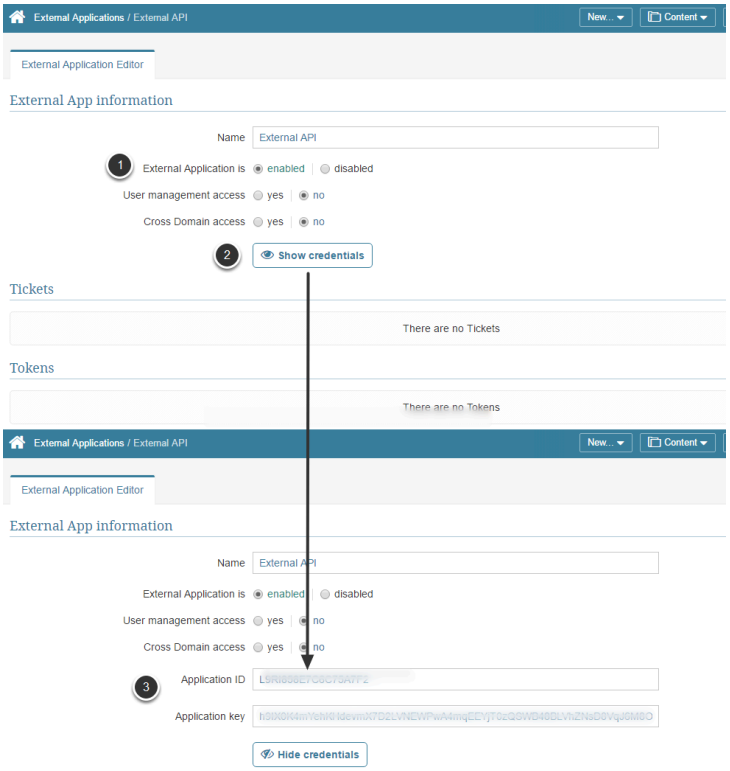
## 1. Access Admin Menu > Credentials > External Applications




1. Select **[+ New External Application]** from list of existing *External Applications*
2. Define the name of an External Application
3. **Save** your entry.

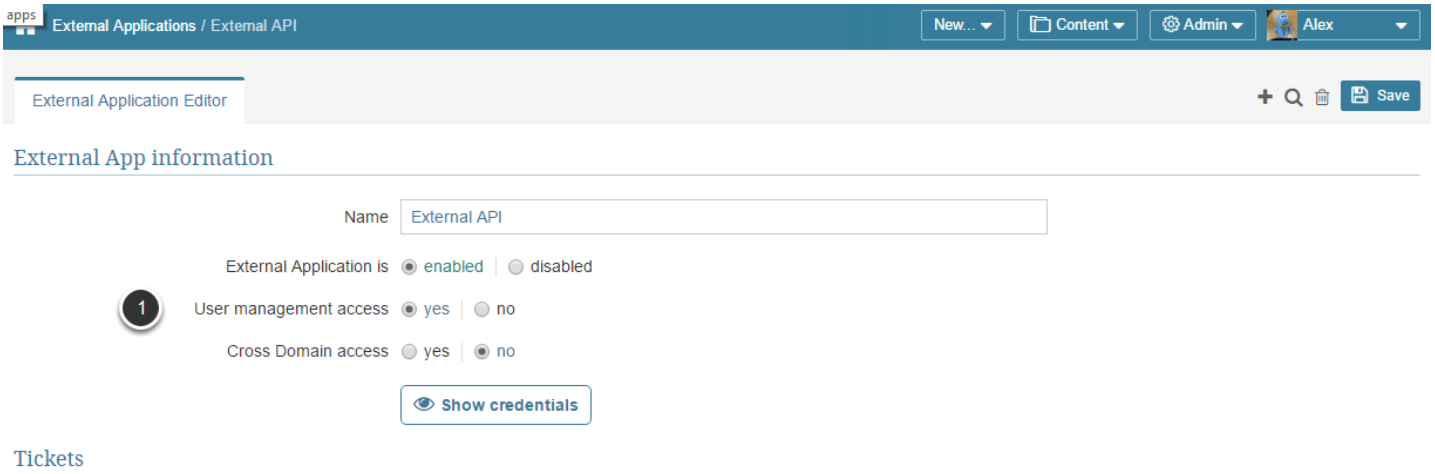
The *External Application Editor* opens.

# 2. Define the External Application



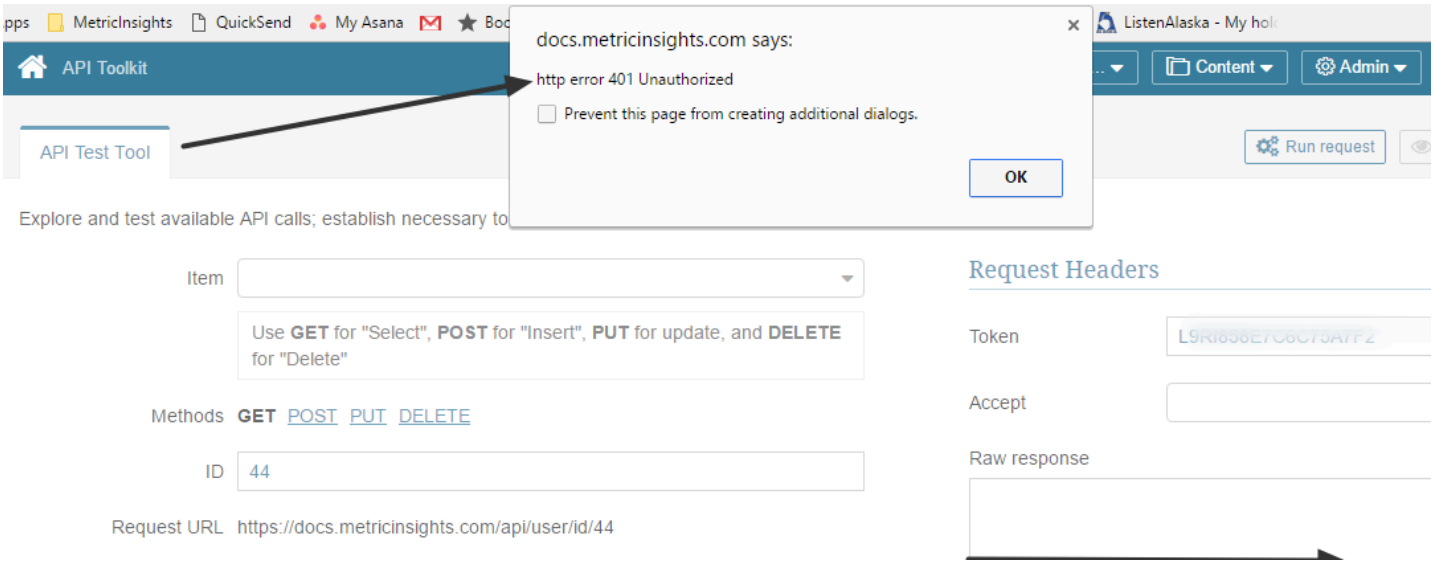

1. Enable your API
2. Click **Show Credentials**
3. The **Application ID** and **Application key** are now available to you

# 3. To enable User API calls




1. Set **User management access** to 'yes'

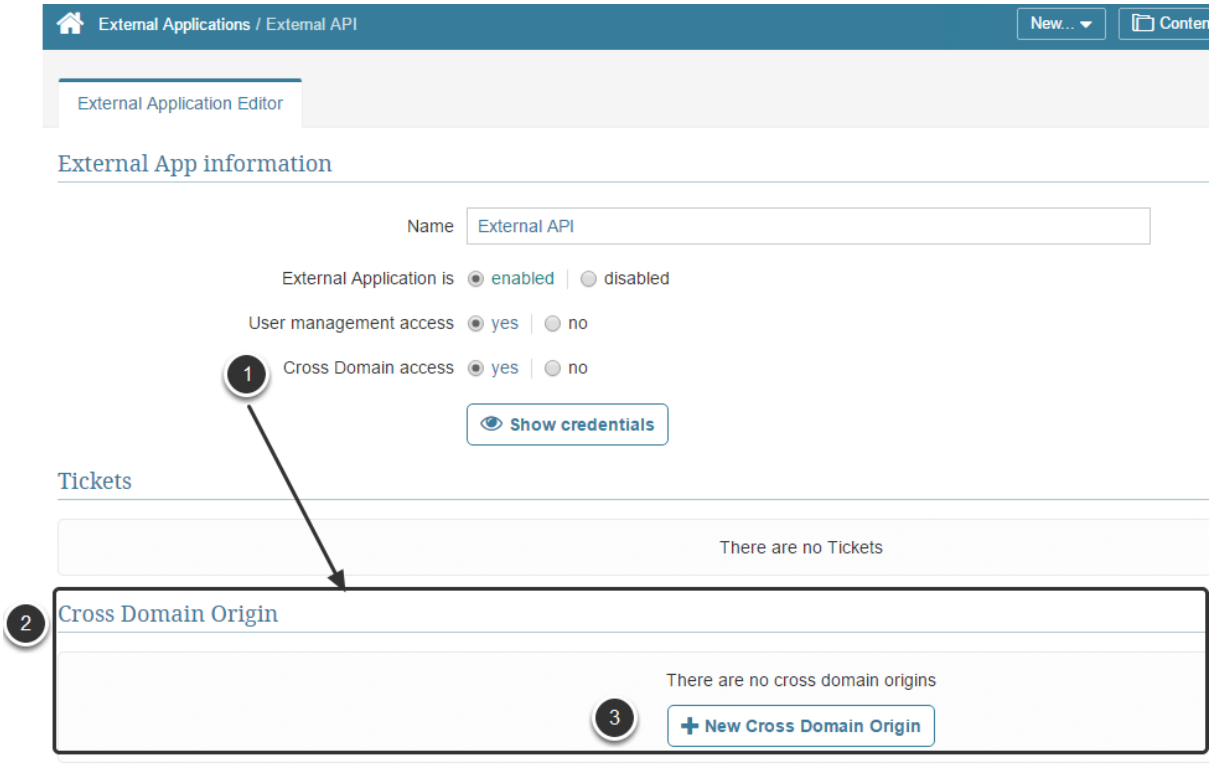
## 3.1. If 'User management access' is inactive




If **User management access** is not enabled (as shown in Step 3), then **user** API calls will fail.

A sample error message returned in the API Toolkit with status code **401 Unauthorized**

# 4. To enable API calls from External Web Sites

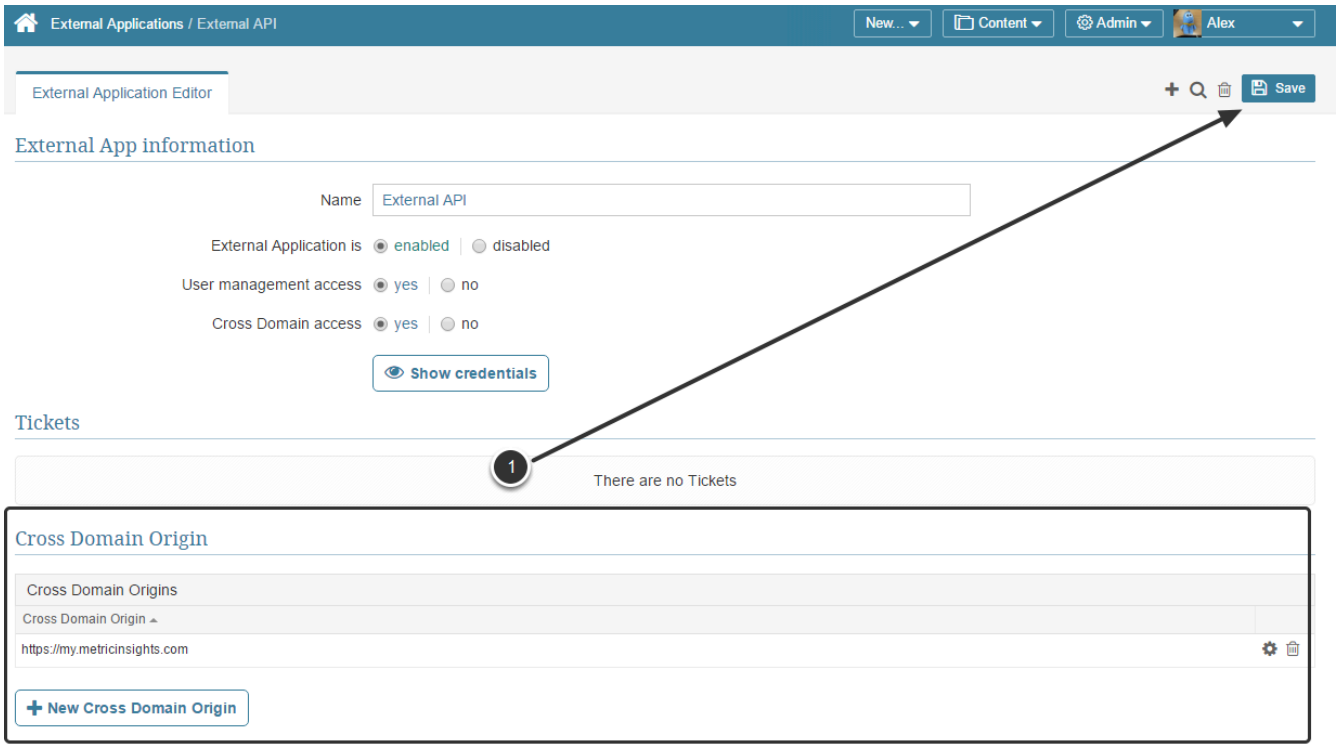



1. Set **Cross Domain Access** to 'yes'

2. Once this option is enabled, a new dynamic field appears on the page.
3. Click **[+ New Cross Domain Origin]** to add specific web sites that will access the API

For more information see the [Overview of Hosting JavaScript charts externally] and [How to Host your own JavaScript charts]

## 4.1. Cross Domain Origins added




1. **Save**

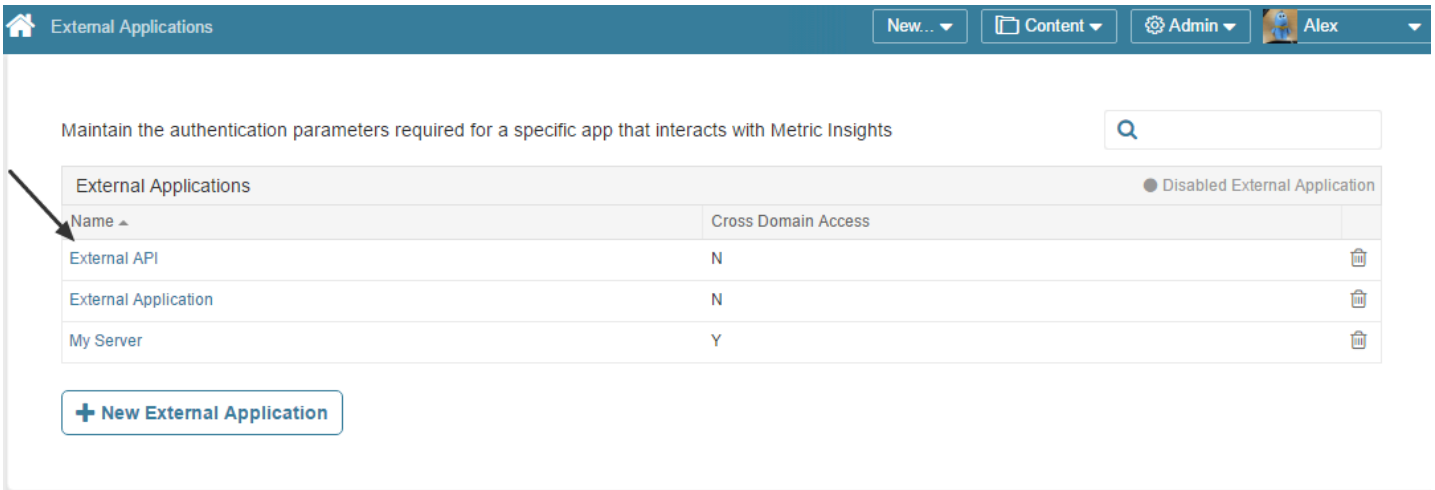
# 2. Test API Access

# 2.1 Test API Access / Get API Token via MI API Test Tool

This article describes how you can quickly test your API access.

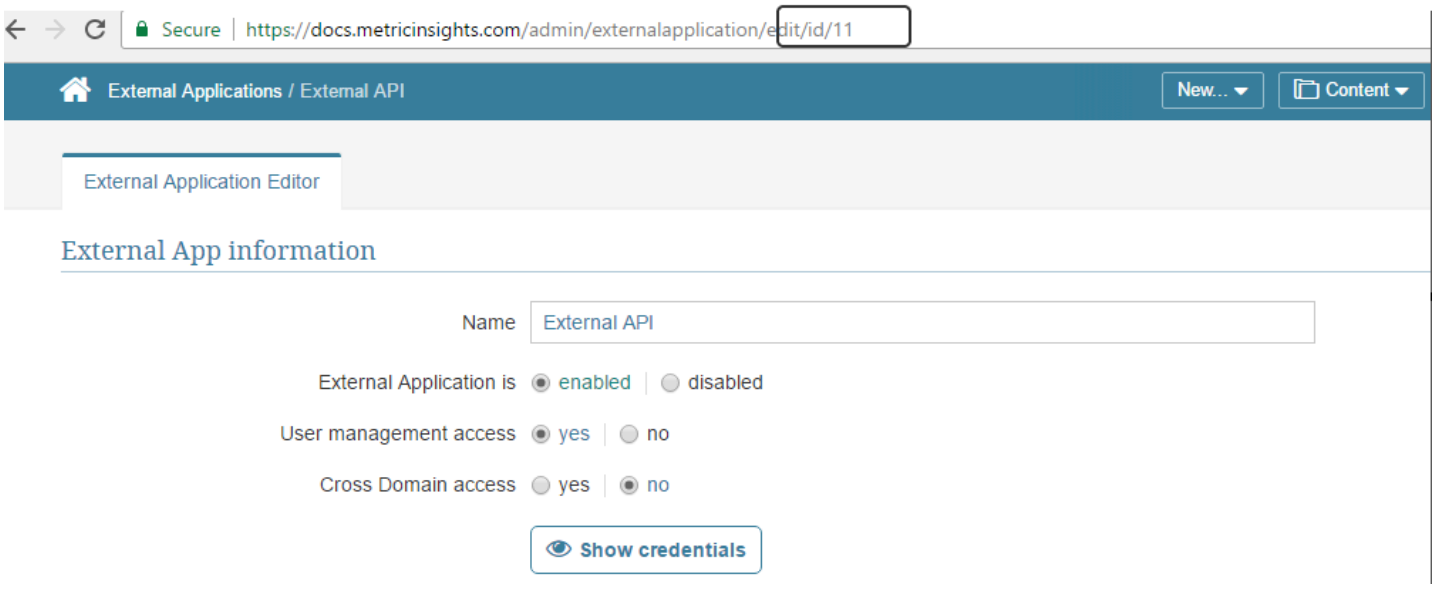
**NOTE:** The format of the *API Test Tool* is slightly different in Version 3.

## 1. Access Admin Menu > Credentials > External Applications




Open the External Application to be tested.

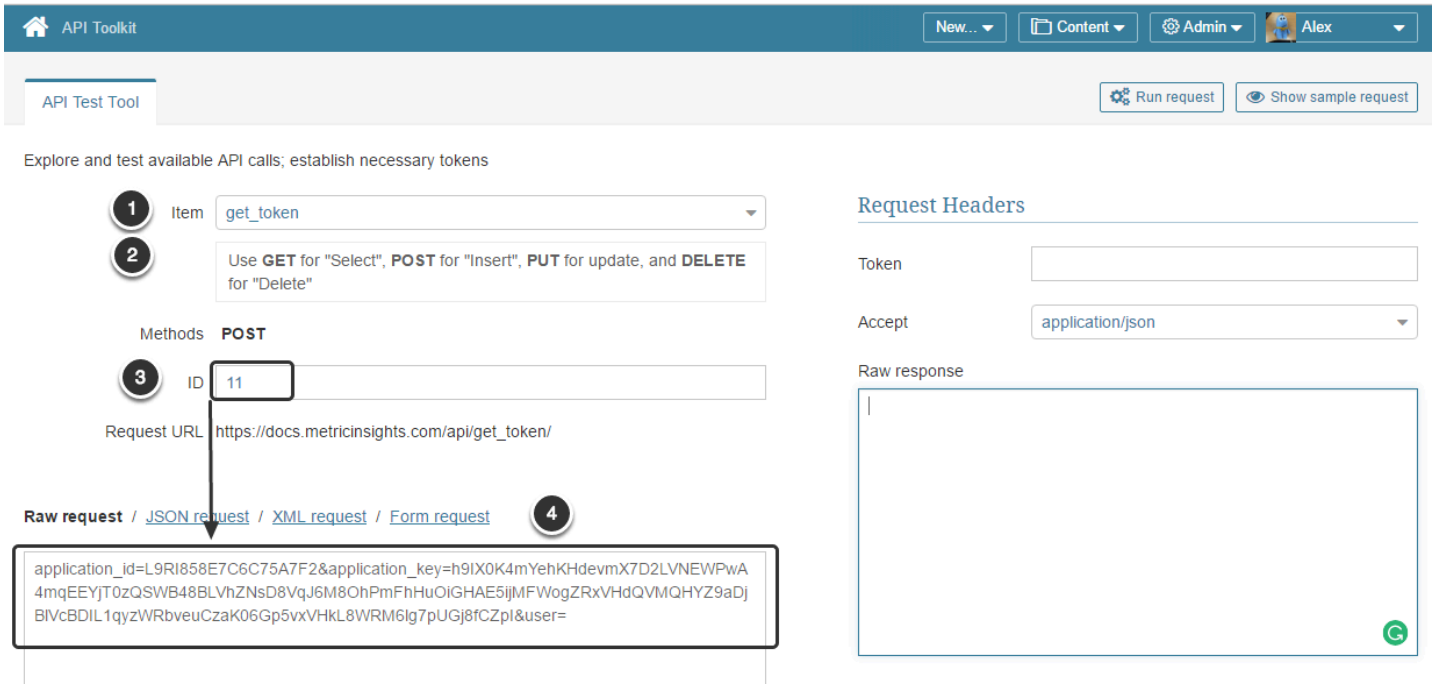
# 2. Review External Application Editor to obtain ID




The **ID value** of your External Application can be obtained from the URL in its editor.

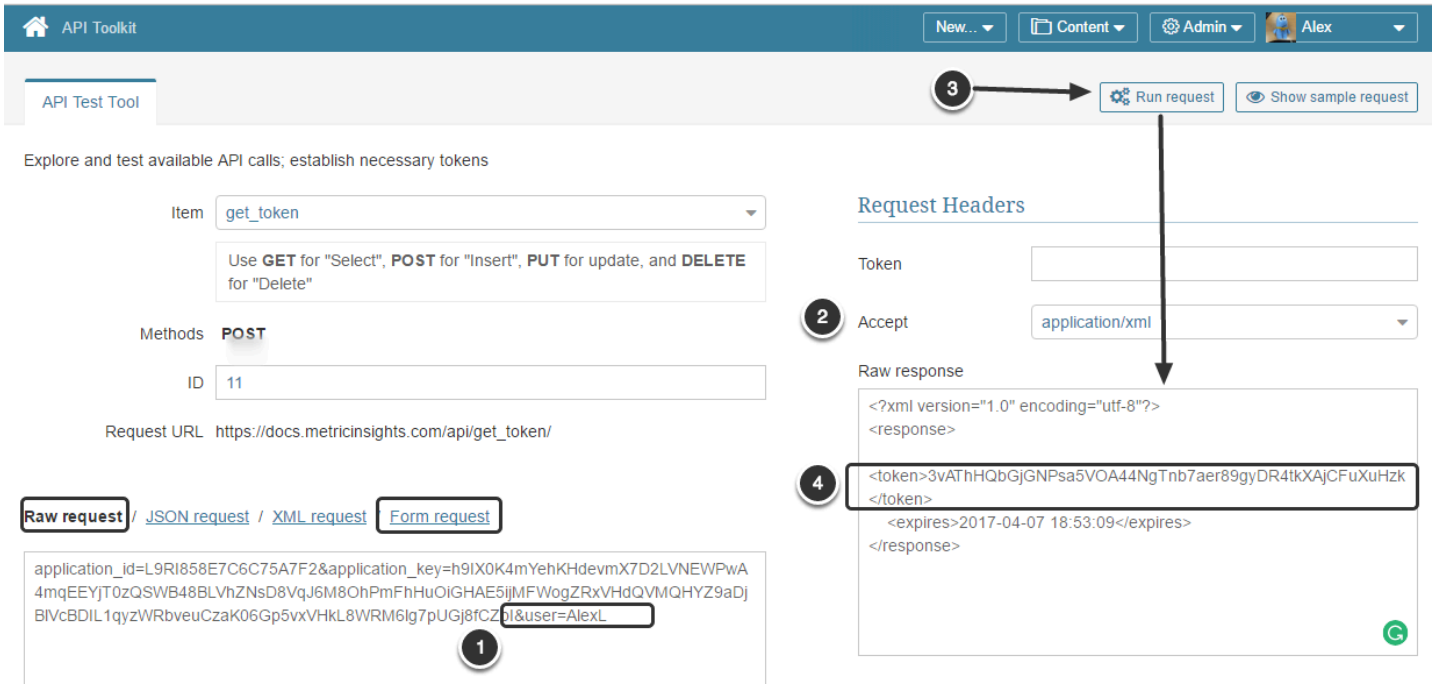
In our example the **ID value** of the 'External API' application is **11**, as seen in the URL of the *External Applications* page.

# 3. Open API Test Tool (access Admin menu > Utilities > API Toolkit)



1. Select the item depending on the request you need to create. This example demonstrates how to obtain token via API request.
2. All Methods available for this item are shown in this field (in our example only the POST method is available)
3. Enter **ID value** of your External Application. In our example the **ID value** of the 'External API' application is **11**, as seen in the URL of the *External Application Editor* page.
4. Once the **ID** is entered, the system automatically generates a **Raw request**

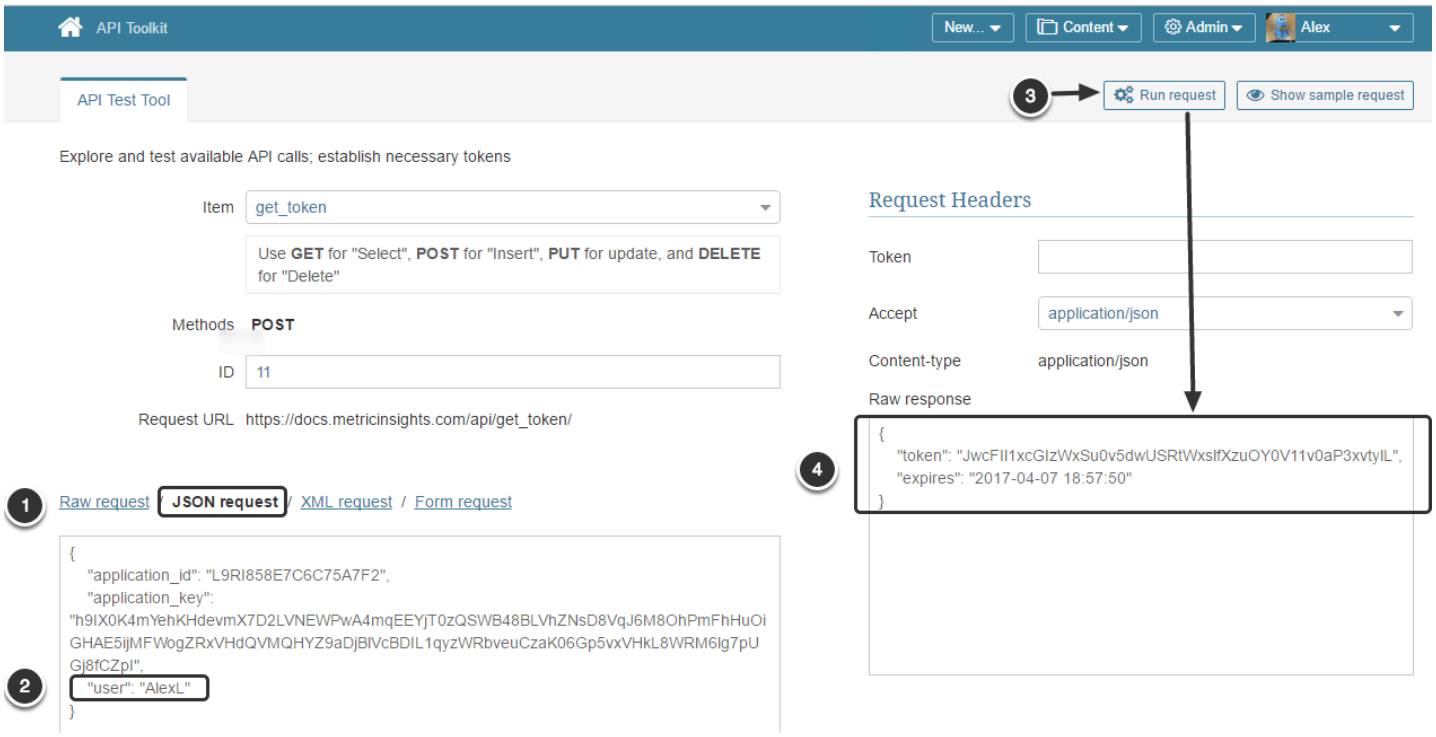
# 4. Complete and Run the Raw request to get API Token




1. Input **Userid** after '**user=**' in the request
2. *Optional.* When performing **Raw request** or **Form Request**, you can define the response format (XML or JSON) in the **Accept** field. **NOTE:** In the **Accept** field, the 'text/csv' setting is only applicable for 'metric_data' and 'report_data' **Items**.
3. Click **Run request** at the top of the page
4. The Token is returned and displayed in the **Raw response** field

**NOTE:** The test is successful since the API Token was obtained.

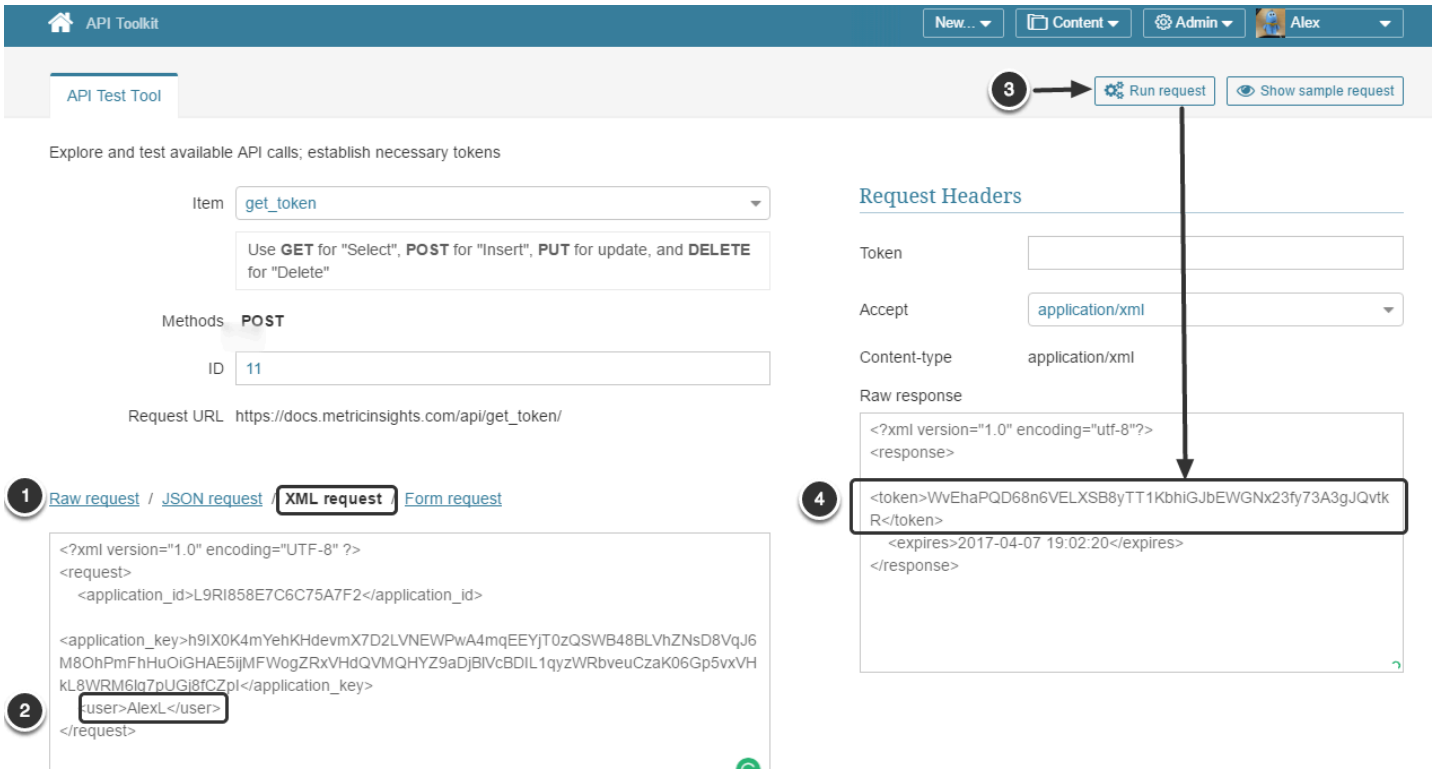
# 5. Get API Token using JSON




Alternately, you can use other formats such as JSON.

1. Select **JSON request** format
2. Both Application ID and the Application Key already provided but you must supply the **user** as previously described
3. **Run Request**
4. The request returns the API Token in **Raw response** field

# 6. Get API Token using XML format




Alternately you can use XML format for the get token request.

1. Select **XML request** format
2. The Application ID and the Application Key already provided but you must supply the **user** as previously described
3. **Run Request**
4. The API Token is returned in **Raw response** field

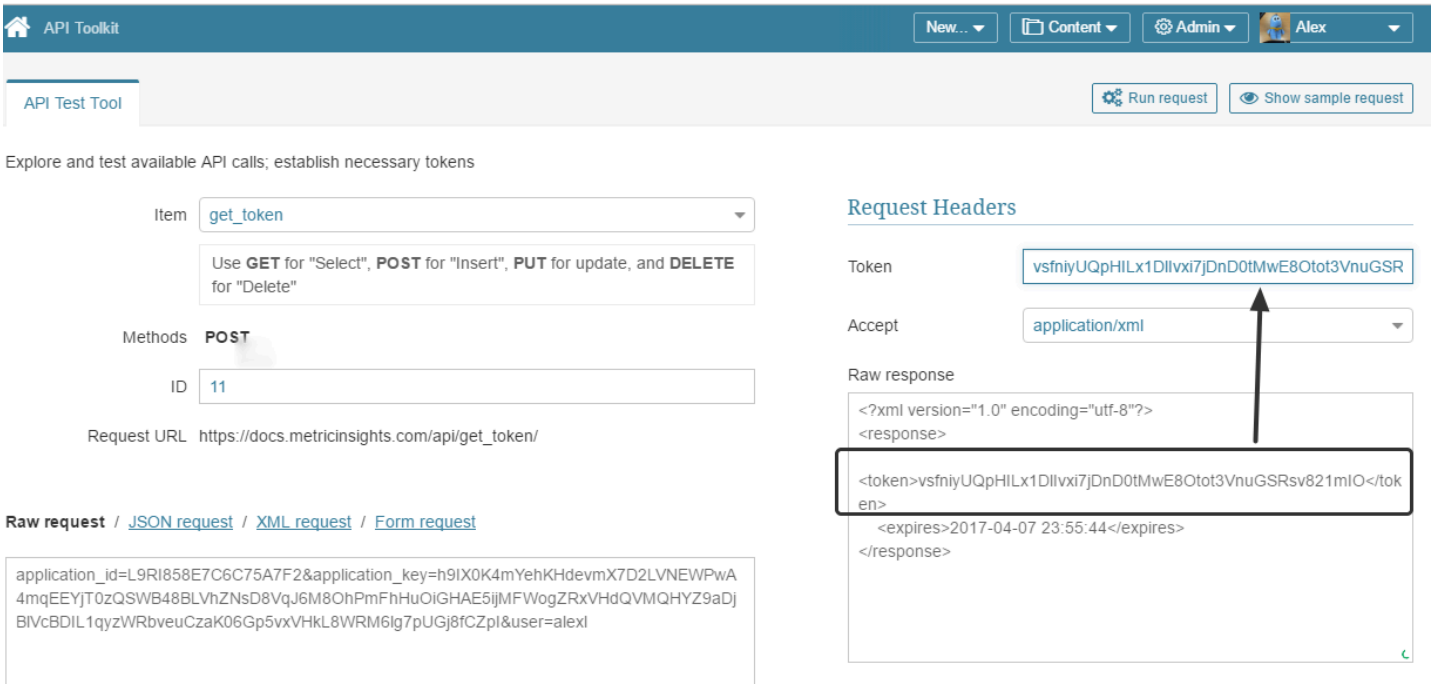
# 2.2 API Test Tool

The API Test Tool can be used to:

- Verify that you have API access
- Discover other API calls available and test each one

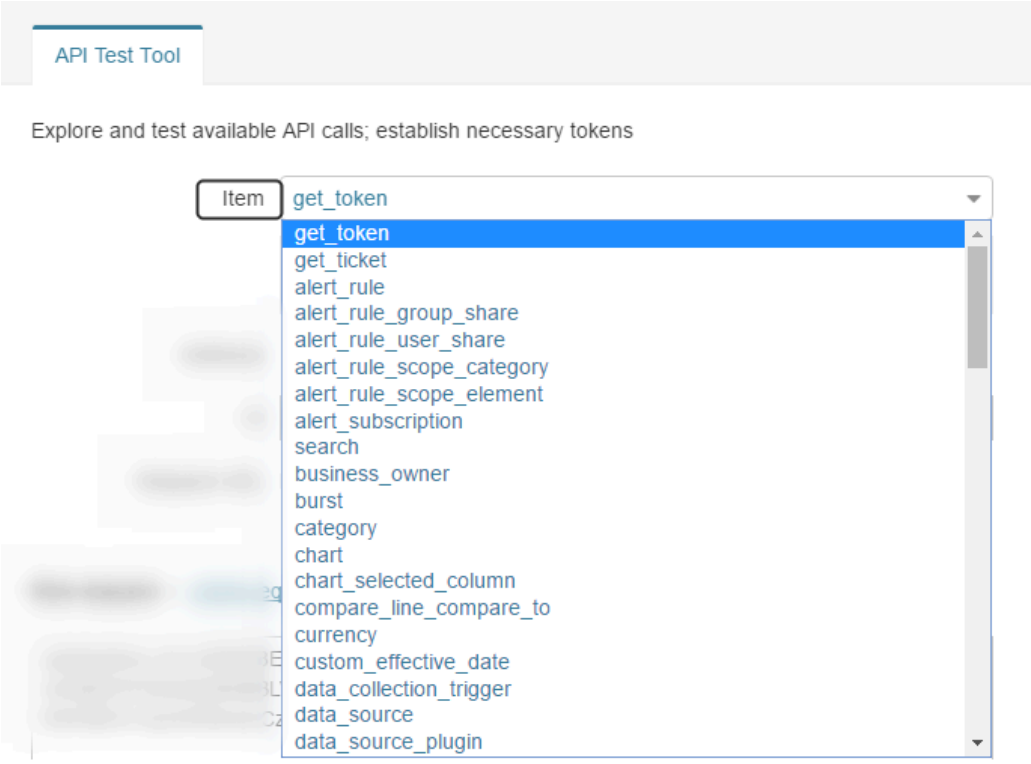
This article describes the process used in one example API call.

## 1. Obtain API token for your session



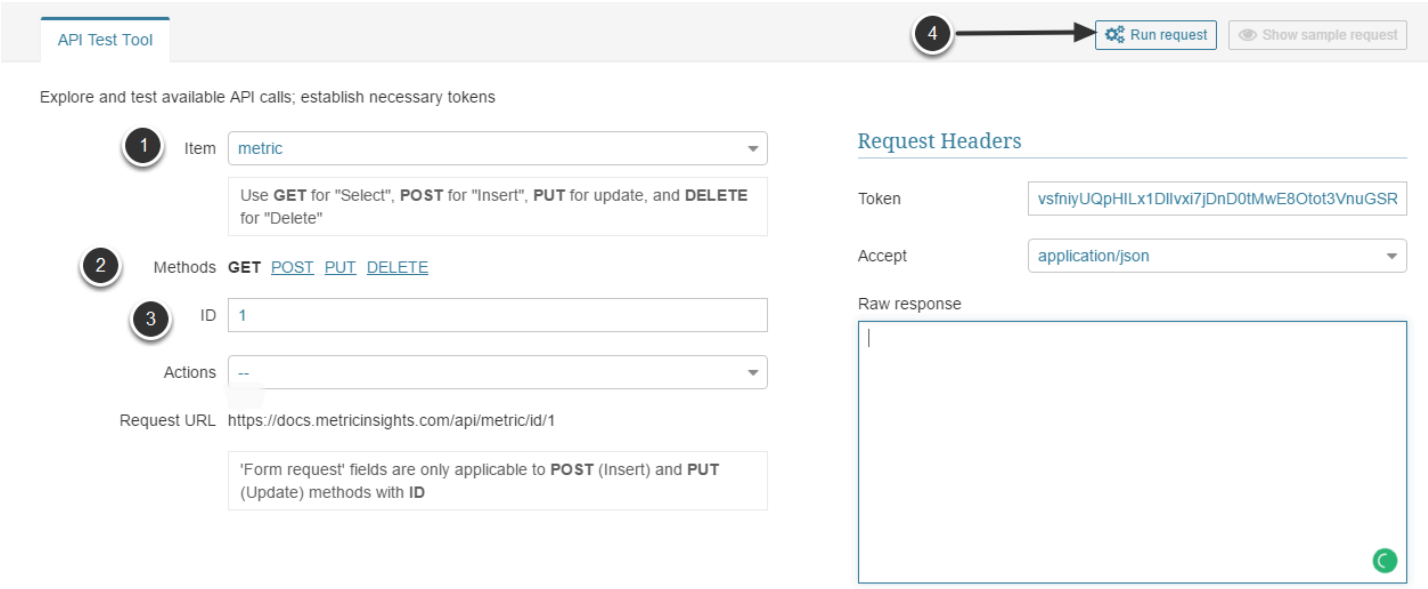

The API Token is required for all subsequent API calls. This token is short lived so you will have to repeat this step to get a new token when the previous token expires.For detailed instruction on getting an API token, refer to this article.

# 2. Explore API calls




The **Item** drop-down list contains the available API calls.

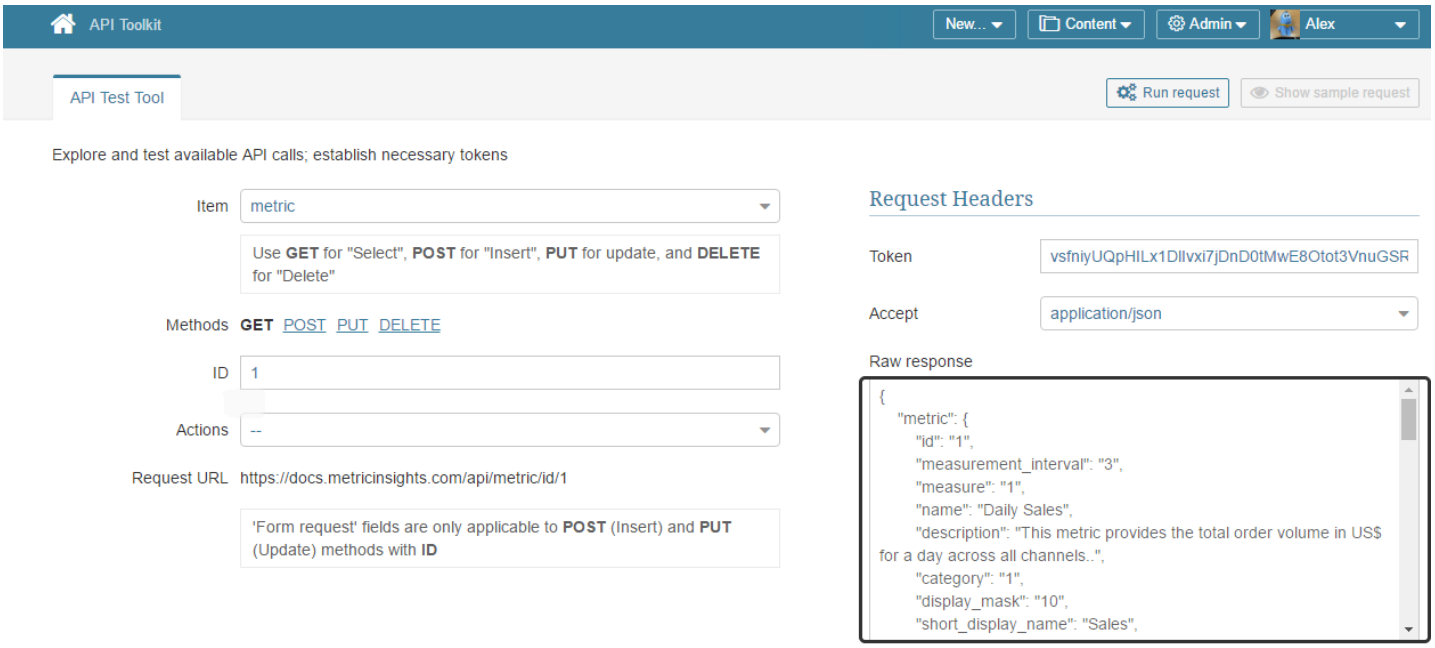
# 3. Choose API Item




1. Choose the API to call. In this example, we choose 'metric' from the drop down list

2. Each API call has a set of available methods. Choose the **Method** in the API call. In this example we choose a **GET** method.
3. Specify the metric **ID**
4. **Run request**

# 4. View results in the 'Response' field



# 2.3 Get Token API via curl

The 'Get Token' call returns a token that is used in all subsequent API calls. Since a token is short-lived, it will be necessary to repeat this process to obtain a new token when the previous token expires.

## Issue the API call

To obtain the secure token, you make a 'get token' API call in which you supply the 'application_id' and 'application_key' generated when you set up API access. You must also supply an existing User value. The token value is then returned.

Below are examples of the API call using ***curl*** at unix command line for JSON (application/json), XML (application/xml), Form (application/x-www-form-urlencoded).

## JSON example

**JSON (Content-Type: application/json)**

Here is an example of JSON request with JSON response:

**Request:**

```
curl --header "Content-Type:application/json" --header "Accept:application/json" -X
POST --data '{"application_id":"ABC","application_key":"XYZ","user":"admin"}'
https://yourinstance.com/api/get_token
```

**Response:**

```
{"token":"UClCUUKxUlkdbhE1cHLz3kyjbIZYVh9eB34A5Q21Y3FPqKGSJs","expires":"2014-02-17
18:46:08"}
```

## XML example

**XML (Content-Type: application/xml)**

Here is an example of XML request with XML response:

**Request:**

```
curl --header "Content-Type:application/xml" --header "Accept:application/xml" -X POST -
```

```
-data
'<request><application_id>ABC</application_id><application_key>XYZ</application_key><use
r>admin</user></request>' https://yourinstance.com/api/get_token
```

**Response:**

```
<?xml version="1.0" encoding="utf-8"?>
<response><token>UClCUUKxUlkdbhE1cHLz3kyjbIZYVh9eB34A5Q21Y3FPqKGSJs</token><expires>2014
-02-17 18:46:08</expires></response>
```

# Form example

### Form (Content-Type: application/x-www-form-urlencoded)

Here is an example of Form request with XML response:

**Request:**

```
curl --header "Content-Type:application/x-www-form-urlencoded" -X POST --data
"application_id=ABC&application_key=XYZ&user=admin" https://yourinstance.com/api/
get_token
```

**Response:**

```
<?xml version="1.0" encoding="utf-8"?>
<response><token>UClCUUKxUlkdbhE1cHLz3kyjbIZYVh9eB34A5Q21Y3FPqKGSJs</token><expires>2014
-02-17 18:46:08</expires></response>
```

Here is an example of Form request with JSON response:

**Request:**

```
curl --header "Content-Type:application/x-www-form-urlencoded" --header
"Accept:application/json" -X POST --data "application_id=ABC&application_key=XYZ&
user=admin" https://yourinstance.com/api/get_token
```

**Response:**

```
{"token":"UClCUUKxUlkdbhE1cHLz3kyjbIZYVh9eB34A5Q21Y3FPqKGSJs","expires":"2014-02-17
18:46:08"}
```

# 2.4 API via curl

You can use curl via the unix command line to submit API requests.

**PRE-REQUISITES:**

Obtain a token via a get token call since all API calls require a token

## Accessing API via curl

Here are some test examples for fetching (GET) data and inserting (POST) data.

Fetch data

```
curl -X GET "https://yourserver.metricinsights.com/api/user_group" -H
"Accept:application/json" -H "Content-Type:application/json" -H
"Token:CaA3cEEkNp8R5k9t1QNHkMxdGRjigzrtYPnLWWZTwyvazbFEtu"
```

Insert data

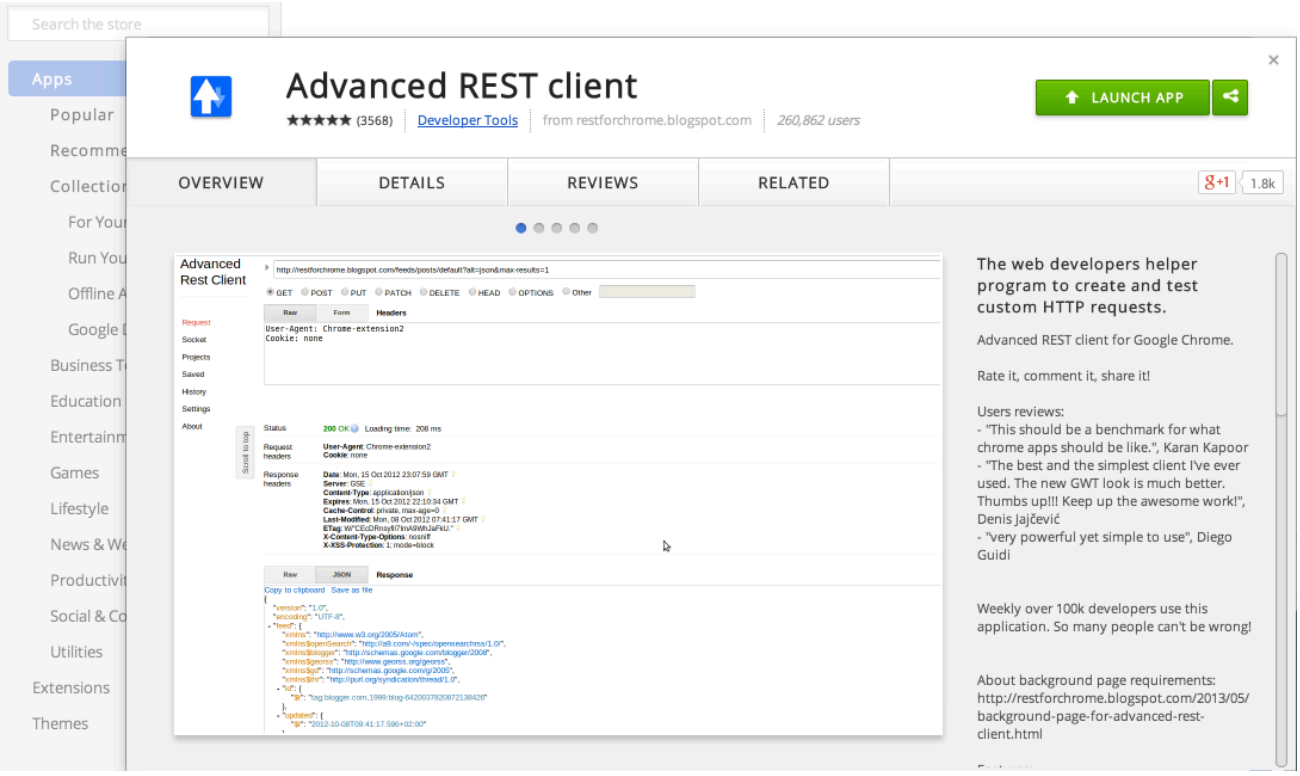
```
curl -H "Accept:application/json" -H "Content-Type:application/json" -H
"Token:CaA3cEEkNp8R5k9t1QNHkMxdGRjigzrtYPnLWWZTwyvazbFEtu" -d '{"name":"group
123","description":"created via API","all_access_group":"No"}' "https://yourserver.
metricinsights.com/api/user_group"
```

# 2.5 Chrome browser REST client

There are a number of client applications that you can use to test calls to the Metric Insights API. Options include the API Tool that comes with the Metric Insights application or use one of your favorites.

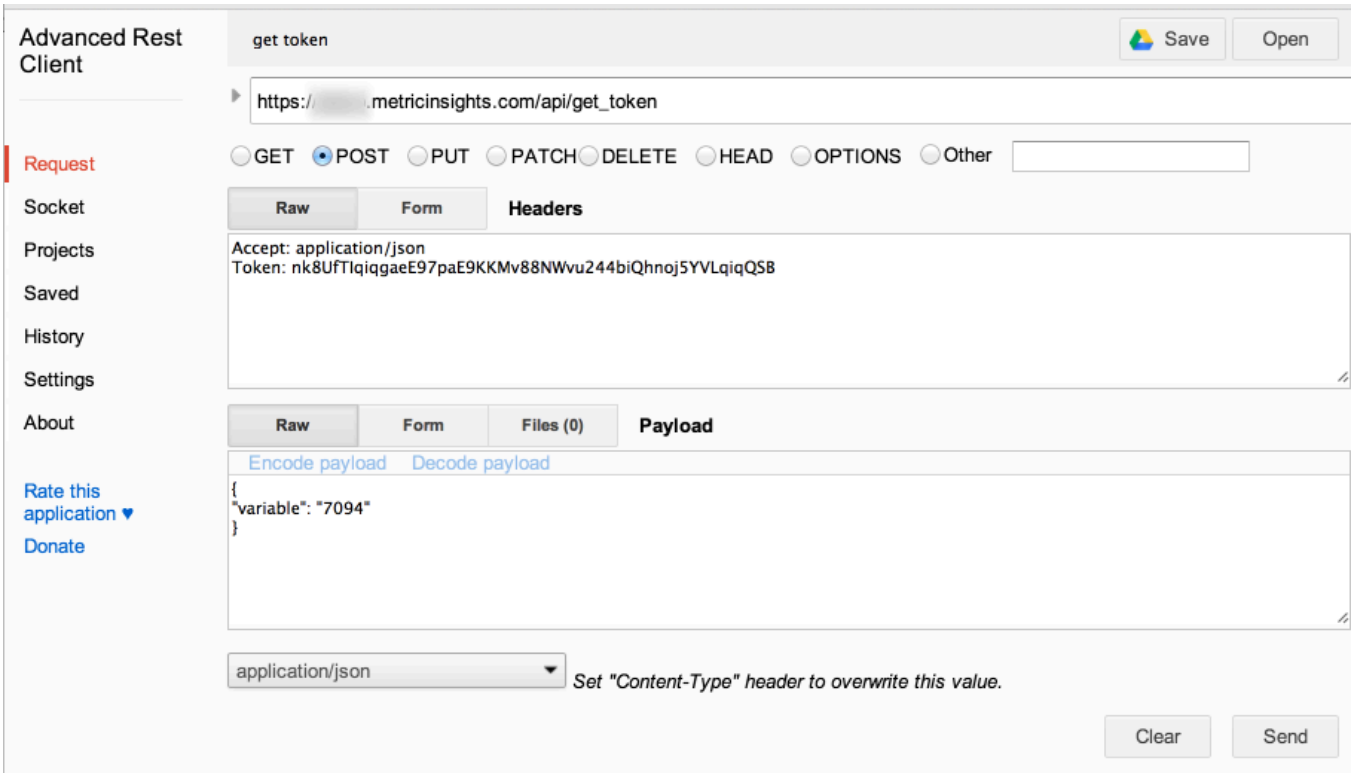
This article gives brief introduction to using the Chrome browser Advanced REST Client.

## 1. Download the Chrome Advance REST Client




1. Search for **Chrome Advanced REST Client**
2. Add it as a plugin to your Chrome browser

# 2. Advanced Rest Client




From this client you can specify

1. URL
2. HTTP Method (GET, POST, ...)
3. HTTP Headers
4. Payload HTTP Body

Typically when testing the Metrics Insight API,  provide the following items:

1. URL you are testing (e.g., /api/get_token or /api/user)
2. GET, POST, PUT, or DELETE
3. Headers such as: Accept: application/json, Token: <your token value>
4. JSON Body for your request if POST or PUT

# 2.6 Python example of using API

The following is an example Python program that illustrates use of the API.

## Python example

You can copy the following code to a file, and run it where Python is installed on your machine:

```
python <filename.py>
```

You will need to fill in your specific environment information in this python file, such as Metric Insights URl, and API credentials

```
#!/opt/mi/.python/bin/python


import requests
import json


url = “<your metric insights URL>/api/get_token/”


values = {
  “application_id”: “<your application ID>“,
  “application_key”: “<your application key>“,
  “user”: “<username>”
}
headers = {“Accept” : “application/json”}


def httpRequestToAPI():
  data = json.dumps(values)
  print ‘Body: ’ + data
  r = requests.post(url,data=values,headers=headers)
  response = (r.text)
  return json.loads(response)


response = httpRequestToAPI()
print response
print ‘Token for use: ’ + response[“token”]
print ‘Expiration : ’ + response[“expires”]
   try:
      response = urllib2.urlopen(req)
```

```
  except urllib2.URLError as e:
      if hasattr(e, 'reason'):
          print 'We failed to reach a server.'
          print 'Reason: ', e.reason
      elif hasattr(e, 'code'):
          print 'The server couldn\'t fulfill the request.'
          print 'Error code: ', e.code
  else:
      # everything is fine


# return the response data
      result = response.read()
      return result

##########################################

# call get token
# TODO fill in with your Metric Insights server:
urlMain = "https://localhost"
#
# TODO fill in with your API credentials
# JSON
values = {'application_id' : '<fill in>',
        'application_key' : '<fill in>',
        'user' : '<fill in>' }

# XML
#values = '<?xml version="1.0" encoding="UTF-8" ?><request><application_id>(fill
in)</application_id><application_key>(fill in)</application_key><user>(fill in)</user></request>'

# JSON
data = json.dumps(values)
# Form data
#data = urllib.urlencode(values)
# XML
#data = values

# JSON
headers = { 'Accept' : 'application/json', 'Content-Type' : 'application/json' }
# Form
#headers = { 'Accept' : 'application/json', 'Content-Type' : 'application/x-www-form-urlencoded' }
# XML
#headers = { 'Accept' : 'application/xml', 'Content-Type' : 'application/xml' }

# returns:
#
```

{"token":"N5WDX010V8jrEh8cNXMQLvlnFGLa9J29qv3msWhVsYzHdi1wbX","expires":"2015-02-19 12:05:25"}

```
#headers = { 'Accept' : 'application/xml' }
# returns:
# <?xml version="1.0" encoding="utf-8"?>
#
<response><token>srhbCkGavdV9I1TNMiO5e33clW79L9BH6WNsRZSIUMmXAiGYGm</token><expire
12:06:06</expires></response>


# Get Token
result = httpRequestToAPI("get_token", data, headers);
print 'result:'
print result

# get returned token value
resultJson = json.loads(result)
print resultJson
token = resultJson["token"]
print 'Token to use: ' + token


# Get Dimensions
headers = { 'Accept' : 'application/json', 'Token' : token }
result = httpRequestToAPI("dimension", None, headers);
print 'result:'
print result
```

# 3. API calls

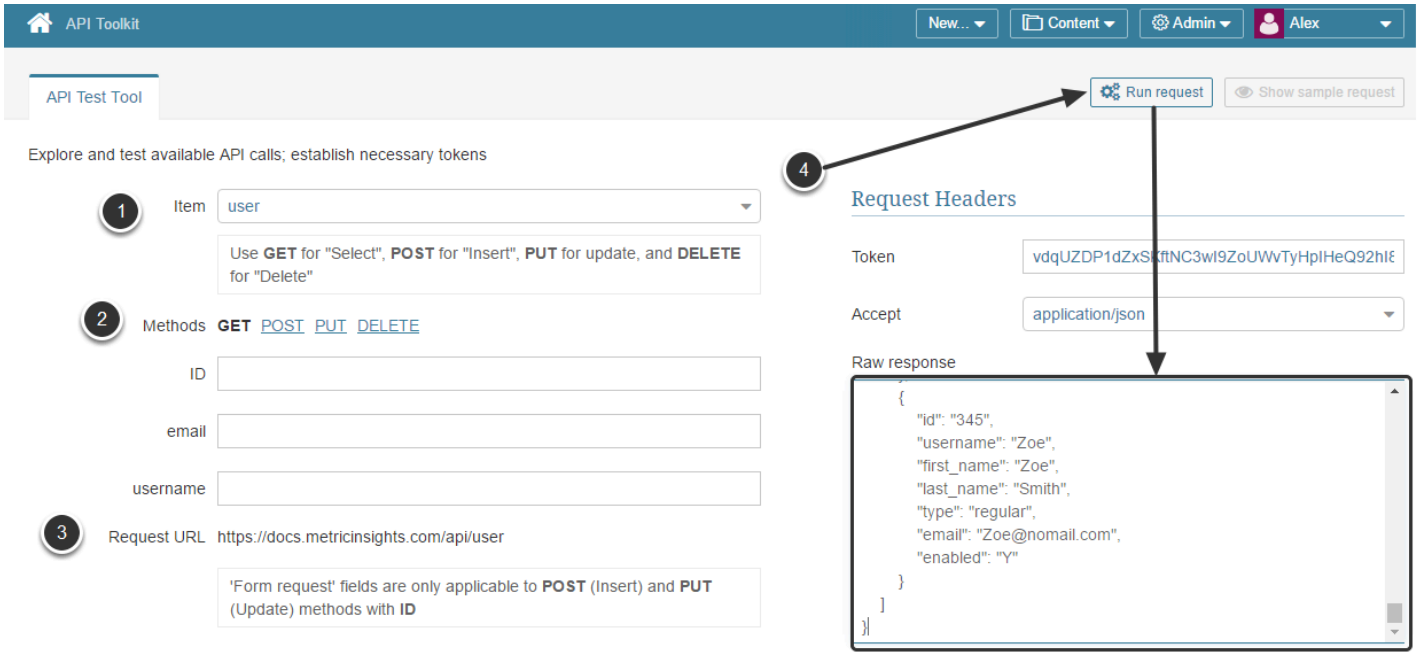
# 3.1 User and Group API

This article provides sample API calls to add **Users** and **Groups** and access to **Dimensions**.

**PRE-REQUISITES:**

- Set up API access. **NOTE:** For the **User** API you must also enable User Management access.
- Verify that you have API access and obtain a token via a since all API calls require a token

## 1. Fetch Users



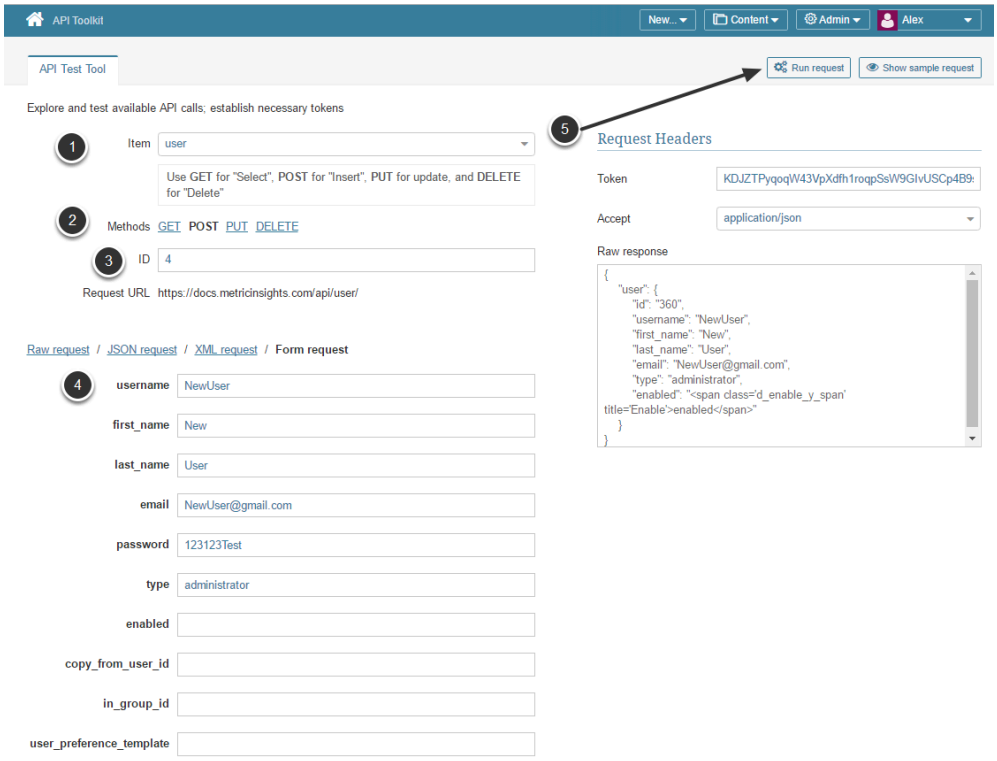

You can fetch the list of all users created in the system.

1. **Item:** user
2. **Method:** GET
3. **Request URL** (*generated automatically*):

```
https://docs.metricinsights.com/api/user
```

4. **Run Request**

# 1.1. Create User




This call creates a new user. After this user is created, Login credentials are going to be sent to a specified email address.

1. **Item:** user
2. **Method:** POST
3. **ID:** Specify ID of any other user previously created in the system. This user's data is going to serve as a basis for creating a new user. User ID can be found at the *User Editor* link (*Admin* > *Users* > select a user):

```
https://docs.metricinsights.com/admin/user/edit/id/4
```

4. Define the following required user info:
   - **username, first_name, last_name, email, password**
   - **type:** regular, power or administrator

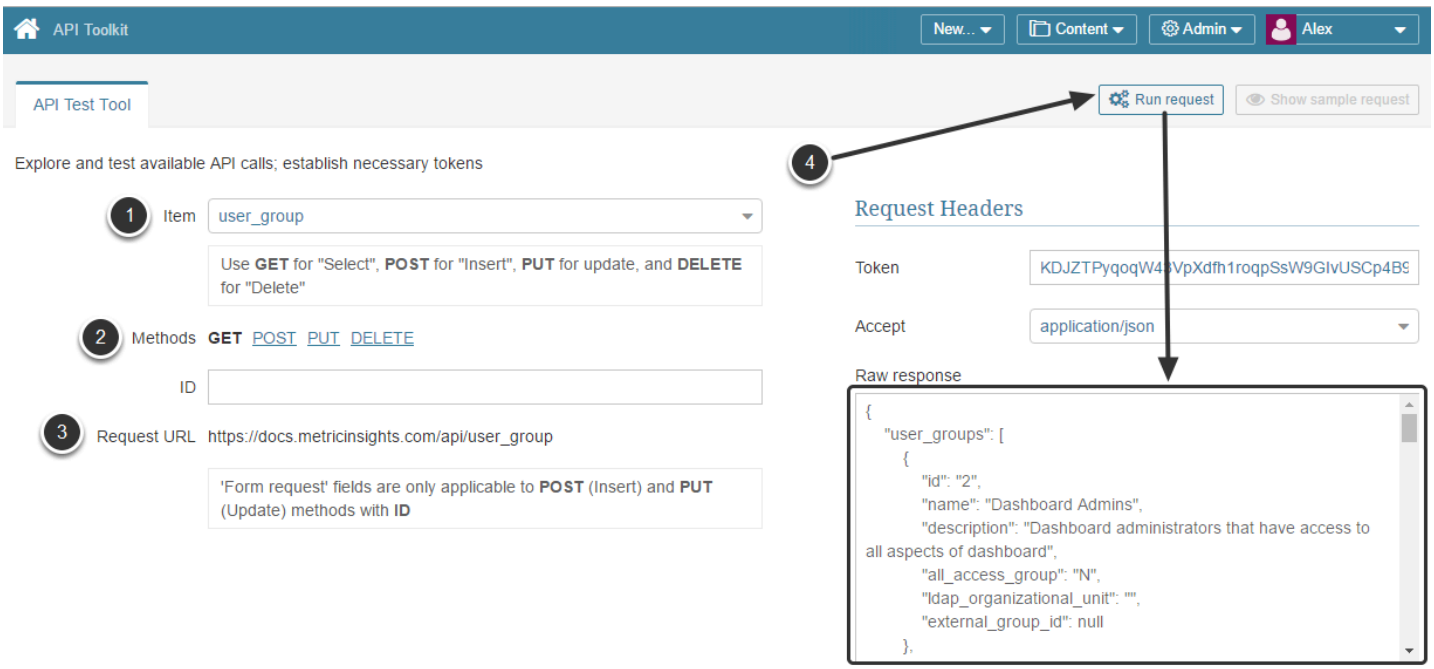
   *Optional*:

- **copy_from_user_id:** Specify ID of a user you want to copy all permissions from.
- **in_group_id**: If this value is not specified, the created user is going to be assigned to a default User Group. If you want to assign a new user to a specific Group instead of a Default group (for example, to the "Finance" group), specify ID of the required Group in this field. Group ID can be found at the *Group Editor* link (*Admin* > *Groups* > select a group):

```
https://docs.metricinsights.com/admin/group/edit/id/2
```

5. **Run Request**

# 2. Fetch Groups



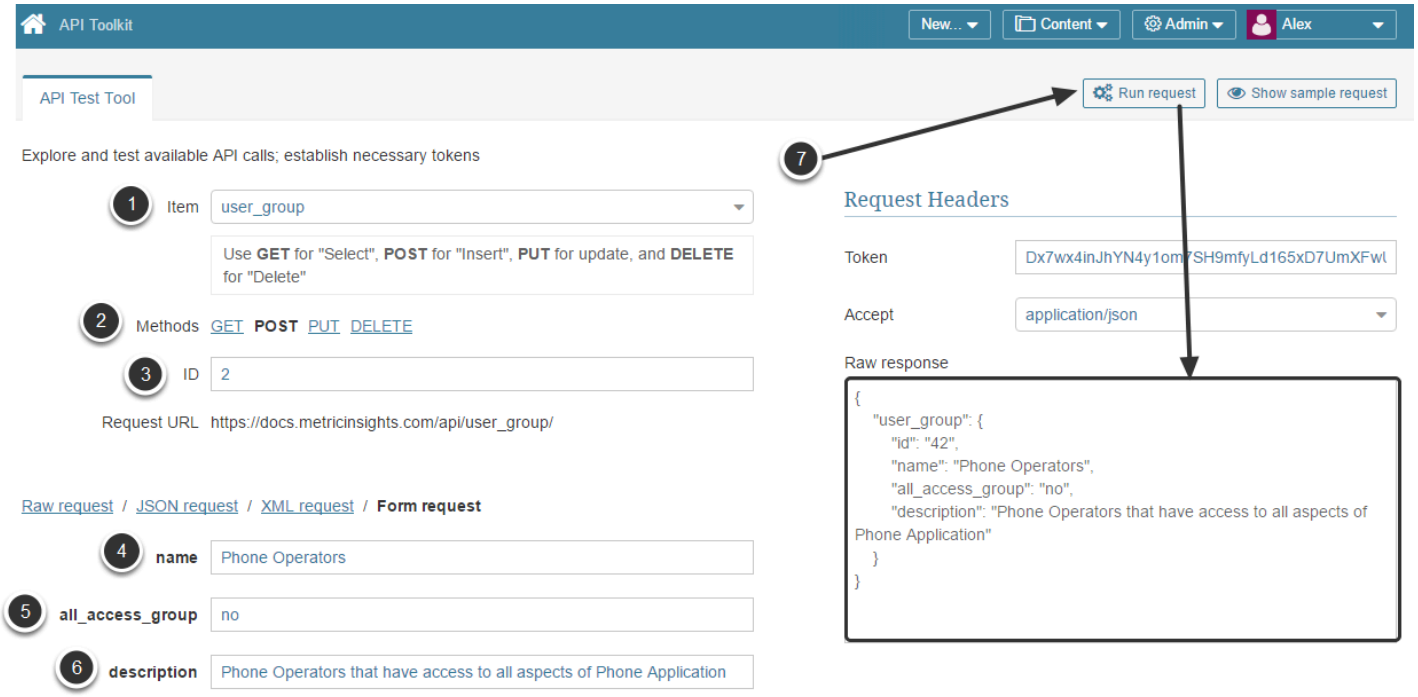

You can fetch the list of all users created in the system.

1. **Item:** user_group
2. **Method:** GET
3. **Request URL** (*generated automatically*):

```
https://docs.metricinsights.com/api/user
```

4. **Run Request**

## 2.1. Create Group



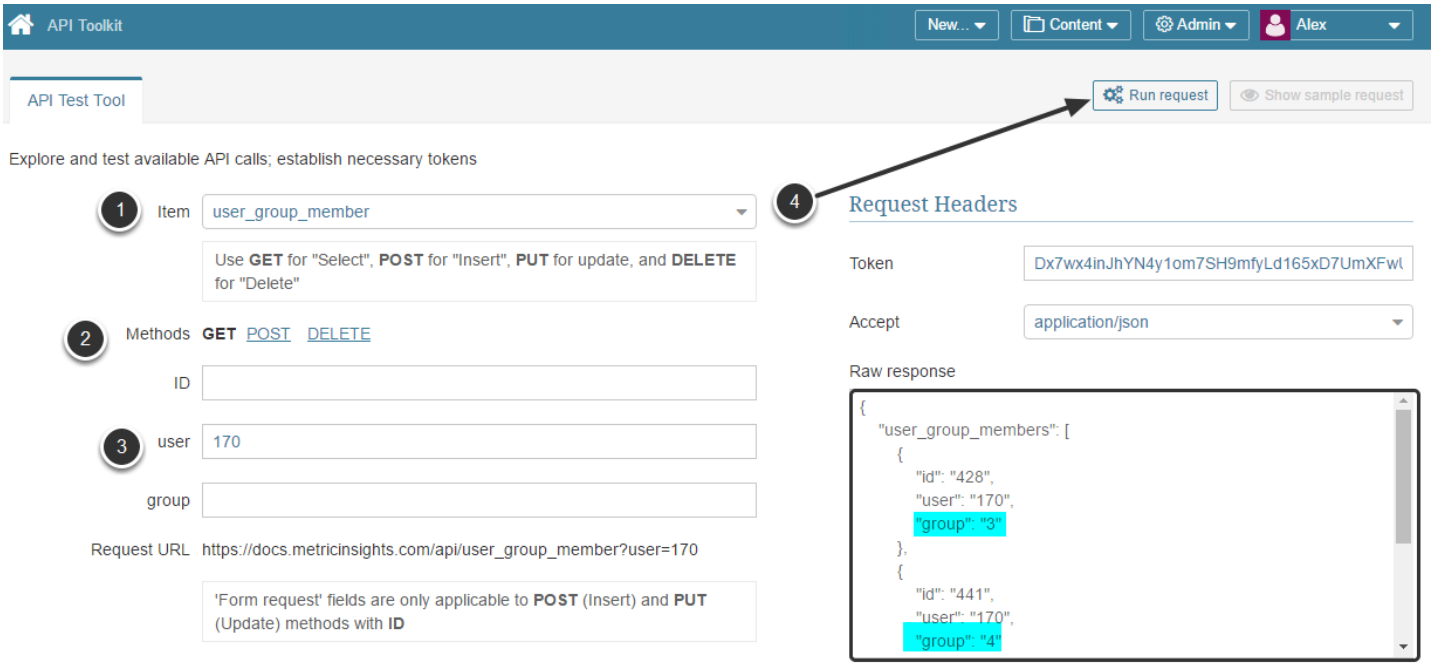

This call creates a new User Group.

1. **Item:** user_group
2. **Method:** POST
3. **ID:** Specify ID of any other User Group previously created in the system. This Group's data is going to serve as a basis for creating a new Group. Group ID can be found at the Group *Editor* link (*Admin* > *Groups* > select a group):

```
https://docs.metricinsights.com/admin/group/edit/id/2
```

4. Provide a meaningful **Group name**
5. **all_access_group:** Specify "yes" / "no". If creating an All Access Group, its members inherit access to all available elements, Dimensions, Dimension Values, Categories and Datasets
6. Optionally, enter a **description** for the Group
7. **Run Request**

# 3. Review User's Group membership




You can get a list of all Groups a certain User is a member of.

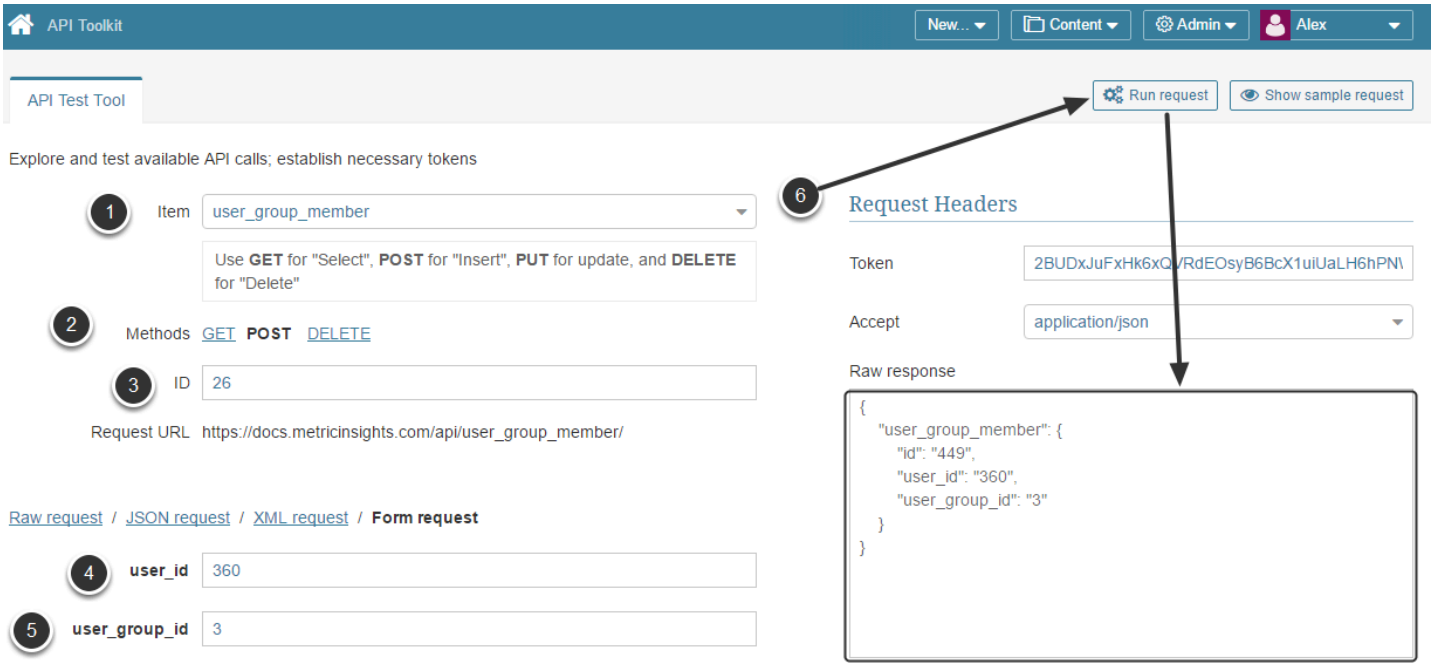
1. **Item:** user_group_member
2. **Method:** GET
3. **user:** Specify ID of a User you want to make a member of a specific Group. User ID can be found at the *User Editor* link (*Admin* > *Users* > select a user):

```
https://docs.metricinsights.com/admin/user/edit/id/170
```

4. **Run Request**

**NOTE:** In this example User belongs to several groups.

## 3.1. Add User to Group




This procedure allows adding a user to a specified group (one user at a time).

1. **Item:** user_group_member
2. **Method:** POST
3. **ID:** Enter the ID of the previously created "user+group" combination. To see the list of all such pairs, choose GET method, leave other fields and Run Request. All "user+group" combinations are going to be displayed in the **Raw response** field. Choose the one that is supposed to serve as a foundation for the new combination.
4. **user_id:** Specify ID of a User you want to make a member of a specific Group. User ID can be found at the *User Editor* link (*Admin* > *Users* > select a user):

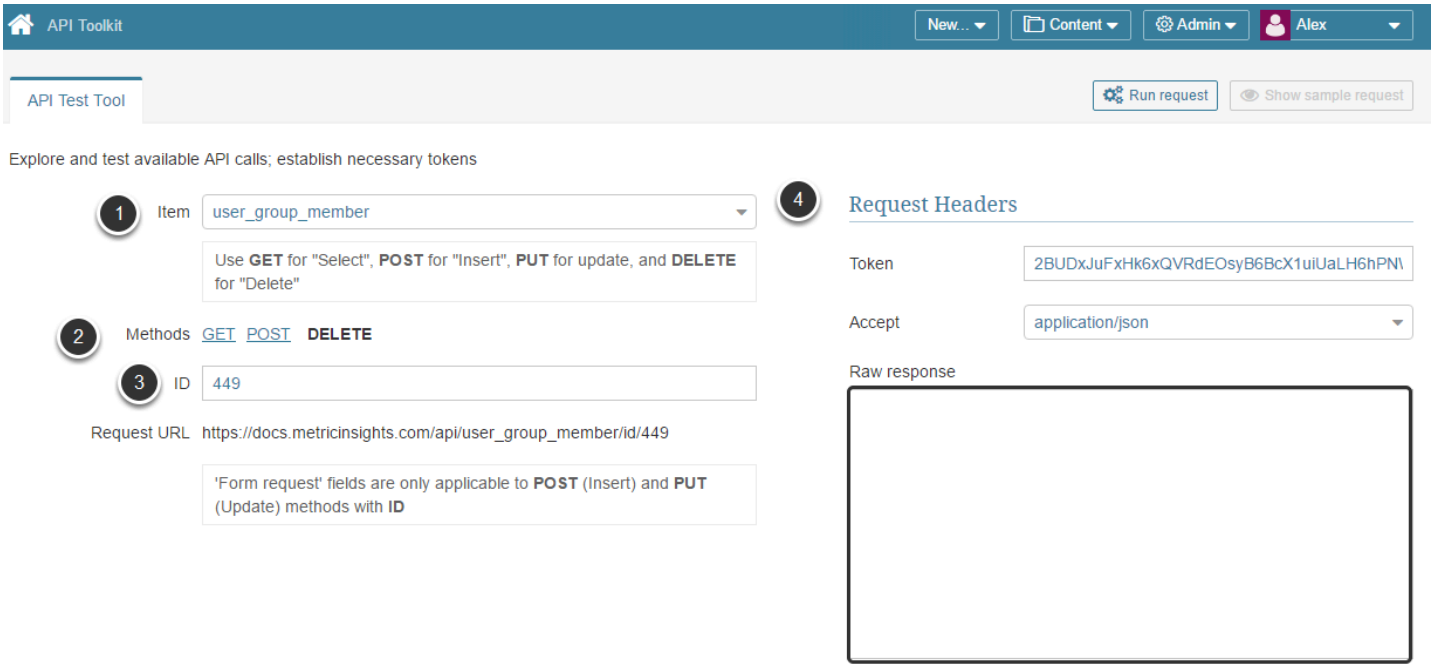
```
https://docs.metricinsights.com/admin/user/edit/id/360
```

5. **user_group_id:** Specify ID of the User Group you want this User to be assigned to. Group ID can be found at the Group *Editor* link (*Admin* > *Groups* > select a user):

```
https://docs.metricinsights.com/admin/group/edit/id/3
```

6. **Run Request**

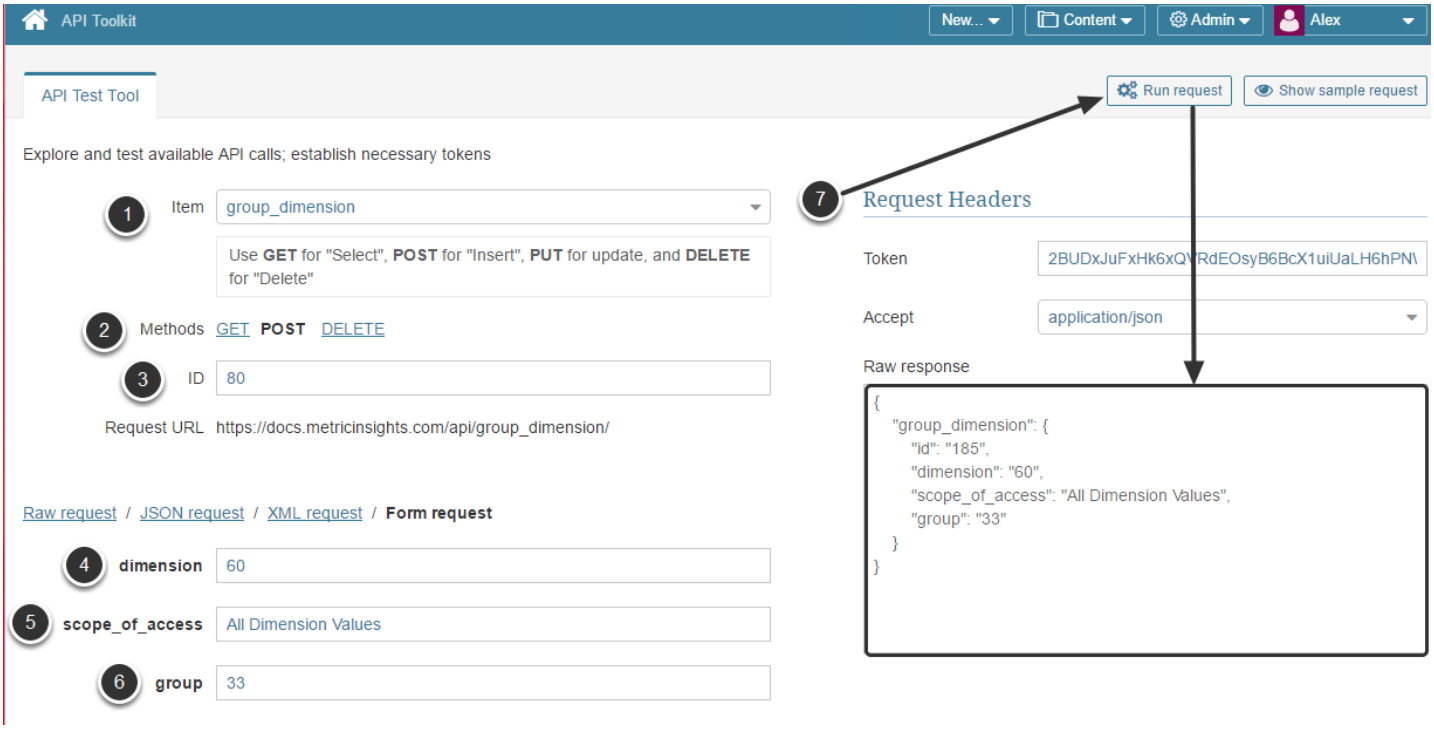
## 3.2. Remove User from the Group




To remove a user from a group, first fetch the Groups that the user is a member of (see Step 3)

1. **Item:** user_group_member
2. **Method:** DELETE
3. **ID:** Enter the ID of the previously created "user+group" combination. To fetch all Groups that the User is a member of, see Step 3 of this article
4. **Run Request**

# 4. Add Dimension access to Group (to all Dimension Values)




This procedure allows adding a user to a specified group (one user at a time).

1. **Item:** group_dimension
2. **Method:** POST
3. **ID:** Enter the ID of the previously created "group+dimension" combination. To see the list of all such pairs, choose GET method, leave other fields empty and **Run Request**. All "group+dimensioned" combinations are going to be displayed in the **Raw response** field. Choose the one that is supposed to serve as a foundation for the new combination. If you want to give all users of a specific group access to all Dimension values, it is better to choose a combination with "All Dimension Access".
4. **dimension:** Specify ID of a Dimension you want to give access to. Dimension ID can be found at the *Dimension Editor* link (*Content* > *Dimensions* > select a Dimension):

```
https://docs.metricinsights.com/editor/segment/edit/segment/60
```

5. **scope_of_access:** All Dimension values
6. **group:** Specify ID of the Group to which access should be given. Group ID can be found at the Group *Editor* link (*Admin* > *Groups* > select a group):

```
https://docs.metricinsights.com/admin/group/edit/id/33
```

7. **Run Request**

## 4.1. Add specific Dimension Value access to Group

API:

```
https://yourinstance.metricinsights.com/api/group_dimension_value
```

Header:

```
Accept:application/json
Token:UClCUUKxUlkdbhE1cHLz3kyjbIZYVh9eB34A5Q21Y3FPqKGSJs
```

POST body payload:

```
{
"group": "14",
"dimension": "55",
"dimension_value": "7094"
}
```

Response:

```
{"id":"2","dimension_value":"7094","dimension":"55","group":"14"}
```

# 3.2 Report Data API

This article provides sample API calls to manipulate Reports.

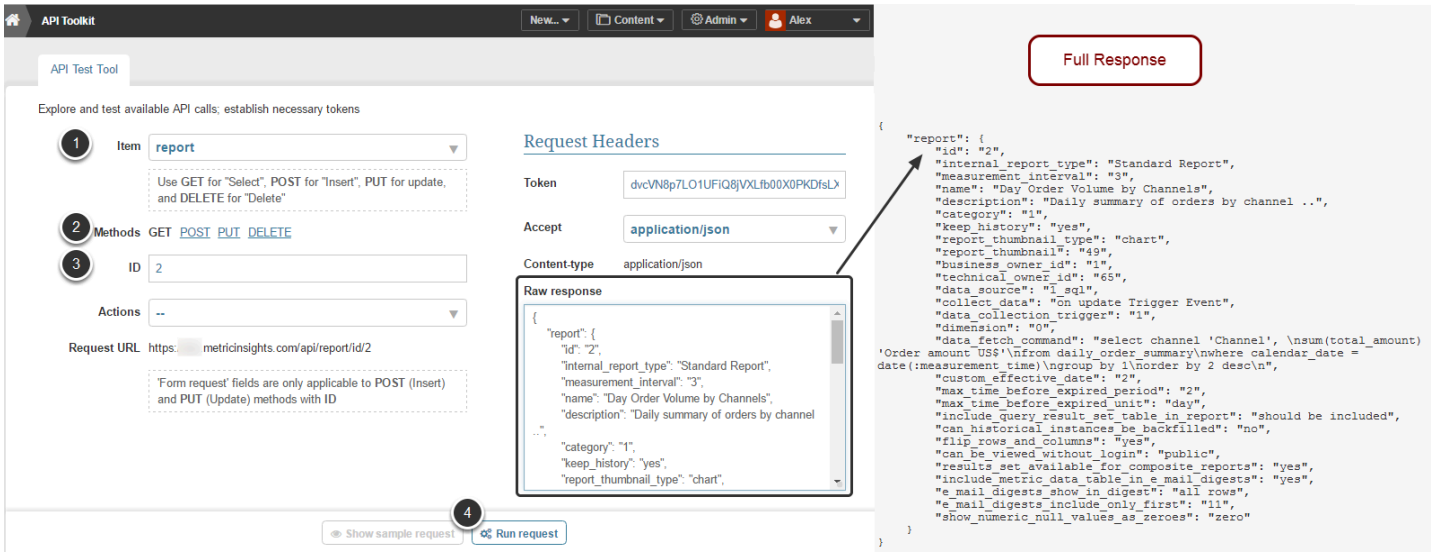
**PREREQUISITES:**

1. Set up API access
2. Verify that you have API access
3. Obtain a token via a get token call since all API calls require a token

## Access Report data for JavaScript charting

See the Overview of externally hosting JavaScript charts and Details of externally hosting JavaScript charts.

## Get Report Information



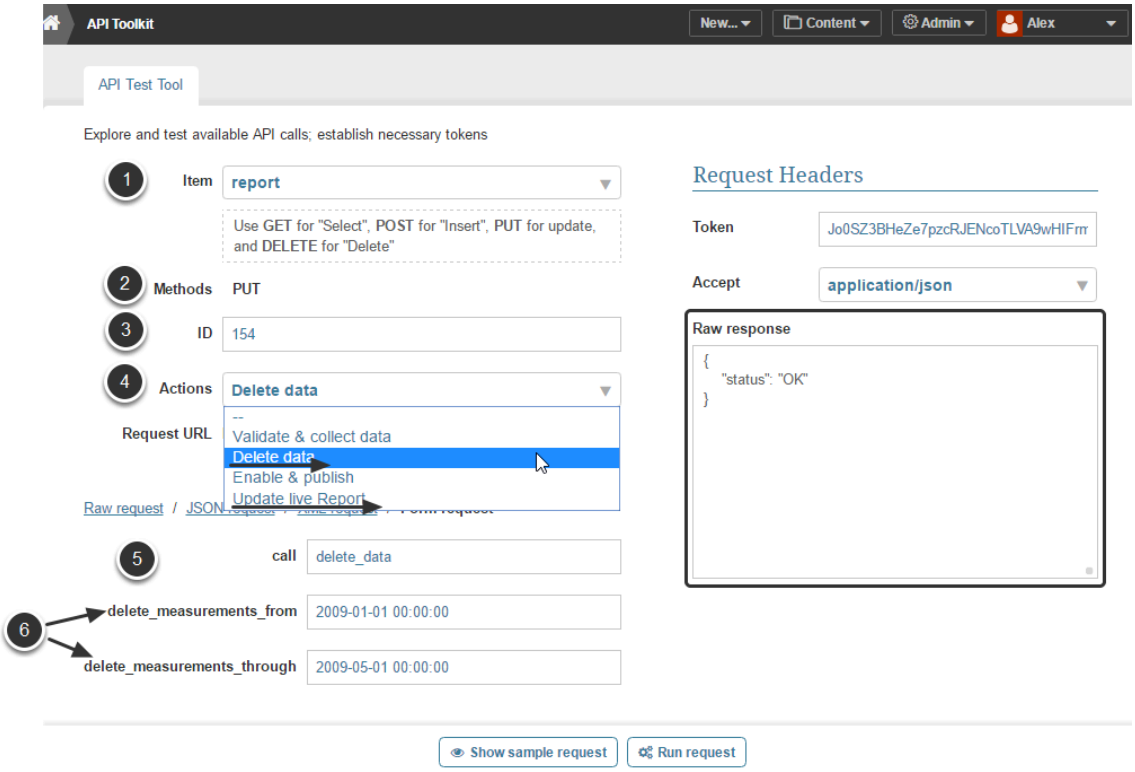

You can obtain all the information about a specific Metric (such as measurement inteval, report type, name, description, category, whether it is a change report, etc.)

1. **Item:** report
2. **Method:** GET
3. **ID:** Specify the ID of a Report you wish to get information about. Report ID can be determined in the web browser URL of the *Report Editor*. For example:

```
https://demo.metricinsights.com/editor/report/edit/element/2
```

4. **Run request**

# Set time range ('from_measurement_time' and 'to_measurement_time' values)



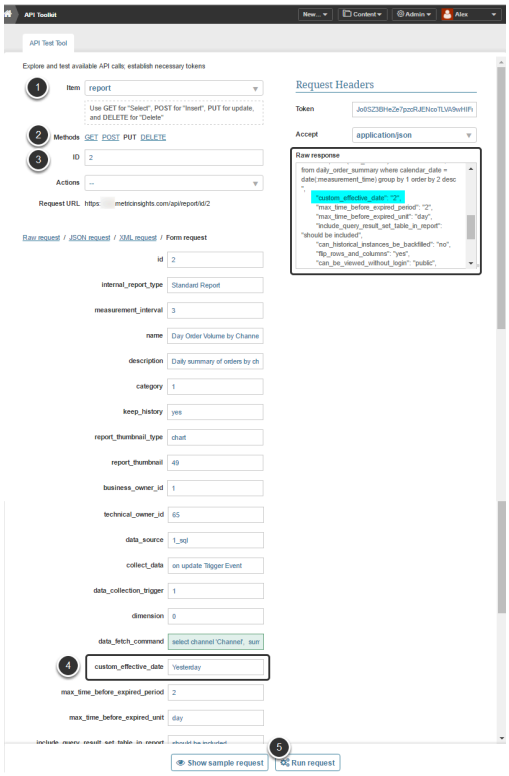

1. **Item:** report
2. **Method:** PUT
3. **ID:** Specify the ID of a Report you wish to modify. Report ID can be determined in the web browser URL of the *Report Editor*. For example:

```
https://demo.metricinsights.com/editor/report/edit/element/154
```

4. **Actions:** Select an action you wish to perform. **NOTE:** Specific time range is defined for "Delete data" and "Update live Report"
5. **call**: This field is going to be automatically populated based on the action you have chosen at the field above
6. Specify the time range by defining the following parameters:
   - **from_measurement_time -** (*measurement time should be in one of the following formats: yyyy-mm-dd or yyyy-mm-dd 00:00:00*) - date from which measurements will be updated
   - **to_measurement_time**- (*measurement time should be in one of the following formats: yyyy-mm-dd or yyyy-mm-dd 00:00:00*) - date to which measurements will be updated

7. **Run request**

# Change ':measurement_time' value



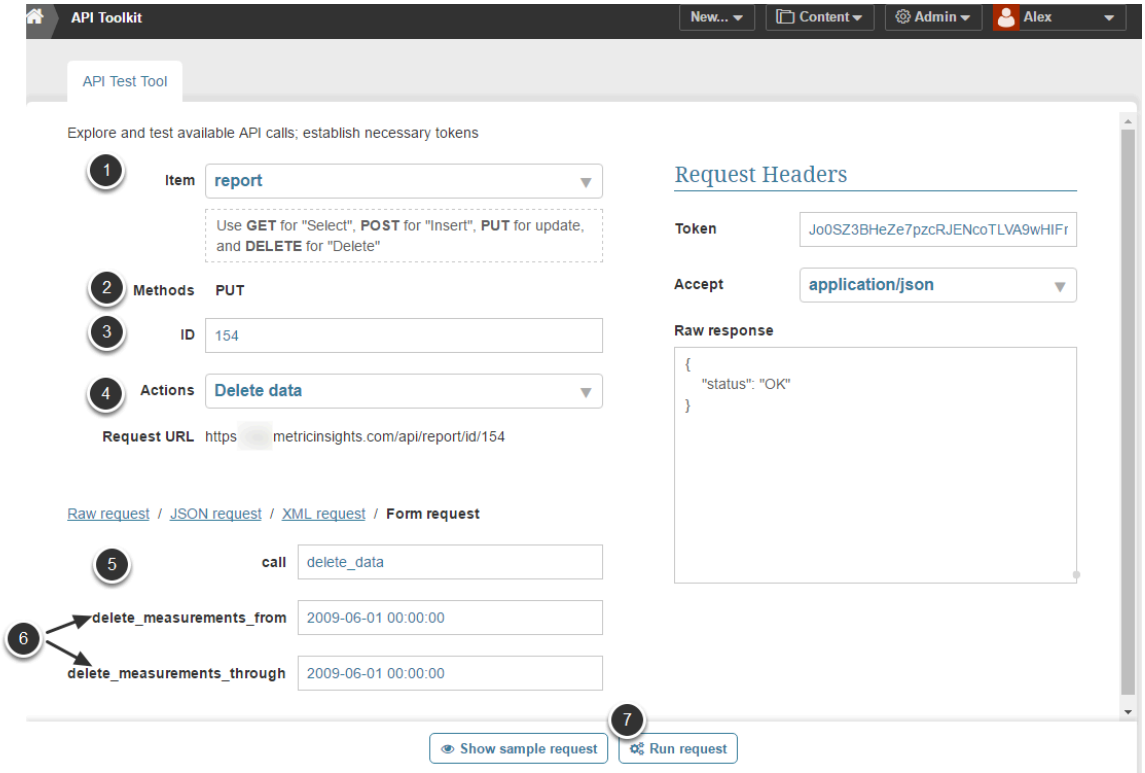

To learn more about the :measurement_time bind variable, [follow this link](#).

1. **Item:** report
2. **Method:** PUT
3. **ID:** Specify the ID of a Report you wish to modify. Report ID can be determined in the web browser URL of the *Report Editor*. For example:

```
https://demo.metricinsights.com/editor/report/edit/element/2
```

4. **custom_effective_date:** (*Value or id of a period. Sample correlation :Last week = 1, Yesterday = 2, Today =7*) - sets ':measurement_time' value
5. **Run request**

# Delete measurements for a certain Time Range




1. **Item:** report
2. **Method:** PUT
3. **ID:** Specify the ID of a Report you wish to modify. Report ID can be determined in the web browser URL of the *Report Editor*. For example:

```
https://demo.metricinsights.com/editor/report/edit/element/154
```

4. **Actions:** "Delete data"
5. **call**: This field is automatically populated based on the action you have chosen at the field above
6. Specify the time range by defining the following parameters:
   - **from_measurement_time -** (*measurement time should be in one of the following formats: yyyy-mm-dd or yyyy-mm-dd 00:00:00*) - date from which measurements will be updated
   - **to_measurement_time**- (*measurement time should be in one of the following formats: yyyy-mm-dd or yyyy-mm-dd 00:00:00*) - date to which measurements will be updated
7. **Run request**

# Get Report's Historical Instance IDs

API:

```
https://yourinstance.metricinsights.com/api/report_instance?element=<element_id>&
dimension_value=<dimension_value_id>
```

Parameters:

- **element** (*numeric value*) - element ID
- **dimension_value** (*numeric value*) - Dimension Value ID (for dimensioned elements only)

Example:

```
Undimensioned Report - https://yourinstance.metricinsights.com/api/
report_instance?element=81
Dimensioned Report - https://yourinstance.metricinsights.com/api/
report_instance?element=81&dimension_value=6
```

Header:

```
Accept:application/json
Token:UClCUUKxUlkdbhE1cHLz3kyjbIZYVh9eB34A5Q21Y3FPqKGSJs
```

Request example :

```
{
"call": "generate",
"from_measurement_time": "2016-04-24",
"to_measurement_time": "2016-04-25"
}
```

Response (without dimension value specified):

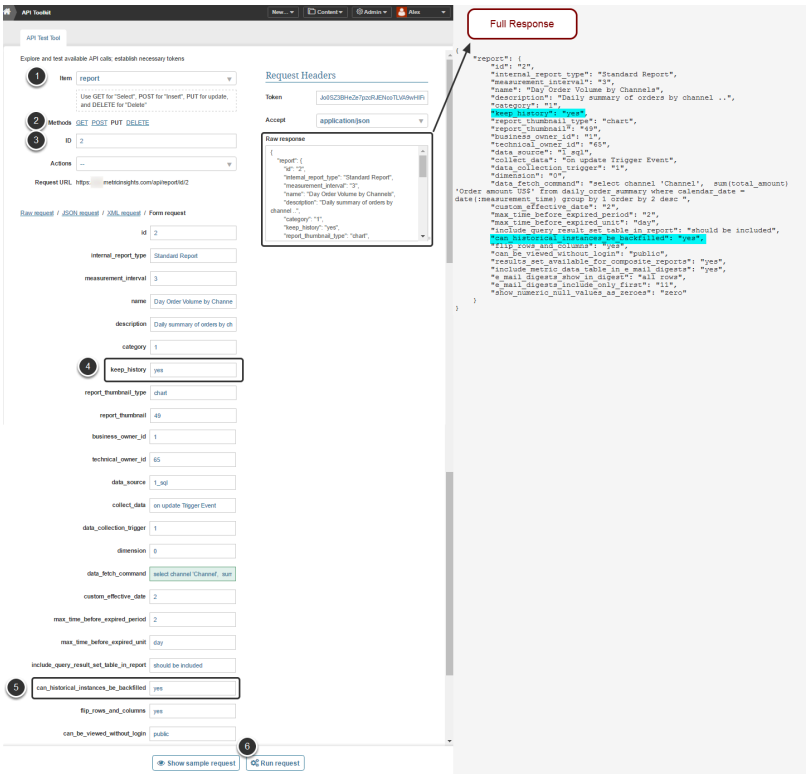
```
{
        "report_instances": [
            {
                "id": "915",
                "element": "81",
                "dimension_value": "5",
                "measurement_time": "2014-01-01 00:00:00"
            },
            {
                "id": "918",
                "element": "81",
                "dimension_value": "8",
                "measurement_time": "2014-01-01 00:00:00"
            },

    ]
}
```

Response (without dimension value specified):

```
{
        "report_instances": [
            {
                "id": "916",
                "element": "81",
                "dimension_value": "6",
                "measurement_time": "2014-01-01 00:00:00"
            },
            {
                "id": "804",
                "element": "81",
                "dimension_value": "6",
                "measurement_time": "2014-02-01 00:00:00"
            },
        ]
}
```

# Set historical instances to be backfilled




1. **Item:** report
2. **Method:** PUT
3. **ID:** Specify the ID of a Report you wish to modify. Report ID can be determined in the web browser URL of the *Report Editor*. For example:

```
https://demo.metricinsights.com/editor/report/edit/element/2
```

4. **keep_history:** (*yes, no*) - When this option is set to 'yes', the Snapshot reporting is enabled (an instance for every data collection run is saved)
5. **can_historical_instances_be_backfilled:** *(yes, no*) - Define whether historical instances of a snapshot report be backfilled when data is recollected. Backfilling is performed within the ':measurement_time' parameter time range
6. **Run request**

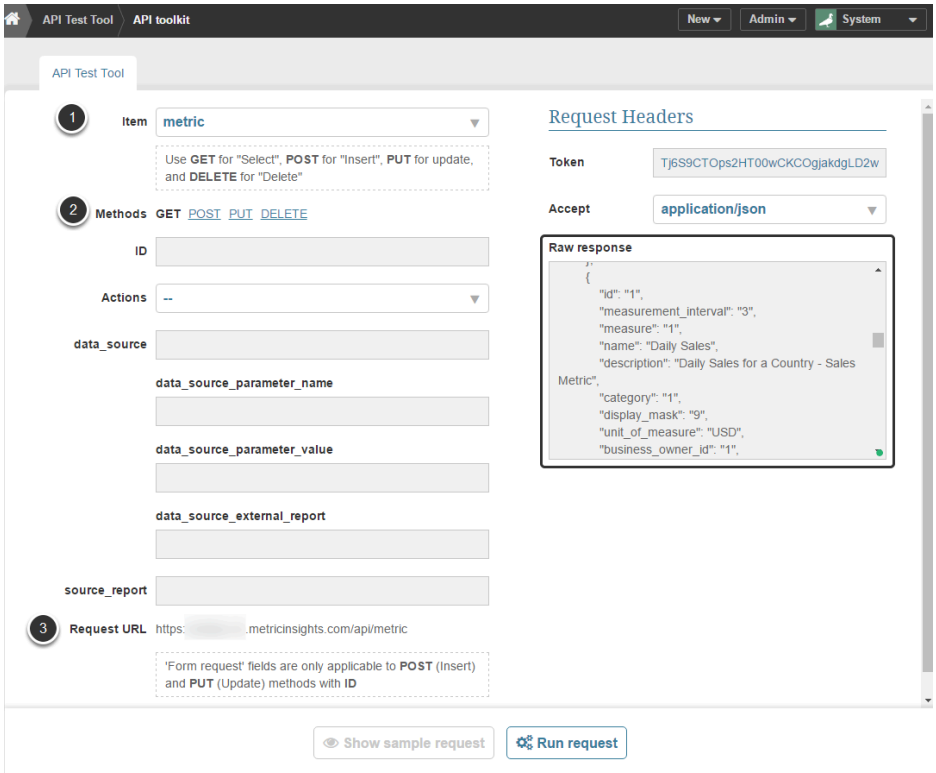
# 3.3 Metric API

This article provides sample API calls to manipulate Metrics.

**PREREQUISITES:**

- Set up API access
- Verify that you have API access
- Obtain a token via a get token call since all API calls require a token

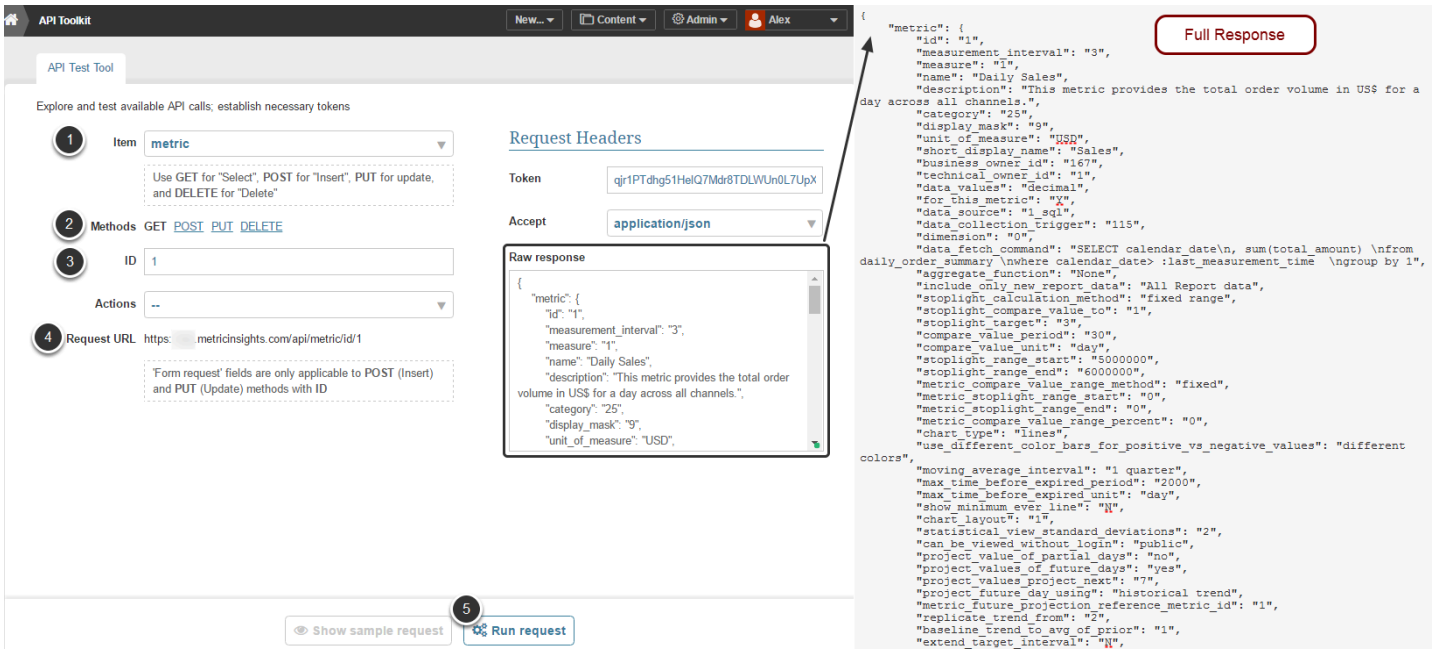
## Get all Metrics




Filter out all Metrics currently existing in the system, followed by the key information per each of them (such as measure, name description, category, data source, dimension, etc.)

1. **Item:** metric
2. **Method:** GET
3. **Request URL** (generated automatically)**:** https://yourcompany.metricinsights.com/api/metric
4. **Run request**

# Return Information about a specific Metric



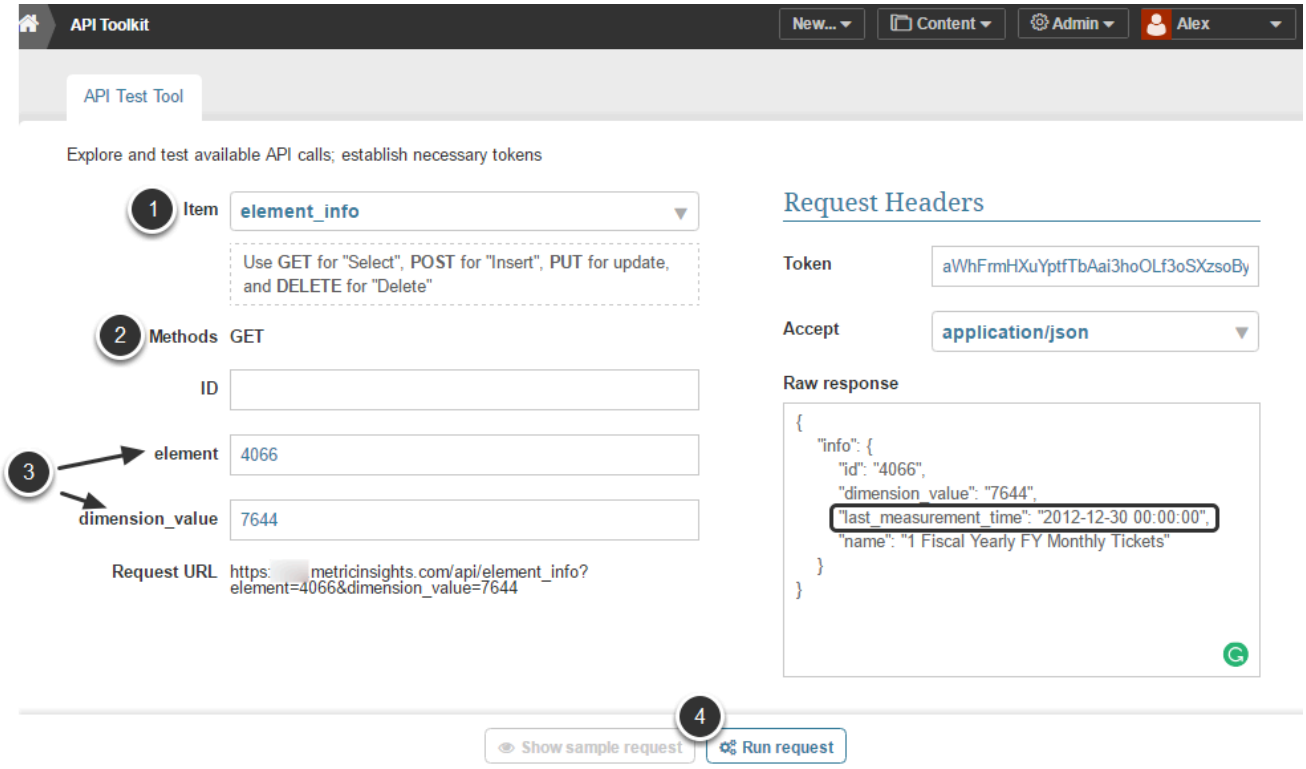

You can obtain all the information about a specific Metric (such as measure, name, description, category, data source, dimension, etc.)

1. **Item:** metric
2. **Method:** GET
3. **ID**: Specify the ID of the Metric you wish to get information about. Metric ID can be determined in the web browser URL of the *Metric Editor*. For example:

```
https://demo.metricinsights.com/chart/index/index/element/1
```

4. **Request URL** (generated automatically): https://yourcompany.metricinsights.com/api/metric/<id>
5. **Run request**

# Get :last_measurement_time




You can obtain all the information about  (such as measure, name description, category, data source, dimension, etc.)

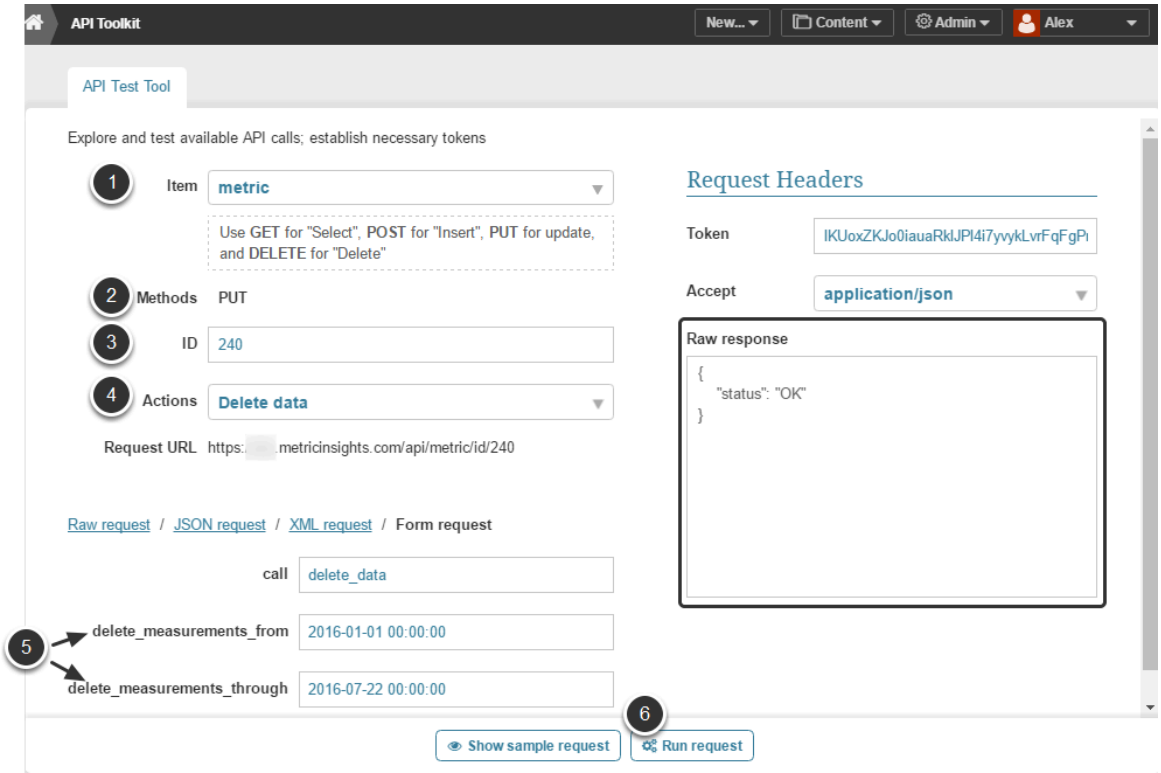
1. **Item:** element_info
2. **Method:** GET
3. Define the following parameters:
   - **element** (*numeric value*) - element id
   - **dimension_value** (*numeric value*) - dimension value id (for dimensioned elements)

These parameters can be determined in the web browser URL of the *Metric Viewer*. For example:

```
https://demo.metricinsights.com/chart/index/index/element/4066/name/
Fiscal+Yearly+FY+Monthly+Tickets/from/editor/segment/7644
```

4. **Run request**

# Delete measurements for a certain Time Range




You can obtain all the information about  (such as measure, name description, category, data source, dimension, etc.)

1. **Item:** metric
2. **Method:** PUT
3. **ID**: Specify the ID of a Metric you wish to get information about. Metric ID can be determined in the web browser URL of the *Metric Viewer*. For example:

```
https://demo.metricinsights.com/chart/index/index/element/240
```

4. **Actions:** Choose to **Delete Data.** Your choice will automatically define a value in the **call** field below
5. Define values for the following parameters:
   - **delete_measurements_from:** (*measurement time should be in one of the following formats: yyyy-mm-dd or yyyy-mm-dd 00:00:00*) - date from which measurements will be deleted
   - **delete_measurements_through:** (*measurement time should be in one of the following formats: yyyy-mm-dd or yyyy-mm-dd 00:00:00*) - date through which measurements will be deleted
6. **Run request**

# 3.4 Multi-Metric API

This article provides sample API calls to manipulate Metrics.

**PREREQUISITES:**

- Set up API access. **NOTE:** For the **User** API you must also enable User Management access.
- Verify that you have API access
- Obtain a token via a get token call since all API calls require a token

## Return Information about a Multi-metric

API:

```
https://yourinstance.metricinsights.com/api/multimetric/id/<multimetric_id>
```

Example:

```
https://yourinstance.metricinsights.com/api/metric/id/61
```

Method: GET

Header:

```
Accept:application/json
Token:UClCUUKxUlkdbhE1cHLz3kyjbIZYVh9eB34A5Q21Y3FPqKGSJs
```

Response:


```
{
        "multimetric": {
            "id": "61",
            "name": "DG MM v1",
            "description": "DG MM v1 Multi-Metric",
            "dimension": "0",
            "category": "14",
            "business_owner_id": "24",
            "technical_owner_id": "24",
            "stoplight_segment": "0",
            "left_y_axis_title": "DG Monthly Customers",
            "right_y_axis_title": "od Monthly Sales",
            "can_be_viewed_without_login": "no",
```

```
            "start_right_axis_from_zero": "no",
            "stack_bars": "no",
            "include_multimetric_data_table_in_e_mail_digests": "no",
            "display_data_table": "no"
        }
}
```

# Multi-metric Associate Elements

API:

```
https://yourinstance.metricinsights.com/api/api/
multimetric_association?element=<element_id>
```

Parameters:

- **element** (*numeric value*) - element id

Example:

```
https://adam.metricinsights.com/api/multimetric_association?element=4422
```

Method: GET

Header:

```
Accept:application/json
Token:UClCUUKxUlkdbhE1cHLz3kyjbIZYVh9eB34A5Q21Y3FPqKGSJs
```

Response:

```
{
        "multimetric_associations": [
            {
                "id": "3458",
                "measurement_interval": "4",
                "measurement": "Weekly",
                "element": "4422",
                "association_element": "5292",
                "name": "Total Weekly Sales"
            },
            {
                "id": "3460",
                "measurement_interval": "6",
                "measurement": "Monthly",
                "element": "4422",
                "association_element": "64",
```

```
                "name": "Profit Analysis II"
            },
            {
                "id": "3462",
                "measurement_interval": "9",
                "measurement": "Quarterly",
                "element": "4422",
                "association_element": "3877",
                "name": "de metric 1"
            },
    ]
}
```

# Delete measurements for a certain Time Range

API:

```
https://yourinstance.metricinsights.com/api/multimetric/id/<multimetric_id>
```

Example:

```
https://mi41.metricinsights.com/api/multimetric/id/61
```

Method: PUT

Call: delete_data

Header:

```
Accept:application/json
Token:UClCUUKxUlkdbhE1cHLz3kyjbIZYVh9eB34A5Q21Y3FPqKGSJs
```

Request example  (simple Multi-metrics):

```
{
        "call": "delete_data",
        "delete_measurements_from": "2014-01-02 00:00:00",
        "delete_measurements_through": "2014-03-26 00:00:00"
}
```

Request example  (dimensioned Multi-metrics):

```
{
        "call": "delete_data",
        "delete_measurements_from": "2014-02-01 00:00:00",
        "delete_measurements_through": "2014-10-01 00:00:00",
```

```
        "dimension_value": "Product Category Total"
}
```

Parameters:

- **delete_measurements_from -** (*measurement time should be in one of the following formats: yyyy-mm-dd or yyyy-mm-dd 00:00:00*) - date from which measurements will be deleted
- **delete_measurements_through**- (*measurement time should be in one of the following formats: yyyy-mm-dd or yyyy-mm-dd 00:00:00*) - date through which measurements will be deleted

Response:

```
{
"status": "OK"
}
```

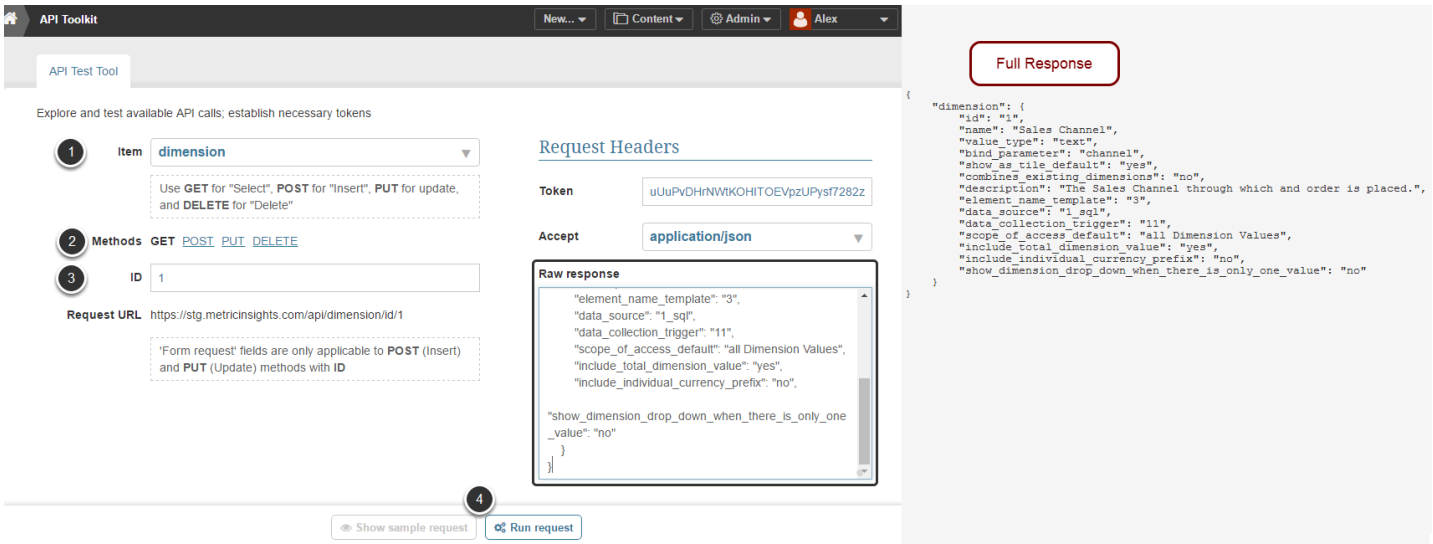
# 3.5 Dimension API

This article provides sample API calls to manipulate **Dimensions**.

**PREREQUISITES:**

- Set up API access. **NOTE:** For the **User** API you must also enable User Management access.
- Verify that you have API access
- Obtain a token via a get token call since all API calls require a token

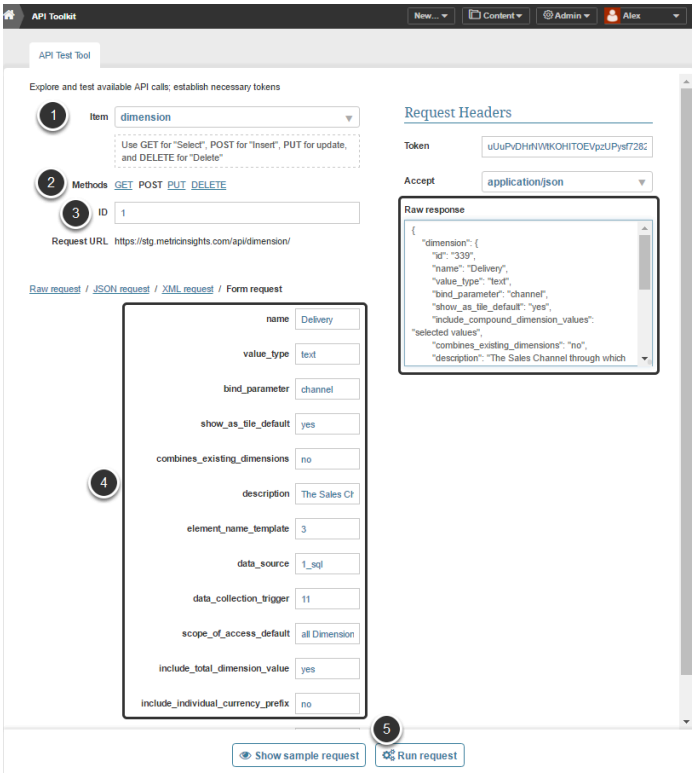
## 1. Fetch Dimension Information




Retrieve key data about specific Dimension.

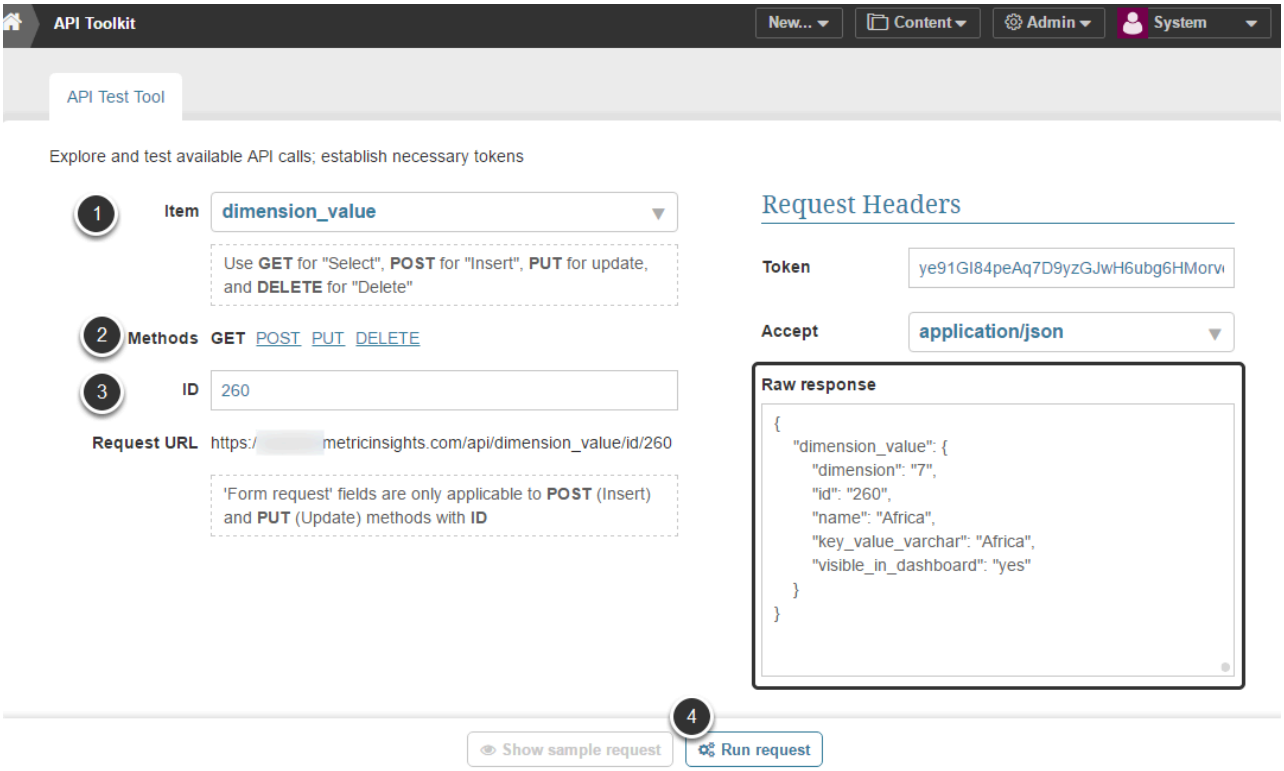
1. **Item:** dimension
2. **Method:** GET
3. **ID:** Specify ID of the dimension you need to get more information about
4. **Run request**

# 1.1. Create Dimension




1. **Item:** dimension
2. **Method:** POST
3. **ID:** Find an existing Dimension that you wish to use as a foundation for new Alert. To see the list of all Dimensions created in the system, select GET method, keep this field empty and **Run request**
4. Define parameters of the new dimension
5. **Run Request**

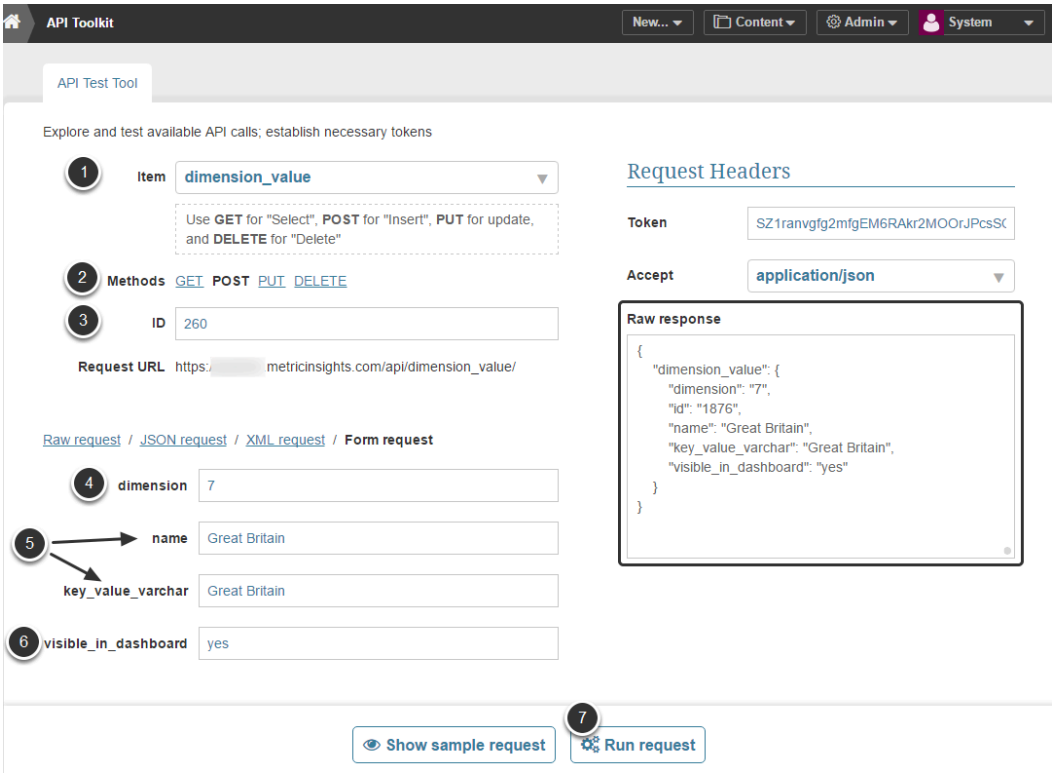
# 2. Fetch Dimension Value Information




Retrieve key data about specific Dimension.

1. **Item:** dimension_value
2. **Method:** GET
3. **ID:** Specify ID of the Dimension Value you need to get more information about
4. **Run request**

## 2.1. Create Dimension Value




1. **Item:** dimension_value
2. **Method:** POST
3. **ID:** Find an existing Dimension Value that you wish to use as a foundation for new Alert. To see the list of all Dimension Values created in the system, select GET method, keep this field empty and **Run request**
4. **dimension:** Specify ID of the Dimension where you need to create a new Value
5. **name / key_value_varchar:** Define a descriptive name of a new Dimension value
6. **visible_in_dashboard:** *yes / no*
7. **Run Request**

## 2.2. Create Compound Dimension Value

API:

```
https://yourinstance.metricinsights.com/api/dimension_value
```

Header:

```
Accept:application/json
Token:UClCUUKxUlkdbhE1cHLz3kyjbIZYVh9eB34A5Q21Y3FPqKGSJs
```

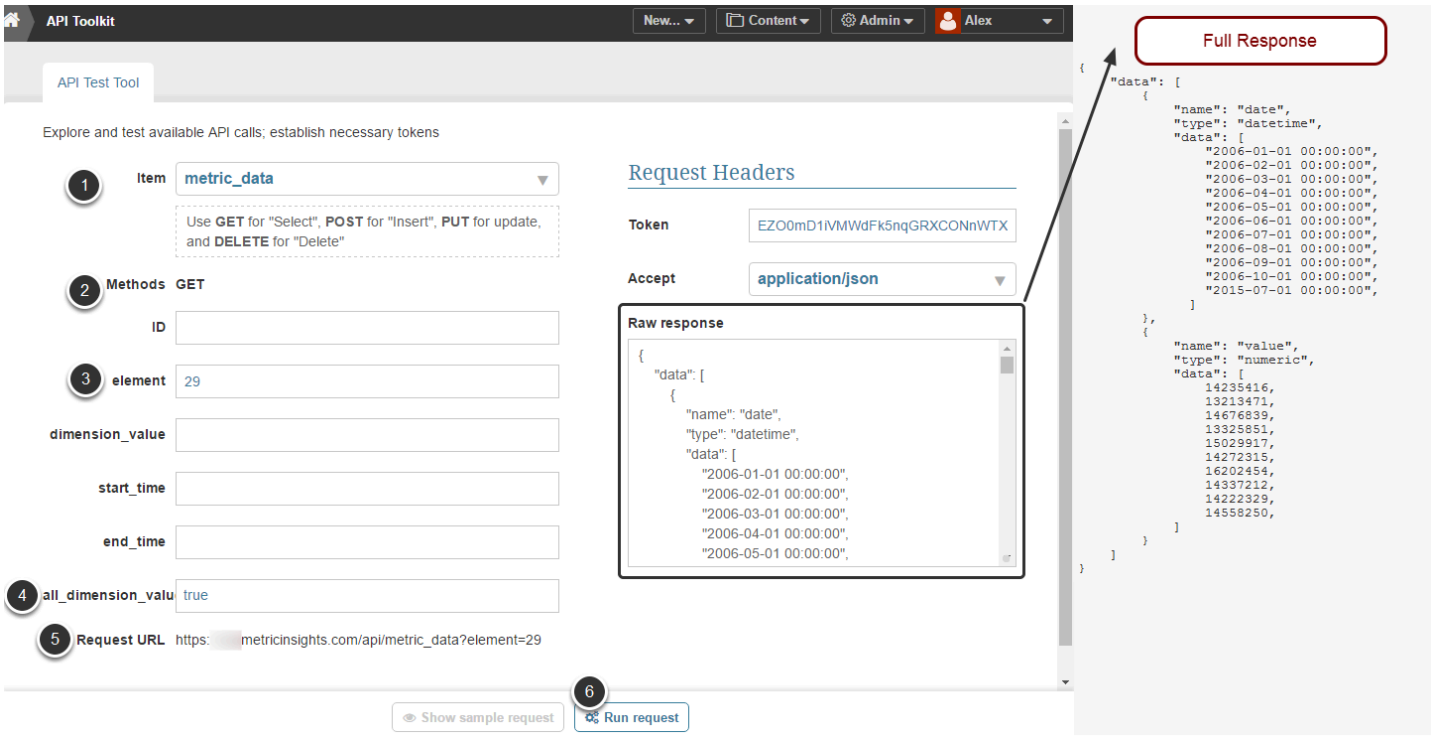
POST body payload:

```
{
"dimension": "17",
"primary_dimension_value": "16345",
"secondary_dimension_value": "13560",
"visible_in_dashboard": "Yes"
}
```

Response:

```
{"dimension_value":{"dimension":"17","primary_dimension_value":"16345","secondary_dimens
ion_value":"13560","visible_in_dashboard":"Yes"}}
```

# 3. Get all Dimension information (for Metric / Multi-metric / Report)




This API call can be used to get metric data along with all data related to all Dimension Value.

**NOTE:** Dimension Values are returned as dimension IDs.

1. **Item:** metric_data / multimetric_data / report_data
2. **Method:** GET
3. **element:** (*numeric value*) Specify the ID of an you wish to get information about. Metric ID can be determined in the web browser URL of the *Metric Viewer (Multi-Metric Viewer* or *Report Viewer)*. For example:

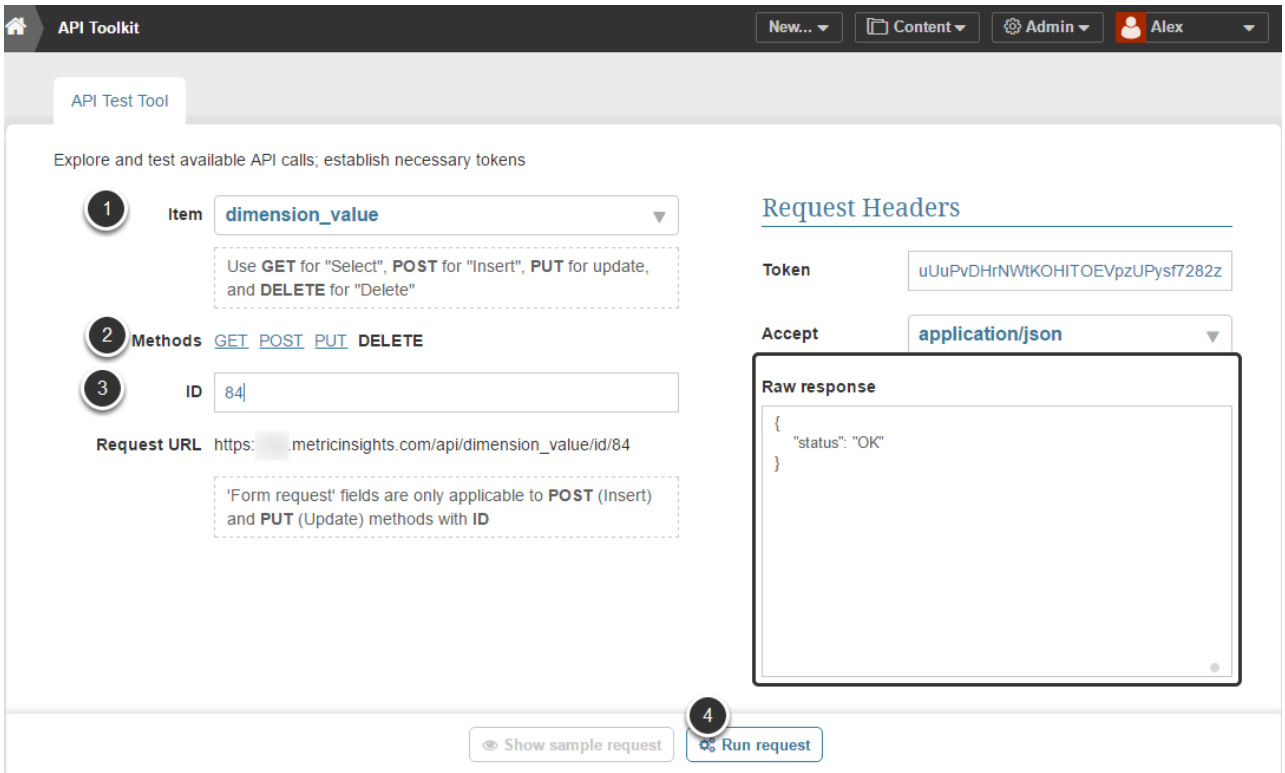
```
https://demo.metricinsights.com/chart/index/index/element/29
```

4. **all_dimension_values:** "true". This parameter retrieves all data related to a specific dimension value
5. **API**:

```
Dimensioned Metric - https://demo.metricinsights.com/api/
metric_data?element=<metric_id>&all_dimension_value=true
Dimensioned Report -  https://demo.metricinsights.com/api/
report_data?element=<report_id>&all_dimension_value=true
Dimensioned Multi-metric - https://demo.metricinsights.com/api/
multimetric_data?element=<multimetric_id>&all_dimension_value=true
```

6. **Run request**

# 4. Delete Dimension value




1. **Item:** dimension_value
2. **Method:** DELETE
3. **ID:** Specify the Dimension Value you need to delete
4. **Run request**

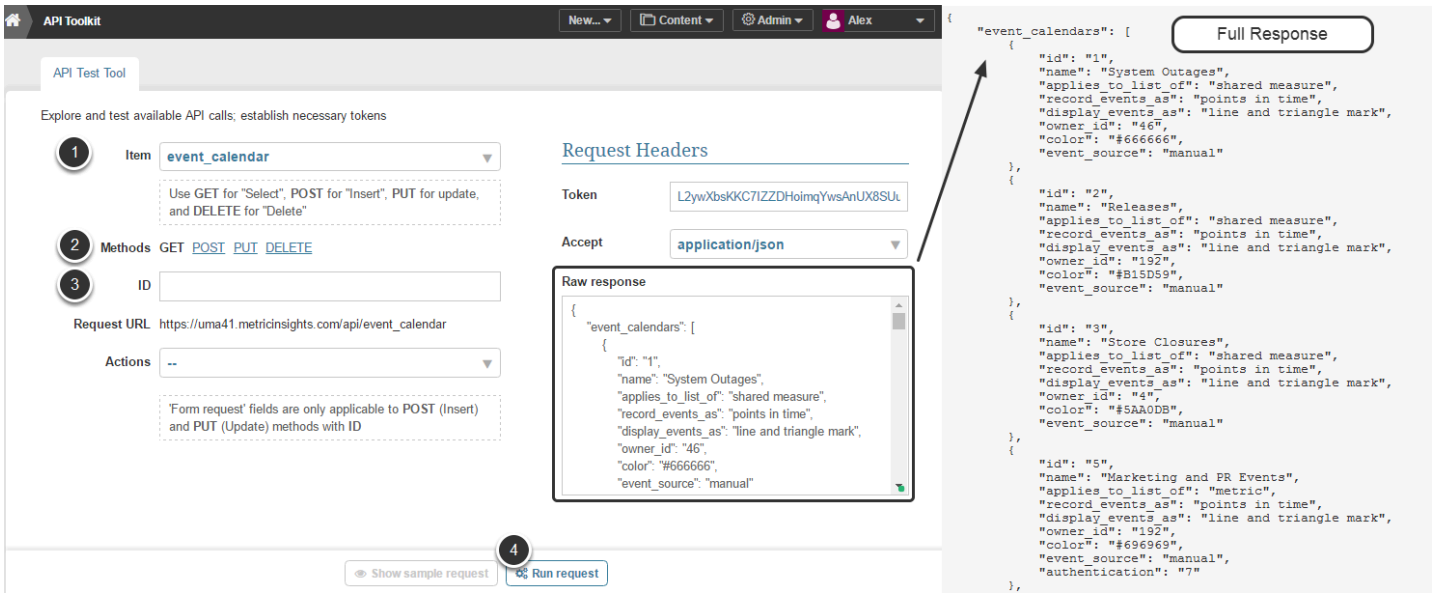
# 3.6 Calendar Event API

This article provides sample API calls to manipulate **Calendar Events**.

**PRE-REQUISITES:**

- Set up API access.
- Verify that you have API access
- Obtain a token via a get token call since all API calls require a token
- Event Calendar must exist, where you can then add, fetch delete events

## 1. Fetch Event Calendar information



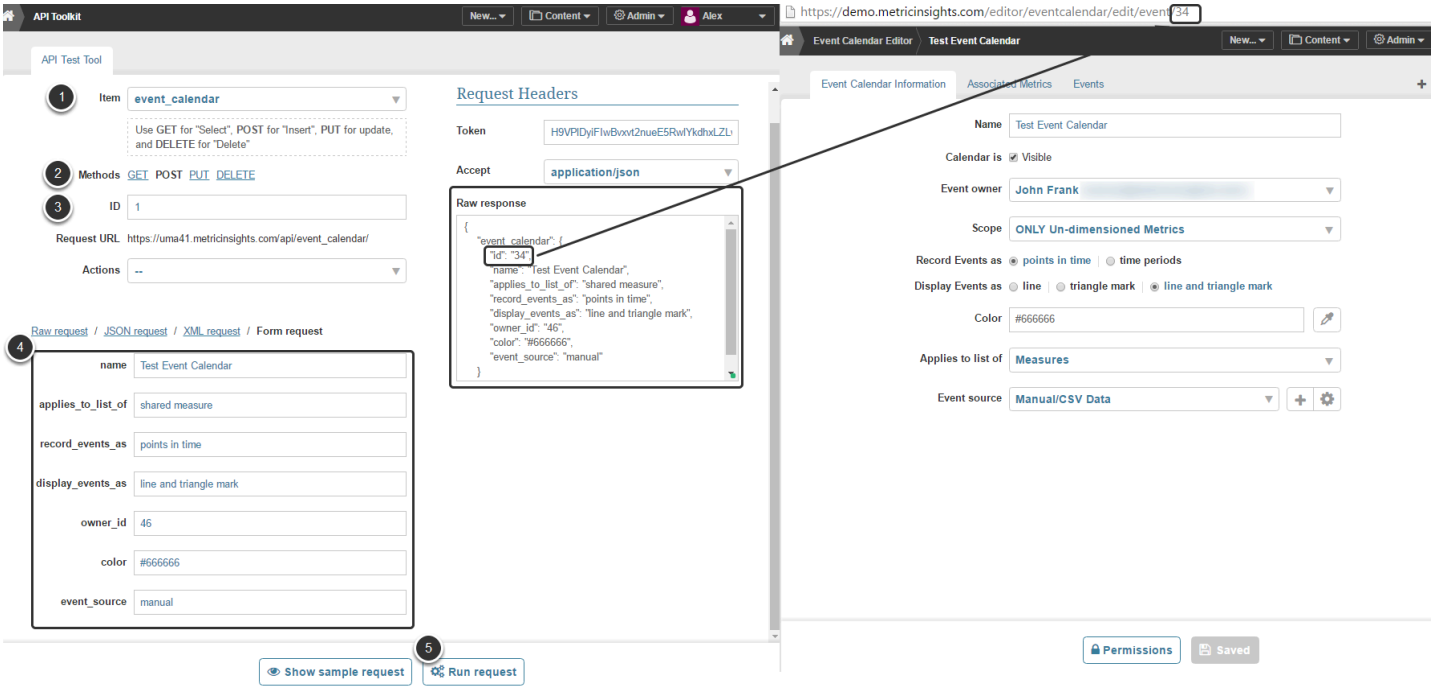

Retrieve information about all or specific Event Calendars created in the system.

1. **Item:** event_calendar
2. **Method:** GET
3. **ID:** Leave this field empty to get the list of all event calendars created in the system or define the ID of a specific Event Calendar. The ID of the Event Calendar can be found at the *Event Calendar Editor* web link *(at Admin > Event Calendars >* select an Event Calendar from the list):

```
https://demo.metricinsights.com/editor/eventcalendar/edit/event/7
```

4. **Run request**

# 1.1. Create Event Calendar

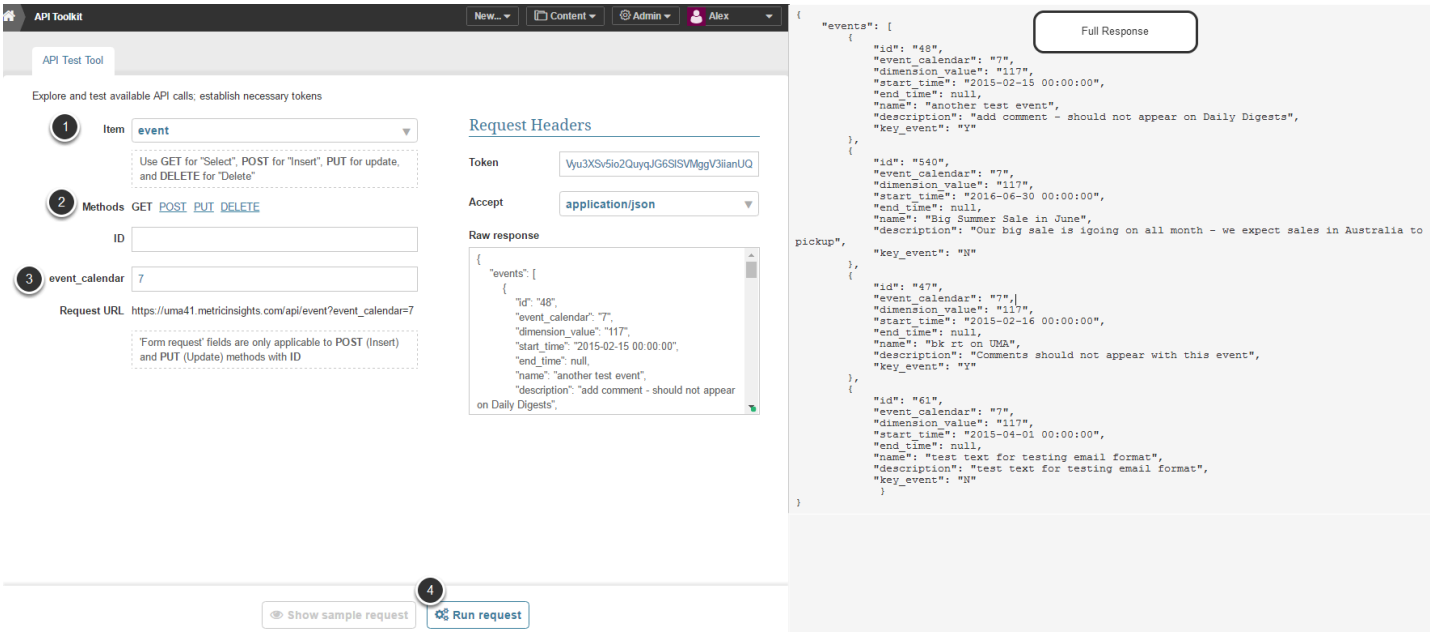



1. **Item:** event_calendar
2. **Method:** POST
3. **ID:** Define ID of an Event Calendar that should serve as a foundation for the new calendar. The ID of the Event Calendar can be found at the *Event Calendar Editor* web link *(at Admin > Event Calendars* > select an Event Calendar from the list):

```
https://demo.metricinsights.com/editor/eventcalendar/edit/event/7
```

4. Define parameters of a new Event Calendar
5. **Run request**

# 2. Fetch all Events of the specific Event Calendar



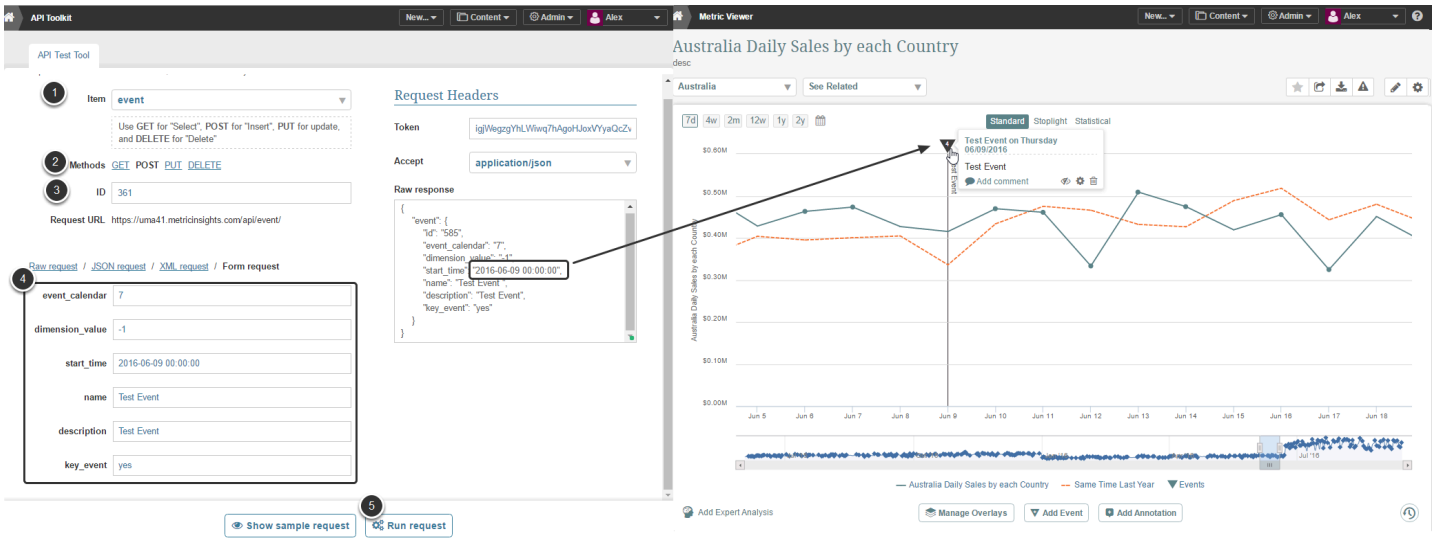

Retrieve information about all or specific Event Calendars created in the system.

1. **Item:** event
2. **Method:** GET
3. **event_calendar:** Define the ID of the Event Calendar to retrieve its events. The ID of the Event Calendar can be found at the *Event Calendar Editor* web link *(at Admin > Event Calendars* > select an Event Calendar from the list):

```
https://demo.metricinsights.com/editor/eventcalendar/edit/event/7
```

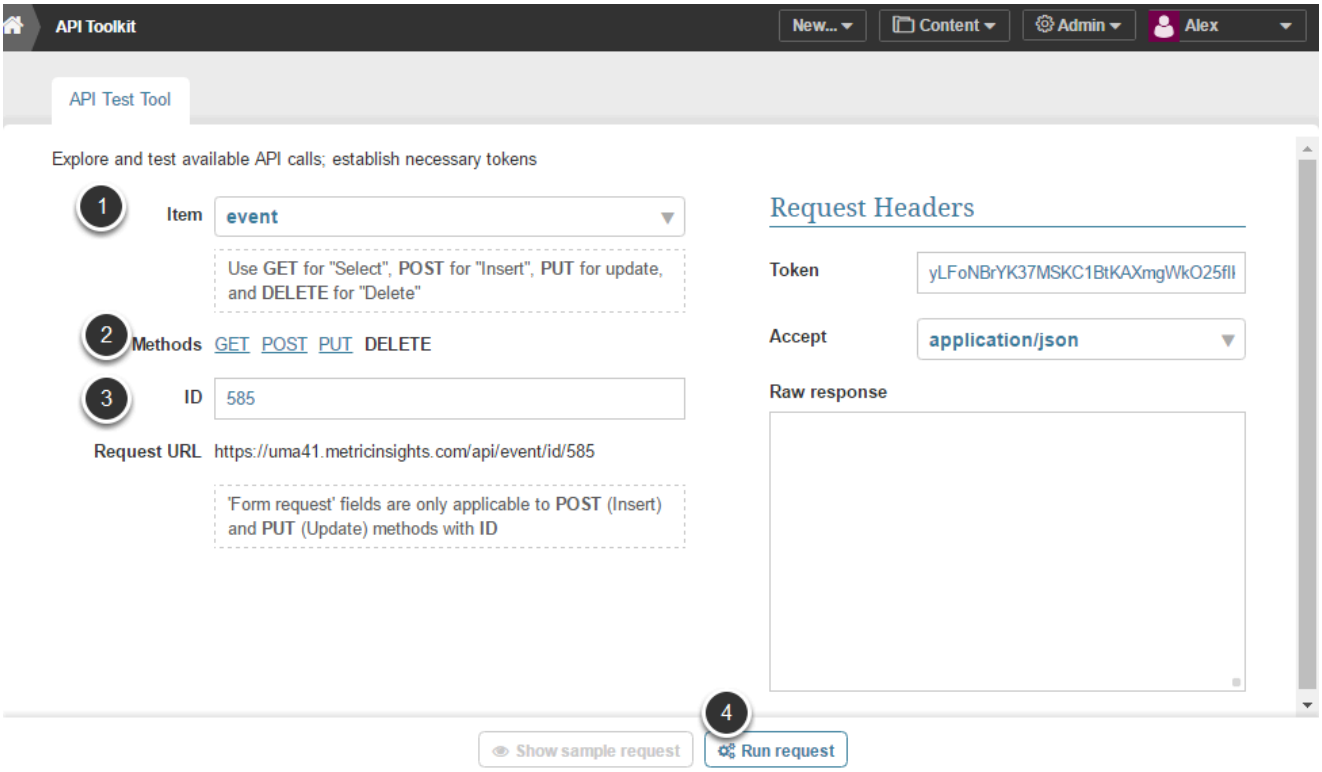
4. **Run request**

# 3. Create Event




1. **Item:** event
2. **Method:** POST
3. **ID:** Define ID of the Event that should serve as a foundation for the new event. To find an ID of an Event, see Step 2.
4. Define parameters of a new Event Event. **NOTE:** Dimension Value "-1" defines ALL values of a given dimension.
5. **Run request**

# 4. Delete Event




1. **Item:** event
2. **Method:** DELETE
3. **ID:** Define ID of the Event you need to delete. To find an ID of an Event, see Step 2.
4. **Run request**

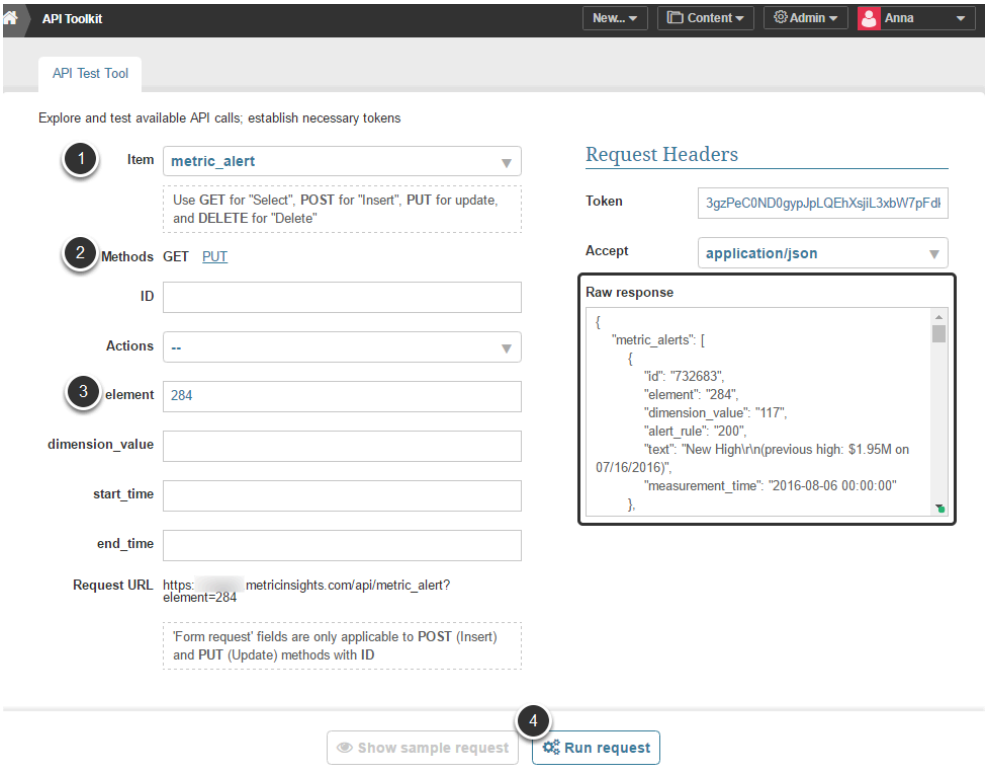
# 3.7 Alerts API

This article provides sample API calls used for obtaining and creating **Alerts**.

**PREREQUISITES:**

- Set up API access
- Verify that you have API access OR get token call via curl since all API calls require a token

## Get all Alerts for a specific Metric



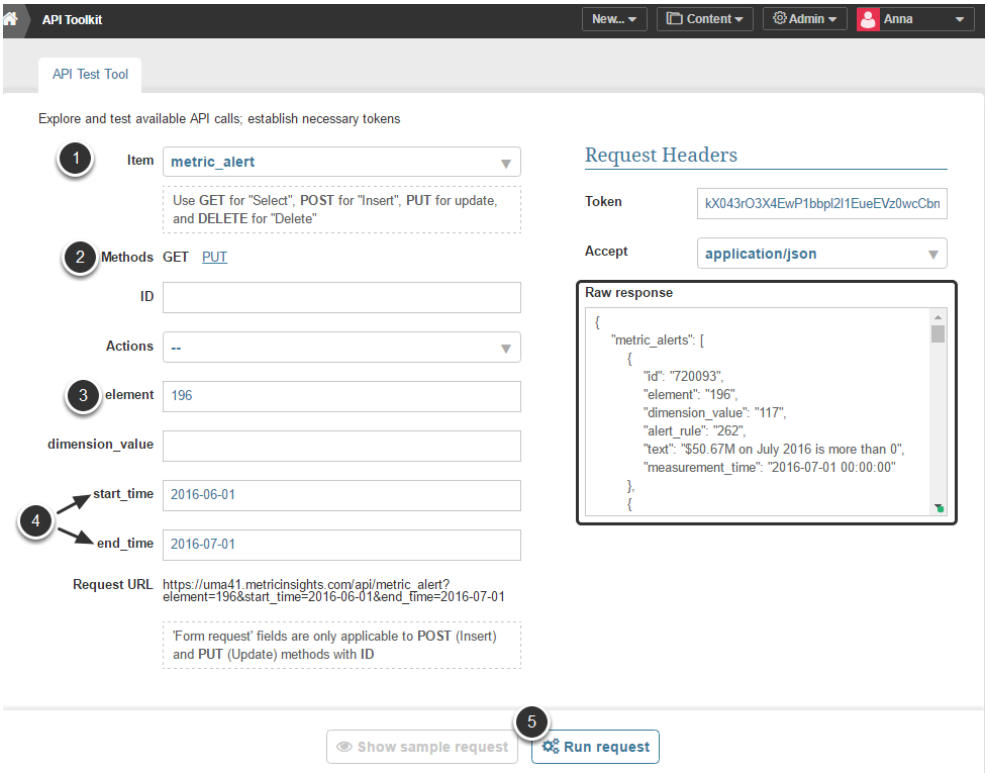

You can fetch the list of all alerts for a specific Metric. **NOTE:** You might get empty result.

1. **Item:** metric_alert
2. **Method:** GET
3. **element**: Specify the element ID in this field. Element ID can be determined in the web browser URL of its editor. For example:

```
https://demo.metricinsights.com/chart/index/index/element/284
```

4. **Run Request**

# Get only recent Alerts (or Alerts for a specific time period)




or qualify each metric by time range

1. **Item:** alert_rule
2. **Method:** POST
3. **element**: Specify the element ID in this field. Element ID can be determined in the web browser URL of its editor. For example:

```
https://demo.metricinsights.com/chart/index/index/element/284
```

4. **start_time/end_time**: Define a date range for which the alerts should be shown.
5. **Run Request**

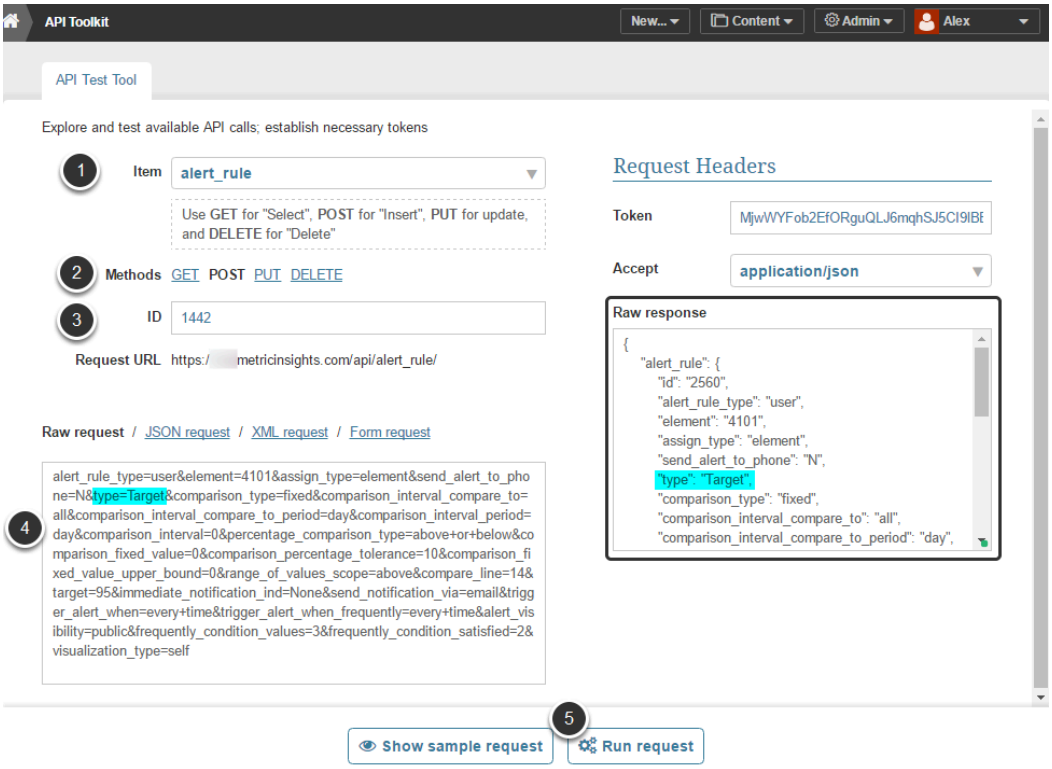
# Create an Alert Rule (multiple types)

| Parameter | Type | Description | Possible Values |
|---|---|---|---|
| name | *string* | A unique name of the alert, defined by a user. | user_defined_name |
| element | *integer* | Specify the unique identification number assigned to the element for which the alert is created. You can find element ID in the link of the element Editor/Viewer. | element_id |

| | | | |
|---|---|---|---|
| report | *integer* | A unique number assigned to the report for which the alert is created. You can find report ID in the link of the report Editor/Viewer. | report_id |
| type | *enum* | Define the type of condition that causes creation of an alert in the system. | alert_type = 'fixed value', 'prior value', 'average', 'target', 'volatility band', 'compound', 'stoplight', 'unusual value', 'best ever', 'worst ever' |
| alert_rule | *enum* | There are two types of alerts in Metric Insights: KPI and global alerts. Specify the "user" value in order to create a KPI alert and global for the second type. | alert_rule_type = 'user', 'global' |
| alert_visibility | *enum* | The alert can be available ether to all users who have access to this metric public or to you only private. | 'public', 'private' |
| assign_type | *enum* | Define the entity to which the given alert is assigned. | 'element', 'segment_value', 'category', 'favorite' |
| send_alert_to_phone | *enum* | Define whether the alert shall be sent as a notification on a mobile device. | 'Y', 'N' |
| comparison_type | *enum* | Specify the type of condition that causes the alert triggering. | 'fixed', 'pct variance', 'less fixed', 'less pct variance' |
| comparison_interval_compare_to_period | *varchar* | Specify the period to which the comparison should be made. | 'day', 'week', 'month', 'quarter', 'year' |
| comparison_interval_period | *varchar* | Define the period for which values should be compared. | 'day', 'week', 'month', 'quarter', 'year' |
| comparison_interval | *integer* | Define how many comparison periods should be taken into consideration. | DEFAULT NULL |
| percentage_comparison_type | *enum* | For more details refer to this article | 'above', 'above or equal', 'below', 'below or equal', 'above or below', 'within', 'between', 'equal' |
| comparison_fixed_value | | Designate the fixed value to which the conditions triggering the alert creation | DEFAULT NULL |

| | | | |
|---|---|---|---|
| | | should be applied. | |
| comparison_fixed_value_upper_bound | | Define the upper bound value in the range of appropriate values. | DEFAULT NULL |
| comparison_percentage_tolerance | *integer* | Define the percantage below or above which the alarm notification shall be created. | DEFAULT NULL |
| range_of_values_scope | *enum* | Designate the value condition in accordance to which the alert should be created. | 'above', 'below', 'above or below' |
| compare_line | *integer* | Specfy the unique identification number of a compare line. | DEFAULT NULL |
| immediate_notification_ind | *enum* | Define whether notification shall be sent: 'Y' - immediately, 'N' - as an email digest, 'None - shown in browser only. | 'Y', 'N', 'None' |
| send_notification_via | *enum* | Specify the means of sending the notification in this parameter. | 'email', 'sms' |
| trigger_alert_when | *enum* | Specify when the alert is supposed to be sent. | 'first time only', 'every time', 'frequently' |
| trigger_alert_when_frequently | *enum* | Define additional conditions if you the 'frequently' value has been chosen for the trigger_alert_when parameter. | 'first time only', 'every time', 'frequently' |
| frequently_condition_values / frequently_condition_satisfied | *integer* | Designate how many times the condition should be met and how many times it should be satisfied to trigger the alert. | NOT NULL |

# Target Alert Rule Type

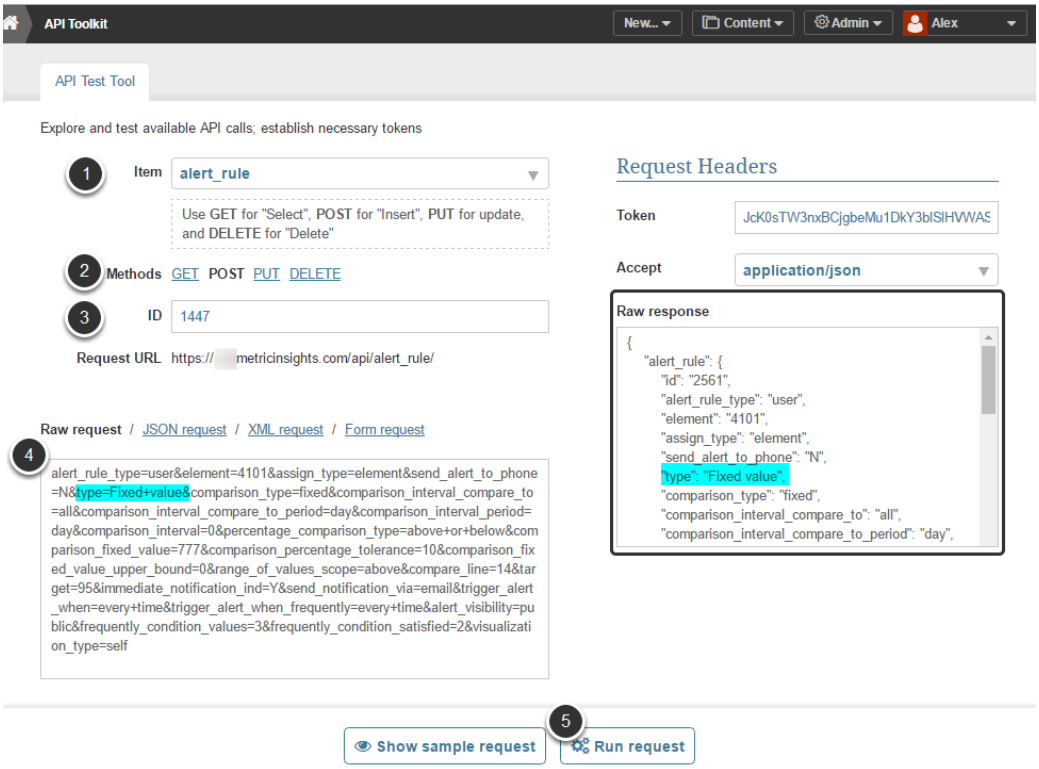



1. **Item:** alert_rule
2. **Method:** POST
3. **ID**: Find an existing Alert of the **Target** type that you wish to use as a foundation for new Alert; open its Editor and find its ID in the web browser URL. For example:

```
https://demo.metricinsights.com/editor/kpialertrule/edit/id/1442
```

4. Modify any parameters of this request according to your requirements
5. **Run Request**

# Fixed Value Rule Type

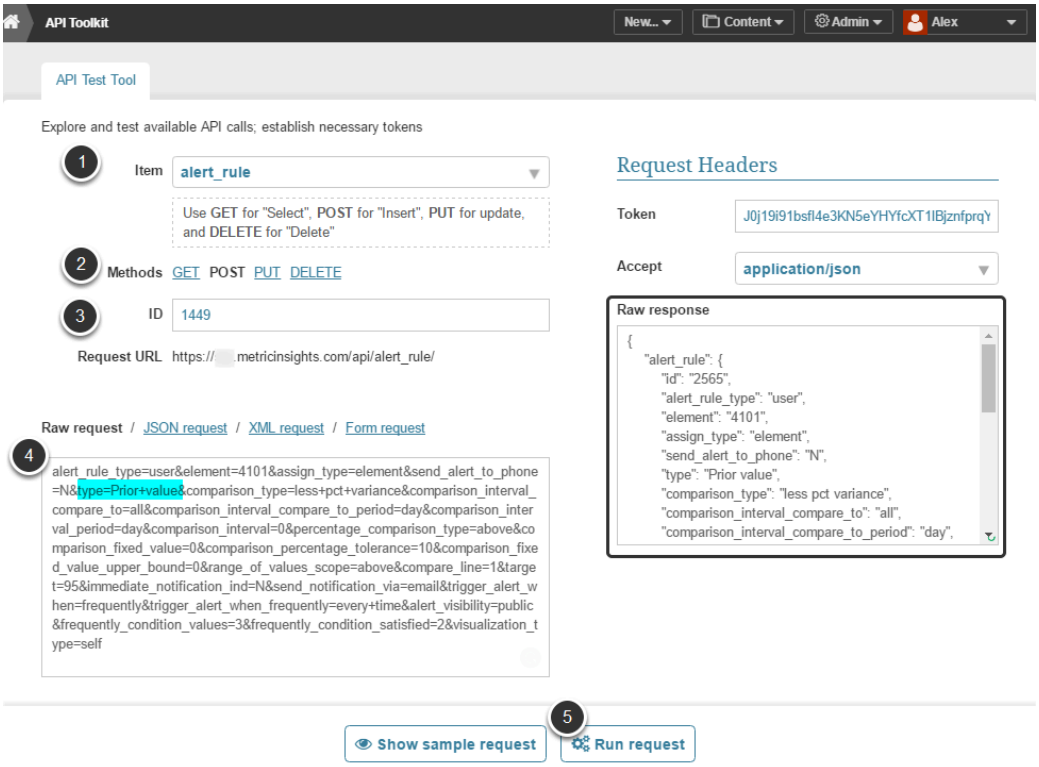



1. **Item:** alert_rule
2. **Method:** POST
3. **ID**: Find an existing Alert of a **Fixed Value** type that you wish to use as a foundation for new Alert; open its Editor and find its ID in the web browser URL. For example:

```
https://demo.metricinsights.com/editor/kpialertrule/edit/id/1444
```

4. Modify any parameters of this request according to your requirements
5. **Run Request**

# Prior Value Rule Type

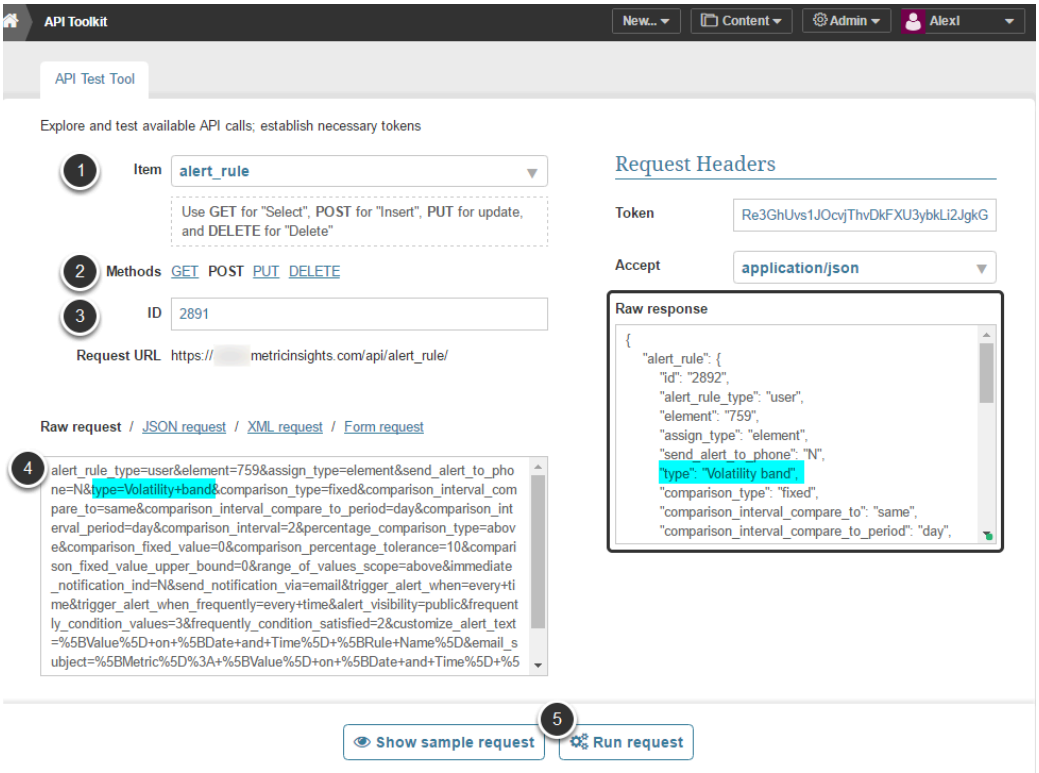



1. **Item:** alert_rule
2. **Method:** POST
3. **ID**: Find an existing Alert with a **Prior Value** type that you wish to use as a foundation for new Alert; open its Editor and find its ID in the web browser URL. For example:

```
https://demo.metricinsights.com/editor/kpialertrule/edit/id/1449
```

4. Modify any parameters of this request according to your requirements
5. **Run Request**

# Recent Volatility Rule Type

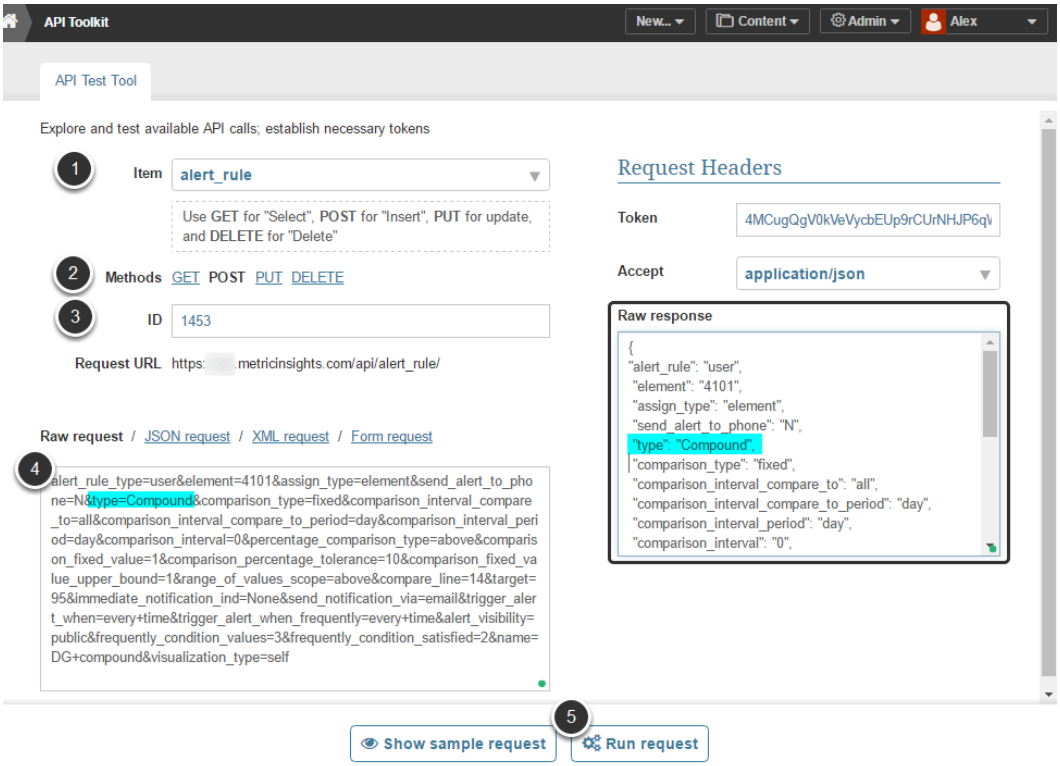



1. **Item:** alert_rule
2. **Method:** POST
3. **ID**: Find an existing Alert with a **Volatility Levels** type that you wish to use as a foundation for new Alert; open its Editor and find its ID in the web browser URL. For example:

```
https://demo.metricinsights.com/editor/kpialertrule/edit/id/2891
```

4. Modify any parameters of this request according to your requirements
5. **Run Request**

# Combination of Multiple Alerts Rule Types

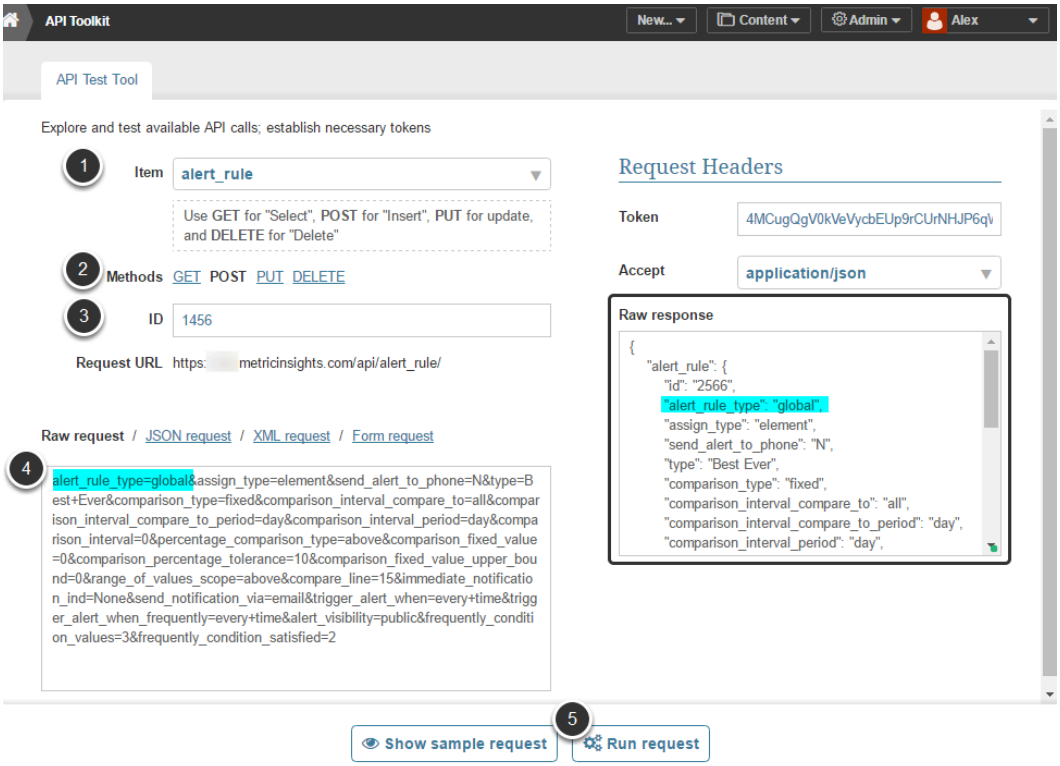



1. **Item:** alert_rule
2. **Method:** POST
3. **ID**: Find an existing Alert with a **Compound** type that you wish to use as a foundation for the new Alert; open its Editor and find its ID in the web browser URL. For example:

```
https://demo.metricinsights.com/editor/kpialertrule/edit/id/2891
```

4. Modify any parameters of this request according to your requirements
5. **Run Request**

# Global Alert Rule Type




1. **Item:** alert_rule
2. **Method:** POST
3. **ID**: Find an existing **Global** Alert Rule that you wish to use as a foundation for the new Alert; open its Editor and find its ID in the web browser URL. For example:

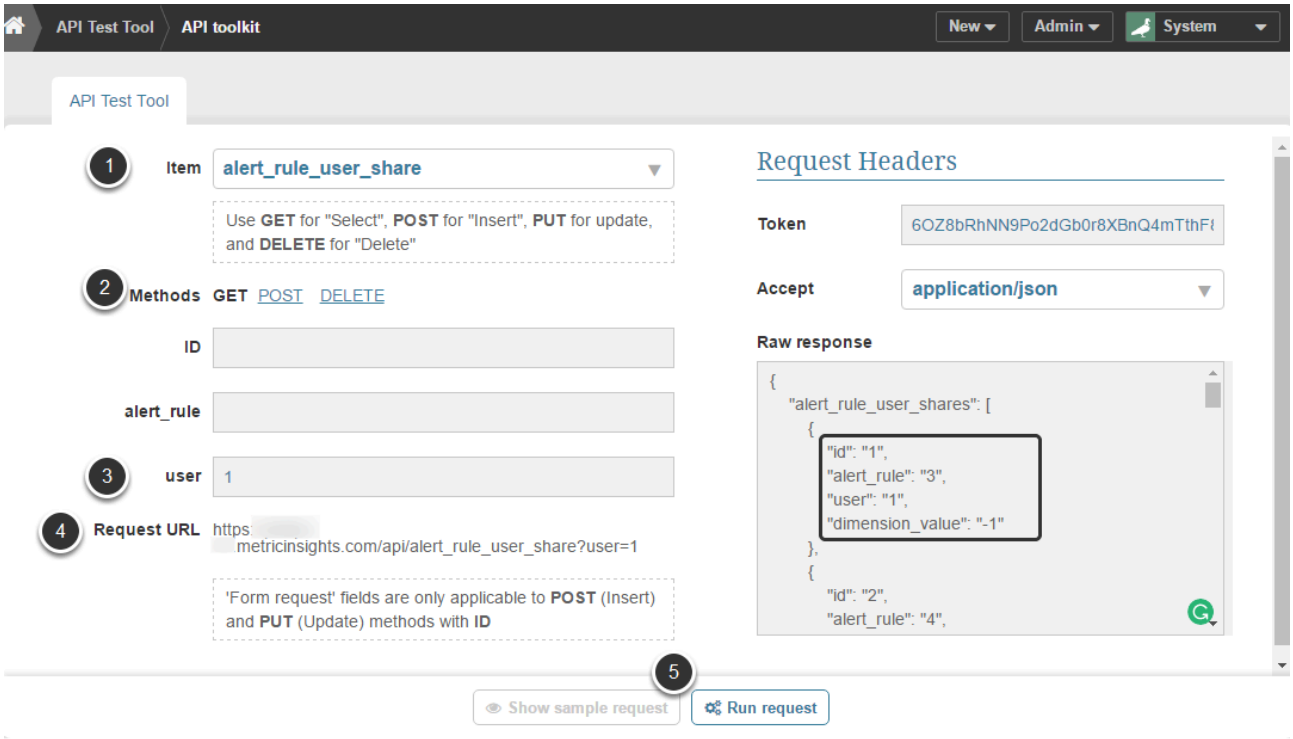
```
https://demo.metricinsights.com/editor/alertrule/edit/id/1456
```

4. Modify any parameters of this request according to your requirements
5. **Run Request**

# User Alert Subscriptions

You can use API toolkit to easily manage users' Alert subscriptions.

# View all User Subscriptions to Alerts




You can review all the Alerts certain user is subscribed to.

1. **Item:** alert_user_share
2. **Method:** GET
3. **User**: Specify user ID. To determine the ID of a specific user, open the *User Editor* and find ID in the web browser URL. For example:

```
https://demo.metricinsights.com/admin/user/edit/id/1
```

4. **Request URL** (generated automatically)**:**

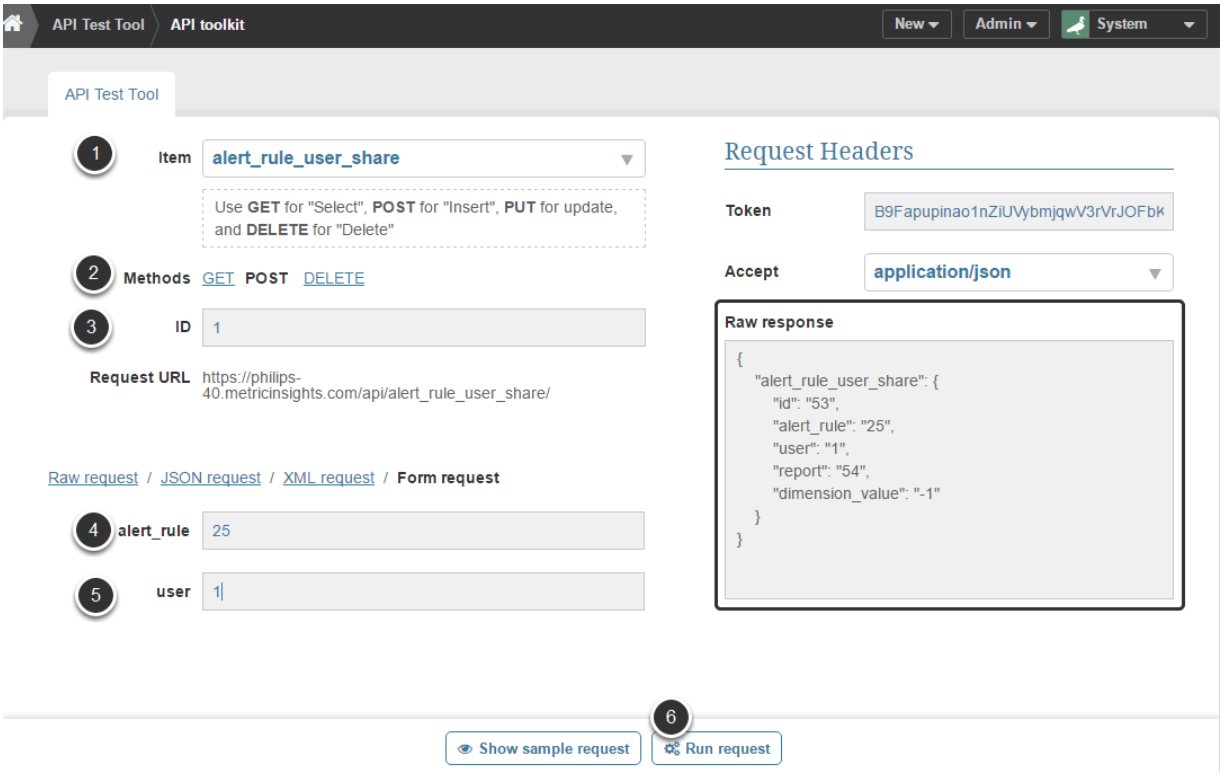
```
https://demo.metricinsights.com/api/alert_rule_user_share?user=1
```

5. **Run Request**

**NOTE:**

Dimension value **"-1"** stands for all dimension values.

# Create User Subscription to an Alert Rule




You can subscribe users to specific Alert Rules (one user to one Alert rule at a time).

1. **Item:** alert_rule_user_share
2. **Method:** POST
3. **ID:** Enter the ID of previously created "alert_rule+user" combination. To see the list of all such pairs, choose GET method, leave **ID**, **alert_rule** and **user** fields empty and **Run Request.** All "alert_rule+user" combinations are going to be displayed in the **Raw response** field. Choose the one that is supposed to serve as a foundation for the new combination.
4. **alert_rule**: Specify the alert rule ID. To determine the ID of a specific Alert, open the *Alert Editor* and find ID in the web browser URL. For example:

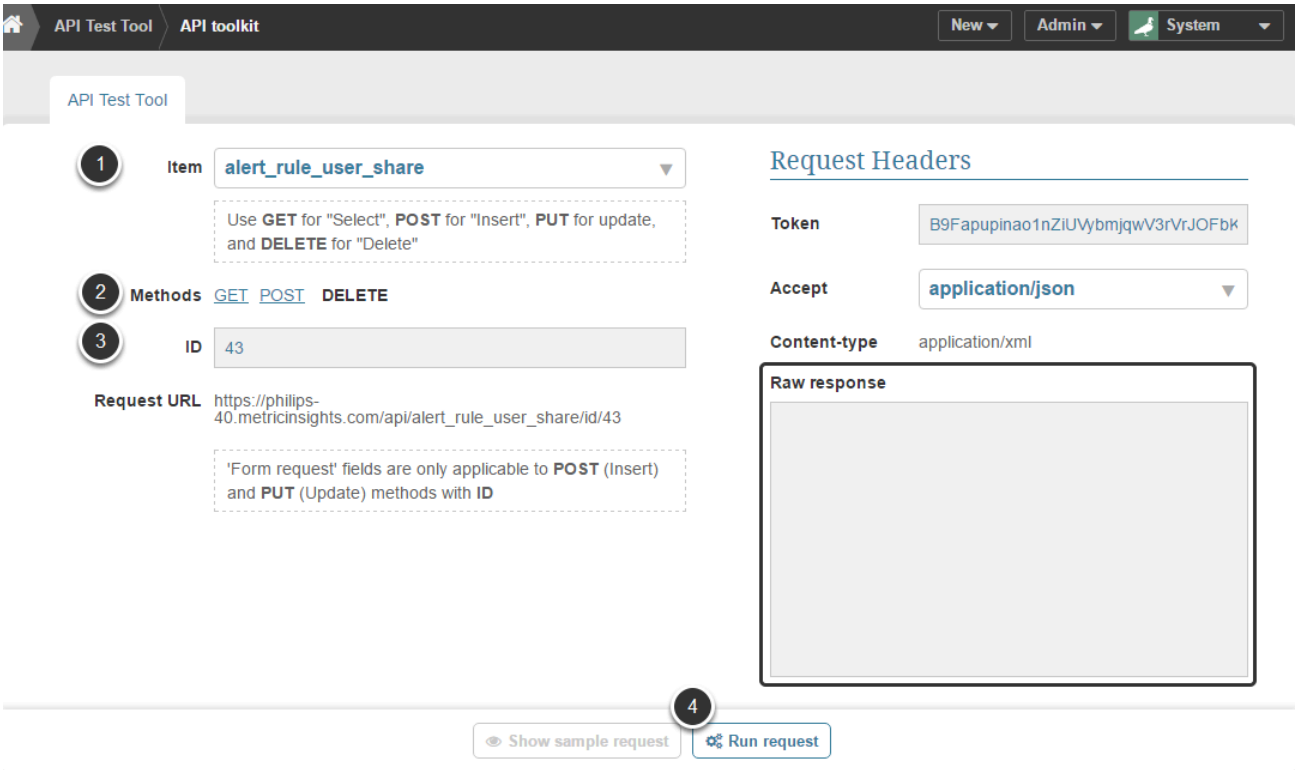
```
https://demo.metricinsights.com/editor/kpialertrule/edit/id/9
```

5. **User**: Specify user ID. To determine the ID of a specific user, open the *User Editor* and find ID in the web browser URL. For example:

```
https://demo.metricinsights.com/admin/user/edit/id/1
```

6. **Run Request**

# Unsubscribe a User from the Alert Rule




This procedure allows deleting certain alert rule subscriptions.

1. **Item:** alert_rule_user_share
2. **Method:** DELETE
3. **ID:** Enter the ID of the "alert_rule+user" combination to be deleted. To see the list of all such pairs, choose GET method, leave **ID**, **alert_rule** and **user** fields empty and **Run Request.** All "alert_rule+user" combinations are going to be displayed in the **Raw response** field. Choose the one which is supposed to be deleted.
4. **Run Request**

**NOTE:** If the "alert_rule+user" combination is successfully deleted, the **Raw response** field is going to be empty.

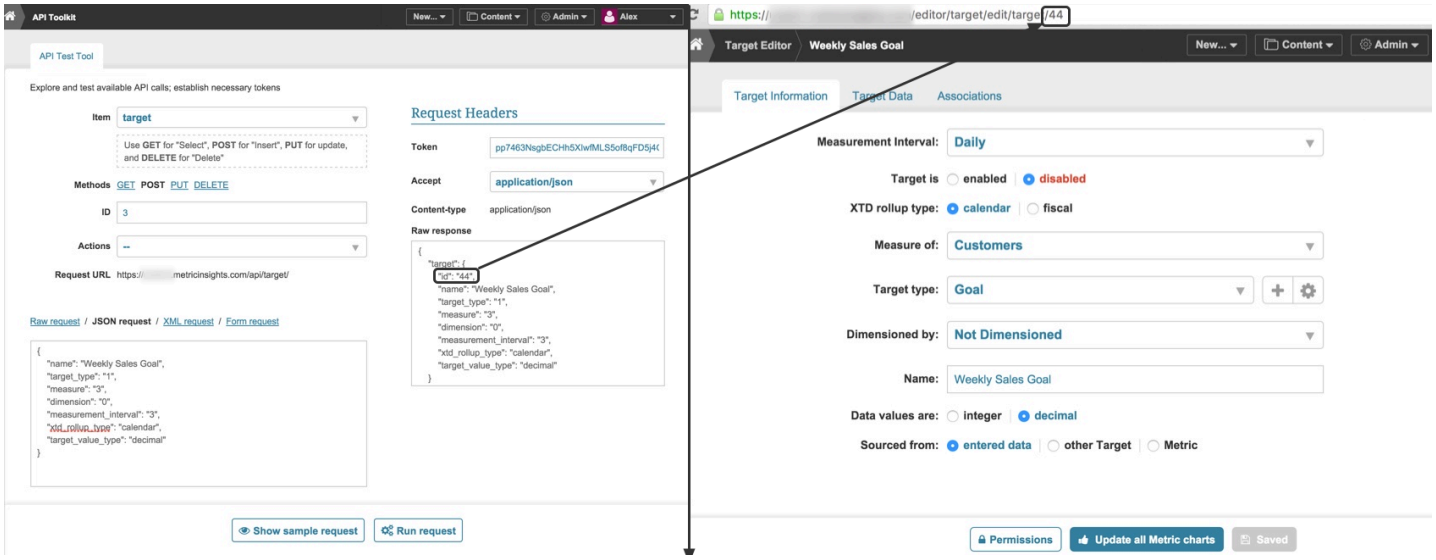
# 3.8 Target API

This article provides sample API calls to manipulate **Targets**.

**PRE-REQUISITES:**

- Set up API access
- Verify that you have API access OR get token call via curl since all API calls require a token.

## Create a new Target




Item: **target**

Method: **POST**

Header:

```
Accept:application/json
Token:pp7463NsgbECHh5XIwfMLS5of8qFD5j4G3QohTCMh72kHyXH3K
```

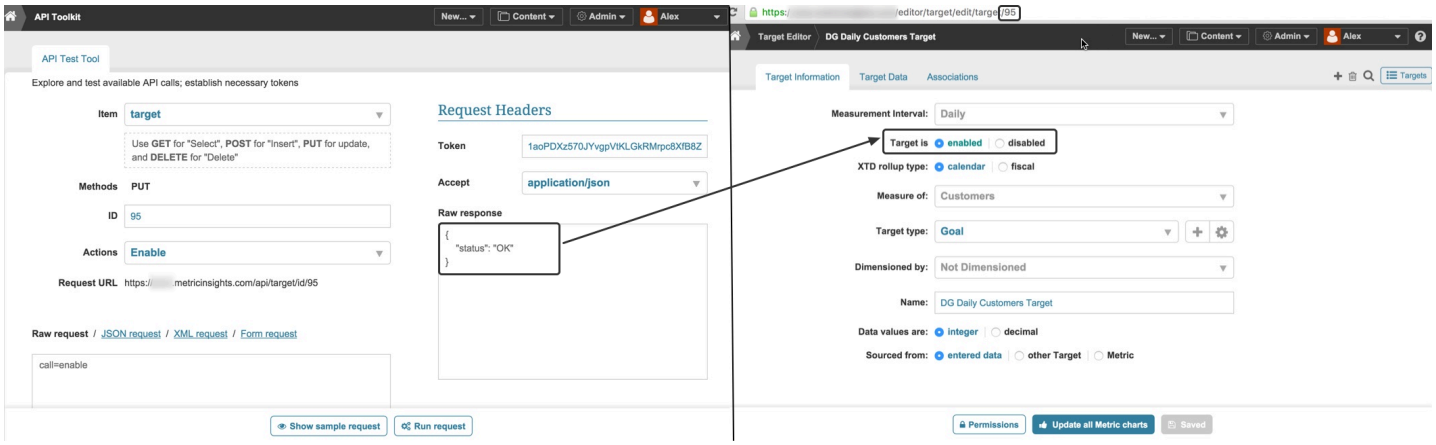
Request example:

```
{
    "name": "Daily Revenue Target",
    "target_type": "1",
    "measure": "3",
    "dimension": "0",
```

```
    "measurement_interval": "3",
    "xtd_rollup_type": "calendar",
    "target_value_type": "integer"
}
```

**NOTE:** Target Name and combination of Target Type, Measure, Dimension and Measurement Interval must be unique across the system.

## Enable Target




Item: **target**

Method: **PUT**

Actions: **Enable**

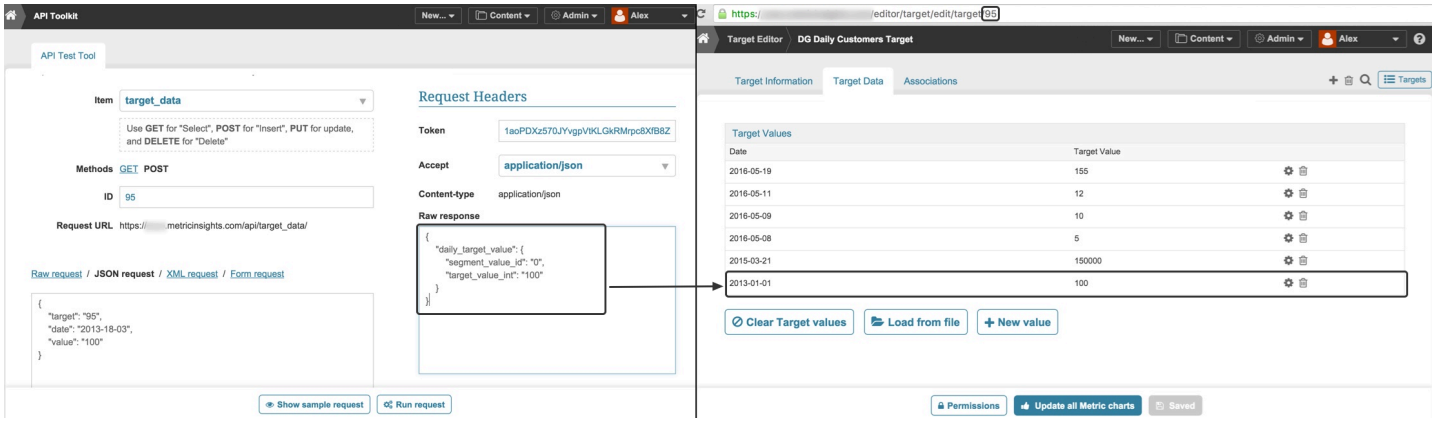
Header:

```
Accept:application/json
Token:1aoPDXz570JYvgpVtKLGkRMrpc8XfB8ZGEYGIteYKjG4hmoIsw
```

Request example:

```
call=enable
```

# Add Datapoint




Item: **target_data**

Method: **POST**

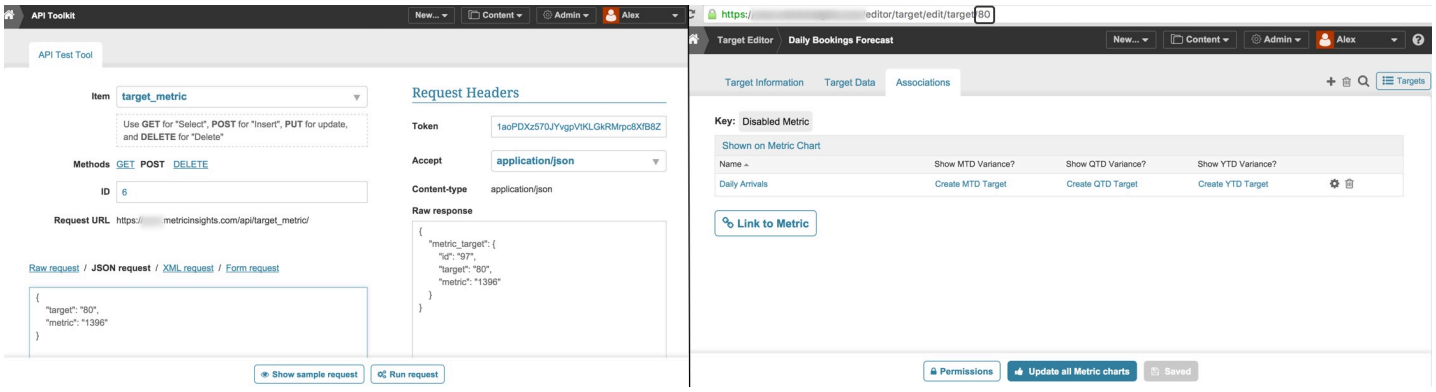
Header:

```
Accept:application/json
Token:1aoPDXz570JYvgpVtKLGkRMrpc8XfB8ZGEYGIteYKjG4hmoIsw
```

Request example:

```
{
    "target": "95",
    "date": "2013-18-03",
    "value": "100"
}
```

# Add Association




Item: **target_metric**

Method: **POST**

Header:

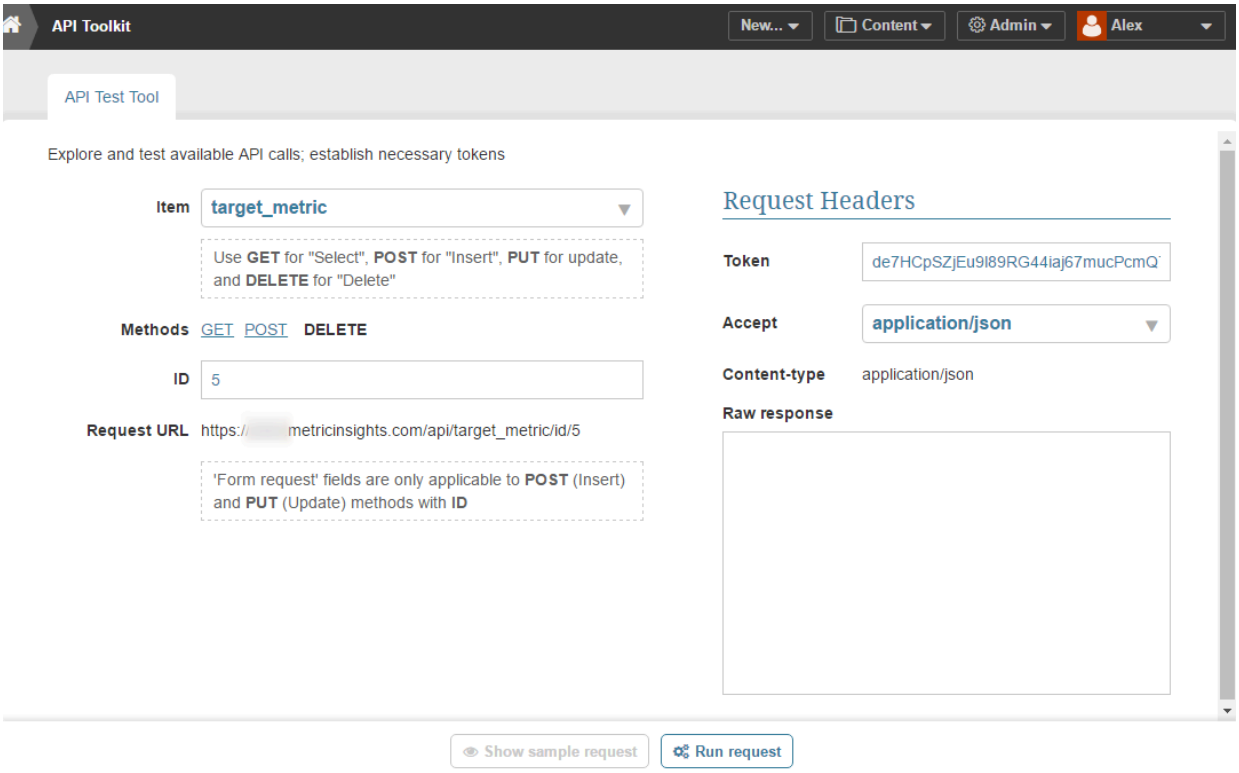
```
Accept:application/json
Token:1aoPDXz570JYvgpVtKLGkRMrpc8XfB8ZGEYGIteYKjG4hmoIsw
```

Request example:

```
{
    "target": "80",
    "metric": "1396"
}
```

In this example an association between a Target 'Daily Bookings Forecast' (id 80) and a metric 'Daily Arrivals' (id 1396) has been established. In the **Raw Response** screen the id of this association is given ("id": "97").

## Delete Association




Item: **target_metric**

Method: **DELETE**

Header:

```
Accept:application/json
```

```
Token:de7HCpSZjEu9l89RG44iaj67mucPcmQ7dUF2bV3WwNs4obbf2P
```

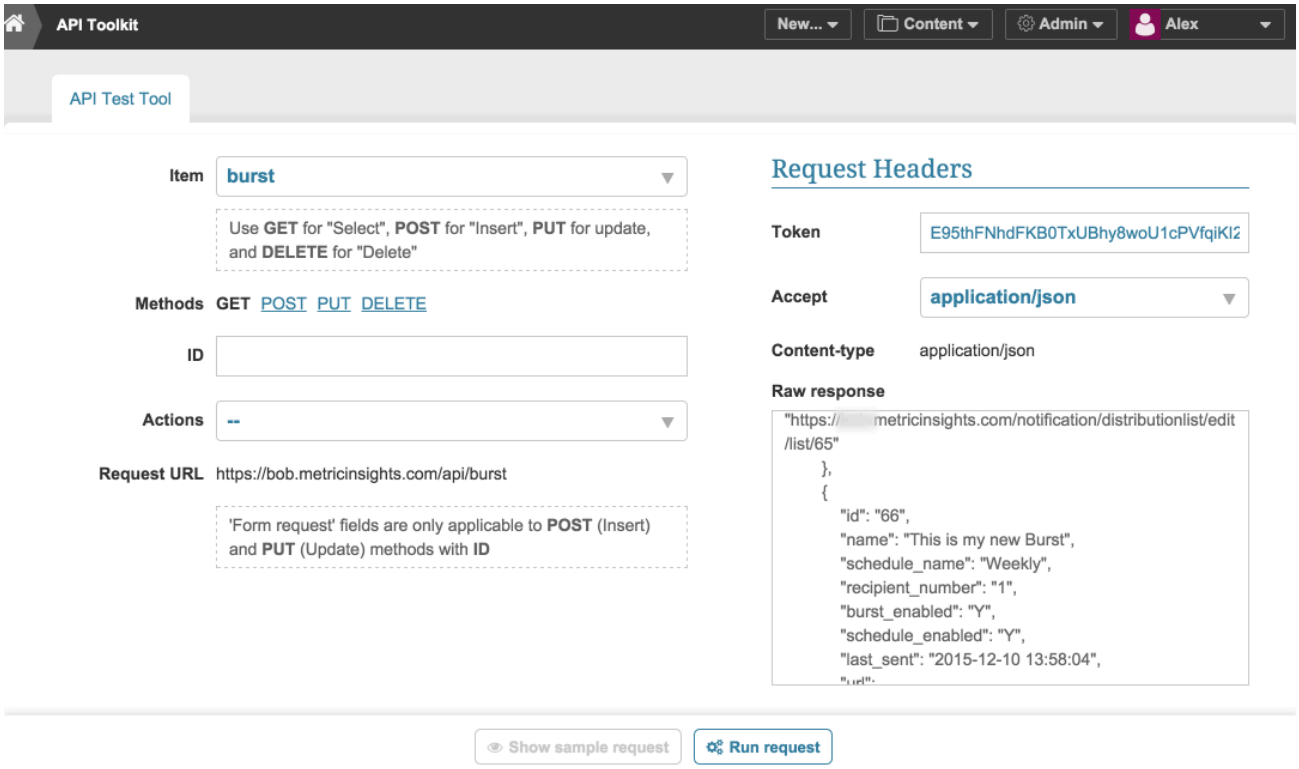
# 3.9 Bursts API

This article provides sample API calls to manipulate **Bursts**.

**PRE-REQUISITES:**

- Set up API access
- Verify that you have API access OR get token call via curl since all API calls require a token

## Get all Bursts




This procedure allows getting a list of all Bursts available to be sent:

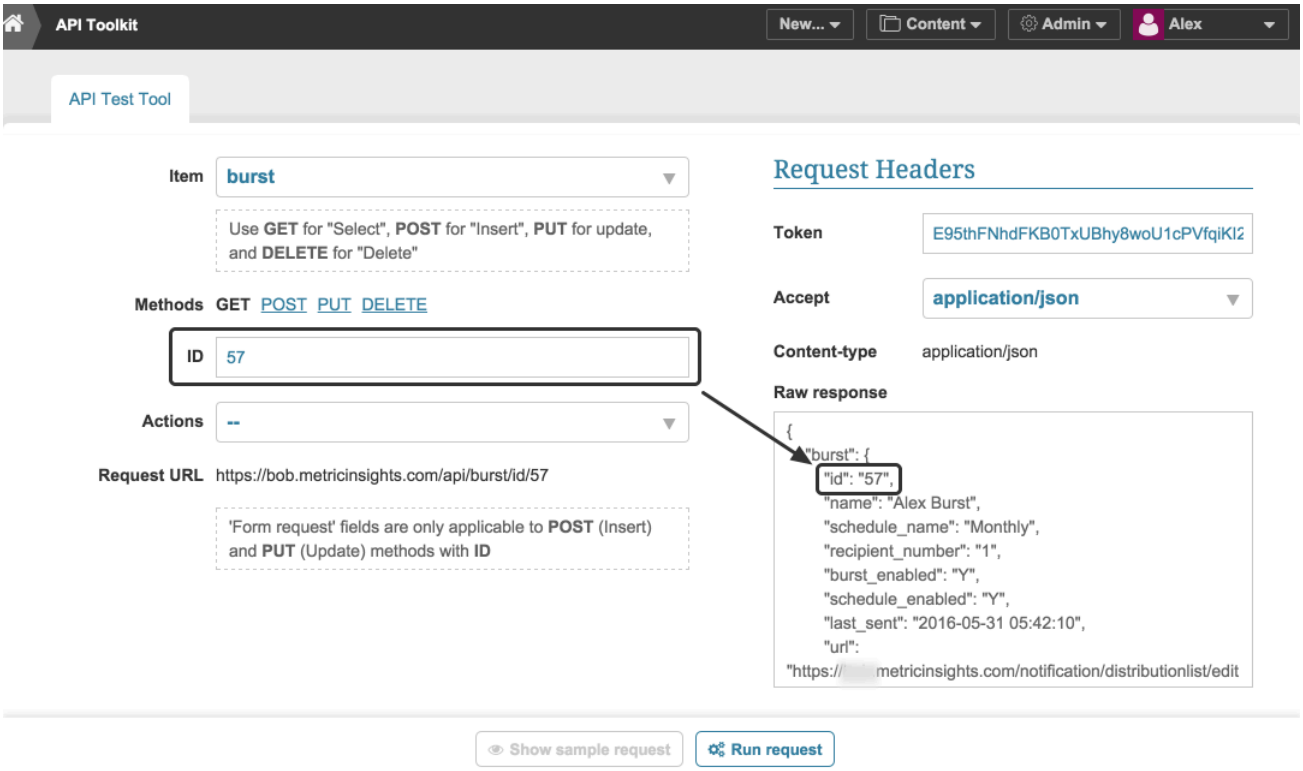
Item: **burst**

Method: **GET**

Header:

```
Accept:application/json
Token:pp7463NsgbECHh5XIwfMLS5of8qFD5j4G3QohTCMh72kHyXH3K
```

# Get details of a specific Burst




This procedure allows getting information on a specific burst by entering its **ID**:

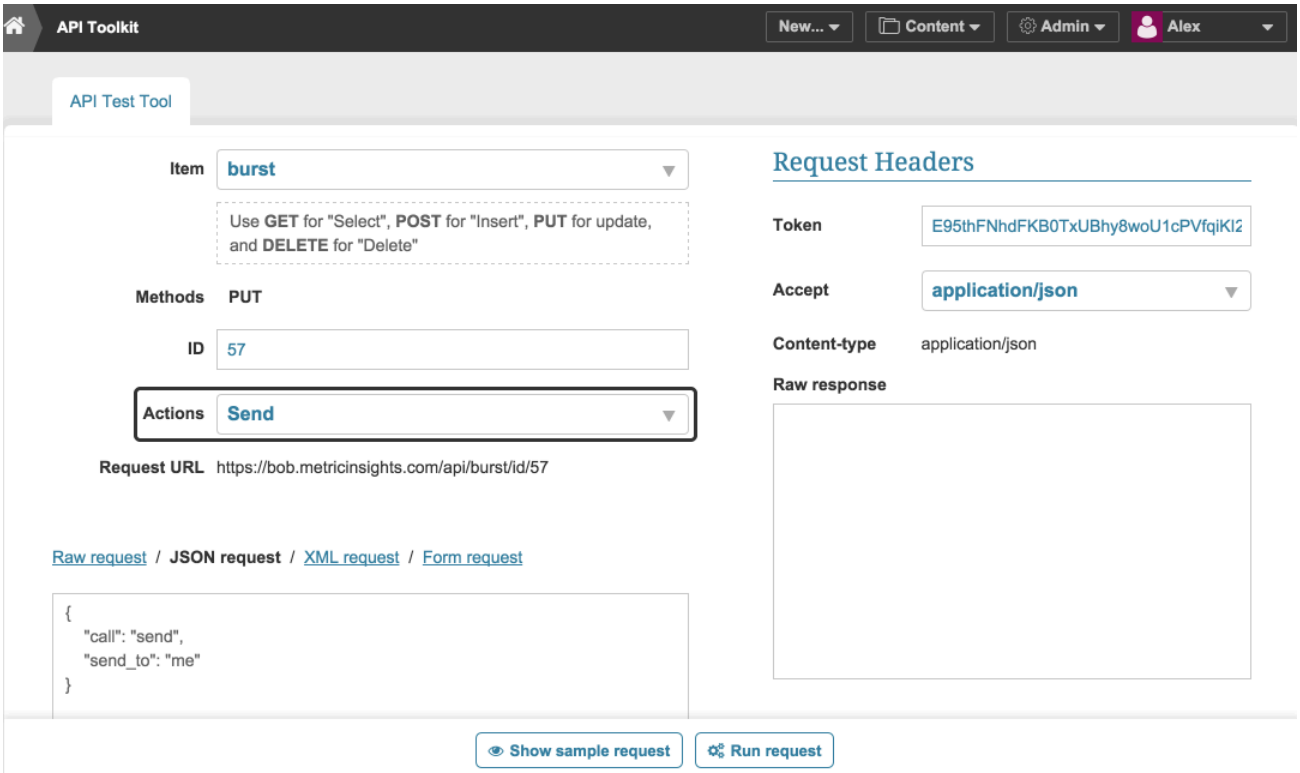
Item: **burst**

Method: **GET**

Header:

```
Accept:application/json
Token:pp7463NsgbECHh5XIwfMLS5of8qFD5j4G3QohTCMh72kHyXH3K
```

# Send a Burst




Item: **burst**

Method: **GET**

Header:

```
Accept:application/json
Token:pp7463NsgbECHh5XIwfMLS5of8qFD5j4G3QohTCMh72kHyXH3K
```

Request example:

```
{
    "call": "send",
    "send_to": "all"
}
```

- **"send_to"**: specify whether the burst shall be sent to "all" or to "me"

# 3.10 Dataset API

This article provides sample API calls used for obtaining a list of Datasets, creating and modifying individual Datasets, as well as obtaining and manipulating their data.

To query Datasets data and metadata, the following **Item** functions are used:

1. **dataset** (to fetch metadata, to edit and to apply Access Maps to Datasets)
   - *available since Version 5.4.0*
2. **dataset_data** (to obtain data from specific Datasets)
   - *available since Version 5.3.1*
3. **dataset_instance** (to get a list of Dataset Instances)
   - *available since Version 5.4.0*

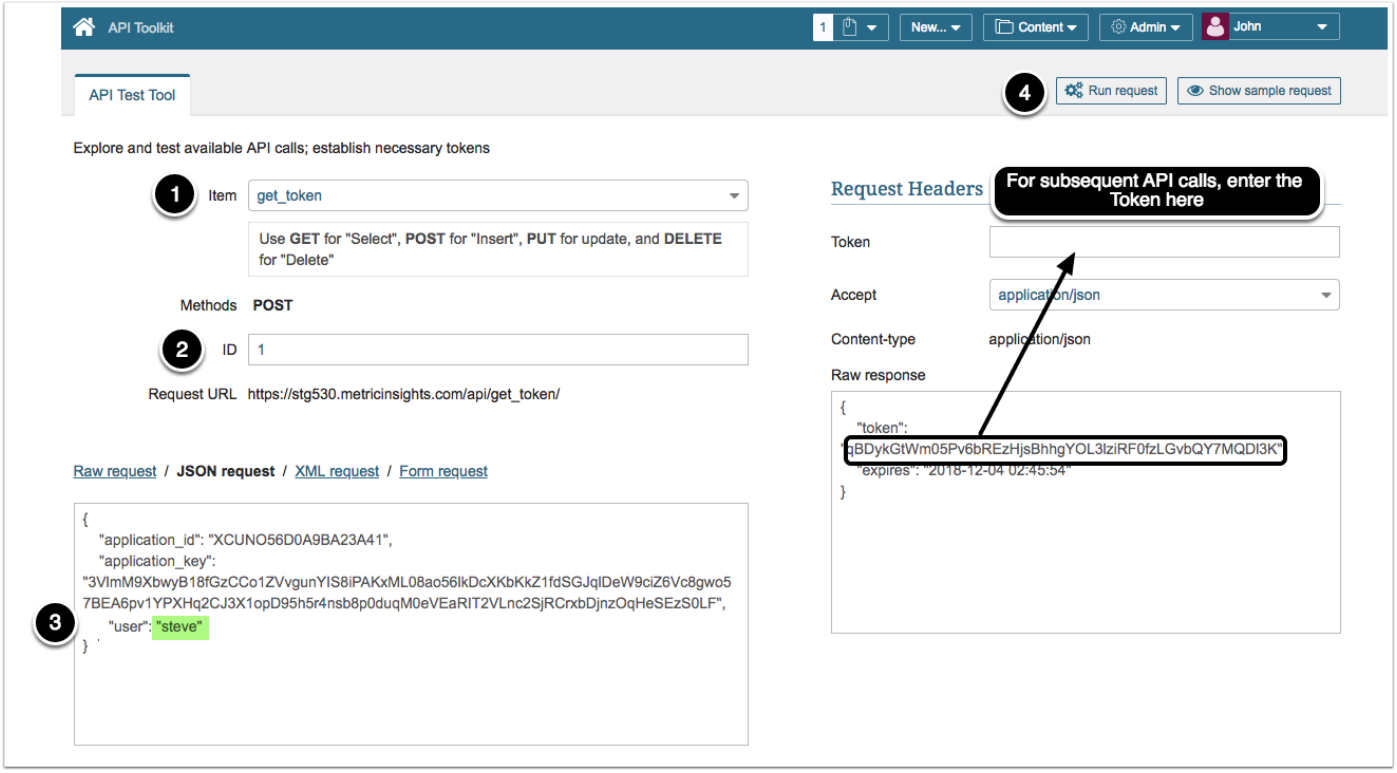
**PREREQUISITES:**

To be able to make API calls, you need to obtain an access Token

For more information, see:

- Set up API access
- Verify that you have API access OR Get Token API via curl

# Get a Token using API Toolkit




Go to **Admin > Utilities > API Toolkit > API Test Tool**

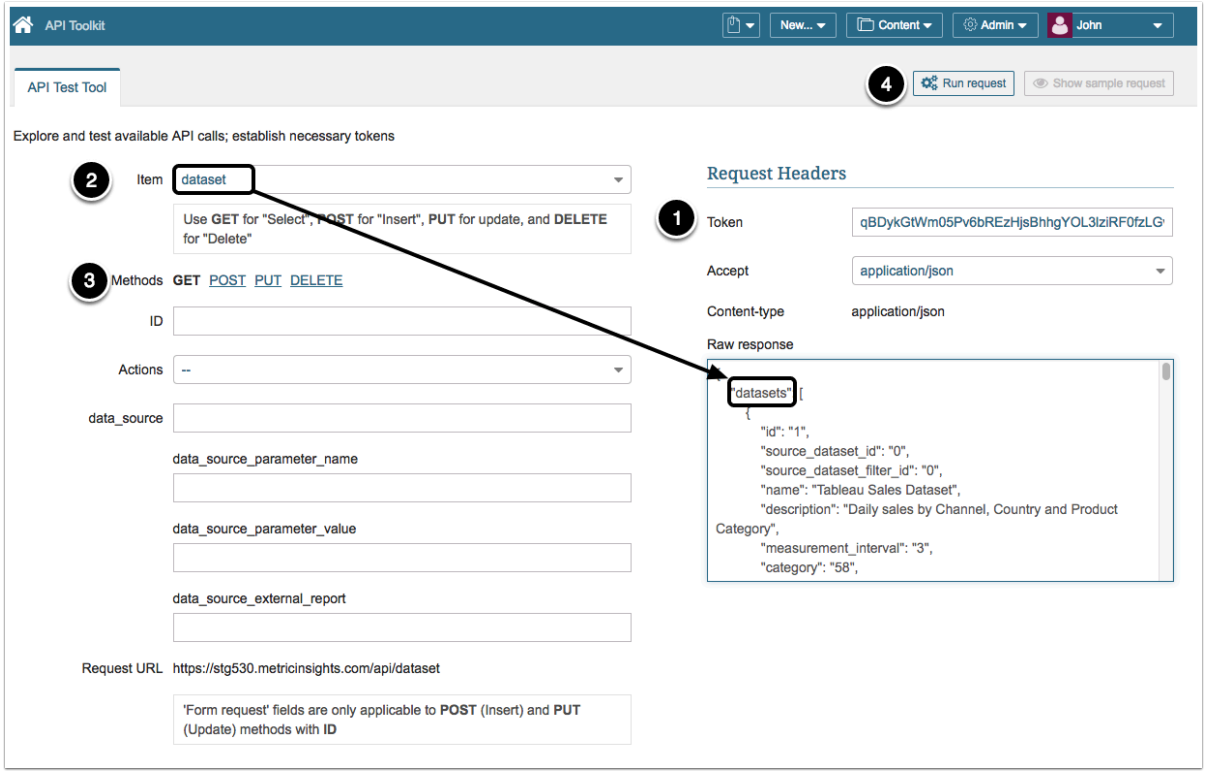
1. **Item:** from the dropdown, select **get_token**
2. **ID:** enter External Application ID
   - For details, refer to [Set up API Access]() and [Test API Access / Get API Token via MI API Test Tool]()
3. **Request:** input User Name after **"user":** in the Request
4. **Run request:** click to get the Token

> ⚠ Since all API calls require a Token, you need to copy the token from the **Raw Response field** and enter it in the provided **Token field**.

# Obtain a list of Datasets in Metric Insights

You can use any of the available Datasets as a model for a new Dataset.

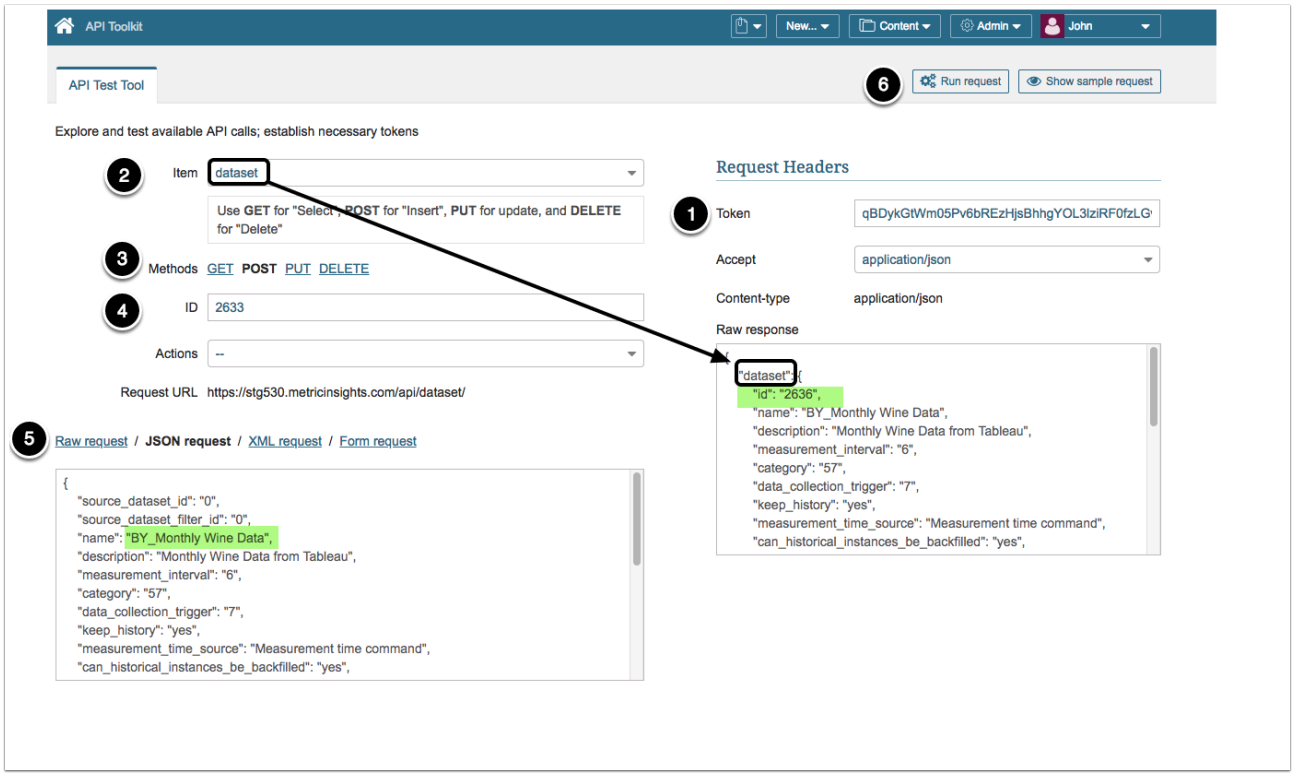
The example provided below shows how create a new Dataset from an existing Dataset.




1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** from the dropdown, select **dataset**
3. **Methods:** choose GET
4. **Run request:** click to get a list of all Datasets and their metadata

*Raw response* will contain information about Datasets in Metric Insights.

# 1. Create a new Dataset




1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** from the dropdown, select **dataset**
3. **Methods:** choose POST
4. **ID:** input the ID of an existing Dataset that you want to use as a model for a new Dataset
   - After you input the Dataset ID, the Request field will display the settings for that Dataset
5. **Request:** type in the **name** for the new Dataset
   - Optionally, you can change the Dataset description, snapshotting options and other settings
6. **Run request:** click to create a new Dataset

*Raw response* will contain metadata for the new Dataset. The newly created Dataset will be assigned an ID that can be used in subsequent API calls (e.g.: when validating, enabling and updating your Dataset)

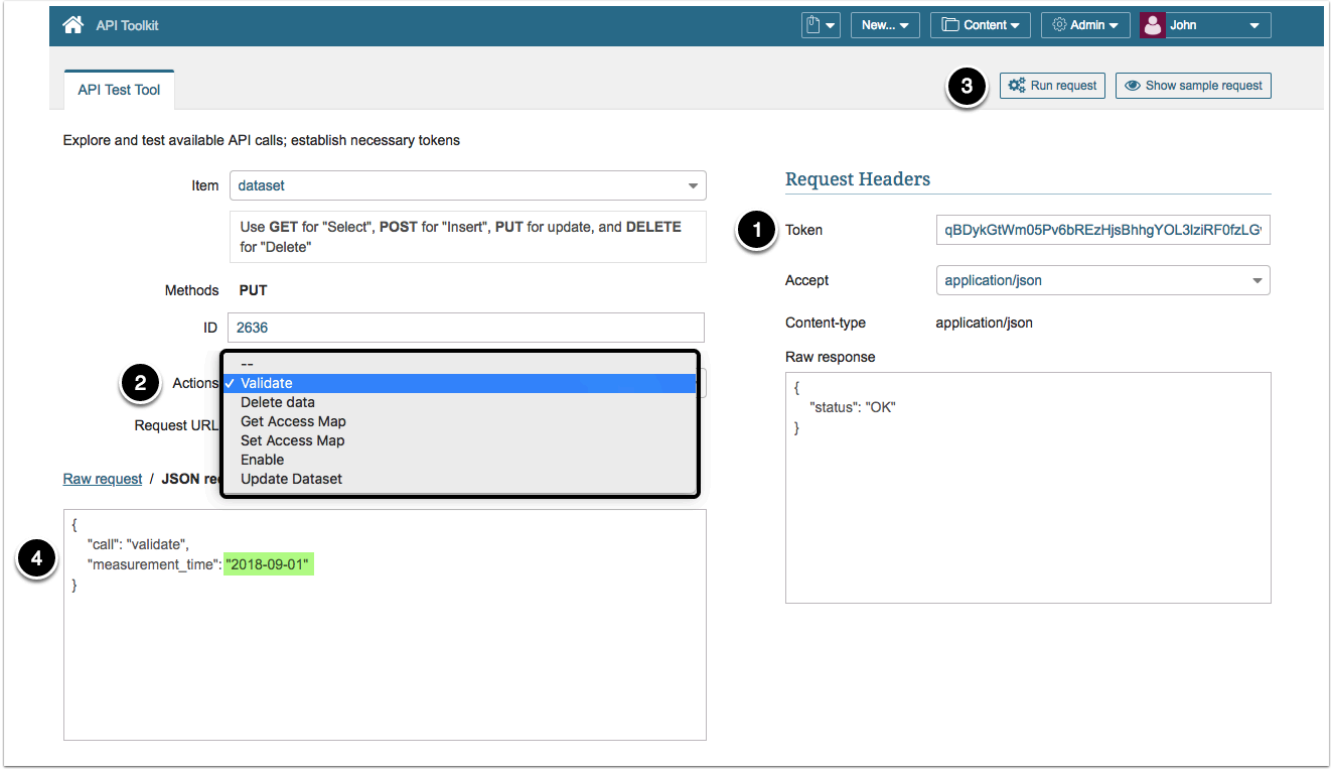
# 2. Modify your Dataset via the API action calls

PUT Method (used in API calls to Metric Insights) supports the following actions:

1. Validate

2. Delete data
3. Get Access Map
4. Set Access Map
5. Enable
6. Update Dataset

## 2.1. Validate and enable your Dataset

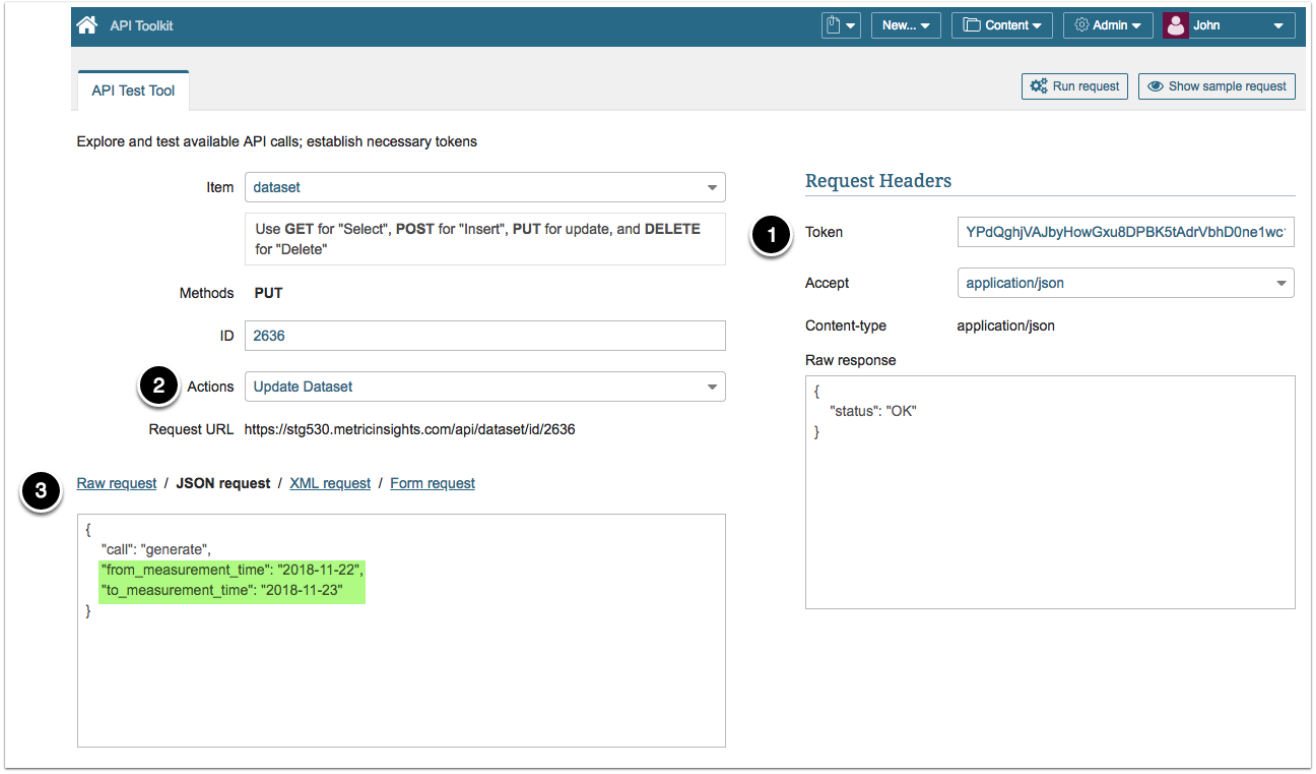



1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
   - The same Token can be used for a series of subsequent API calls
2. **Actions:** from the dropdown, select the relevant option
3. **Request:** if you choose to validate, enter the date against which Dataset data will be validated
   - ***measurement_time*** parameter must be in the following format: *year-month-day*
4. **Run request:** click to run the API action call

After you ***validate*** and ***enable*** your Dataset, you can get the Dataset data by following the steps described in Section 4 of this article.

## 2.2. Update a Dataset

After modifying your Dataset, update the Dataset to make the changes effective.

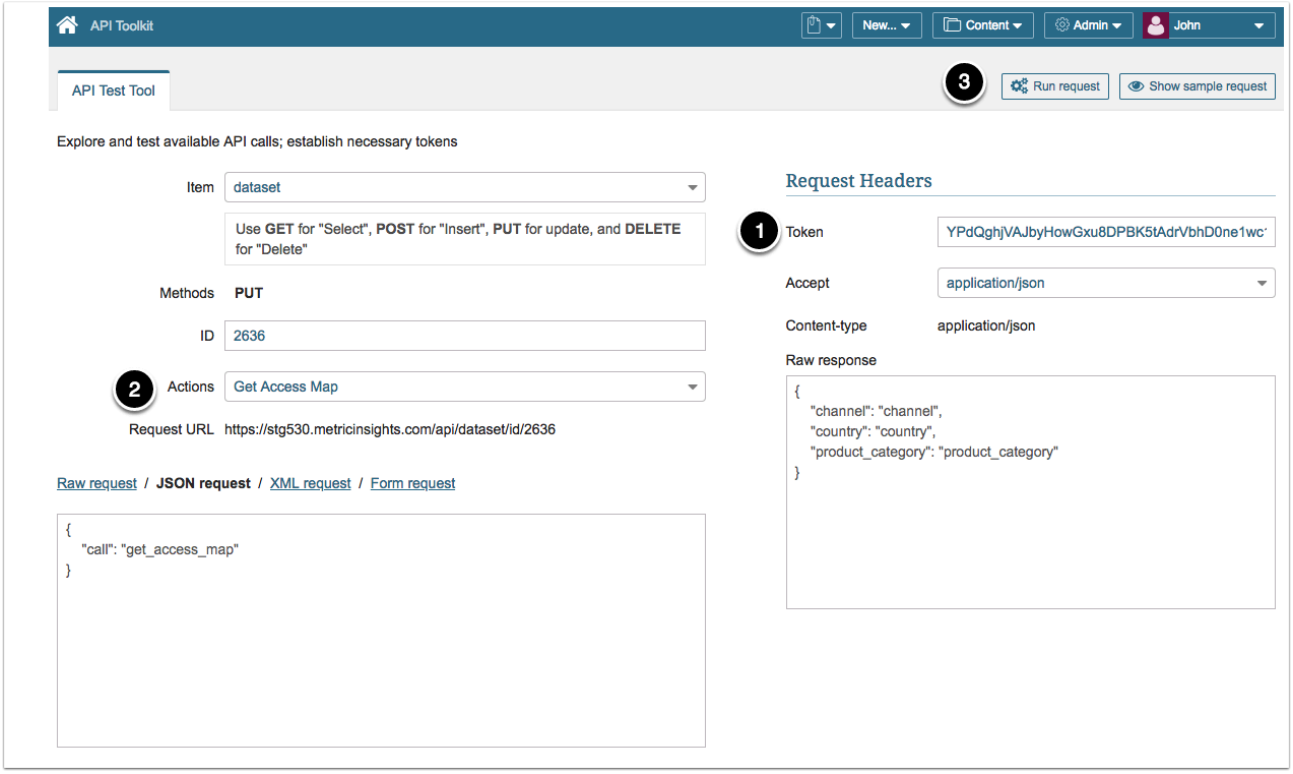



1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Actions:** from the dropdown, select ***Update Dataset***
3. **Request:** measurement_time can be changed from default
   - *measurement_time* parameter must have the following format: *year-month-day*
4. **Run request:** click to run the API action call

If the API update call is successful, the **"status":"OK"** message will be displayed in the *Raw response*

## 2.3. Query a Dataset Access Map

Dataset API calls allow you to query the Access (User) Map of any existing Dataset that has an Access (User) Map.



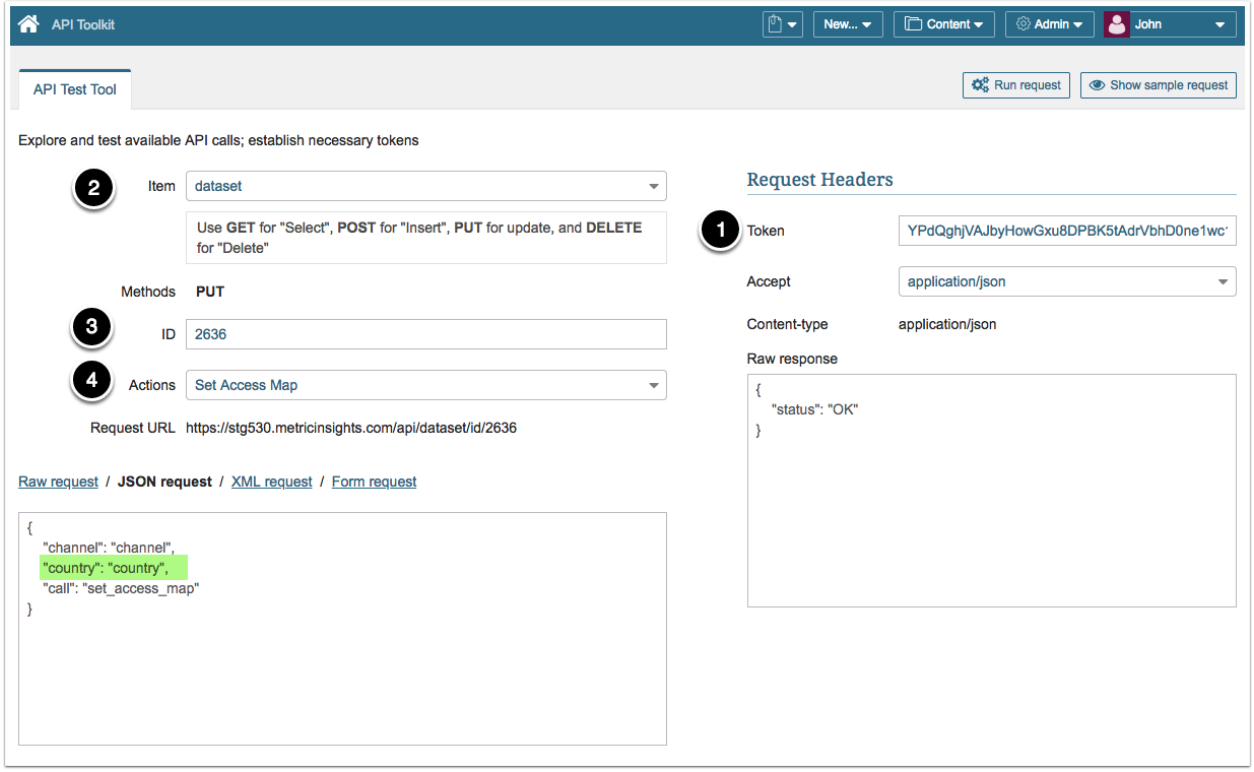

1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Actions:** from the dropdown, select ***Get Access Map***
3. **Run request:** click to run the API action call

*Raw response* will contain the Dataset columns mapped to Access Map columns.

For more information on Access (User) Maps, refer to:

- Create a User Map
- Apply a User Map to a Dataset

## 2.4. Apply an Access Map to a Dataset

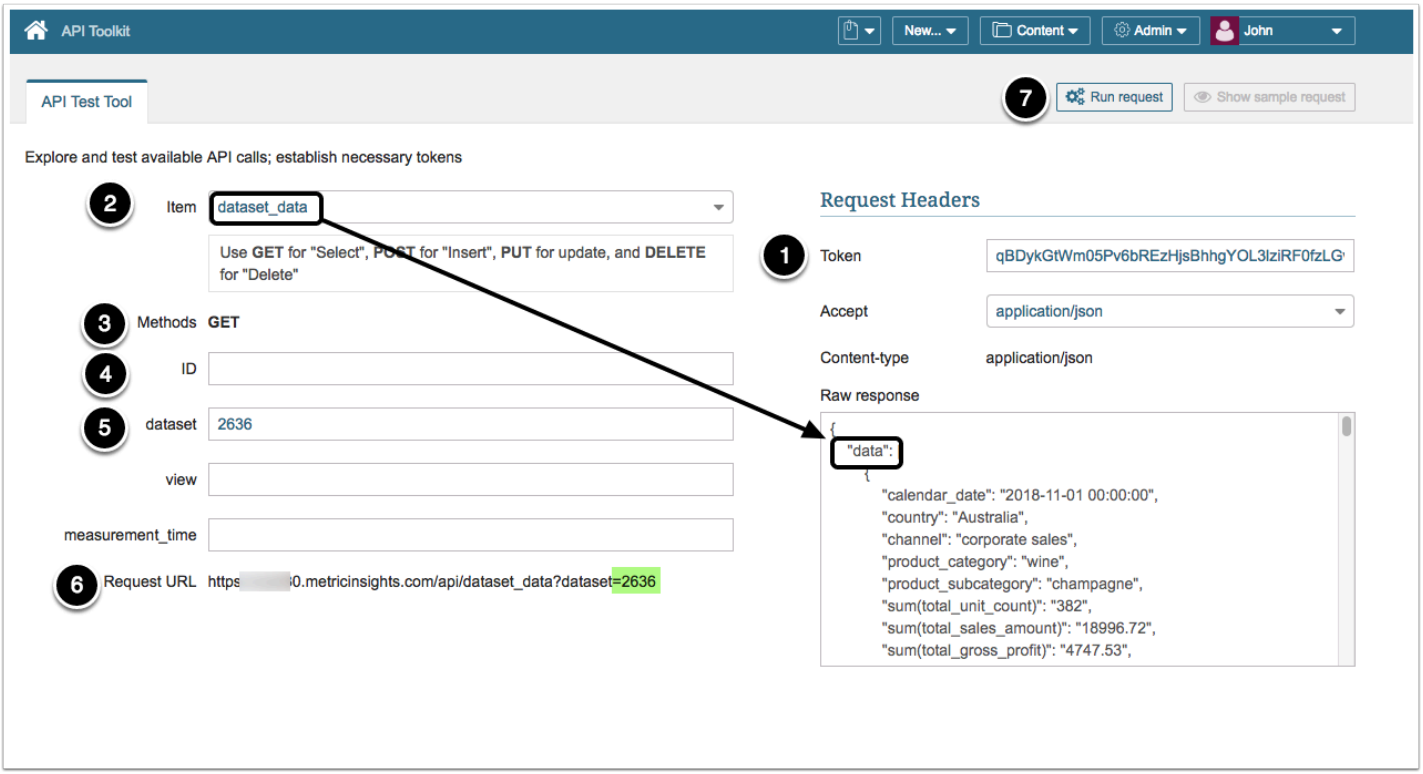



1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **ID:** input the ID of a Dataset
3. **Actions:** choose ***Set Access Map***
   - After you input the Dataset ID, the **Request field** will display Dataset columns that have already been mapped to Access (User) Map columns
   - **Note:** for the Set Access Map action to work, at least one Access (User) Map column should already be mapped to the corresponding Dataset column
4. **Request:** input additional columns from the Dataset and User Map that will be mapped against one another
   - To get a list of all available columns in an Access (User) Map, use the **dataset_data** item since Acces Maps are treated as Datasets by the system. For more information, check Fetch Dataset Data
5. **Run request:** click to run the API action call

# 3. Fetch Dataset Data

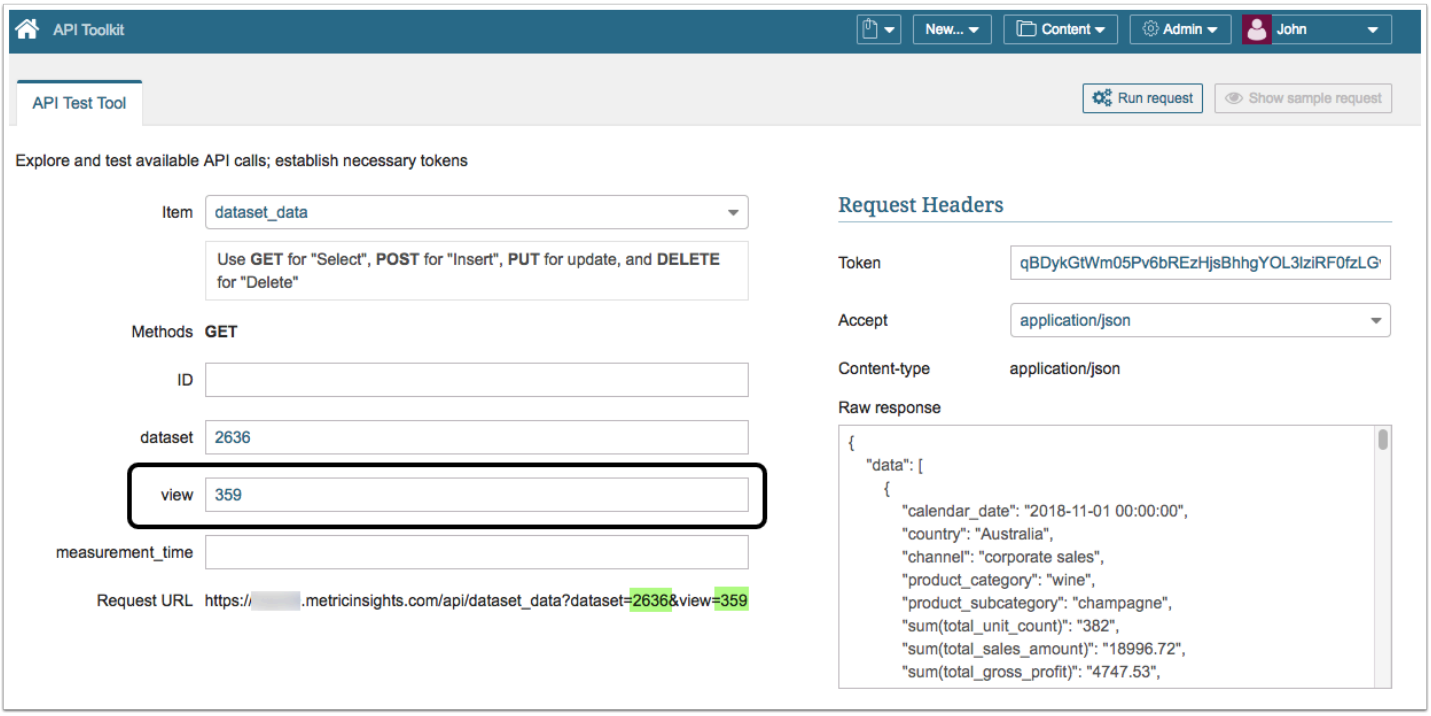
> You can either query all the data contained in that (All Data) View of the Dataset or specify a particular View and/or Time Instance.

## 3.1. Get data from an "All Data" View




1. **Token:** enter a Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** select "*dataset_data*" from the drop-down
3. **Methods:** GET
4. **ID:** must be left blank
5. **dataset:** enter the Dataset ID. [Learn how to find your Dataset ID]
6. **Request URL** (generated automatically): https://yourcompany.metricinsights.com/api/dataset_data?dataset=<id>
7. **Run request**

## 3.2. Get data from a Specific View

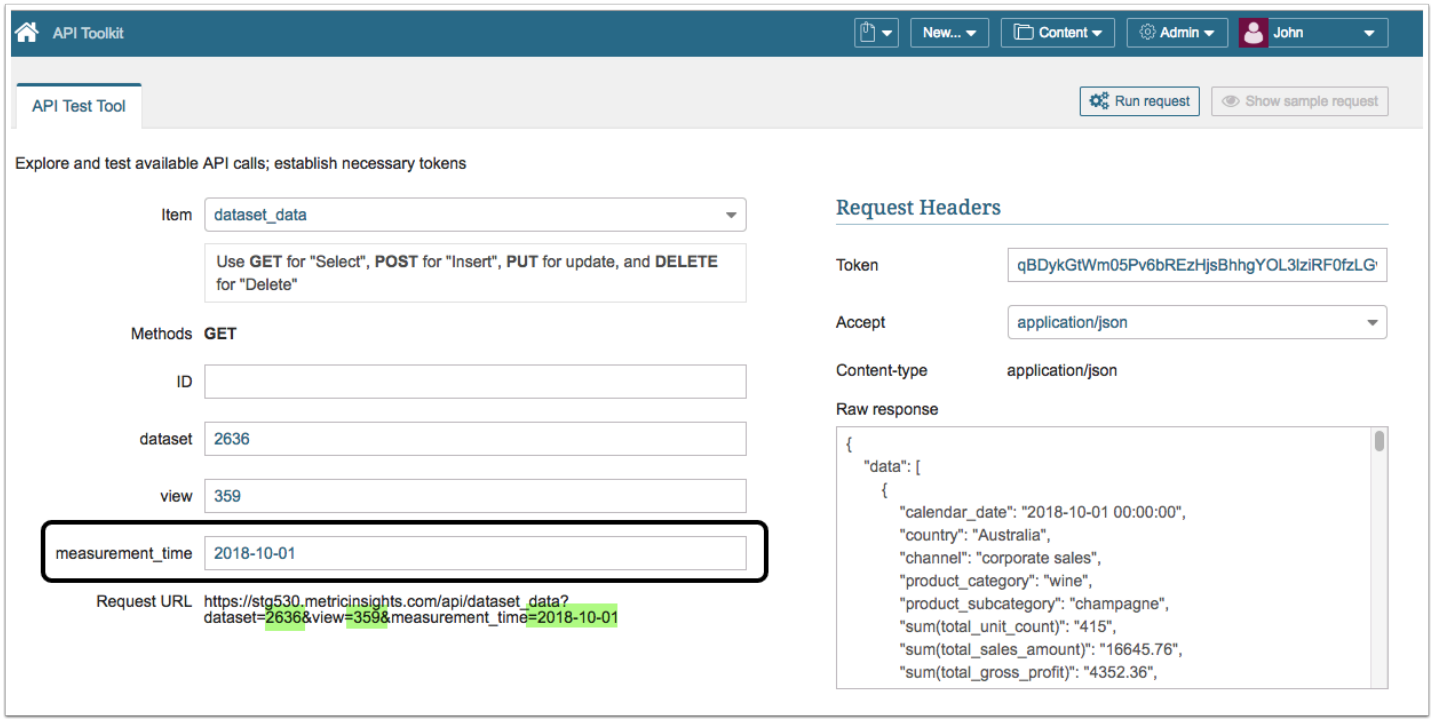



To see a specific View, additionally enter a View ID. Learn how to find you Dataset View ID.

**Request URL** (generated automatically): https://yourcompany.metricinsights.com/api/dataset_data?dataset=<id>&view=<id> .

## 3.3. Get data from a Specific Dataset Instance

In the example below, the data displayed in the *Raw Response* was pre-filtered for a specific View and Date.




1. **dataset:** required
2. **View ID:** optional
3. **Measurement_time:** type in the date that you need to fetch your information for (only the following format is supported: *yyyy-mm-dd*). If Dataset is NOT *"set to keep history"*, the *"measurement_time"* parameter is ignored during the API call. The *Raw Response* returns data exclusively for the last time the data was collected.

**Request URL** (generated automatically): https://yourcompany.metricinsights.com/api/dataset_data?dataset=<id>&view=<id>&measurement_time=yyyy-mm-dd .

# Where can I find Dataset ID?

Dataset ID can be found in the the web browser URL of the *Dataset Viewer or Dataset Editor*:

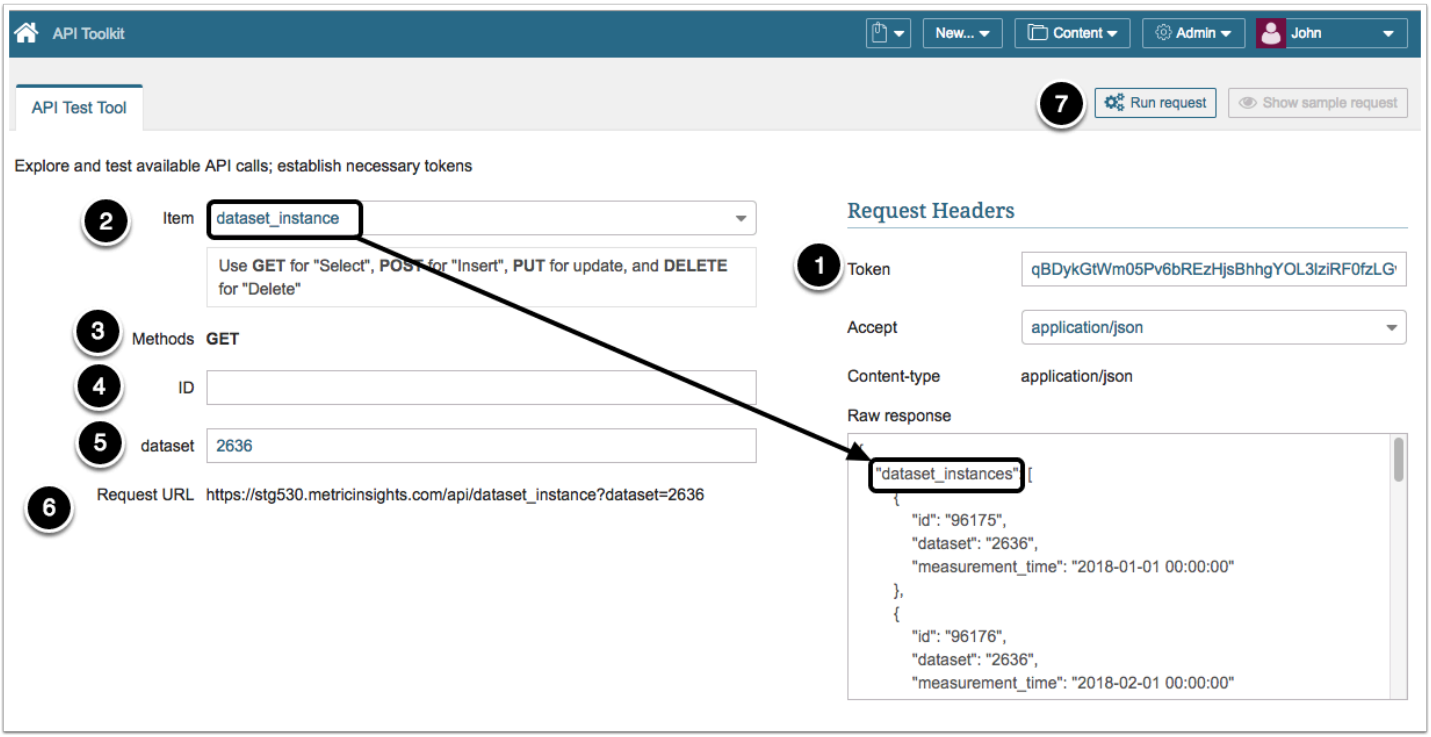
**https://stg530.metricinsights.com/dataset/index/index/ds/**474

# Where can I find Dataset View ID?

Dataset View ID can be found in the the web browser URL of the *Dataset Viewer.*

**https://stg530.metricinsights.com/dataset/index/index/ds/474/filter/**211

# 4. Query all Dataset Instances




1. **Token:** enter a Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** select "*dataset_data*" from the drop-down
3. **Methods:** only GET is currently supported
4. **ID:** must be left blank
5. **dataset:** enter the Dataset ID
6. **Request URL** (generated automatically): [https://yourcompany.metricinsights.com/api/dataset_data?dataset=<id>](https://yourcompany.metricinsights.com/api/dataset_data?dataset=<id>)
7. **Run request**

*Raw response* will contain a list of Dataset Instances

# 3.11 Data Source API

This article provides sample API calls for Data Sources in Metric Insights.

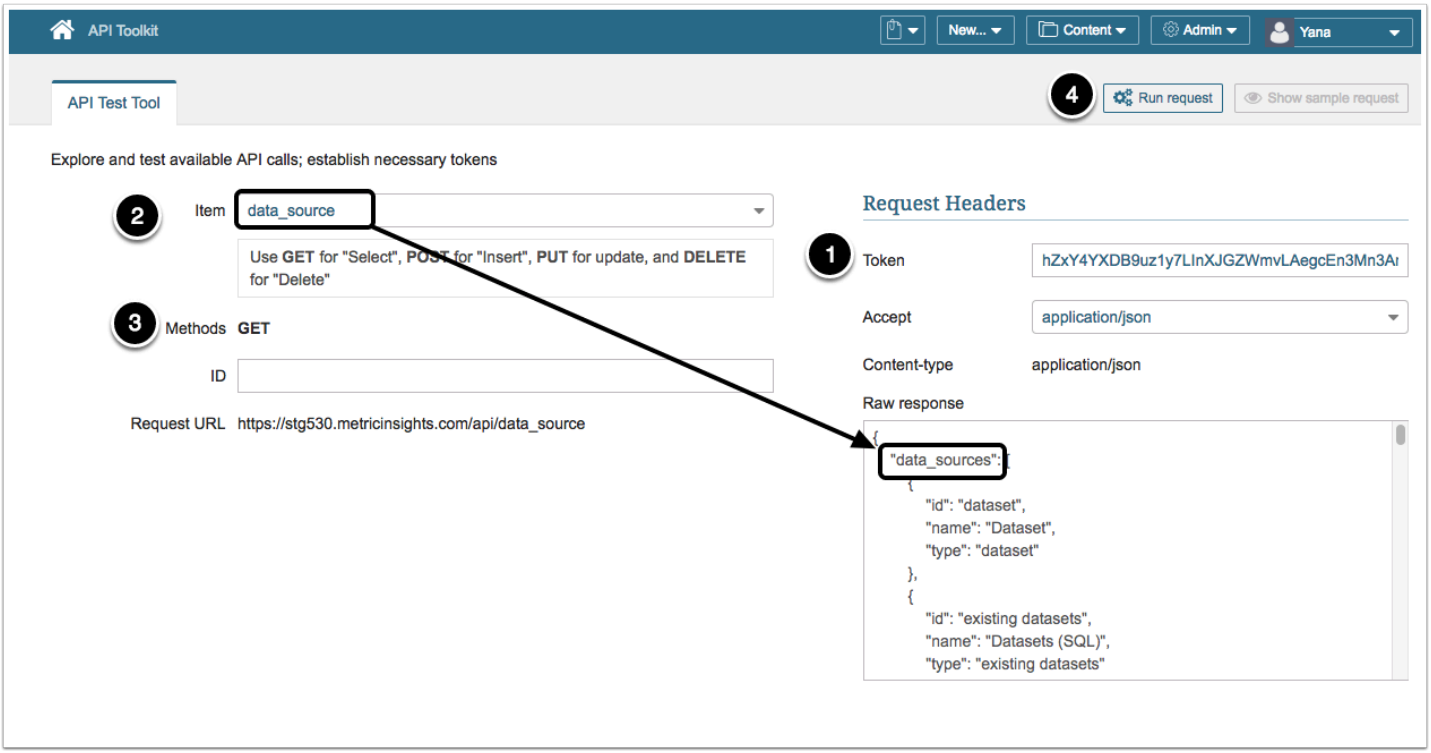
You can use the following **Item** functions:

1. **data_source** (to fetch a list of Data Sources and their metadata)
2. **data_source_sql** (to create, delete, edit SQL Data Sources and obtain a list of SQL Data Sources)
3. **data_source_plugin** (to obtain a list of Plugin Data Sources)
4. **data_source_parameter** (to query Parameters used for a particular Plugin Data Source)
5. **data_source_external_report** (to obtain a list of External Reports for a particular Plugin Data Source)

**PREREQUISITES:**

1. Set up API access
2. Verify that you have API access
3. Obtain a token via a get token call since all API calls require a token

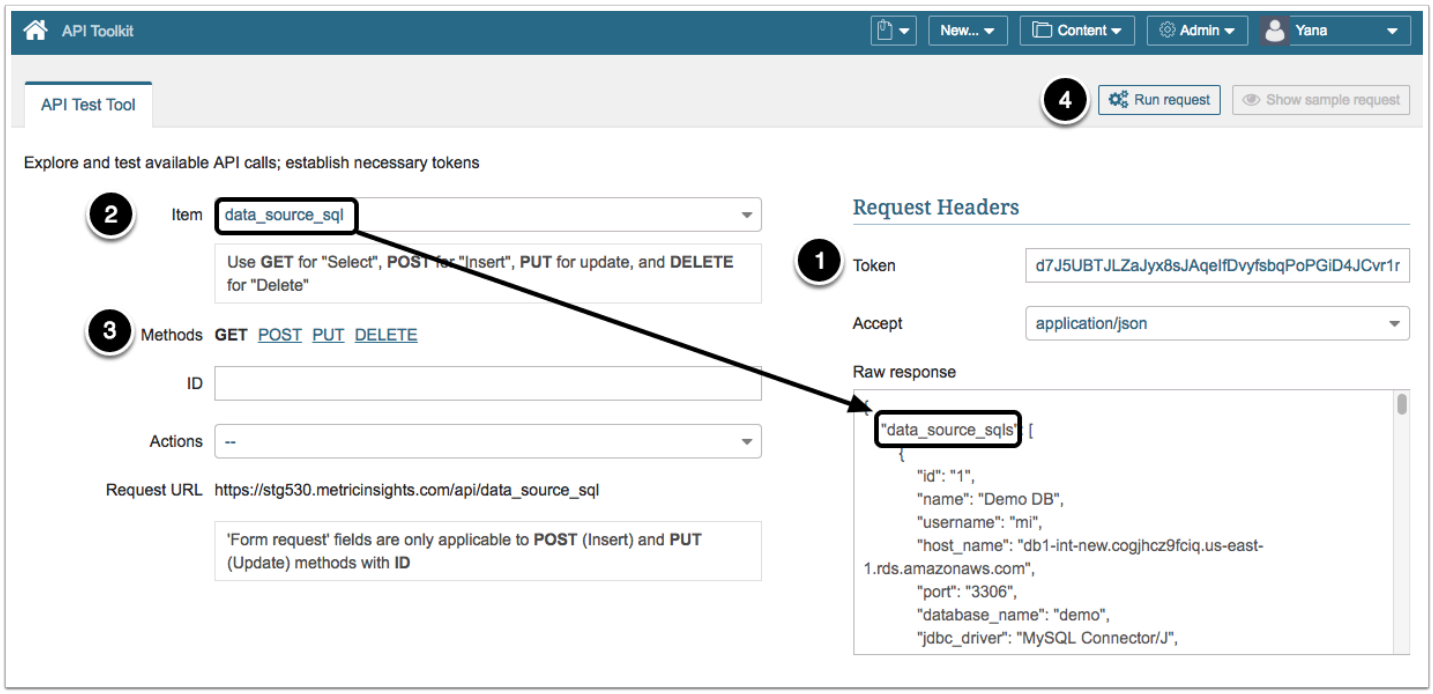
## Obtain a list of all Data Sources



1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** from the dropdown, select **data_source**
3. **Methods:** GET
4. **Run request:** click to get a list of all Data Sources and their metadata

*Raw response* will contain information about Data Sources in Metric Insights.

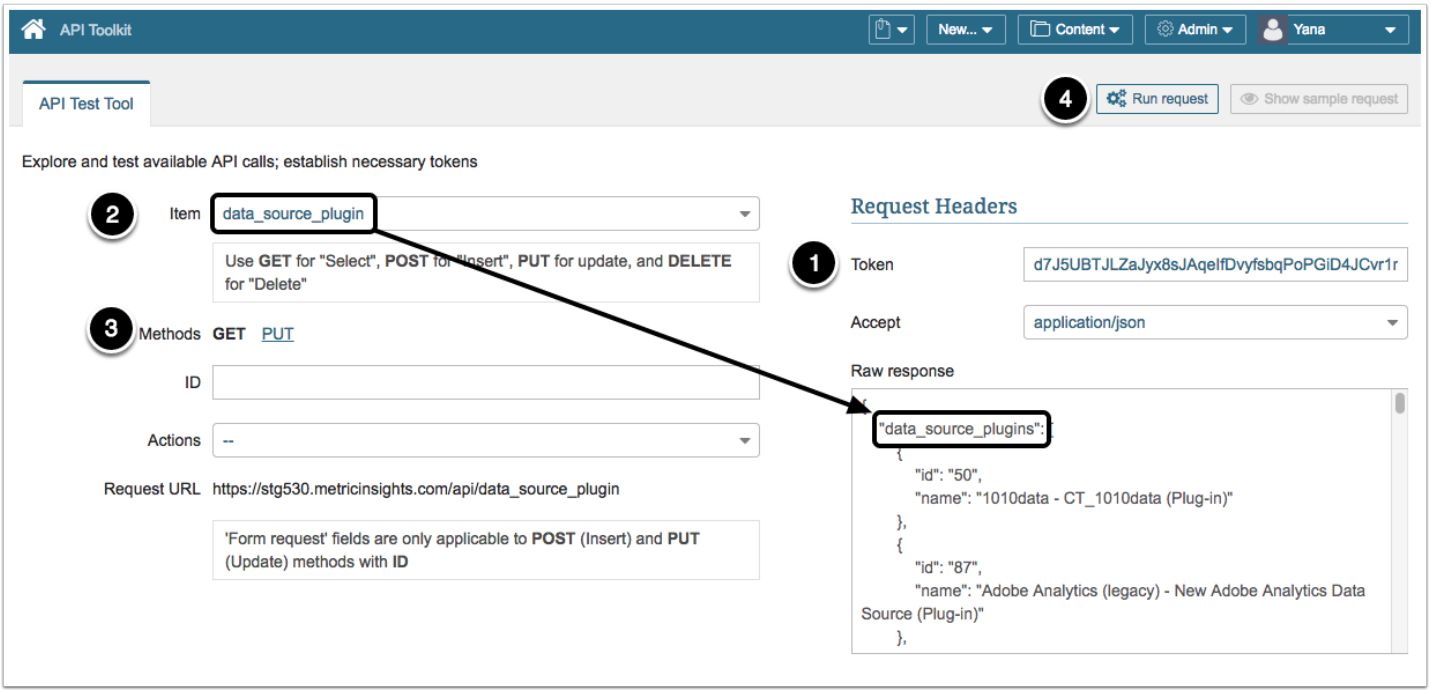
# Obtain a List of SQL Data Sources




1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** from the dropdown, select **data_source_sql**
3. **Methods:** GET
4. **Run request:** click to get a list of all SQL Data Sources and their metadata

*Raw response* will contain information about SQL Data Sources in Metric Insights.

# Obtain a list of Plugin Data Sources




1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** from the dropdown, select **data_source_plugin**
3. **Methods:** GET
4. **Run request:** click to get a list of all Plugin Data Sources and their metadata

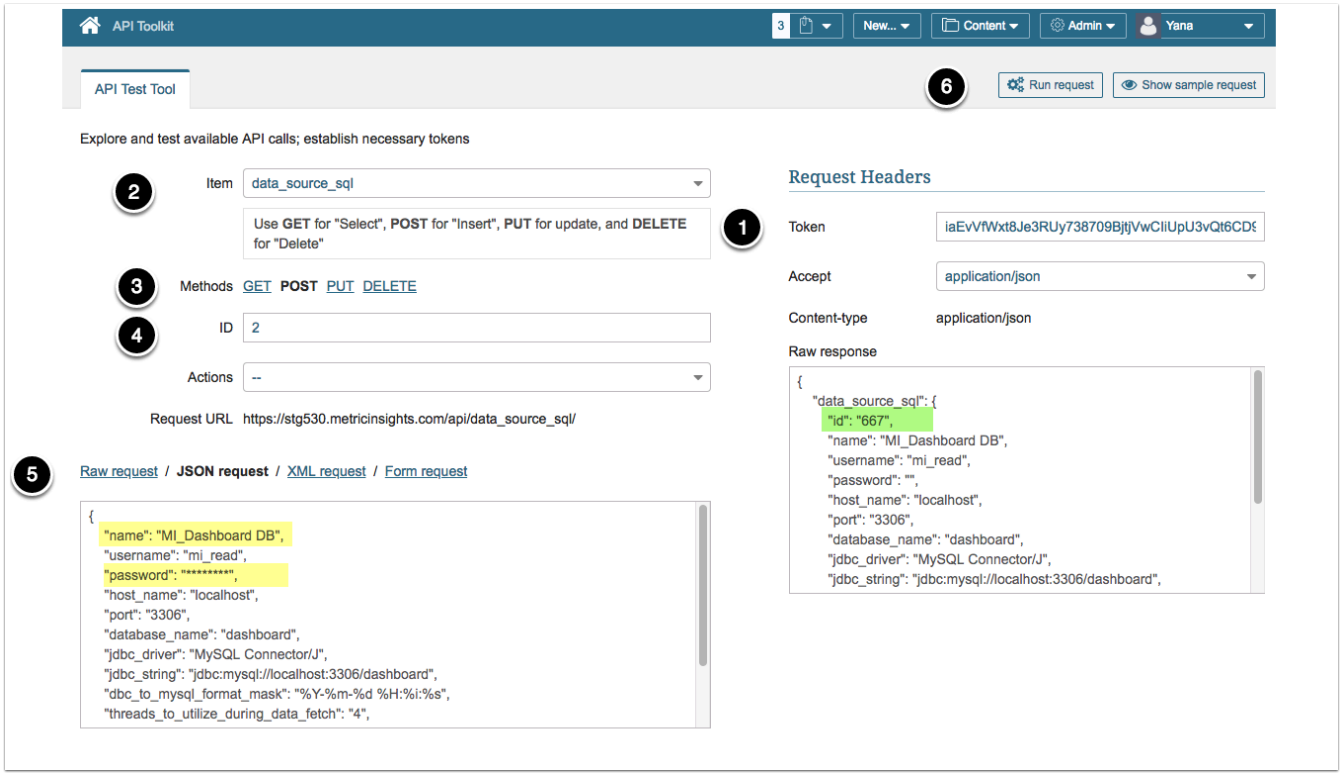
*Raw response* will contain information about Plugin Data Sources in Metric Insights.

# 1. Create a Data Source (applicable to SQL Data Sources)

You can use any of the available Data Sources as a model for a new Data Source.

The example provided below shows how create a new Data Source from an existing Data Source.

Currently **creating** and **editing** options are supported for SQL Data Sources only.




1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** select **data_source_sql**
3. **Methods:** choose POST
4. **ID:** input the ID of an existing Data Source that will be used to build a new Data Source
   - After you input the Data Source ID, the Request field will display the settings for that Data Source

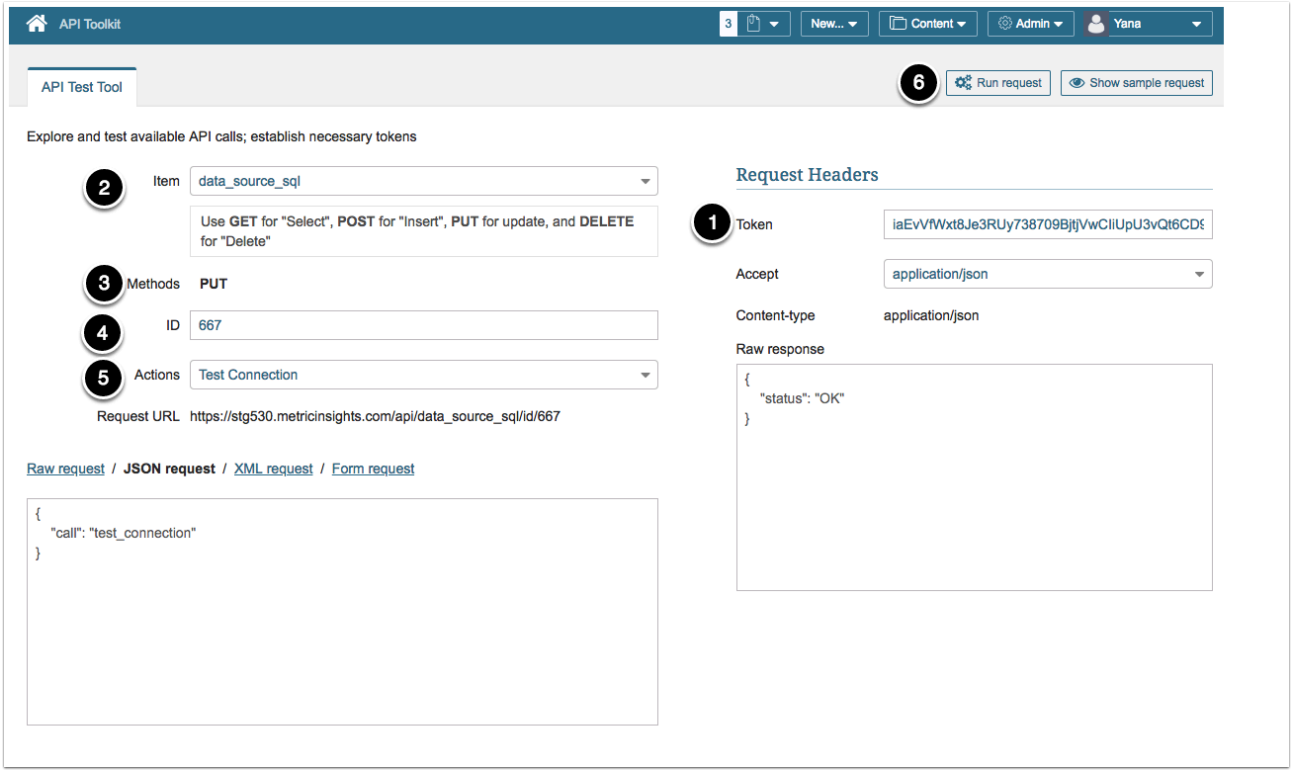
5. **Request:**
   - If you want to make a copy of an existing Data Source, input a meaningful **Name** and **Password** and keep the rest of the settings as they are
   - In case you want to use an existing Data Source as a model and query data from your own database, additionally input connection parameters that are relevant to you

6. **Run request:** click to create a new Data SourceAsok

*Raw response* will contain the new Data Source settings.

The newly created Data Source will be assigned an **ID** that can be used to Test Connection during a subsequent API call.

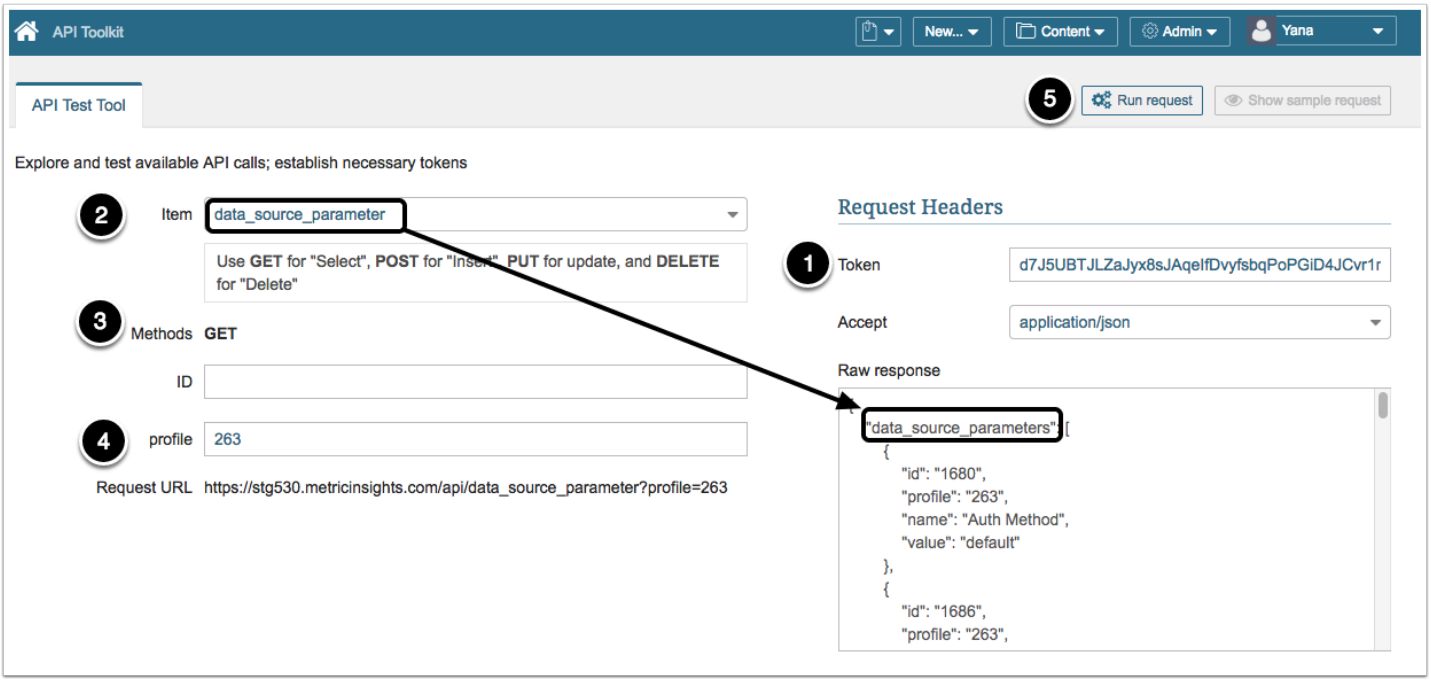
# 2. Test Connection (applicable to SQL Data Sources)




1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** select **data_source_sql**
3. **Methods:** PUT is used
4. **ID:** input the ID of the Data Source whose connection will be tested
5. **Actions:** from the dropdown, select Test Connection
6. **Run request:** click to run the API action call

If the call is successful, the Raw response will display **"status: OK"** message

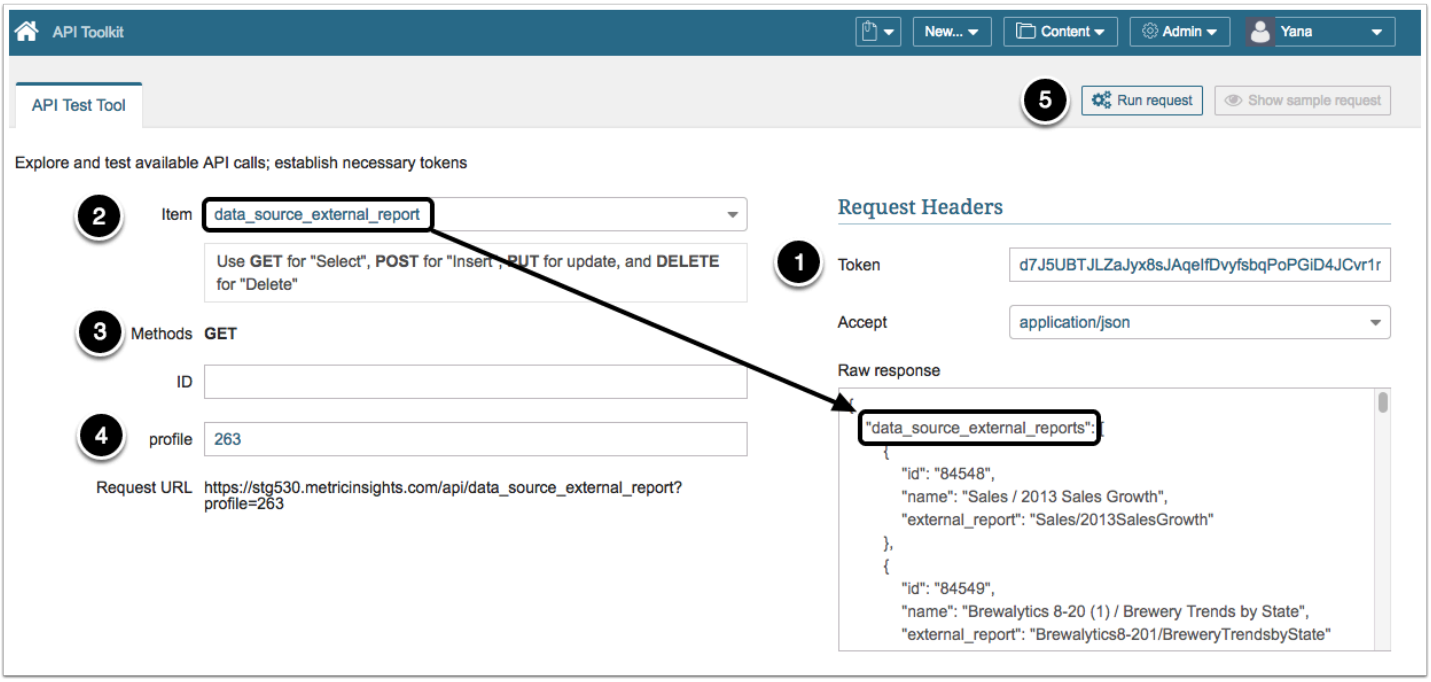
# 3. Obtain a list of Data Source Parameters (applicable to Plugin Data Sources)




1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** select **data_source_parameter**
3. **Methods:** GET is used
4. **Profile:** input the ID of the Data Source (connection profile) whose connection will be tested
5. **Run request:** click to query a list of Data Source Parameters

*Raw response* will contain information about Parameters for the specified Data Source.

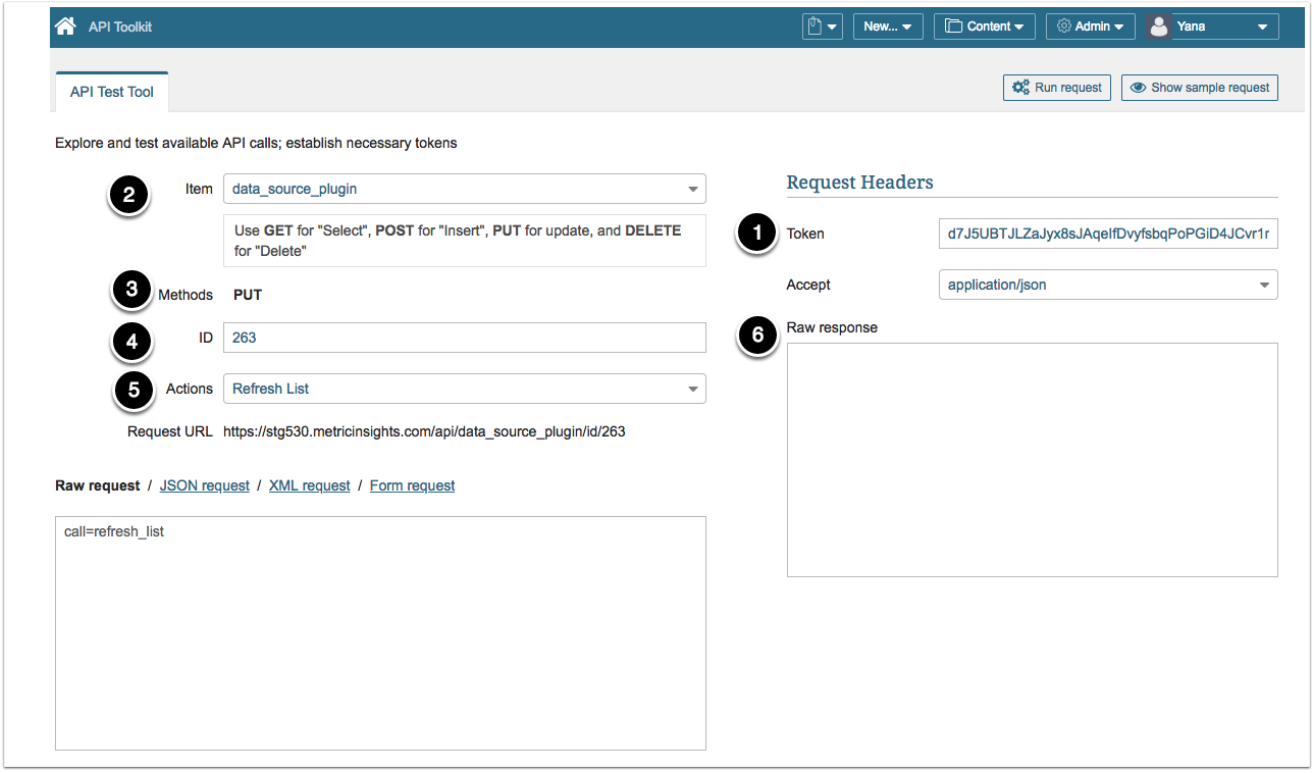
# 4. Query a list of External Reports (applicable to Plugin Data Sources)




1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** select **data_source_external_report**
3. **Methods:** GET is used
4. **Profile:** input the ID of the Data Source (connection profile) whose connection will be tested
5. **Run request:** click to query a list of External Reports (Data Source Objects)

*Raw response* will contain information about External Reports for that Data Source.

# 5. Refresh List (applicable to Plugin Data Sources)




1. **Token:** enter the Token that you have previously obtained as noted in the PREREQUISITES
2. **Item:** select **data_source_plugin**
3. **Methods:** PUT is used
4. **Actions:** select Refresh List
5. **Run request:** click to make an API action call

If the call is successful, the Raw response field will be empty. Otherwise, an error message will be displayed.

# 4. API Calls in JavaScript

# 4.1 Displaying Metric Insights content in web page using API via JavaScript

This article provides an example of using the Metric Insights API via JavaScript to display Alerts and other items in your website.

See the follow-on article to see what items to pull via API such as which Element ID to use for your Metric or Report in Metric Insights.

**PRE-REQUISITES:**

- Set up API access.
- Verify that you have API access
- Obtain a token via a get token call since all API calls require a token

**NOTE:** the format of the *API Test Tool* is slightly different in Version 3.

## 1. Use JavaScript to call the API

You can pull Alerts, Annotations, Notes and other items stored in Metric Insights and display them in your website. Just use the API with some JavaScript coding.

## 1.1. Sample Website that pulls in content from Metric Insights via API

### Alerts

4 on Tuesday 10/12/2010 is more than 0.00
New High 5 on Tuesday 10/19/2010 (previous high: 4 on Tuesday 10/12/2010)
5 on Tuesday 10/19/2010 is more than 10% above 4.00 (Yesterday)
5 on Tuesday 10/19/2010 is more than 0.00
New Low 2 on Tuesday 10/26/2010 (previous low: 4 on Tuesday 10/12/2010)
Unusual low 2 on Tuesday 10/26/2010
2 on Tuesday 10/26/2010 is more than 0.00
3 on Tuesday 11/02/2010 is more than 10% above 2.00 (Yesterday)
3 on Tuesday 11/02/2010 is more than 0.00
New High 6 on Tuesday 11/09/2010 (previous high: 5 on Tuesday 10/19/2010)

### Annotations

Spike was due to previous week issue at 2012-01-31 00:00:00
Expected this higher. Please research why. at 2012-06-13 00:00:00

### Notes

2015-04-18 09:41:52 : All, Please take at the trend

Example of Alerts, Annotations and Notes pulled from Metric Insights via API and displayed in a separate website.

## 1.2. API via JavaScript

```
23 <script>
24 var MI_URL = "https://hackathon2015.metricinsights.com";
25 var ELEMENT_ID = "60";
26 var MIToken = "";
27 $(document).ready(function() {
28   $.ajax({
29         url:MI_URL+"/api/index/qlik-token/",
30         method:'GET',
31         dataType: 'json',
32         accepts: {
33             text: 'application/json'
34         }
35     }).then(function (data){
36                 MIToken = data.token;
37                 //console.log("token response",data);
38                 //console.log("token",MIToken);
39
40                 $.ajax({
41                         url:MI_URL+"/api/metric/id/" + ELEMENT_ID,
42                         method:'GET',
43                         dataType: 'json',
44                         accepts: {
45                                 text: 'application/json'
46                         },
47                         beforeSend: function (request){
48                                 request.setRequestHeader("Token", MIToken);
49                         }
50                 }).then(function (data){
51                         //console.log("mi data response",data);
52                 });
53
54                 $.ajax({
55                         url:MI_URL+"/api/metric_alert?element=" + ELEMENT_ID,
56                         method:'GET',
57                         dataType: 'json',
58                         accepts: {
59                                 text: 'application/json'
60                         },
61                         beforeSend: function (request){
62                                 request.setRequestHeader("Token", MIToken);
63                         }
64                 }).then(function (data){
65                         //console.log("mi data response",data);
66                         var txt=document.getElementById("alerts");
67                         if (data && data.metric_alerts) {
```

**NOTE:** All API calls require a Token.

1. To make any API calls from JavaScript you will first need to obtain a token. In the example here we have included a server side function ("/qlik-token") that contains the authentication information for API access. This is just for illustrative and test purposes. You will need to supply your own server side method to obtain a token. And this server side code can be be done on any server; it does not have to be on Metric Insights server as shown in the example. Requiring server side call prevents setting up API access in the web browser. Metric Insights prevents you from making the Get Token (get_token) API call directly from the web browser due to Cross-Origin Resource Sharing (CORS) AJAX restriction. However, all other API calls are allowed for CORS once you obtain a token.

## 1.3. API via JavaScript in HTML

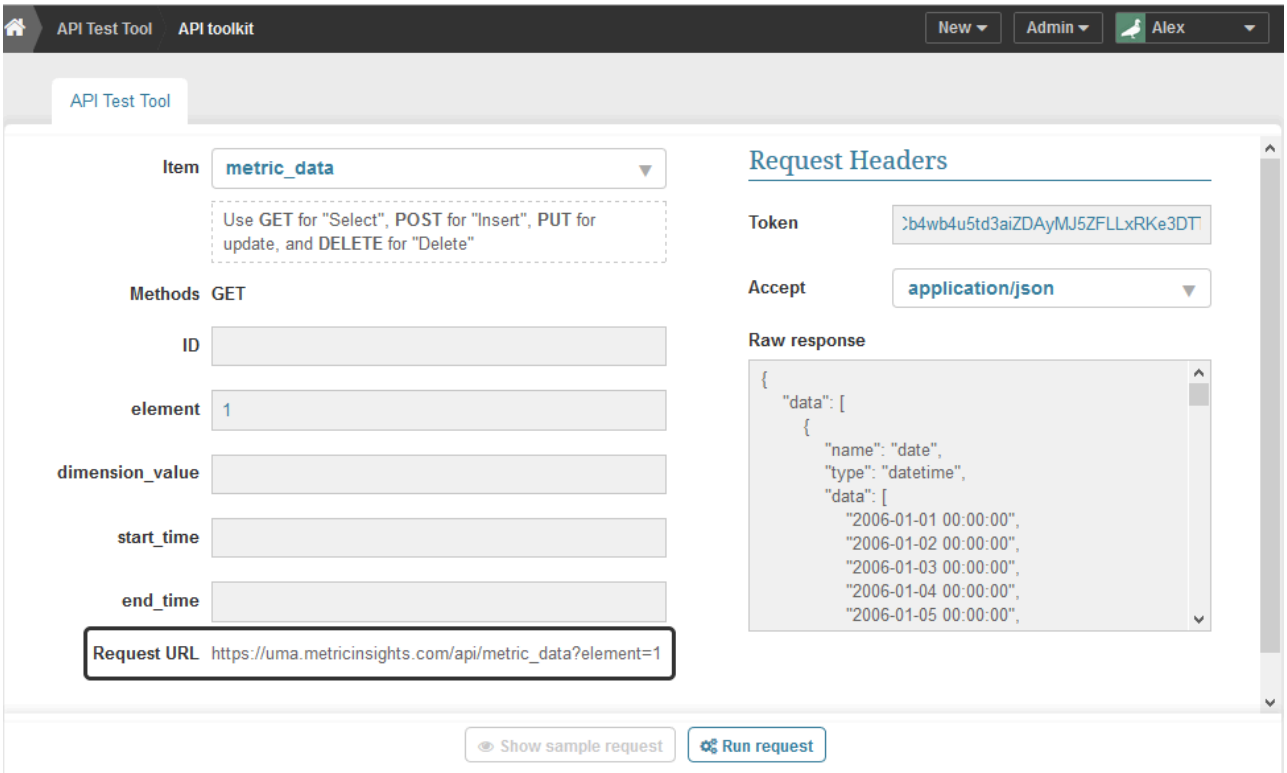
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="EN" lang="EN" dir="ltr">
    <head>
        <title>Metric Insights | Demo</title>
        <script type="text/javascript" src="https://hackathon2015.metricinsights.com/js/
jquery/jquery-1.10.2.min.js"></script>
    </head>
```

```
    <body id="top" style="background-color: white;">
      <div class="wrapper col3">
        <div>
          <h3>Alerts</h3>
          <div id="alerts"/>
        </div>
        <div>
          <h3>Annotations</h3>
          <div id="annotations"/>
        </div>
        <div>
          <h3>Notes</h3>
          <div id="notes"/>
        </div>
      </div>
<script>
var MI_URL = "https://hackathon2015.metricinsights.com";
var ELEMENT_ID = "60";
var MIToken = "";
$(document).ready(function() {
  $.ajax({
        url:MI_URL+"/api/index/qlik-token/",
        method:'GET',
        dataType: 'json',
        accepts: {
            text: 'application/json'
        }
    }).then(function (data){
                MIToken = data.token;
                //console.log("token response",data);
                //console.log("token",MIToken);
                $.ajax({
                        url:MI_URL+"/api/metric/id/" + ELEMENT_ID,
                        method:'GET',
                        dataType: 'json',
                        accepts: {
                                text: 'application/json'
                        },
                        beforeSend: function (request){
                                request.setRequestHeader("Token", MIToken);
                        }
                }).then(function (data){
                        //console.log("mi data response",data);
                });
                $.ajax({
                        url:MI_URL+"/api/metric_alert?element=" + ELEMENT_ID,
                        method:'GET',
                        dataType: 'json',
```

```
                        accepts: {
                                text: 'application/json'
                        },
                        beforeSend: function (request){
                                request.setRequestHeader("Token", MIToken);
                        }
                }).then(function (data){
                        //console.log("mi data response",data);
                        var txt=document.getElementById("alerts");
                        if (data && data.metric_alerts) {
                            var text =  "";
                            var item = "";
                            for(var i=0; i< 10 && i< data.metric_alerts.length; i++) {
                                item = data.metric_alerts[i].text_html?data.
metric_alerts[i].text_html:data.metric_alerts[i].text;
                                text = text + item + "<br/>";
                            }
                            txt.innerHTML=text;
                        };
                });
                $.ajax({
                        url:MI_URL+"/api/metric_annotation?element=" + ELEMENT_ID,
                        method:'GET',
                        dataType: 'json',
                        accepts: {
                                text: 'application/json'
                        },
                        beforeSend: function (request){
                                request.setRequestHeader("Token", MIToken);
                        }
                }).then(function (data){
                        //console.log("mi data response",data);
                        var txt=document.getElementById("annotations");
                        if (data && data.metric_annotations) {
                            var text =  "";
                            var item = "";
                            for(var i=0; i< 10 && i< data.metric_annotations.length;
i++) {
                                item = data.metric_annotations[i].text + " at " + data.
metric_annotations[i].measurement_time;
                                text = text + item + "<br/>";
                            }
                            txt.innerHTML=text;
                        };
                });
                $.ajax({
                        url:MI_URL+"/api/note?element=" + ELEMENT_ID,
                        method:'GET',
```

```
                        dataType: 'json',
                        accepts: {
                                text: 'application/json'
                        },
                        beforeSend: function (request){
                                request.setRequestHeader("Token", MIToken);
                        }
                }).then(function (data){
                        //console.log("mi data response",data);
                        var txt=document.getElementById("notes");
                        if (data && data.notes) {
                            var text =  "";
                            var item = "";
                            for(var i=0; i< 10 && i< data.notes.length; i++) {
                                item = data.notes[i].created_time + " : " + data.
notes[i].text;
                                text = text + item + "<br/>";
                            }
                            txt.innerHTML=text;
                        };
                });
    });
});
</script>
    </body>
</html>
```

# 2. Use API Toolkit to test and discover API calls




Use the [API Test Tool]() to test the API calls and discover the API format for requests and responses.

In this example, we pull all the data for the Metric identified by its Element ID.

The tool gives you the API url to use in your code:

```
https://uma.metricinsights.com/api/metric_data?element=1
```

# 3. Looking in Metric Insights to see what items to pull via API

See the follow-on article to [see what items to pull via API]() such as which Element ID to use for your Metric or Report in Metric Insights.

# 4.2 Looking in Metric Insights to see what items to pull via API

This article is a follow on to the one that provides an example of using the Metric Insights API via JavaScript to display items in your website.

This section gives a quick overview of where to look in Metric Insights to determine which elements (Metrics or Reports) to use in your API calls. In these examples the elements contain Alerts, Annotations, Notes and other items that you might want to pull via API to your website.

**NOTE:** The format of the *API Test Tool* is slightly different in Version 3.

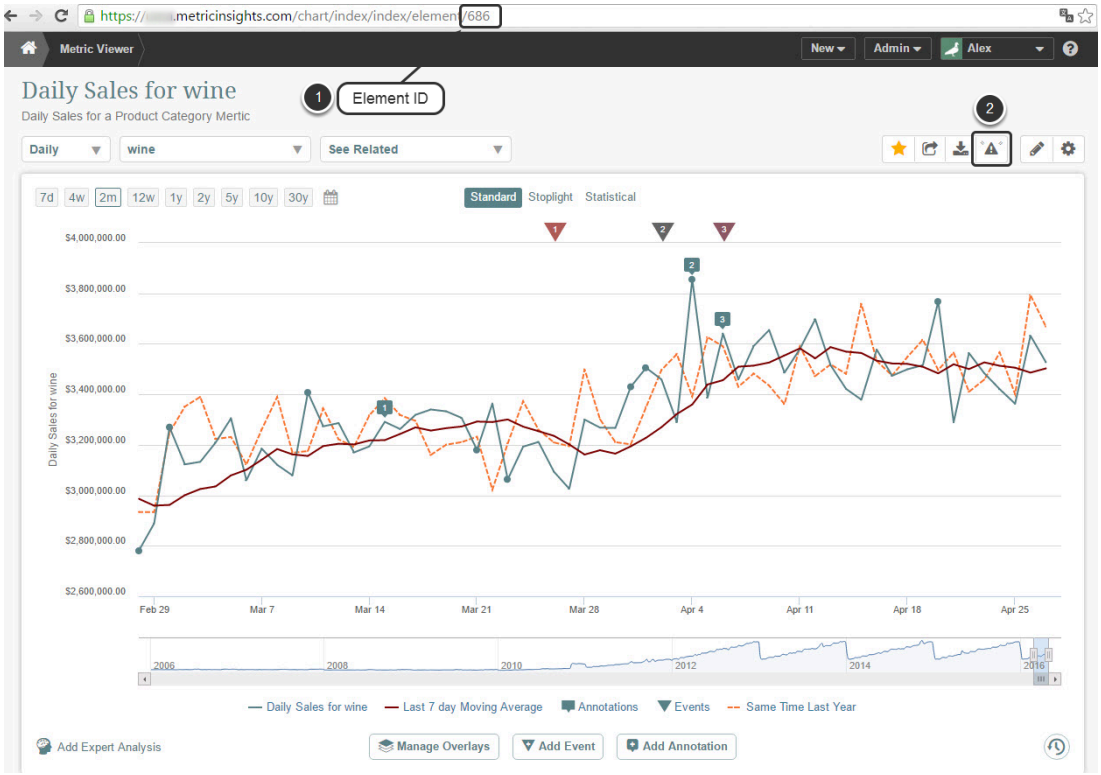
## 1. API via JavaScript - Determine which Element ID to use

```
23 <script>
24 var MI_URL = "https://hackathon2015.metricinsights.com";
25 var ELEMENT_ID = "60";
26 var MIToken = "";
27 $(document).ready(function() {
28   $.ajax({
29         url:MI_URL+"/api/index/qlik-token/",
30         method:'GET',
31         dataType: 'json',
32         accepts: {
33             text: 'application/json'
34         }
35     }).then(function (data){
36                 MIToken = data.token;
37                 //console.log("token response",data);
38                 //console.log("token",MIToken);
39
40                 $.ajax({
41                         url:MI_URL+"/api/metric/id/" + ELEMENT_ID,
42                         method:'GET',
43                         dataType: 'json',
44                         accepts: {
45                                 text: 'application/json'
46                         },
47                         beforeSend: function (request){
48                                 request.setRequestHeader("Token", MIToken);
49                         }
50                 }).then(function (data){
51                         //console.log("mi data response",data);
52                 });
53
54                 $.ajax({
55                         url:MI_URL+"/api/metric_alert?element=" + ELEMENT_ID,
56                         method:'GET',
57                         dataType: 'json',
58                         accepts: {
59                                 text: 'application/json'
60                         },
61                         beforeSend: function (request){
62                                 request.setRequestHeader("Token", MIToken);
63                         }
64                 }).then(function (data){
65                         //console.log("mi data response",data);
66                         var txt=document.getElementById("alerts");
67                         if (data && data.metric_alerts) {
```

In the example here we have a web page that calls the API via JavaScript. In the page, we need to determine which element ID to use for pulling content from Metric Insights.

1. In this example we just hardcode the Metric Insights element ID that contains the items we are interested in such as Alerts, Annotations and Notes. The next sections give you tips on how to determine which Element ID to use.

# 1.1. Element ID on a Metric Viewer




The Metric is what is used in Metric Insights to trigger the Alerts; so use this element in your API calls to get the Alerts, Annotations, Notes and other items.

To determine the Element ID for a Metric Insights element:

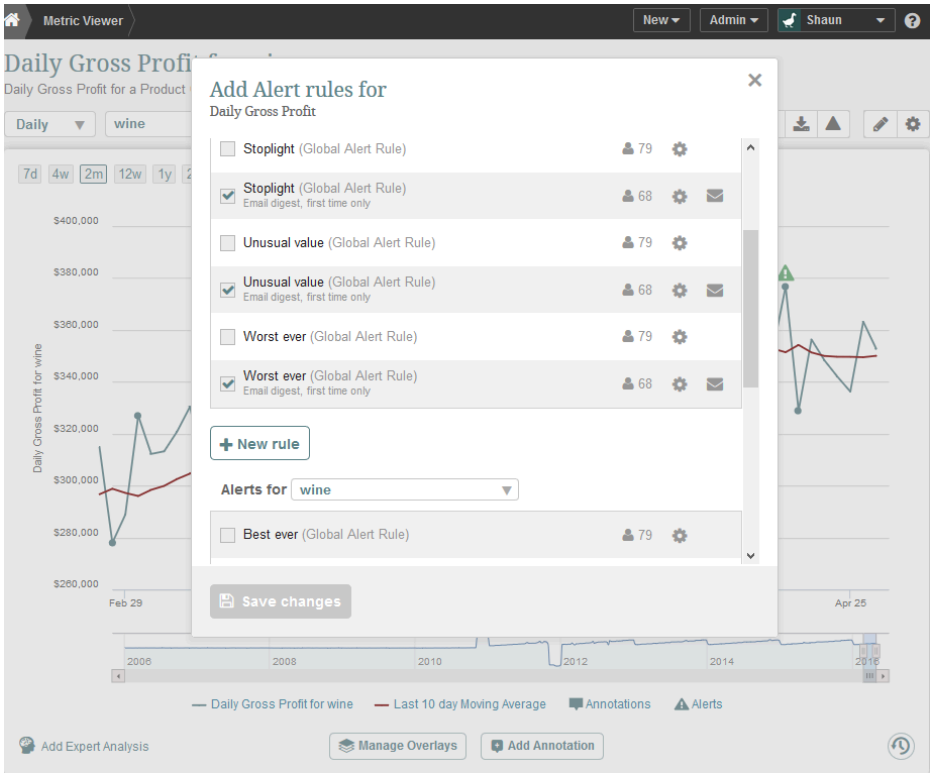
1. The Element ID for the Metric and any other element can be found in the web browser URL. For example, this Metric element ID is "686" which is seen in the URL:

```
https://demo.metricinsights.com/chart/index/index/element/686
```

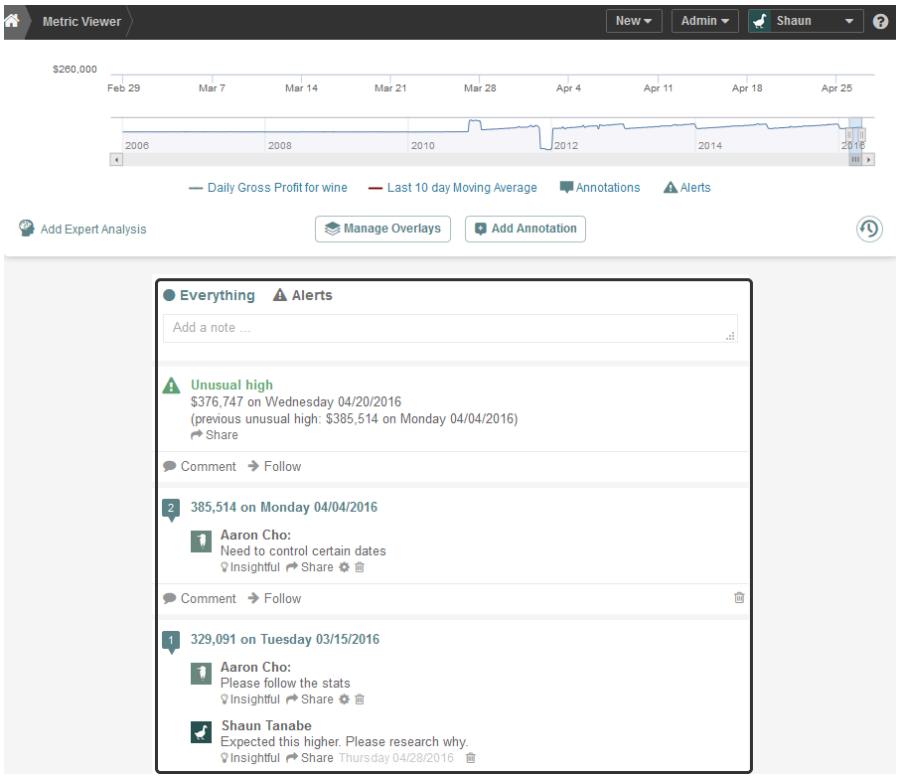
Additionally, if you want to see the Alert Rules:

2. Click on the **Alert** icon to display the **Alert Rules** for this Metric

# 2. Metric with Alert Rules

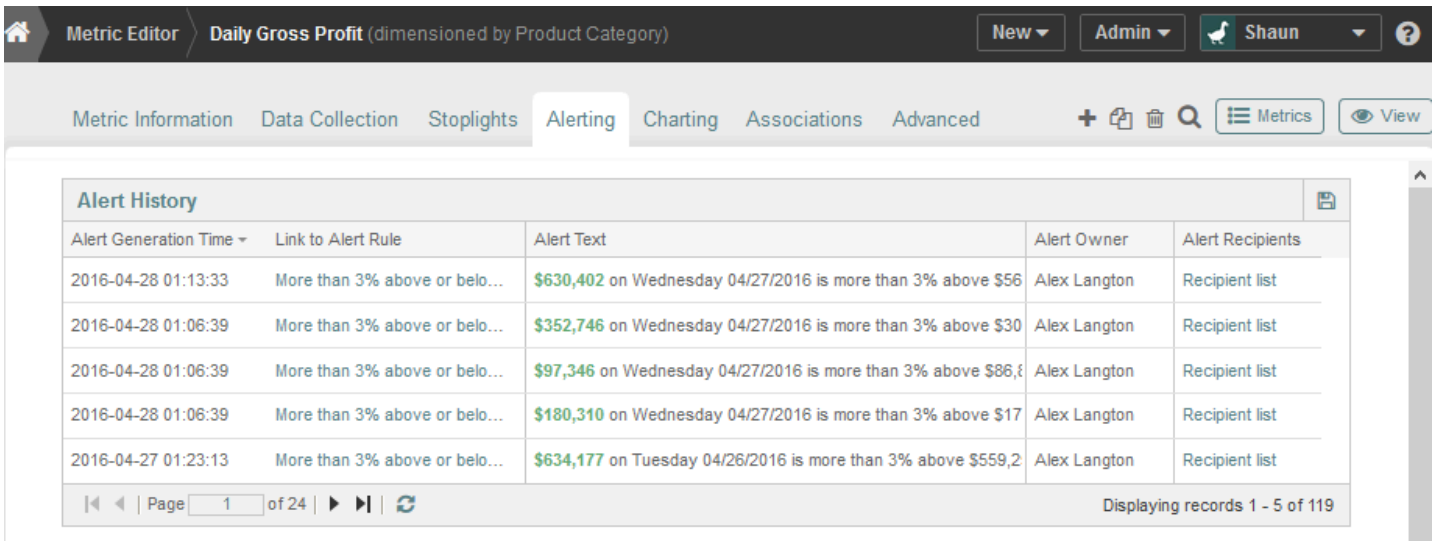



## 2.1. Viewing Alerts, Annotations, Notes



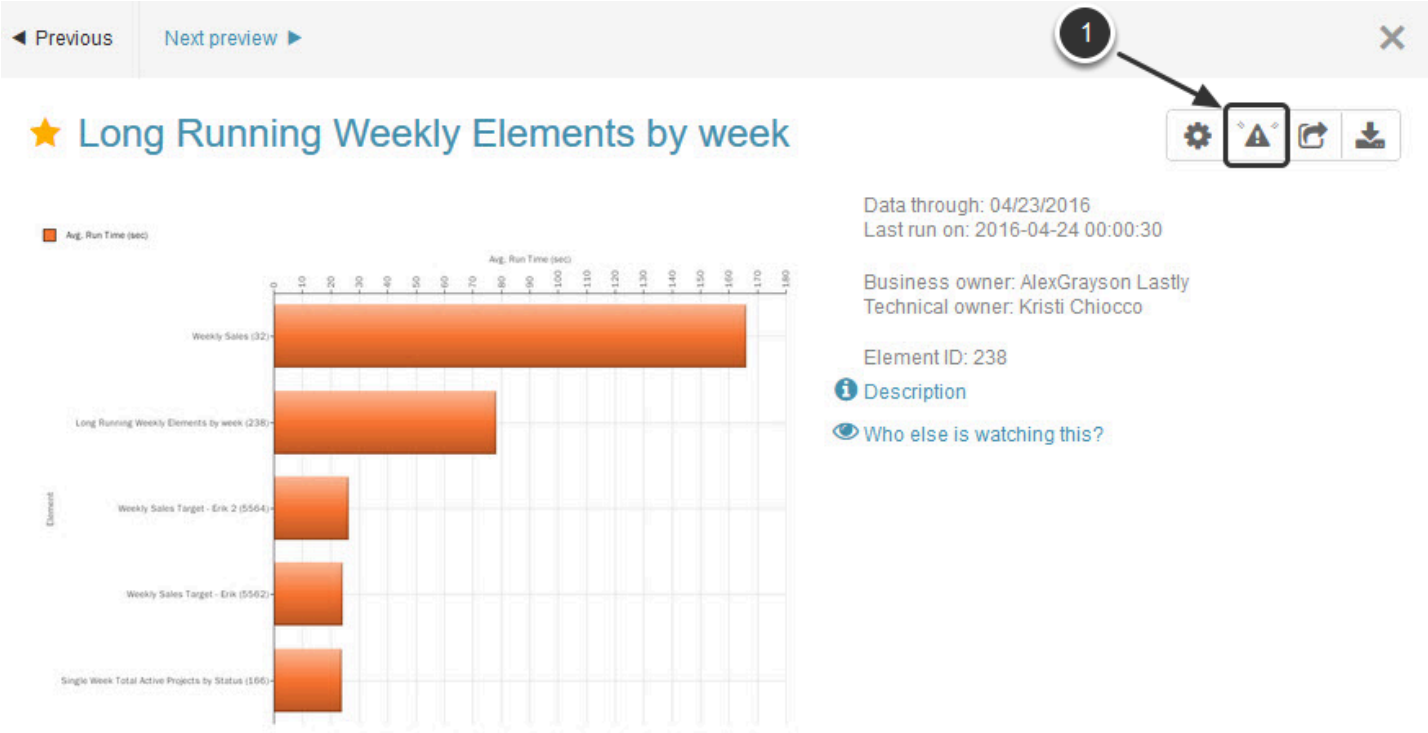
In the Metric Viewer, scroll down to see the Alerts, Annotations, and Notes added to your Metric. These are the same items you can pull into your website using the API.

## 2.2. Viewing Alerts




In the Metric Editor, switch to the *Alerting* tab to see the **Alert History** for your Metric. These are items you can pull into your website using the API.

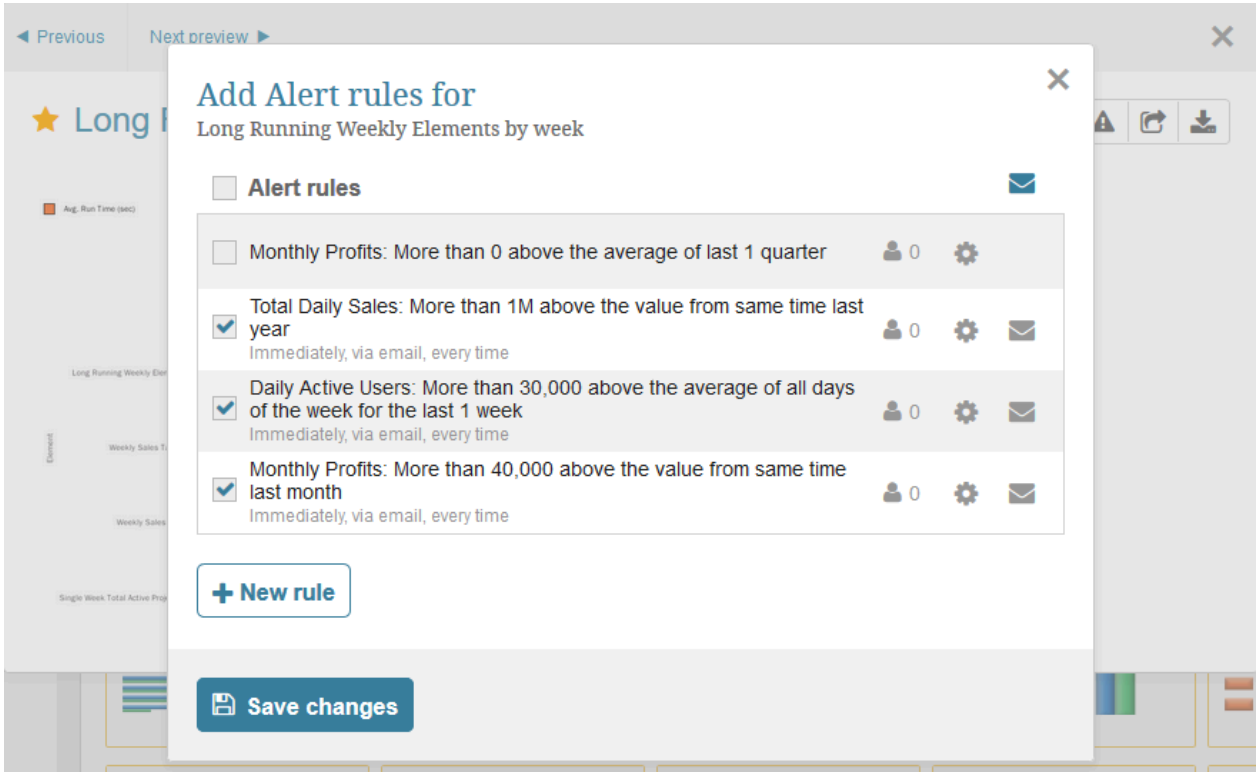
# 3. Report with Alert Rules



If your Alerts are sent via an Image of your dashboard from your Business Intelligence application, then you can view the corresponding Alert Rules directly from the Metric Insights dashboard.

1. Click the **Alert** icon to display the Alert Rules for this Report

## 3.1. Report with Alert Rules

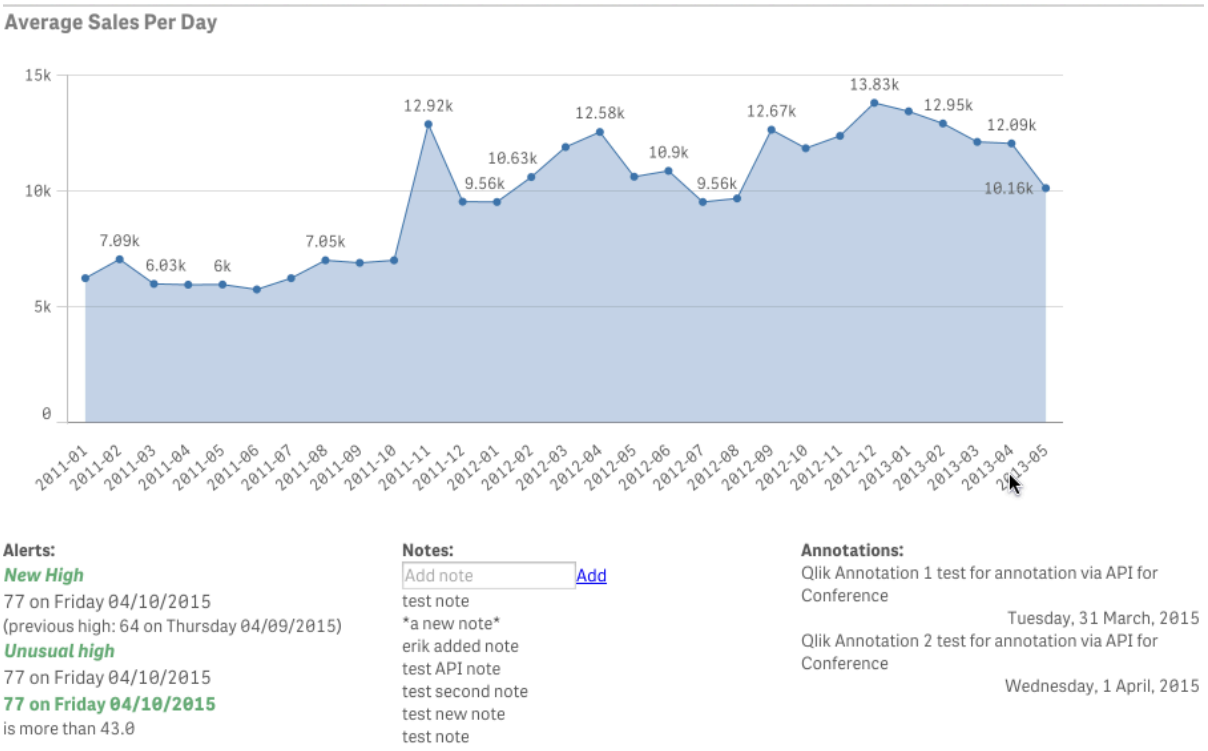


# 4.3 Qlik Sense mashup using Metric Insights API

This article shows a working example of using the Metric Insights API from a Qlik Sense mashup.

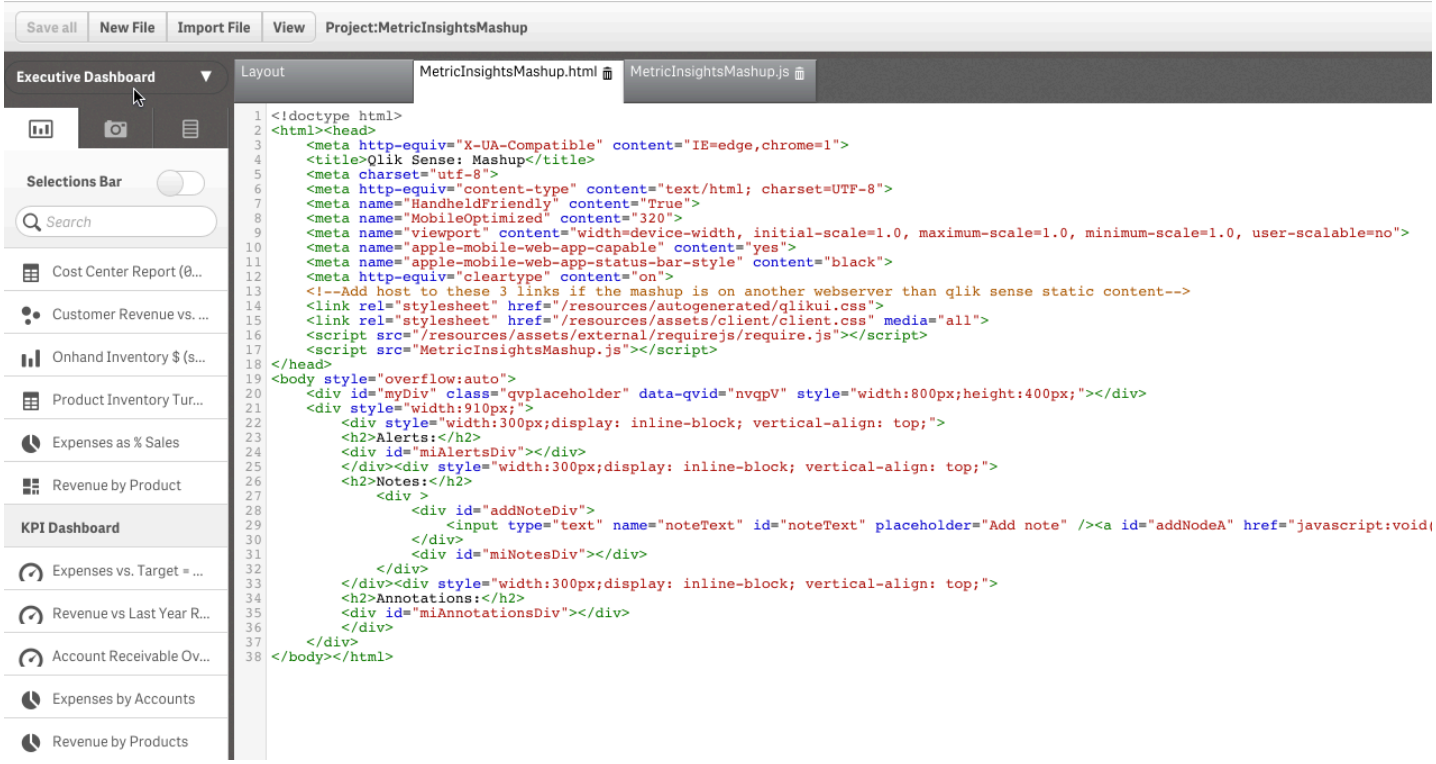
You can refer to more details on integrating the Metric Insights API call into your web app.

## Qlik Sense mashup




This Qlik Sense mashup illustrates using the Metric Insights API. In this example, the Alerts, Notes and Annotations at the bottom are the result of calling the Metric Insights API.

# Layout in Qlik Sense workbench




Example mashup as viewed from Qlik Sense workbench

[https://qliksense-test.metricinsights.com/workbench/mashup.html?extension=MetricInsightsMashup](https://qliksense-test.metricinsights.com/workbench/mashup.html?extension=MetricInsightsMashup)

# HTML file for mashup

```
<!doctype html>
<html><head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <title>Qlik Sense: Mashup</title>
    <meta charset="utf-8">
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <meta name="HandheldFriendly" content="True">
    <meta name="MobileOptimized" content="320">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0, user-scalable=no">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-status-bar-style" content="black">
    <meta http-equiv="cleartype" content="on">
    <!--Add host to these 3 links if the mashup is on another webserver than qlik sense
static content-->
```

```
    <link rel="stylesheet" href="/resources/autogenerated/qlikui.css">
    <link rel="stylesheet" href="/resources/assets/client/client.css" media="all">
    <script src="/resources/assets/external/requirejs/require.js"></script>
    <script src="MetricInsightsMashup.js"></script>
</head>
<body style="overflow:auto">
    <div id="myDiv" class="qvplaceholder" data-qvid="nvqpV"
style="width:800px;height:400px;"></div>
    <div style="width:910px;">
        <div style="width:300px;display: inline-block; vertical-align: top;">
        <h2>Alerts:</h2>
        <div id="miAlertsDiv"></div>
        </div><div style="width:300px;display: inline-block; vertical-align: top;">
        <h2>Notes:</h2>
            <div >
                <div id="addNoteDiv">
                    <input type="text" name="noteText" id="noteText" placeholder="Add
note" /><a id="addNodeA" href="javascript:void(0)">Add</a>
                </div>
                <div id="miNotesDiv"></div>
            </div>
        </div><div style="width:300px;display: inline-block; vertical-align: top;">
        <h2>Annotations:</h2>
        <div id="miAnnotationsDiv"></div>
        </div>
    </div>
</body></html>
```

HTML file used for the Qlik Sense mashup

https://qliksense-test.metricinsights.com/extensions/MetricInsightsMashup/MetricInsightsMashup.html

# JavaScript file for calling Metric Insights API

```
/*global require, alert*/
/*
 *
 * @owner Enter you name here (xxx)
 */
/*
 *    Fill in host and port for Qlik engine
 */
var config = {
        host: window.location.hostname,
        prefix: "/",
        port: window.location.port,
```

```
        isSecure: window.location.protocol === "https:"
};
require.config( {
        baseUrl: ( config.isSecure ? "https://" : "http://" ) + config.host + (config.
port ? ":" + config.port: "") + config.prefix + "resources"
} );
var MIHostName = "https://adam.metricinsights.com";
var MIElement = 3210;
var MIToken = '';
function loadAlerts(){
    $.ajax({
        url:MIHostName+"/api/metric_alert?element="+MIElement+"&html=Y",
        method:'GET',
        dataType: 'json',
        accepts: {
            text: 'application/json'
        },
        beforeSend: function (request){
            request.setRequestHeader("Token", MIToken);
        },
        statusCode:{
            401:function(){ $.getScript(MIHostName+"/api/index/get-qlik-token/token.
js",function(){ if (MIToken) loadAlerts(); }); }
        }
    }).then(function (data){
        $('#miAlertsDiv').html('');
        if (data && data.metric_alerts)
            for(var i=0;i<data.metric_alerts.length;i++)
                $('#miAlertsDiv').append('<p>'+(data.metric_alerts[i].text_html?data.
metric_alerts[i].text_html:data.metric_alerts[i].text)+'</p>');
        console.log("alerts",data);
    });
}
function loadAnnotations(){
    $.ajax({
        url:MIHostName+"/api/metric_annotation?element="+MIElement+"&html=Y",
        method:'GET',
        dataType: 'json',
        accepts: {
            text: 'application/json'
        },
        beforeSend: function (request){
            request.setRequestHeader("Token", MIToken);
        },
        statusCode:{
            401:function(){ $.getScript(MIHostName+"/api/index/get-qlik-token/token.
js",function(){ if (MIToken) loadAnnotations(); }); }
        }
```

```
    }).then(function (data){
        $('#miAnnotationsDiv').html('');
        if (data && data.metric_annotations)
            for(var i=0;i<data.metric_annotations.length;i++)
                $('#miAnnotationsDiv').append('<p>'+data.metric_annotations[i].
text+'</p>'+'<p style="text-align:right">'+($.datepicker.formatDate('DD, d MM, yy', new
Date(data.metric_annotations[i].measurement_time)))+'</p>');
        console.log("annotations",data);
    });
}
function loadNotes(){
    $.ajax({
        url:MIHostName+"/api/note?element="+MIElement,
        method:'GET',
        dataType: 'json',
        accepts: {
            text: 'application/json'
        },
        beforeSend: function (request){
            request.setRequestHeader("Token", MIToken);
        },
        statusCode:{
            401:function(){ $.getScript(MIHostName+"/api/index/get-qlik-token/token.
js",function(){ if (MIToken) loadNotes(); }); }
        }
    }).then(function (data){
        $('#miNotesDiv').html('');
        if (data && data.notes)
            for(var i=0;i<data.notes.length;i++)
                $('#miNotesDiv').append('<p>'+data.notes[i].text+'</p>');
        console.log("notes",data);
    });
}
function addNote(){
    var el = $('#noteText');
    if (!el.val()) {
        alert('Please enter text');
        el.focus();
        return;
    }
    $.ajax({
        url:MIHostName+"/api/note?element="+MIElement,
        method:'POST',
        dataType: 'json',
        data:{"element":MIElement, text:el.val()},
        accepts: {
            text: 'application/json'
        },
```

```
        beforeSend: function (request){
            request.setRequestHeader("Token", MIToken);
        },
        statusCode:{
            401:function(){ $.getScript(MIHostName+"/api/index/get-qlik-token/token.
js",function(){ if (MIToken) addNote(); }); }
        }
    }).then(function (data){
        el.val('');
        loadNotes();
    });
}
require( ["js/qlik"], function ( qlik ) {
        qlik.setOnError( function ( error ) {
                alert( error.message );
        });
    $.ajax({
        url:MIHostName+"/api/index/qlik-token/",
        method:'GET',
        dataType: 'json',
        accepts: {
            text: 'application/json'
        }
    }).then(function (data){
        MIToken = data.token;
                console.log("token",data);
        loadAlerts();
        loadNotes();
        loadAnnotations();
    });
    var app = qlik.openApp("8913ed6b-587d-4e4e-ac2e-5f304fe27dd9",config);
    $('.qvplaceholder').each(function(){
        var qvid = $(this).data('qvid');
        app.getObject(this,qvid);
    });
    $('#addNodeA').click(function(){
        addNote();
    });
});
```

JavaScript file used for Qlik Sense mashup

https://qliksense-test.metricinsights.com/extensions/MetricInsightsMashup/MetricInsightsMashup.js

The JavaScript file contains the API calls to Metric Insights.

Key points in using the API are discussed in more detail on integrating the Metric Insights API call into your web app. Reference that to see how:

1. All API calls require a token
2. Determine which elements in Metric Insights to pull data from.