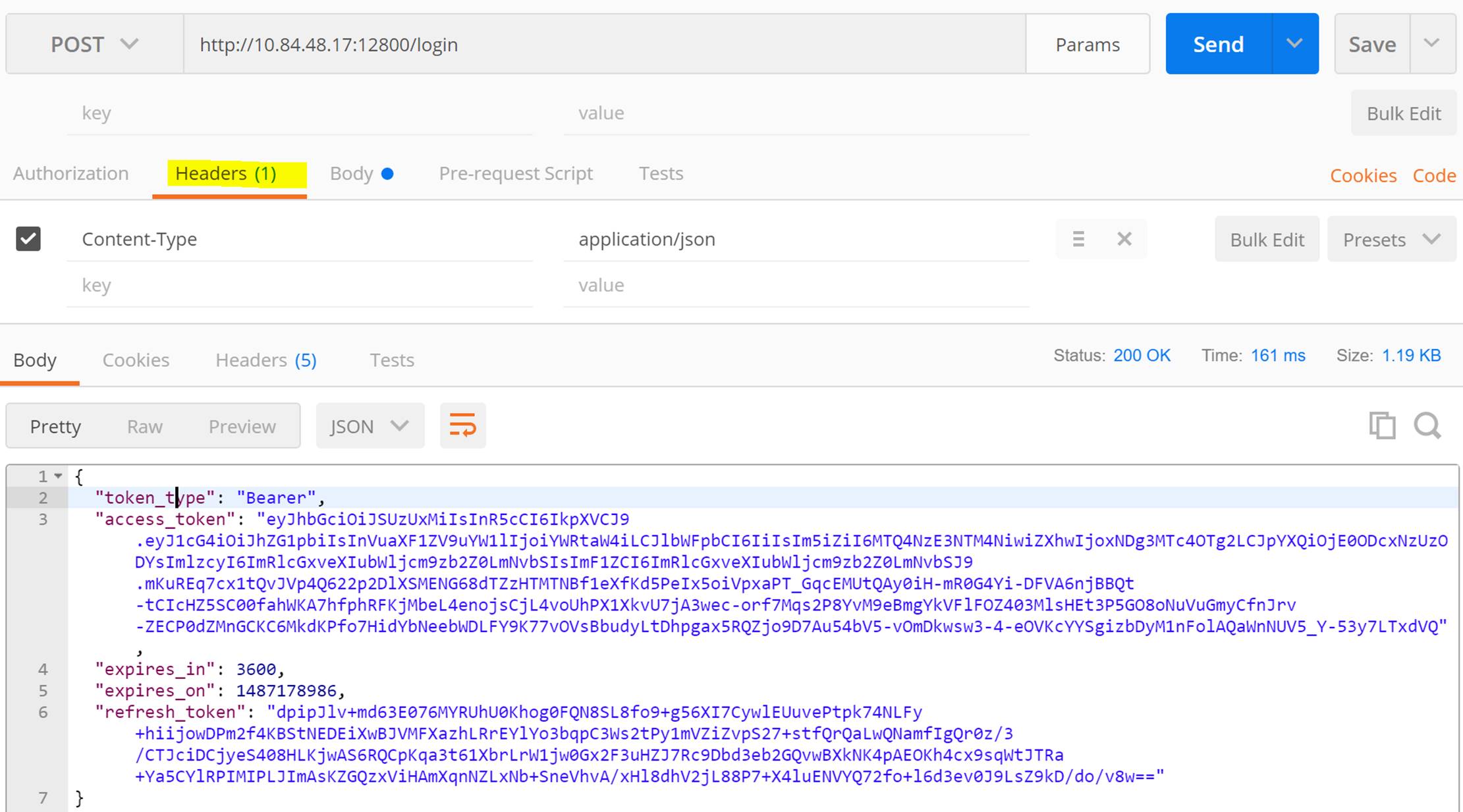
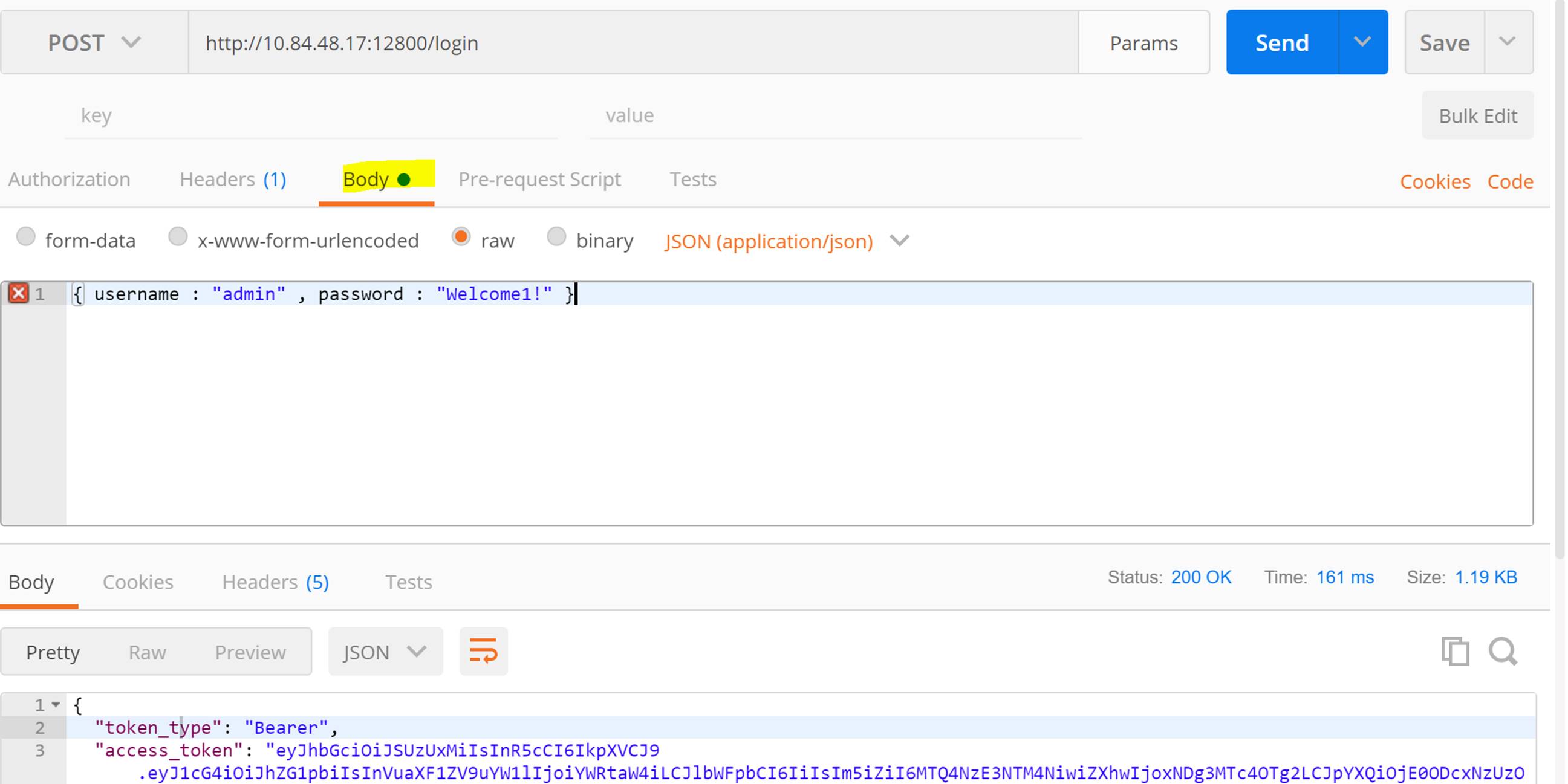
# Connecting Postman, Swagger and Microsoft R Server

[Swagger](http://swagger.io/) is the most popular framework for Rest APIs.

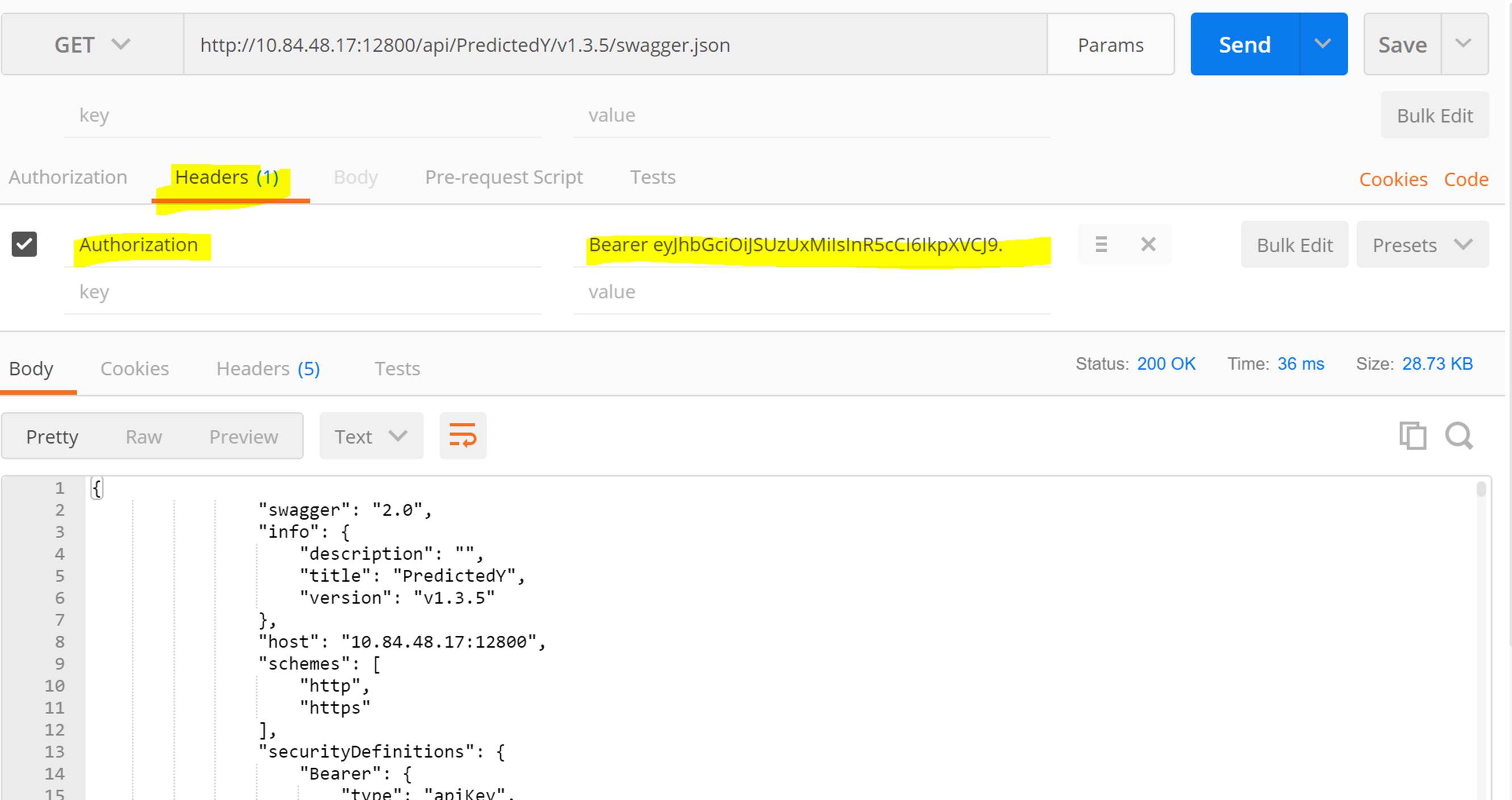
Importing Swagger files: <https://www.getpostman.com/docs/importing_swagger>

**Login Request:**





**Access the JSON with your secure token:**



[Postman](https://www.getpostman.com/) is a useful tool to try out REST calls and debug. There are similar tools of course that do the same thing.

I do like its ability to easily pull in our  [swagger doc](https://microsoft.github.io/deployr-api-docs/swagger/rserver-swagger-9.0.1.json)   and immediately have all APIs registered for interacting/trying.

I quickly compiled some setup steps should you ever want to try it out, mostly around authentication/LDAP and the Bearer Token:

@@@NOTE – There might be a better way to do this @@@

**1. Download** [**Postman**](https://www.getpostman.com/)**.  (I installed the App/OS not the chrome extension although I heard that was ok too)**

**2. Create an account with a un/pw**

**3. Import our swagger**

- First change the [swagger.json](https://microsoft.github.io/deployr-api-docs/swagger/rserver-swagger-9.0.1.json) to add your server’s `host`  For example:

 "host": "10.145.19.6:12800"

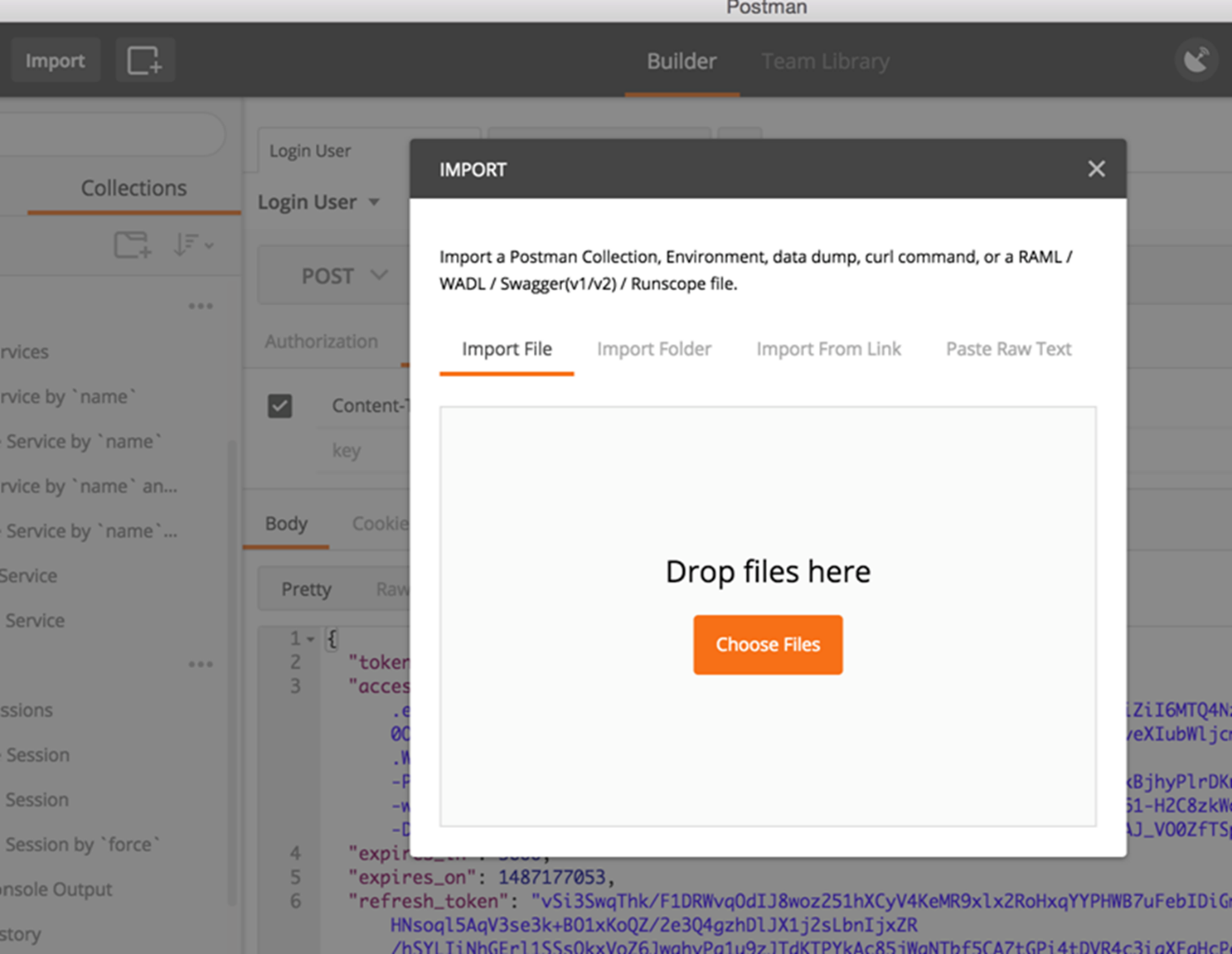
- Next change the `schemes` property to use only the **one** protocol that your server is configured to. This helps the Postman UI.

    "schemes": [

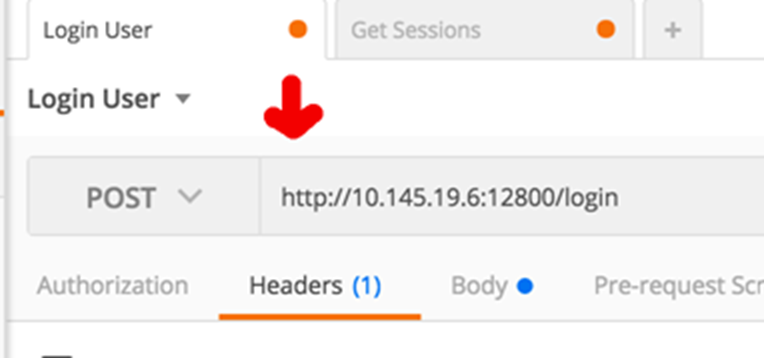
**"http"**

    ]

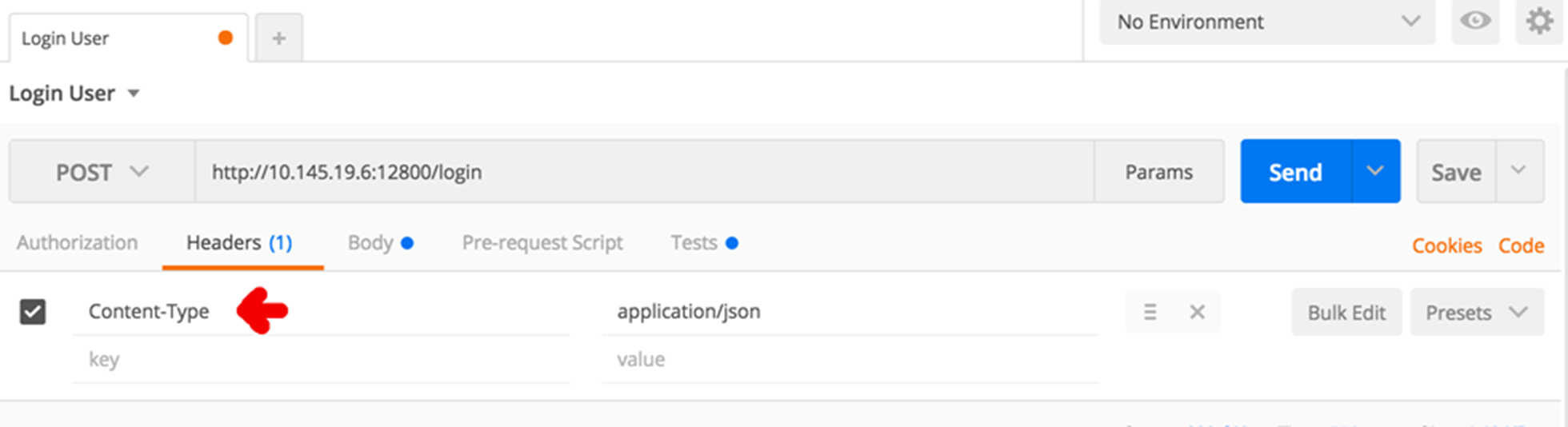
- Now import the swagger using the Postman UI:



**4. Once imported, under the `Login` tags to the left, select the `POST User Login`   (e.g. The /login API). Double check that your `host and protocol` looks correct:**

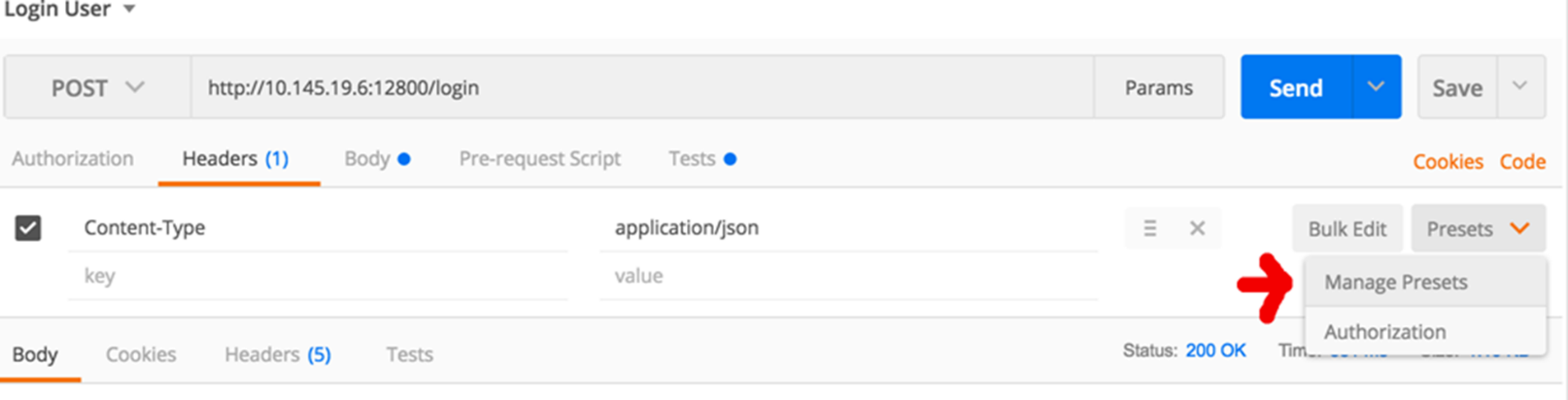


**5. Update your API headers. Select the `Headers` tab and add `Content-Type: application/json`:**



**Prepare for saving the Bearer Token for future API calls on the Header (one time thing)**

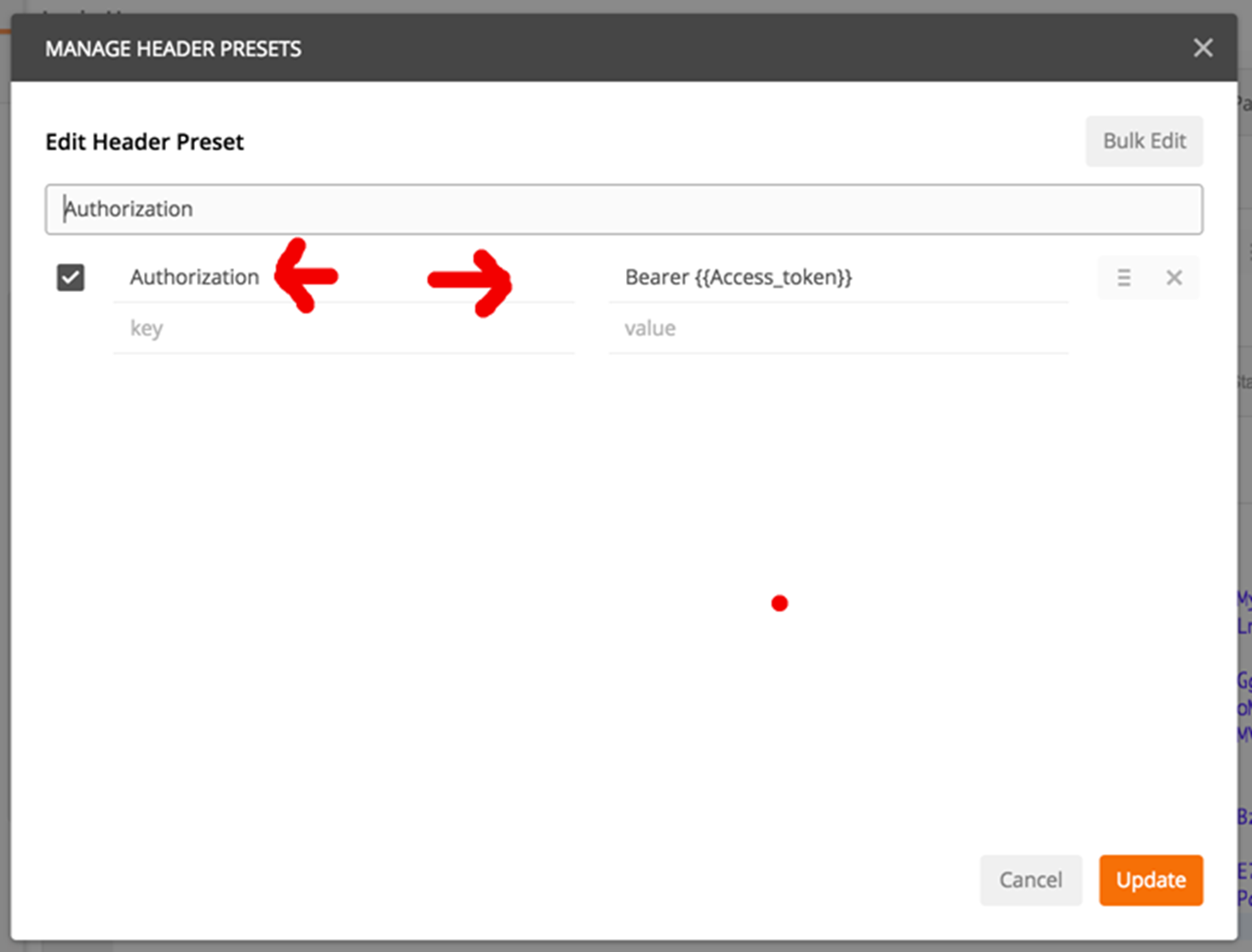
**6. While on the Header tab, select the `Manage Presets` item under the `Presets` menu on the center right:**



**7. Fill out a Postman header preset to store our access token for future API calls. Save it.**

key = Authorization

value = Bearer {{Access\_token}}



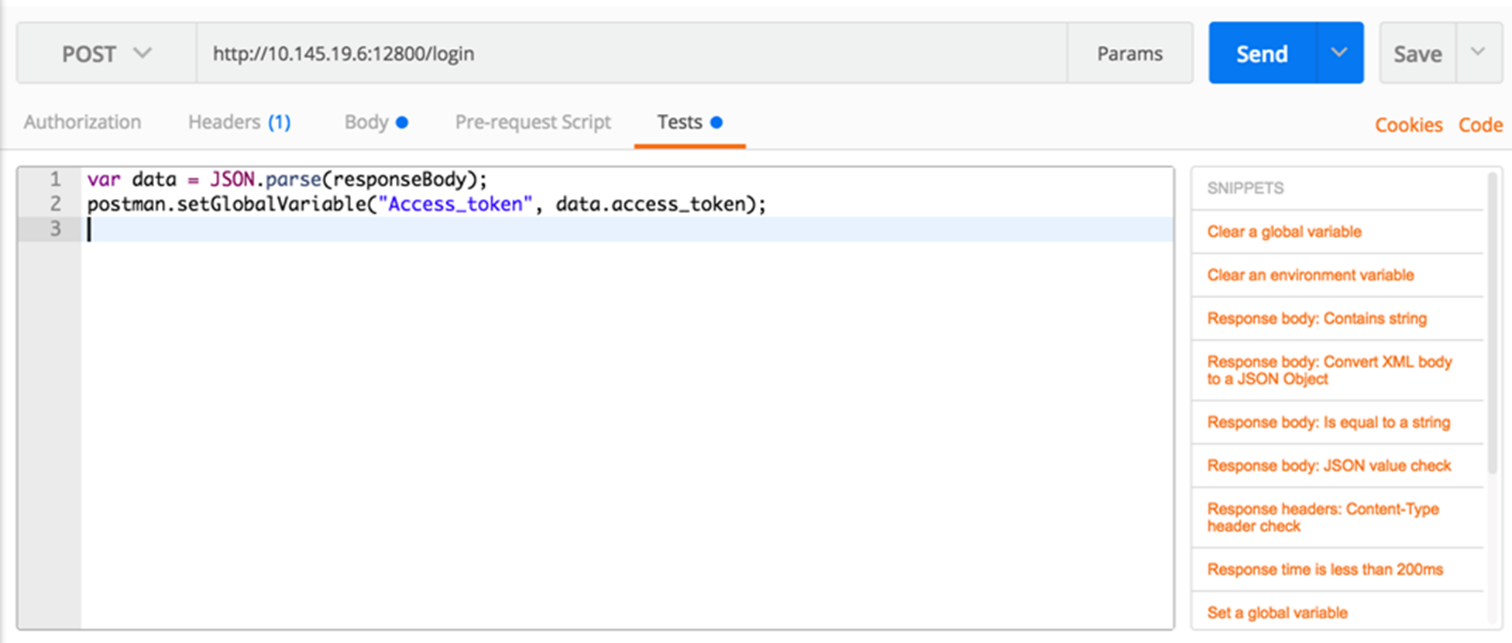
**8. Add post script to parse `access\_token` out of the response**

Under the `Tests` tab on the /Login API, add this script to pluck out the access token from the /login call and save it into a variable called `Access\_token`

var data = JSON.parse(responseBody);

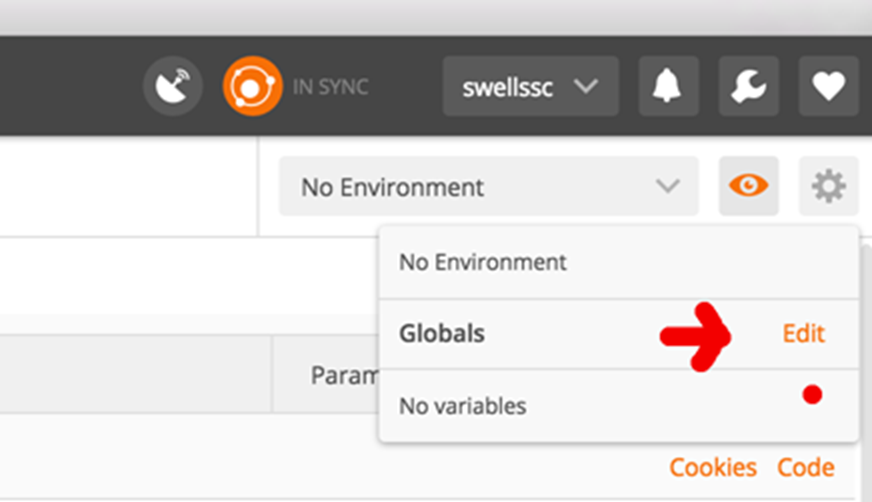
postman.setGlobalVariable("Access\_token", data.access\_token);

IMO – Tests should be called `Post-request Script` as it runs after the API call.



**Add global variable to work across all calls**

**9.  Goto the `Globals menu` on the top right menu bar and select `Edit`:**

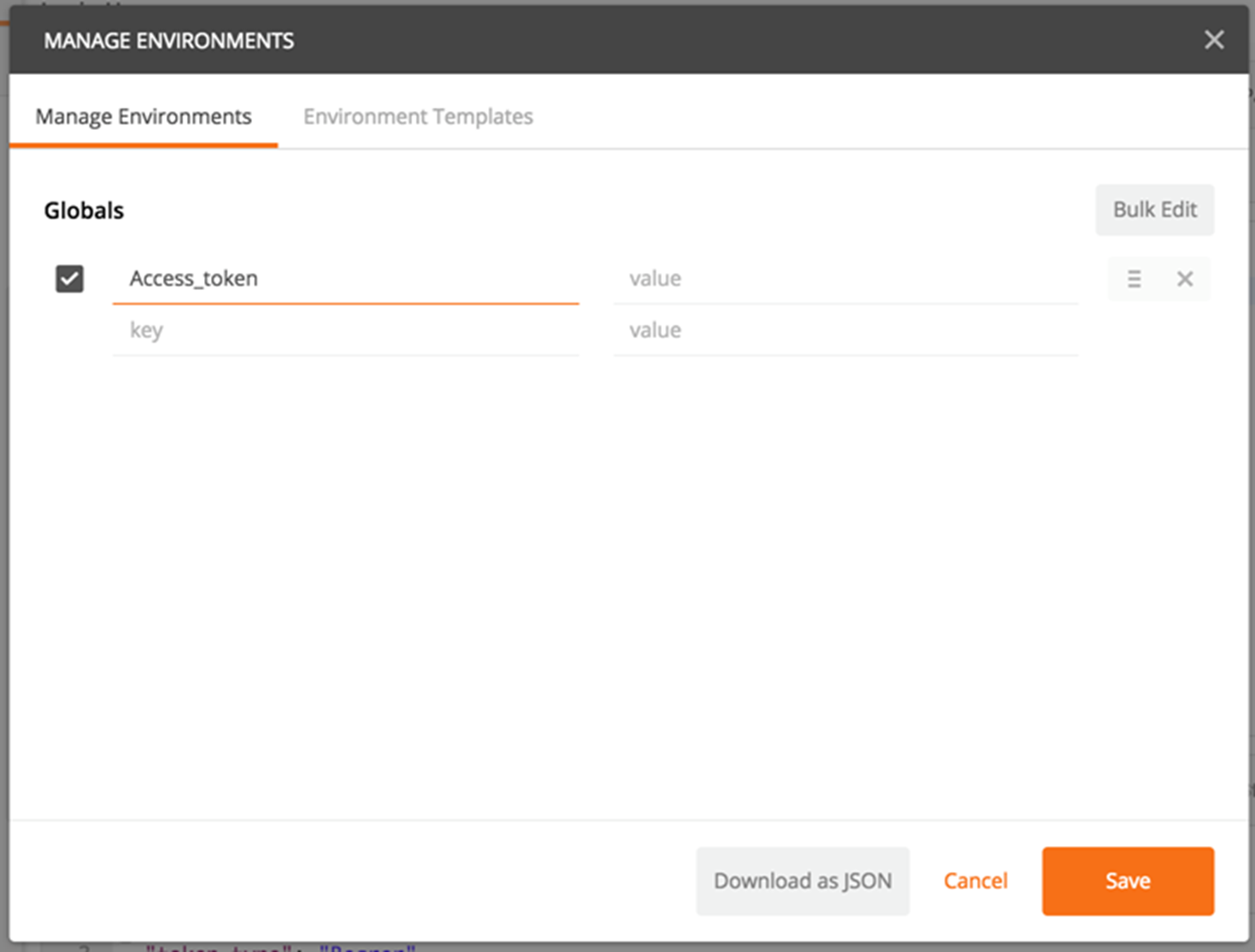


**10. Add the variable `Access\_token` and save**

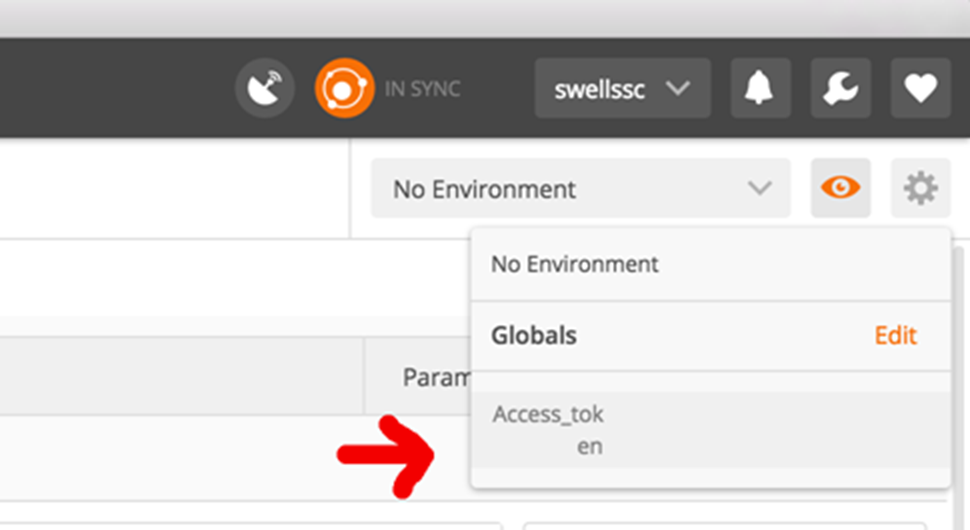
key = Access\_token

value =

Note the value of `Access\_token` is empty. This will be populated with the actual token value from **step 8** when /login is returned



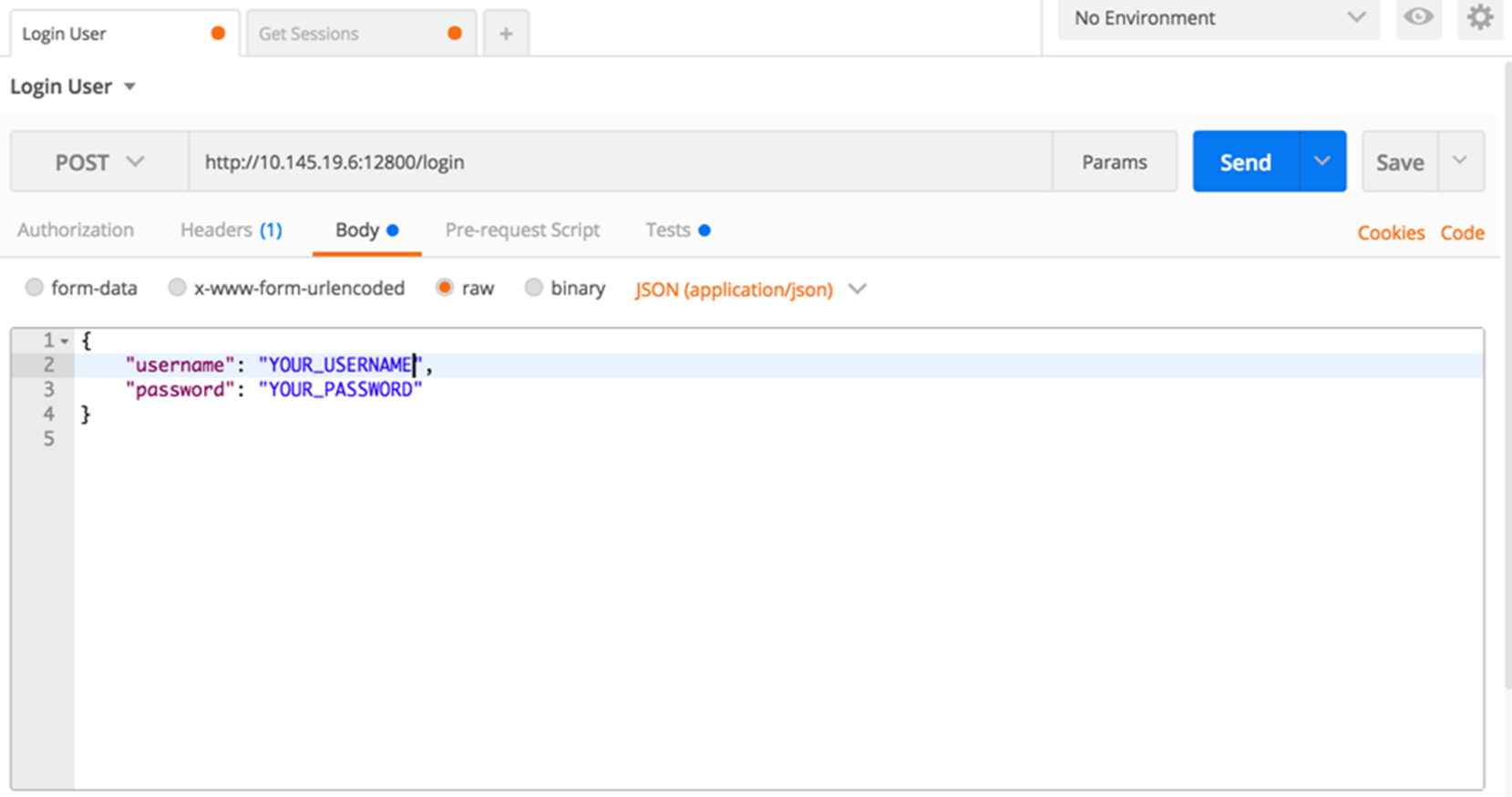
**11.  Check that your Global variable `Access\_token` as been saved:**



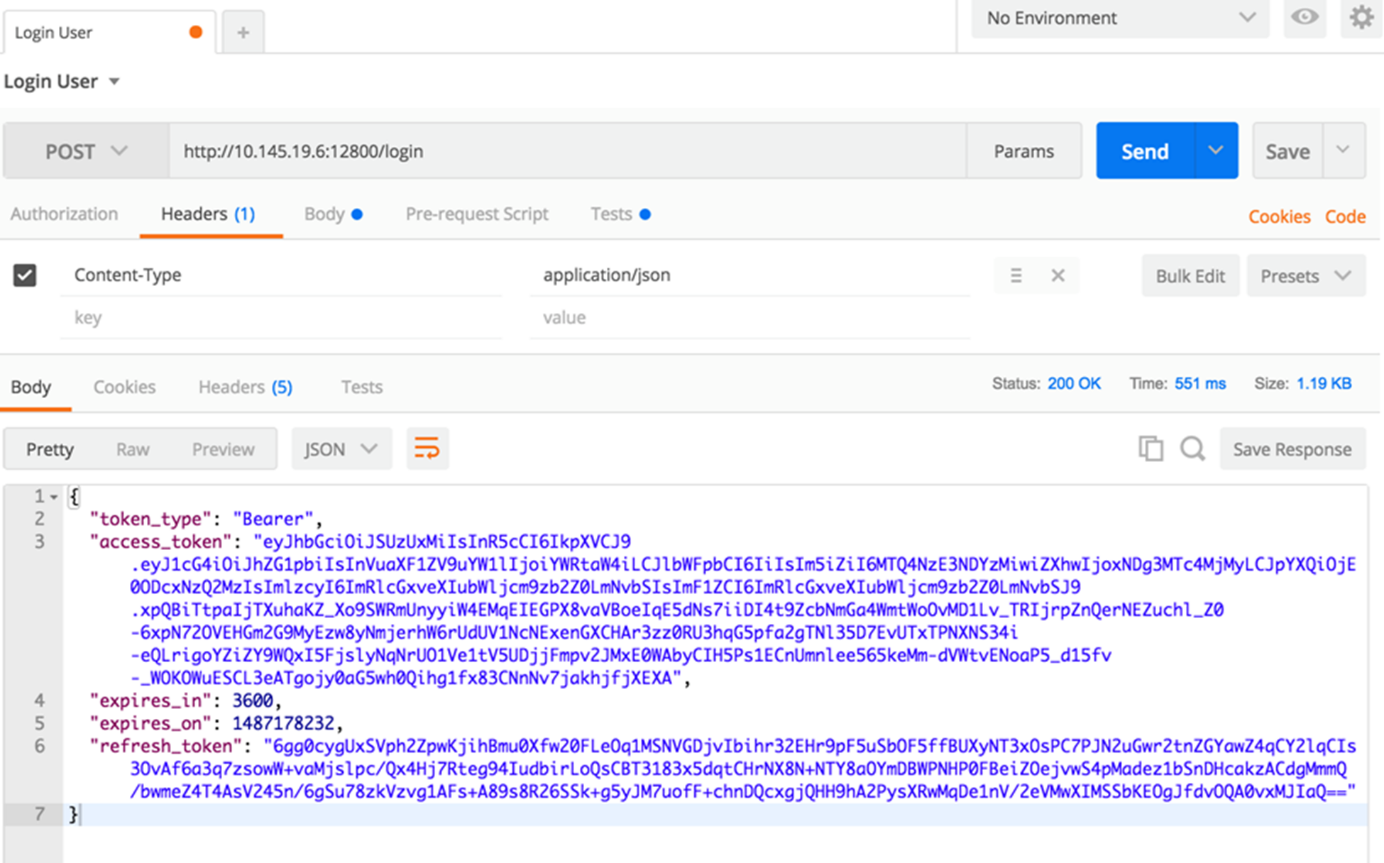
**12. Send your /login API call**

This will call /login and save the `access\_token` for future calls.

**- Fill in the parameters for username/password under the `Body` Tab selecting the `raw` option:**

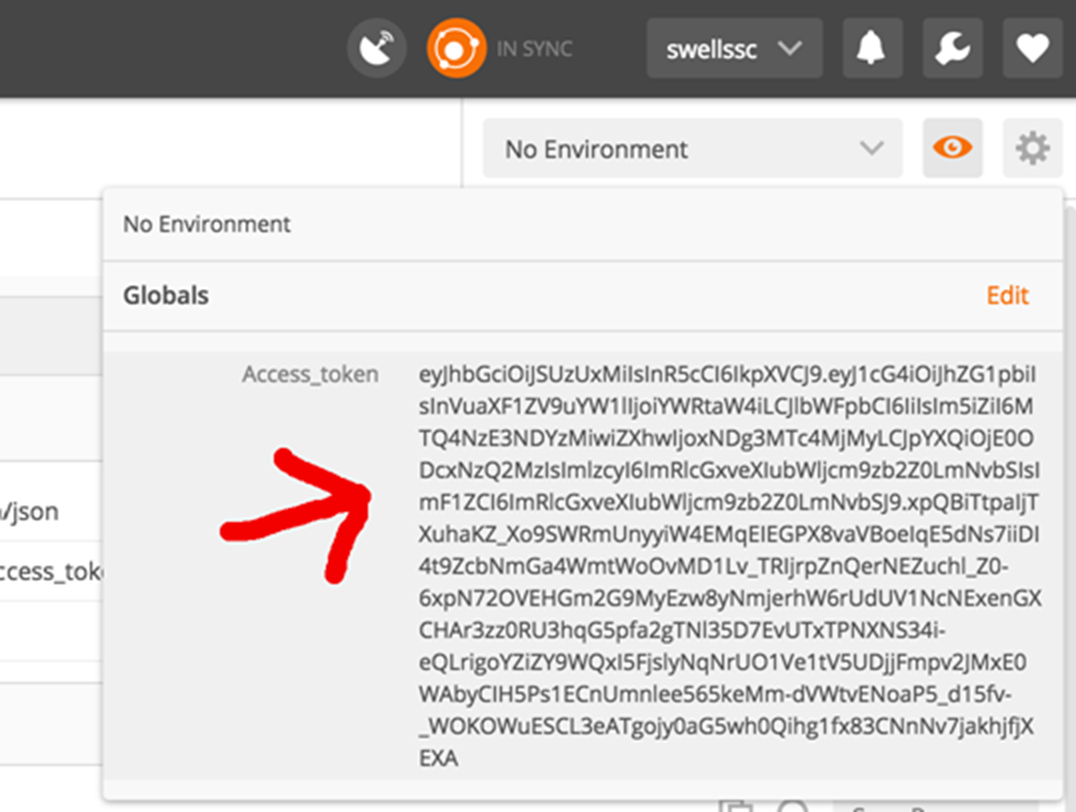


**- Send request**



**13.  Double check that your Global variable `Access\_token` has actually saved the /login call’s `access\_token` on the response:**

If it looks something like this, things worked!



**Now we are authenticated, try out other API calls using this token in the header**

**14. Select an other API, and prepare the headers:**

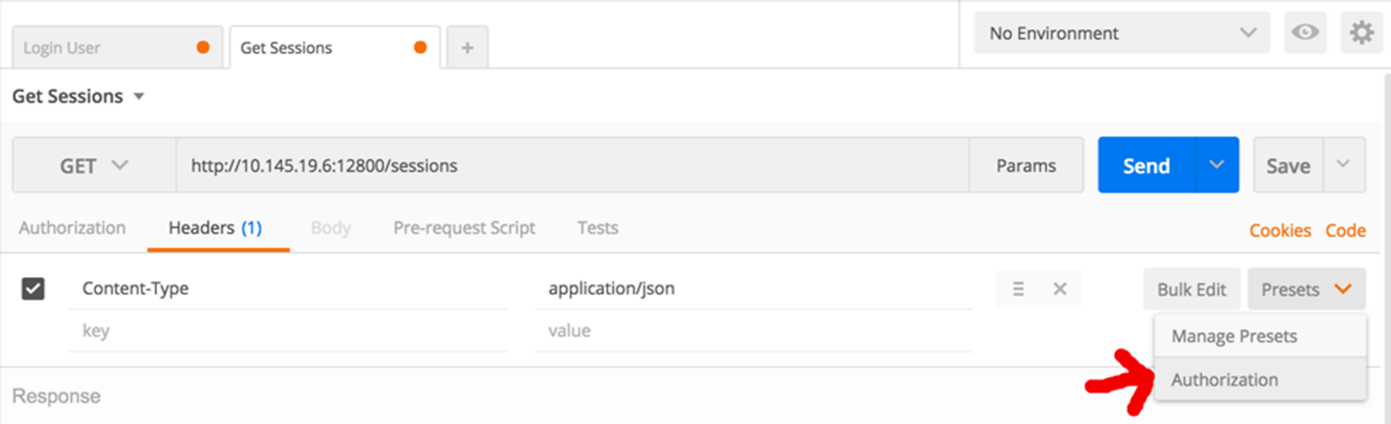
Here I chose `GET /sessions`

We need to add the headers:

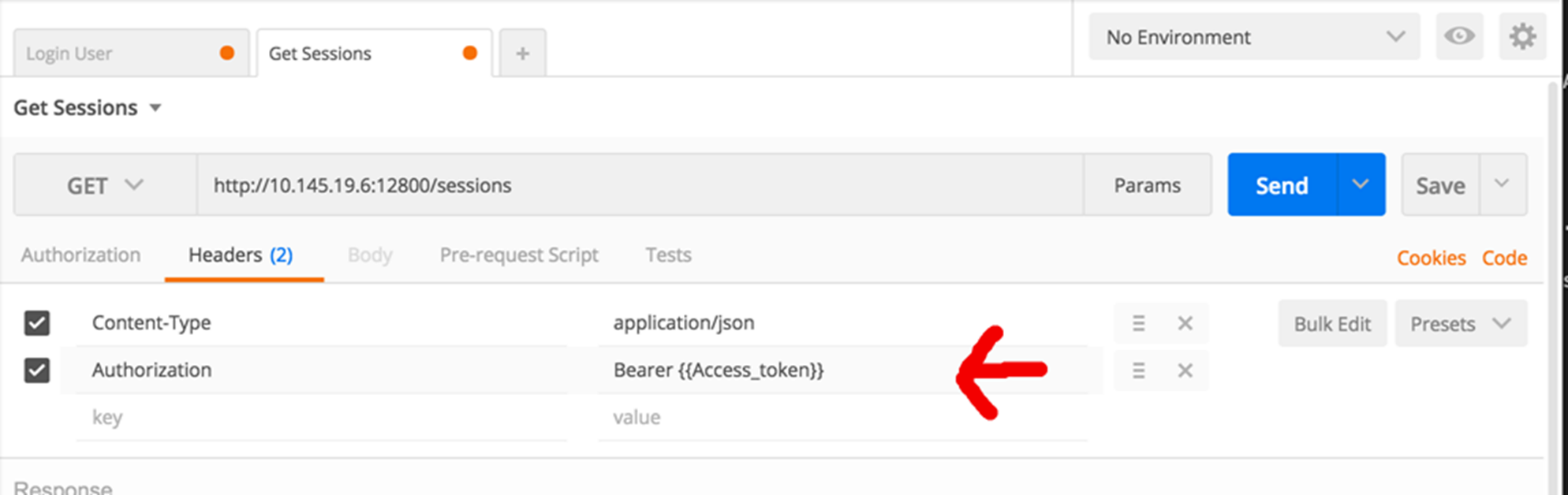
- Content-Type: application/json

- Our new Authorization Header

We can do this by selecting the `Authorization` item under the  `Presets` menu



This should then populate the header with the  `Authorization`  template we setup back in **Step 7.**

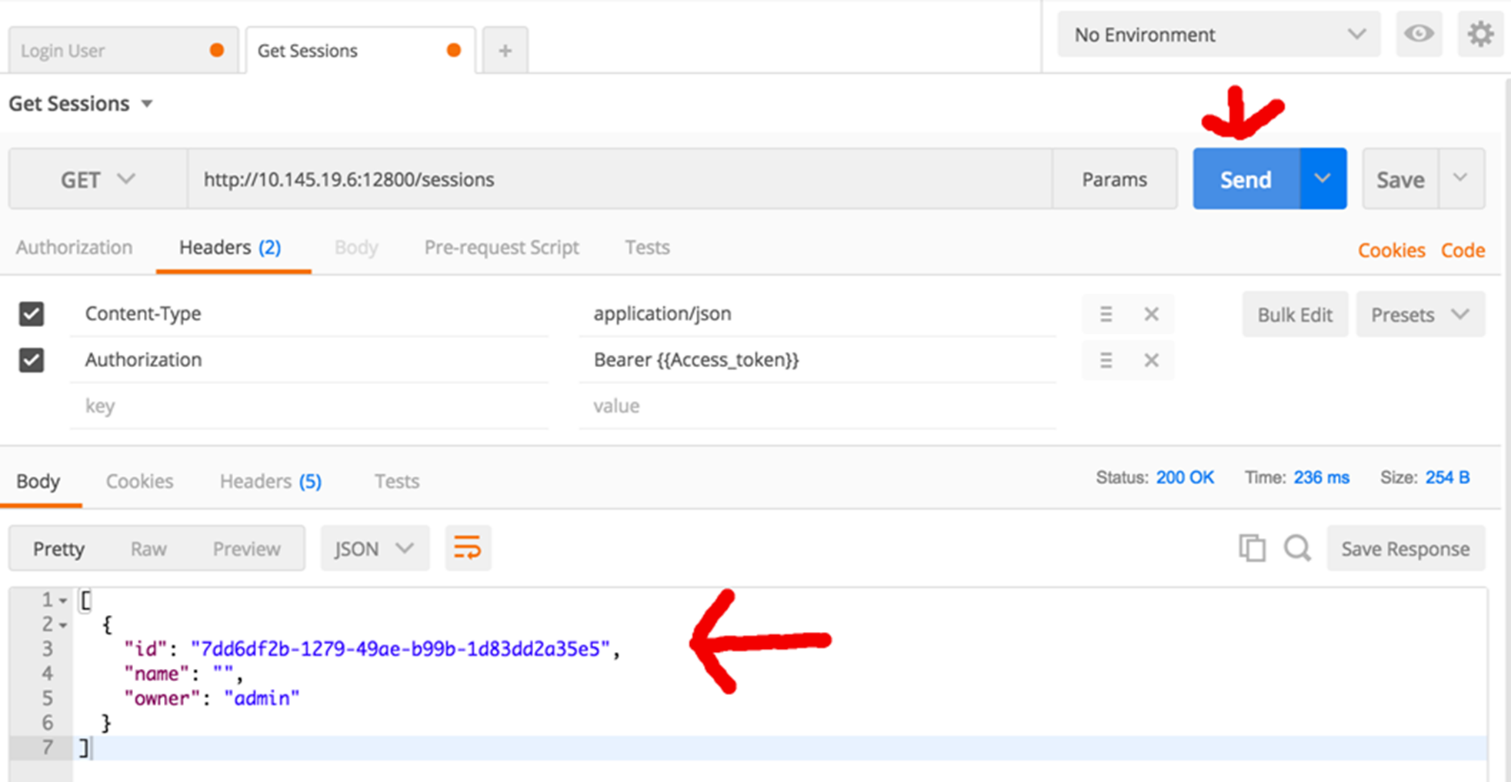


**Try it out**

**15. Select the `Send` button to call `GET /sessions`**

This should automatically populate the header in Postman with the `Access\_token` value from the previous steps.

The result should be a `200 OK` success:



**Repeat Step 14 for any new API call**

The fact that you need to explicitly `add the header` token under the `Presets menu` to all calls is unfortunate. However, it is only 1 step. Plus I think you can save your environment for later

and just repopulate the access\_token token with a new value by calling /login again.