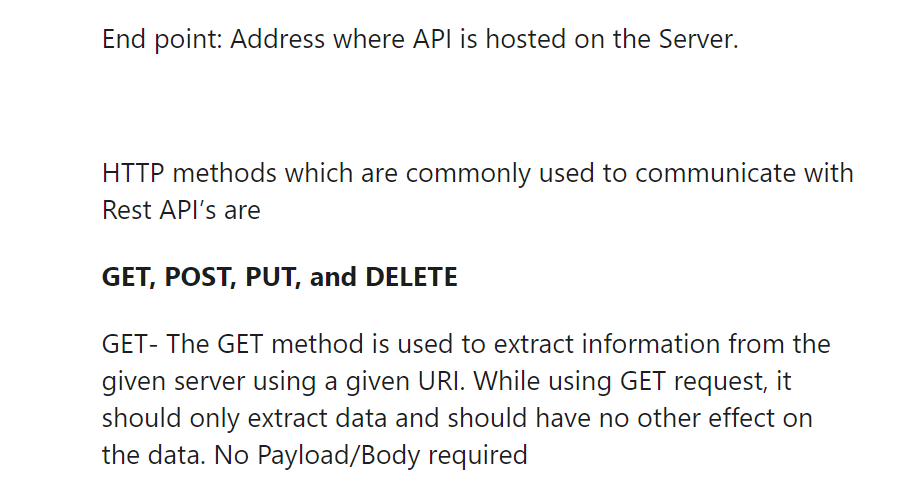
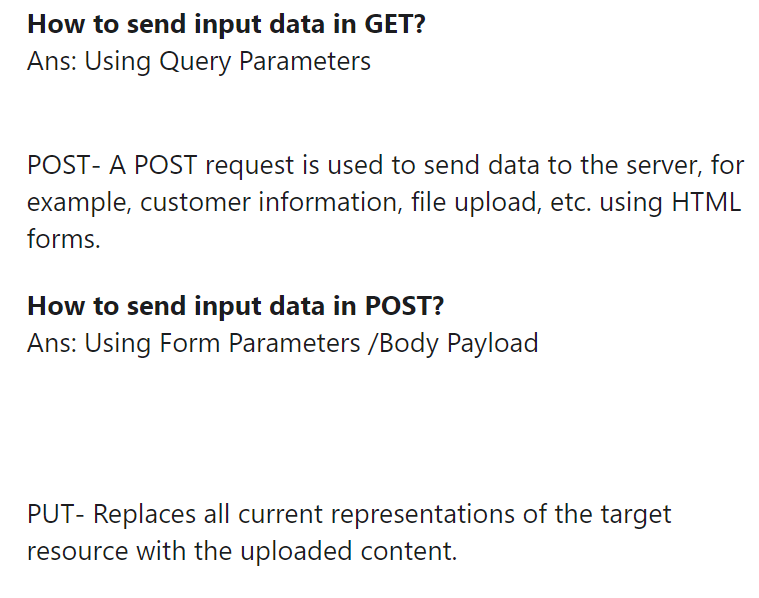
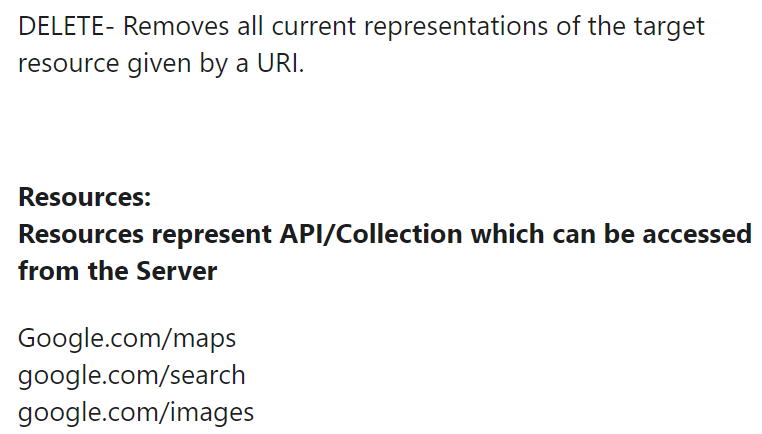
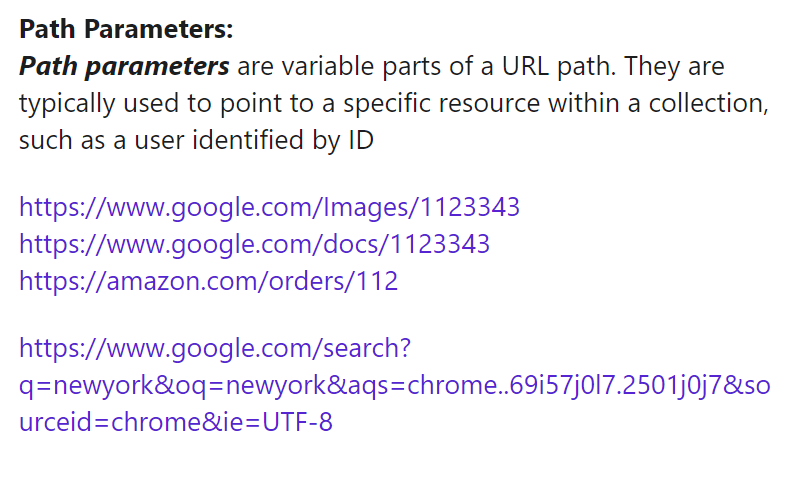
**

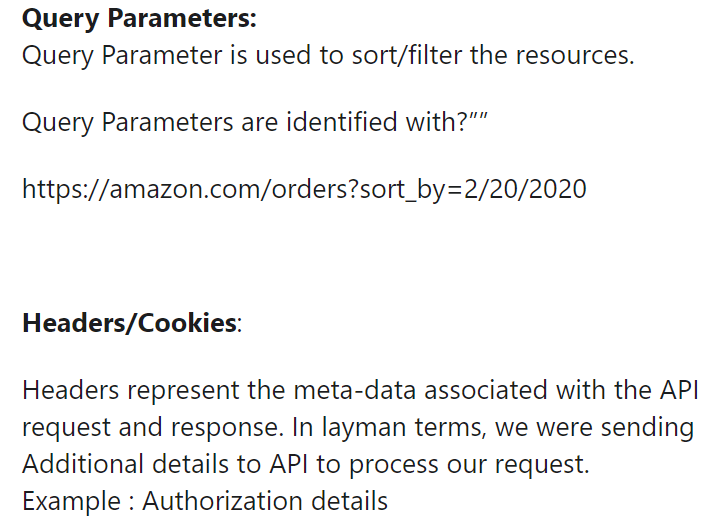
**Patch** : bulk update

**put** :at time update 1 resource

**

**

**

**

**

**Header** : Headers are nothing but additional information we are sending to server. Header may contain Authorization(uname, pwd), cookies, body-type, proxies, additional data for API to work. Request and response both can have associated headers. It posses key-value pairs. We use header() and headers() methods to pass header parameters.

* ***headers()****: returns****Headers***
* ***getHeaders()****: returns****Headers***

few frequently used headers :

1. Authorization : we send credentials in different way
2. Accept : Tells server what is accepted
3. Content type : tells what it contains (generally we use "Content-Type", "application/json")
4. Accept encoding : what kind of encoding we are supporting can mention here
5. Cookies : Sets cookies

Example :

header("Server", "Apache/2.4.18 (Ubuntu)") // checks from which server the response is coming from \*IMP check

RequestSpecBuilder abc = new RequestSpecBuilder();

abc.addHeader("Accept-Language", "en-US");

abc.addHeader("Accept-Encoding", "gzip, deflate, br");

abc.addHeader("amz-sdk-request", "attempt=1; max=3r");

abc.addHeader("Content-Type", "application/x-amz-json-1.1");

RequestSpecification def = abc.build();

RestAssured.given().log().all().header("Content-Type","application/x-amz-json-1.1")

.config(RestAssured.config().encoderConfig(

encoderConfig().encodeContentTypeAs("application/x-amz-json-1.1", ContentType.)))

.spec(def).when().body(request).post(URL).then().log().all();

Few other realtime headers :

Expected header "Content-Type" was not "application/json; charset=UTF-8", was "application/json;charset=UTF-8". Headers are:

Date=Tue, 14 Sep 2021 08:49:02 GMT

Server=Apache/2.4.18 (Ubuntu)

Access-Control-Allow-Origin=\*

Access-Control-Allow-Methods=POST

Access-Control-Max-Age=3600

Access-Control-Allow-Headers=Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With

Content-Length=194

Keep-Alive=timeout=5, max=100

Connection=Keep-Alive

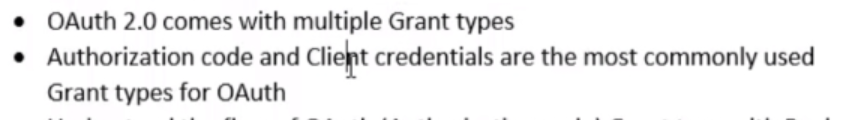
Content-Type=application/json;charset=UTF-8

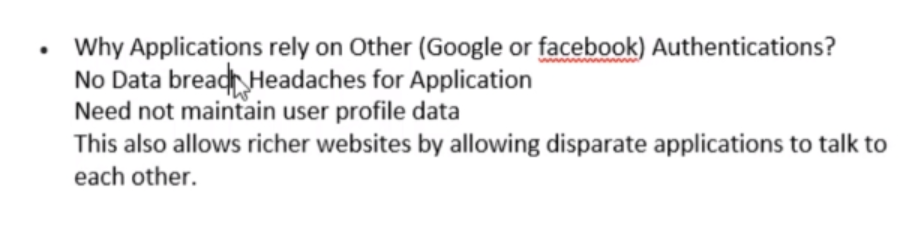
**Authentication** :

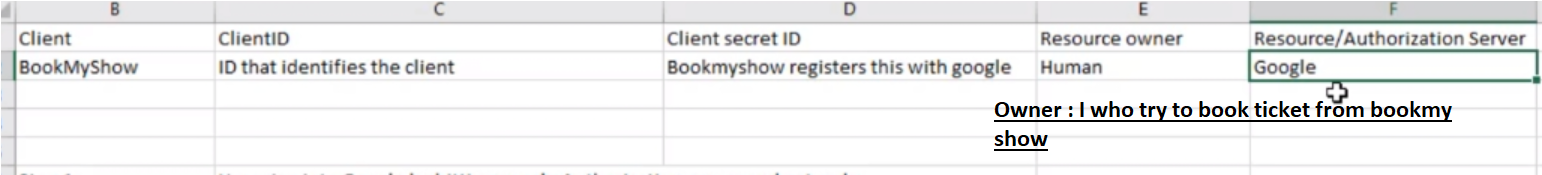
few popular types are :

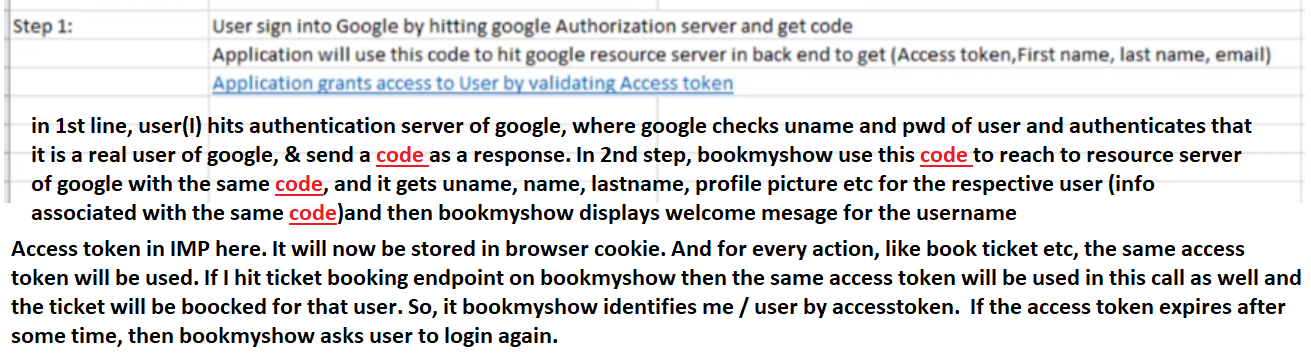
* 1. Basic authentication (use Uname and Pwd)
  2. Digest authentication
  3. Form authentication
  4. Cookie based
  5. OAuth 1 and OAuth 2
* Basic authentication (use Uname and Pwd) : Here we it uses base64 to encrypt credentials and then send it over.
  1. Given().auth().preemptive().basic(“username”,”password”).when().get(“URL/end point”).then () .. .. .
  2. Given().auth().basic(“username”,”password”).when().get(“URL/end point”).then () .. .. .
* Digest : It is same as basic but it uses other than base64 for encryption
  1. Given().auth().digest(“username”,”password”).when().get(“URL/end point”).then () .. .. .

**OAuth 2.0 :** OAuth 2.0 is the industry-standard protocol for authorization. Ex. : If you just hit twitter’s endpoint, you directly can not see every tweet. First we have to authenticate ourselves. If we are a valid and authorized user to see that tweet, then only we are allowed to see the tweet. OAuth comes with multiple grant types.

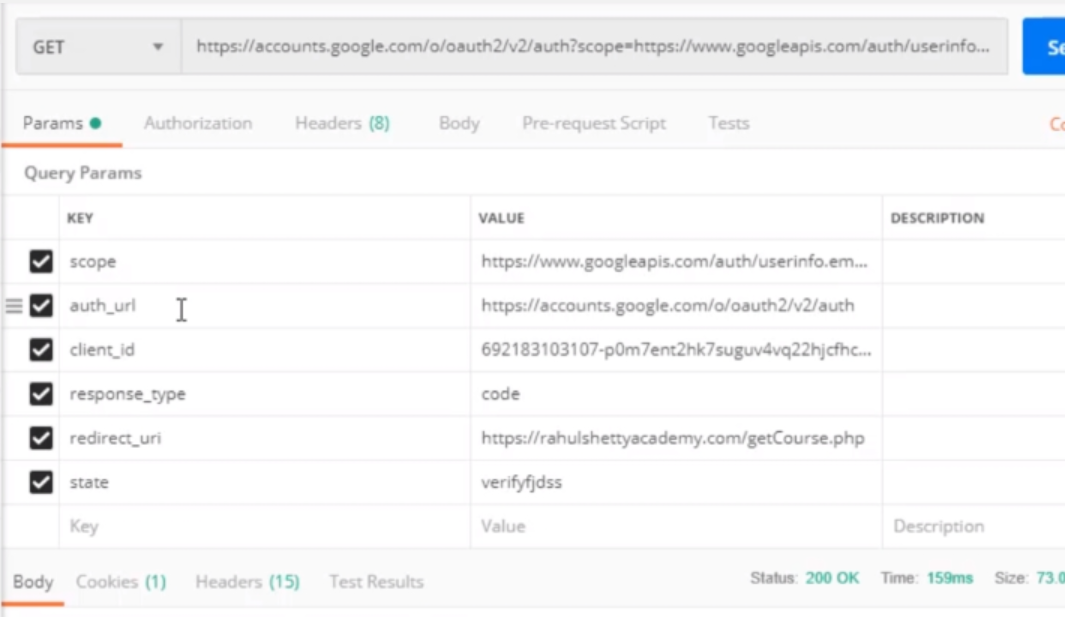








1. **TO GET CODE :** Below are some standard parameters used : 99% developer give us the URL directly (URL to get code). But for understanding purpose, we will see the actual meaning of each section / part / parameter used in the URL.



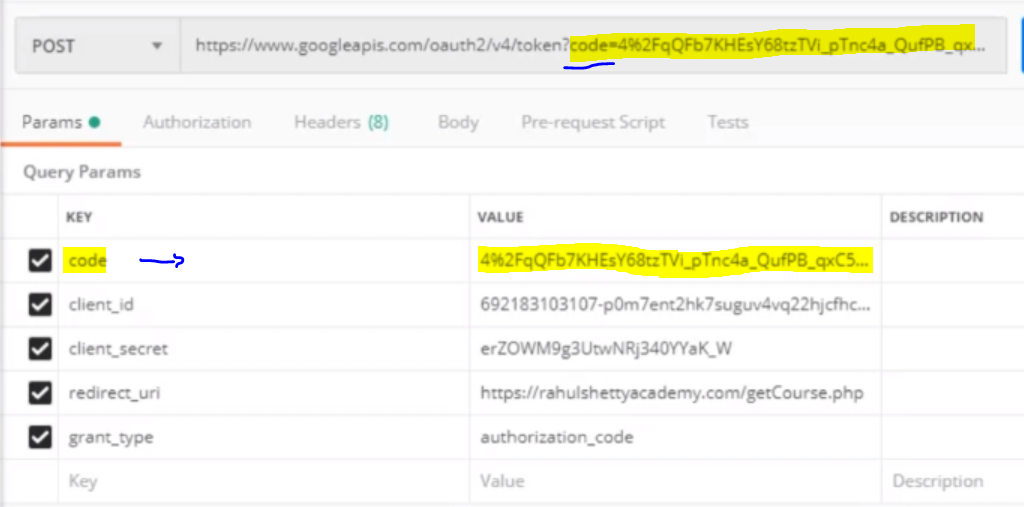
Below info used to form URL. We get URL (URL to get code) from **developer** and these all info also we can get from developer only.

1. Scope : Application / client is asking what all details needed for me (resource owner / I who is logging in)
2. auth\_url : which server we are trying to authorized with (google who authorizes the user)
3. client\_id : When bookmyshow 1st time get registered with google, it gets this id.
4. Response\_type : what response we expect from google (we expect authorization code)
5. Redirect\_url : After login to google, where to go back? (here we come back to bookmyshow)
6. State : it is optional parameter used for security.

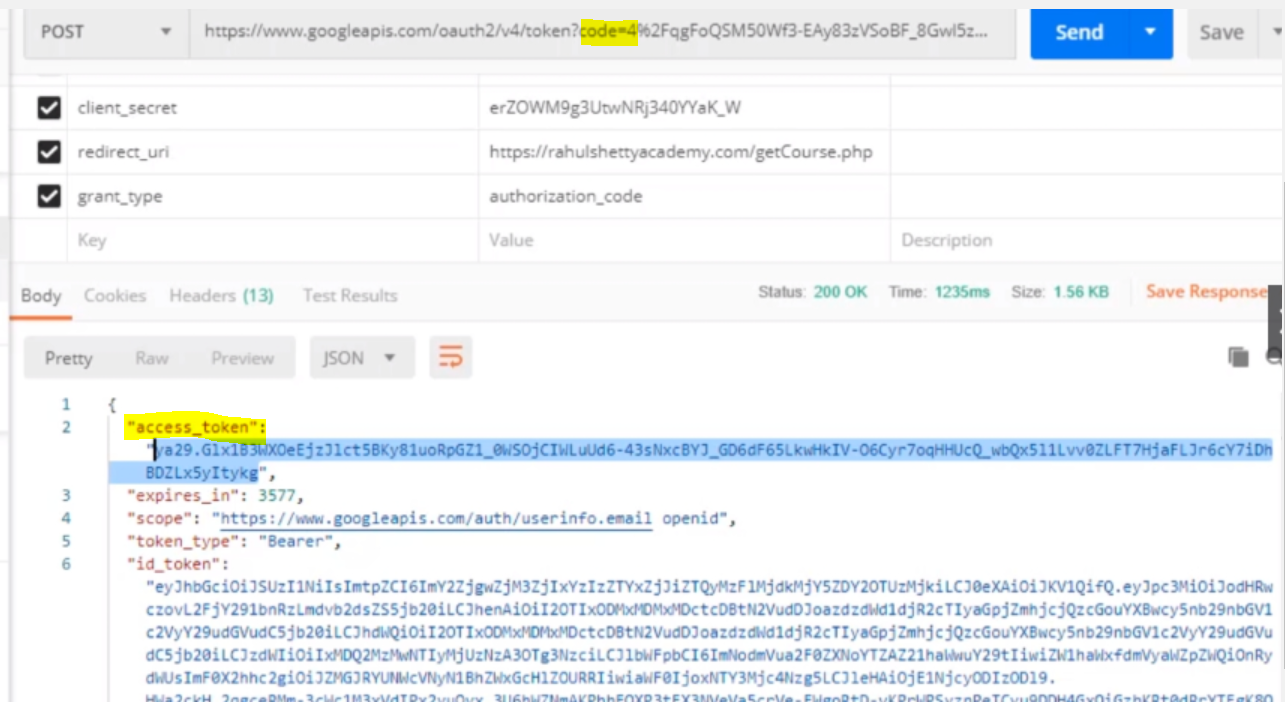
Now get the URL (URL to get code), if we hit the URL, will get the **code.** If we paste it in postman, there also we can see the **code**. It is a **GET** request.

THIS response URL also we get from **DEVELOPER.** So, request and response URLS we get from developers only

1. **Now use the code from above step in 2nd URL to get access token :**



**Now, use the code and add it in 2nd URL given by developer. And hit the 2nd URL after adding the code in it. As a response, we get ACCESS TOKEN. 🡪 it is a POST request**



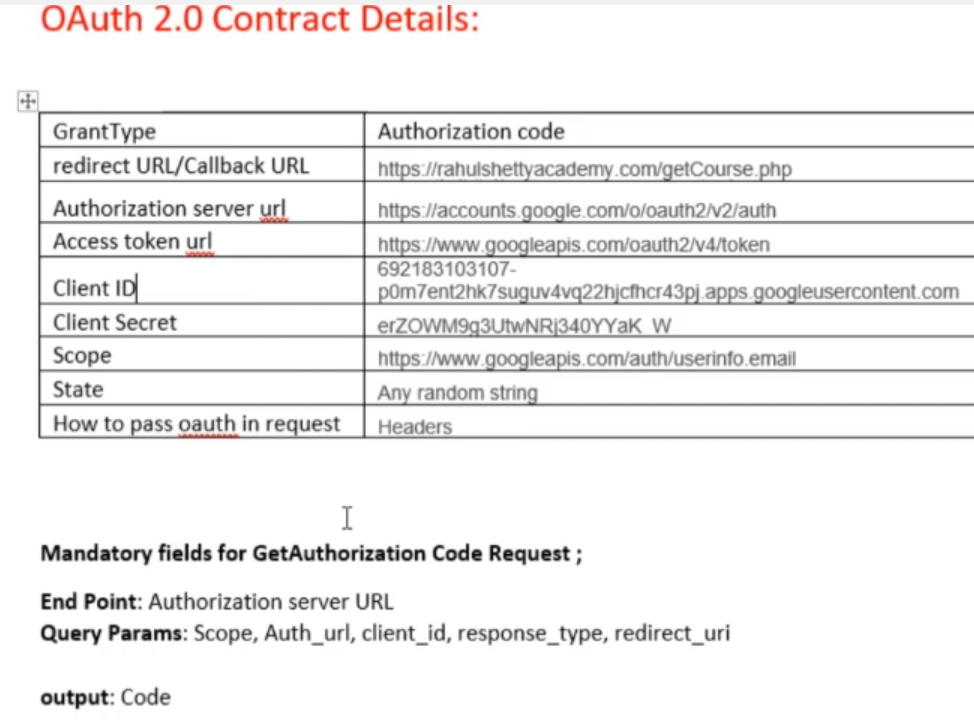
Now, the access token can be used to perform further operations.

**NOTE** : We get 2 URLs from developers

1. 1st URL after hiting we get a code(it’s a GET request). Need selenium to login and then response URL posses the code.
2. 2nd URL + add above code in it and then hit it > we get access token (it’s a POST request)

Now, use access token for all further operations

Contract document :



Redirect url / call back URL : bookmyshow

Authorization server url : google [once authorized, we get code and we use the code in below step]

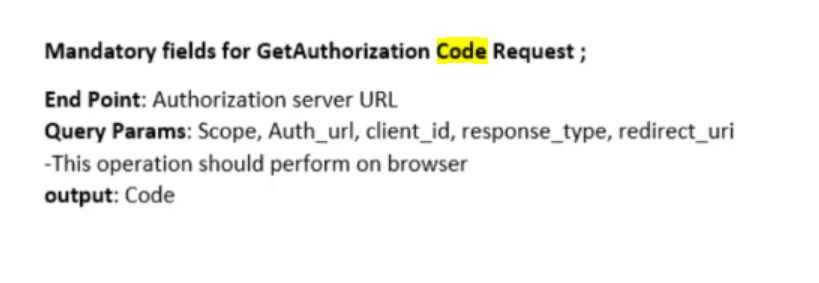
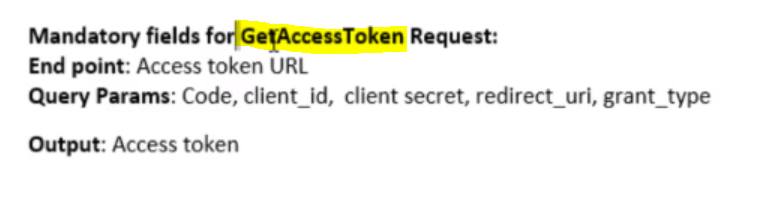
Access token URL : google [use the code from above step to get access toke]

Client id and client secret : unique for client

Scope : what information google shares with us

State : any random string used for security

**VIPM :**



**headers** : (mentioned in contract) USED BY **HIRESH**

content type

caller id

corelation id

date and time

**Authorization** :

header : jwt token (Auth 2.0)