

zebrix REST api starter guide

Introduction

Every zebrix REST API call required a token that has to be included in the request header. The token can be obtained with a login procedure.

login api

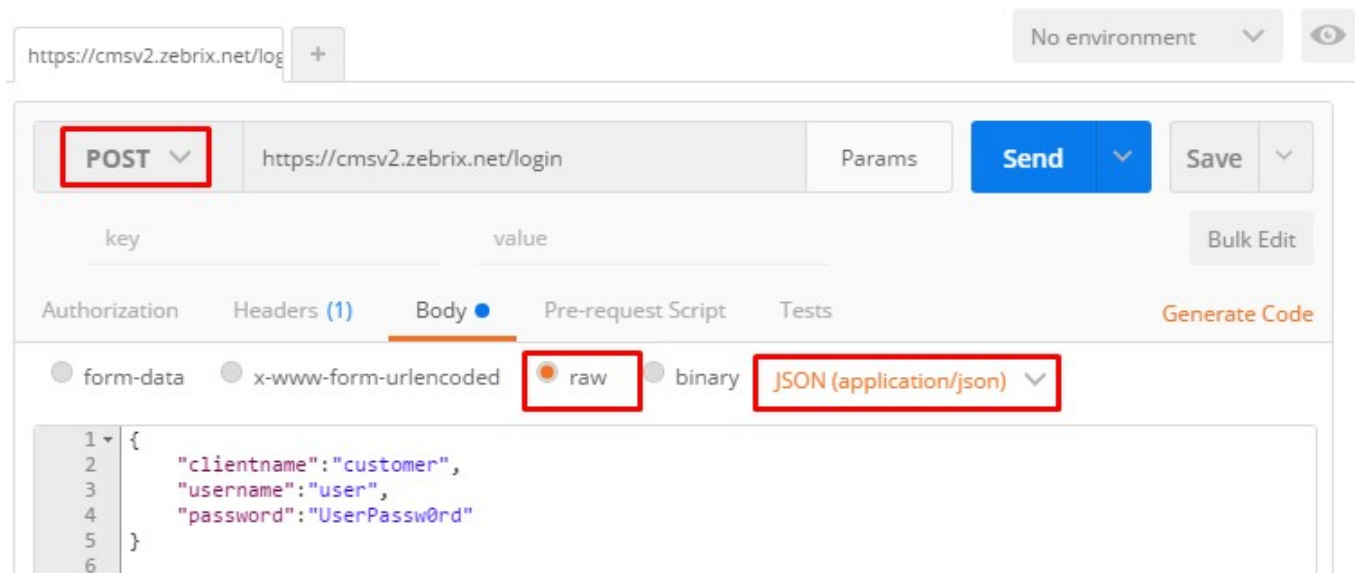
POST /login

Parameters has to be sent in the JSON format

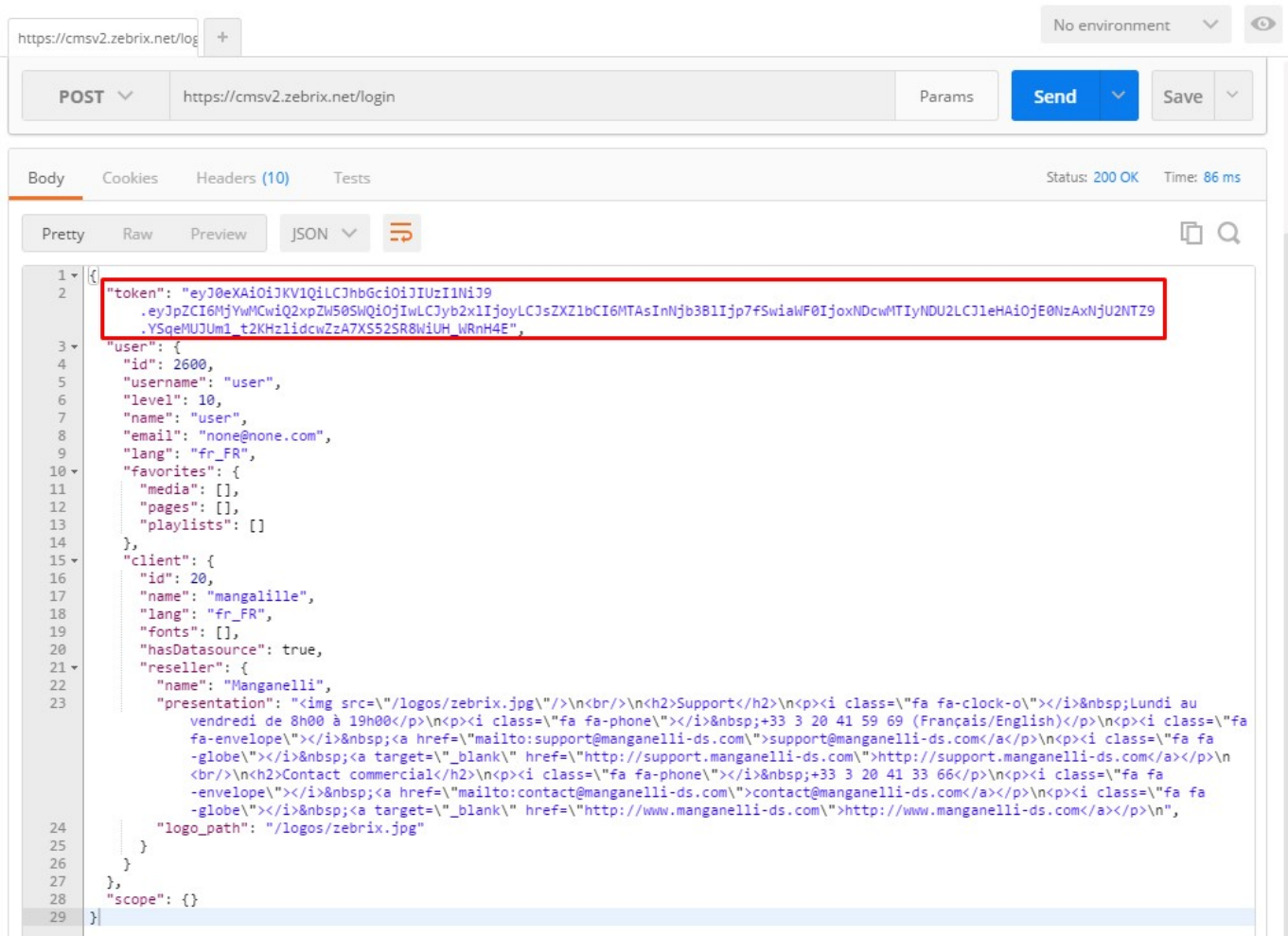
| Parameters | |
|------------|-------------------|
| clientname | the customer name |
| username | the username |
| password | the user password |

login demo with "POSTMAN" (Google Chrome Extension)

Call



Return



Example of GET API on screen

GET /api/screen/:id

This example will demonstrate how to request information with a GET call to the API. We will use the screen api (to get the list of all screens with their properties). It will work the same for all other objects in zebrix (media, pages, playlist, etc.)

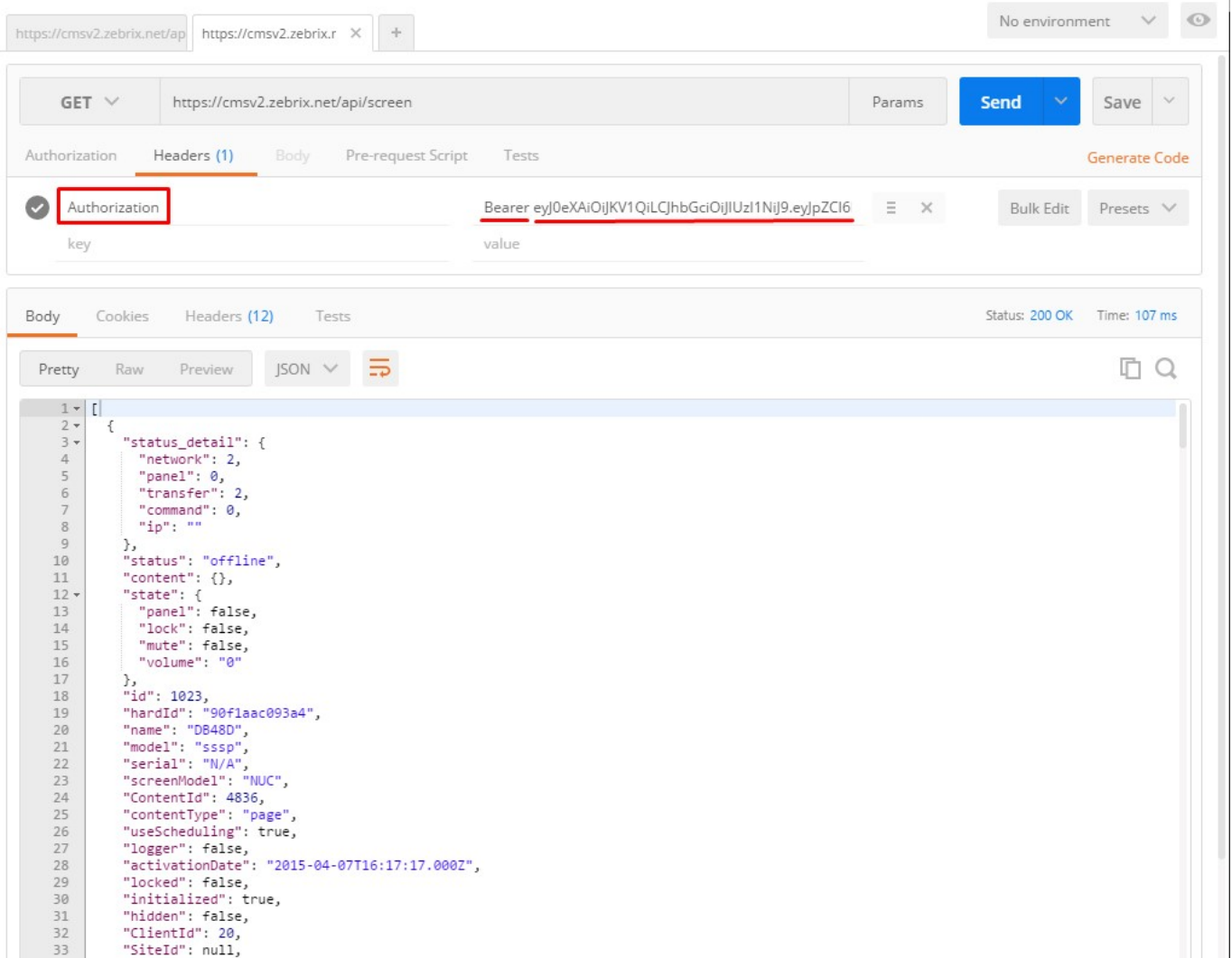
In this request we will need to use the previously requested token in the request header of the API will return a 401 (not authorized) error.

The token must be prefixed with the word "Bearer" and a space char that way :

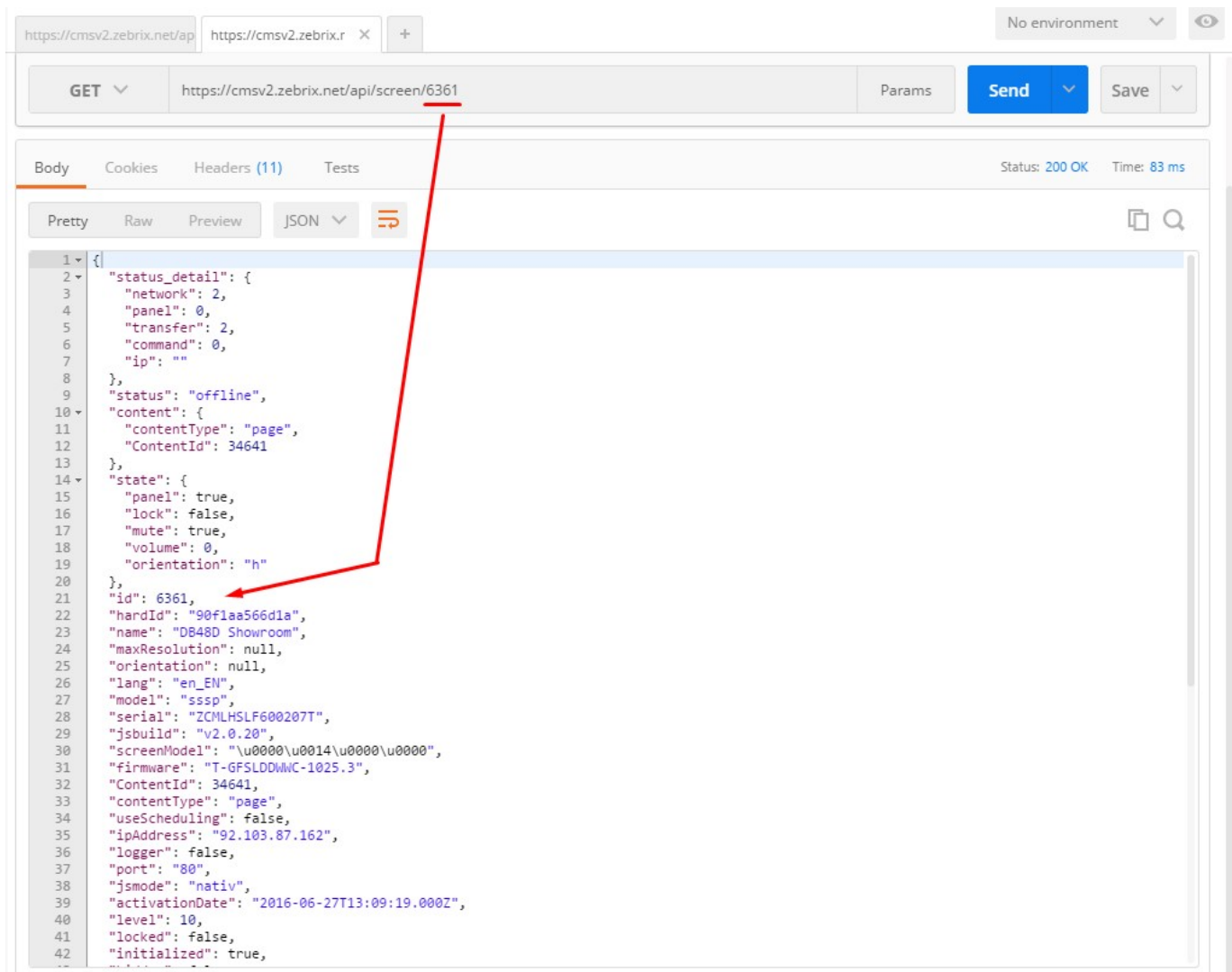
Bearer

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MjYwMCwiQ2xpZW50SWQiOiJwLCJyb2x1Ijo yLCJsczZlIjE6MTAsInNjb3BlIjpb7fSwiaWF0IjoxNDcwMTIyNDU2LCJleHAiOiJE0NzAxNjU2NTZ9.YSqeMUJUm1t2JullidcwZzA7XS52SR8WiUHWnH4E

The call of the API will return a JSON array of all screens :



To get information for a specific screen, its unique id can be added in the URL. For example, we want to get information about the screen with the id 6361 :



How to use API for datasource

Introduction to datasource

In zebrix a **datasource** is basically one row from a database.

As an example let take a table from DB containing information about 4 meeting rooms. Each meeting room has 4 properties :

- ROOM (room name)
- VISITOR (current visitor who uses the meeting room)
- SEATS
- HASBEAMER

| | A | B | C | D |
|---|--------------------|---------------|-------|--------------|
| 1 | ROOM | VISITOR | SEATS | HASABEAMER |
| 2 | <u>Yellow room</u> | MORA | 6 | <u>true</u> |
| 3 | <u>Dark Room</u> | BARCO | 0 | <u>false</u> |
| 4 | <u>White Room</u> | <u>Zebrix</u> | 8 | <u>false</u> |
| 5 | <u>Black Room</u> | <u>Audi</u> | 10 | <u>true</u> |
| 6 | | | | |

In zebrix this table of 4 meeting rooms will requires 4 datasources (i.e. meeting room 1, meeting room 2, meeting room 3, meeting room 4). Each line/datasource will contains all columns / fields in a JSON.

Get all datasources

GET /api/datasource/:id

All datasources will be returned with all their properties (including their content)


Using our meeting rooms example, here is the result you will get :

```

111 {
112   "defaults": {
113     "ROOM": "Black Room",
114     "VISITOR": "Audi",
115     "SEATS": "10",
116     "HASABEAMER": "true"
117   },
118   "id": 1010,
119   "name": "test14",
120   "createdAt": "2016-04-29T14:53:26.000Z",
121   "updatedAt": "2016-04-29T14:57:11.000Z",
122   "ClientId": 20,
123   "TaxoId": null,
124   "taxo": null
125 },
126 {
127   "defaults": {
128     "ROOM": "Yellow room",
129     "VISITOR": "MORA",
130     "SEATS": "6",
131     "HASABEAMER": "true"
132   },
133   "id": 1011,
134   "name": "test21",
135   "createdAt": "2016-04-29T14:59:58.000Z",
136   "updatedAt": "2016-04-29T14:59:58.000Z",
137   "ClientId": 20,
138   "TaxoId": null,
139   "taxo": null
140 },

```

Theses datasource can be used by users in their page by following instructions on that page (section 5) ([Using datasources in Zebrix](#))



Please notice that some datasources are auto-generated by zebrix when the functionality of "zone with variable content"

is used. These datasources can be easily recognized with their name which always begin with double underscore + page + id of the concerned page.



```
{
  "defaults": {
    "__54488": [
      {
        "MediumId": 26368,
        "index": 0,
        "properties": {
          "customCrop": false,
          "backgroundSize": 100,
          "backgroundPosH": 50,
          "backgroundPosV": 50
        }
      }
    ]
  },
  "id": 418,
  "name": "__page_16716",
  "createdAt": "2015-12-02T11:59:45.000Z",
  "updatedAt": "2015-12-02T12:01:32.000Z",
  "ClientId": 20,
  "TaxoId": 589,
  "taxo": {
    "targets": {},
    "id": 589,
    "name": "Sites",
    "color": "info"
  }
},
```

How to add a new datasource to zebrix

POST /api/datasource/

The content of the datasource has to be JSON encoded and put as an object in a defaults variable.

```
1 {
2   "defaults":{
3     "companyName":"zebrix",
4     "trainingTopic":"zebrix api usage",
5     "trainer":"Pierre",
6     "duration":"3 hours",
7     "misc":"test"
8   }
9 }
```

Example of usage :

The screenshot shows a REST client interface with three browser tabs. The active tab is titled 'https://cmsv2.zebrix.net/ap'. The main area shows a POST request to 'https://cmsv2.zebrix.net/api/datasource'. The 'Body' tab is selected, and the content type is set to 'JSON (application/json)'. The JSON body is as follows:

```
1 {
2   "defaults": {
3     "companyName": "zebrix",
4     "trainingTopic": "zebrix api usage",
5     "trainer": "Pierre",
6     "duration": "2 hours"
7   }
8 }
```

JSON which is POSTed to the API

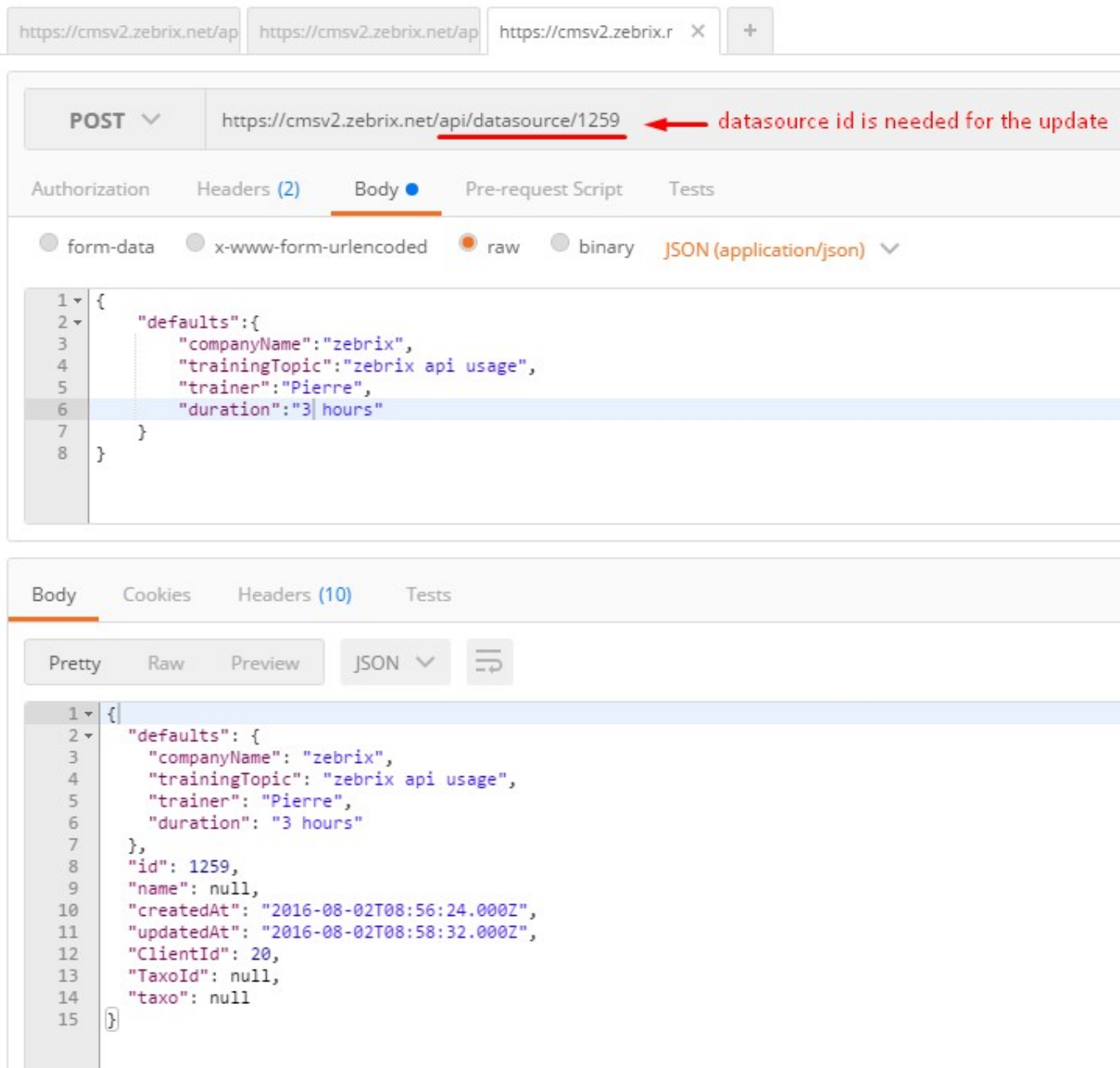
The screenshot shows the response of the POST request. The 'Body' tab is selected, and the content type is set to 'JSON'. The JSON response is as follows:

```
1 {
2   "defaults": {
3     "companyName": "zebrix",
4     "trainingTopic": "zebrix api usage",
5     "trainer": "Pierre",
6     "duration": "2 hours"
7   },
8   "id": 1259,
9   "clientId": 20,
10  "updatedAt": "2016-08-02T08:56:24.000Z",
11  "createdAt": "2016-08-02T08:56:24.000Z"
12 }
```

Datasource JSON which is returned after the add

How to update an existing datasource

POST /api/datasource/:id

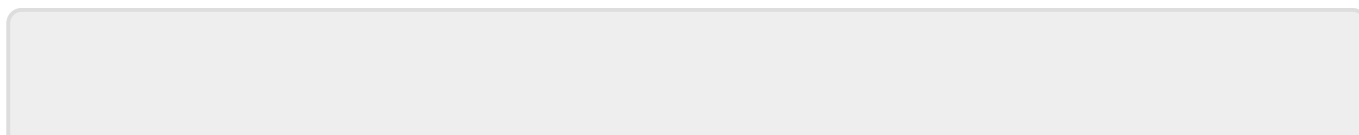


The update process works the same as a creation but the id of the datasource need to be specified.

How to force screens to update

POST /api/screens/updateds

When a datasource is updated screens that display this datasource will not be update until the `updateds` is called. This API is usually called after update of a datasource. Is displayed pages contain multiple datasources it is adviced to call the `updateds` method at the end of the update process of all datasources instead of after every datasource update to limit ressources and bandwidth usage.



From:
<https://documentation.zebrix.net/> - **zebrix documentation**

Permanent link:
<https://documentation.zebrix.net/doku.php?id=en:zebrixrestapi&rev=1537447012>

Last update: **2020/06/22 11:53**

