COBBLESTONE
SOFTWARE

COBBLESTONE SOFTWARE
REST API 2.0 Technical Overview

Version 1.3
April 9,2021

.

Table of Contents

Overview

CobbleStone Software's Contract Insight application offers the functionality to integrate with outside systems. Application Program Interface delivers data and between devices and programs. Representational State Transfer program implementation increases efficiency of communication in computing systems.

What is API

An application programming interface (API) is a messenger that processes requests and ensures seamless functioning of enterprise systems. API enables interaction between data, applications, and devices. It delivers data and facilitates connectivity between devices and programs.

What is REST

Representational state transfer (REST) is a programming architectural implementation intended to increase the efficiency of communication in computing systems. It embodies the idea that the best way to share large amounts of data between multiple parties is to make that data available on-demand by sharing references to that data rather than a complete copy of the data itself. Systems which implement REST are called 'RESTful' systems.

More about REST

An example of a non-RESTful real-world system would be the traditional home movie collection. In order to have access to any given movie, the library owner must obtain a physical copy of it. This results in substantial waste as more copies are in existence than are in use at any given moment. Also, the time required to add new titles to the library is generally non-trivial. Streaming video is the RESTful counterpart to the home library. Instead of having a complete copy of every movie stored in the home, the movie is referred to by its title only and the content of the movie is streamed on demand.

CSS API 2.0

This would be the next generation of APIs from CobbleStone as RESTful implementation where it would be installed as separate application side by side with Contract Insight either on cloud or on-premise.

Consumer of the API (End-User) would be able to access desired endpoints dedicated for each entity, for example to access Contracts table then the consumer would create a web call to the REST endpoint passing entity name as first parameter with any query criteria if needed.

Operations on data through this APIs would reflect the same as Contract Insight where only Insert and Update are available and Delete is not allowed.

***OAuth** is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords.*

How this is authenticated:

- This version of API has its own separated list of authenticated users but linked to the users list in Contract Insight for logging purpose only and the authentication mechanism

How this is secured:

- Our APIs would use SSL to authenticate, send and receive data from the consumer.

How This is triggered:

- Our API does not offer notification or monitoring functionality to the consumer. It does not notify the consumer about a change that happened to any record or field within ContractInsight data so it cannot be used to implement a real time system but only "Near Realtime/Scheduled" and "Batch".

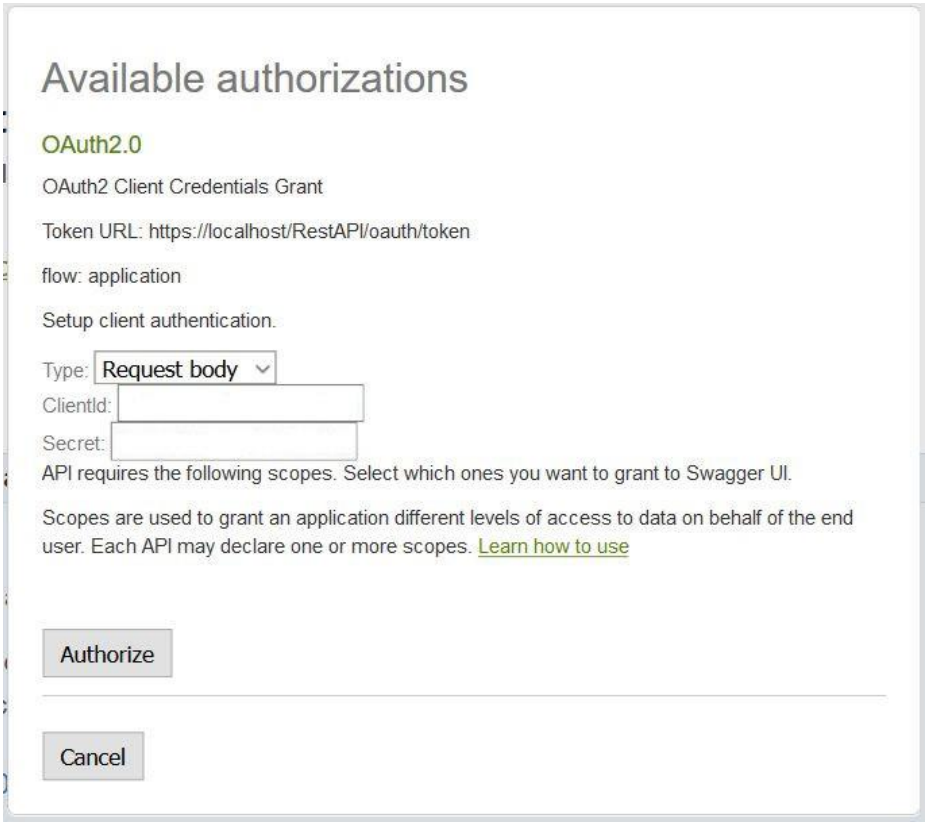
Access Level

- The username and password must belong to an active system administrator.

API Endpoints Review

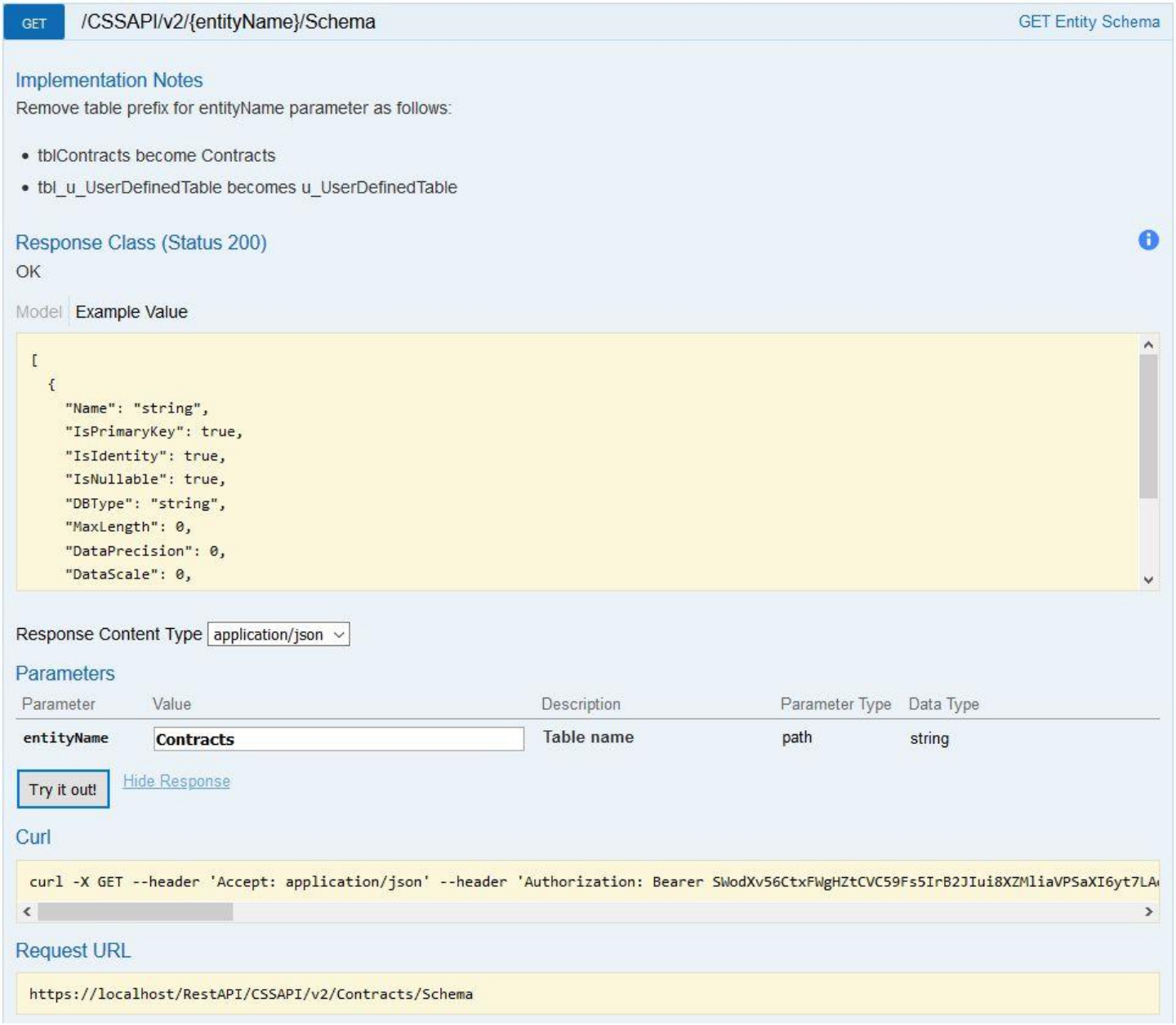
Authentication Endpoint

- This version of API is OAuth2 compliant with client credentials grant type, this endpoint responds with an access token that be default expires after 4 hours and non re-usable refresh token that expires 1 month after the date and time it was issued.



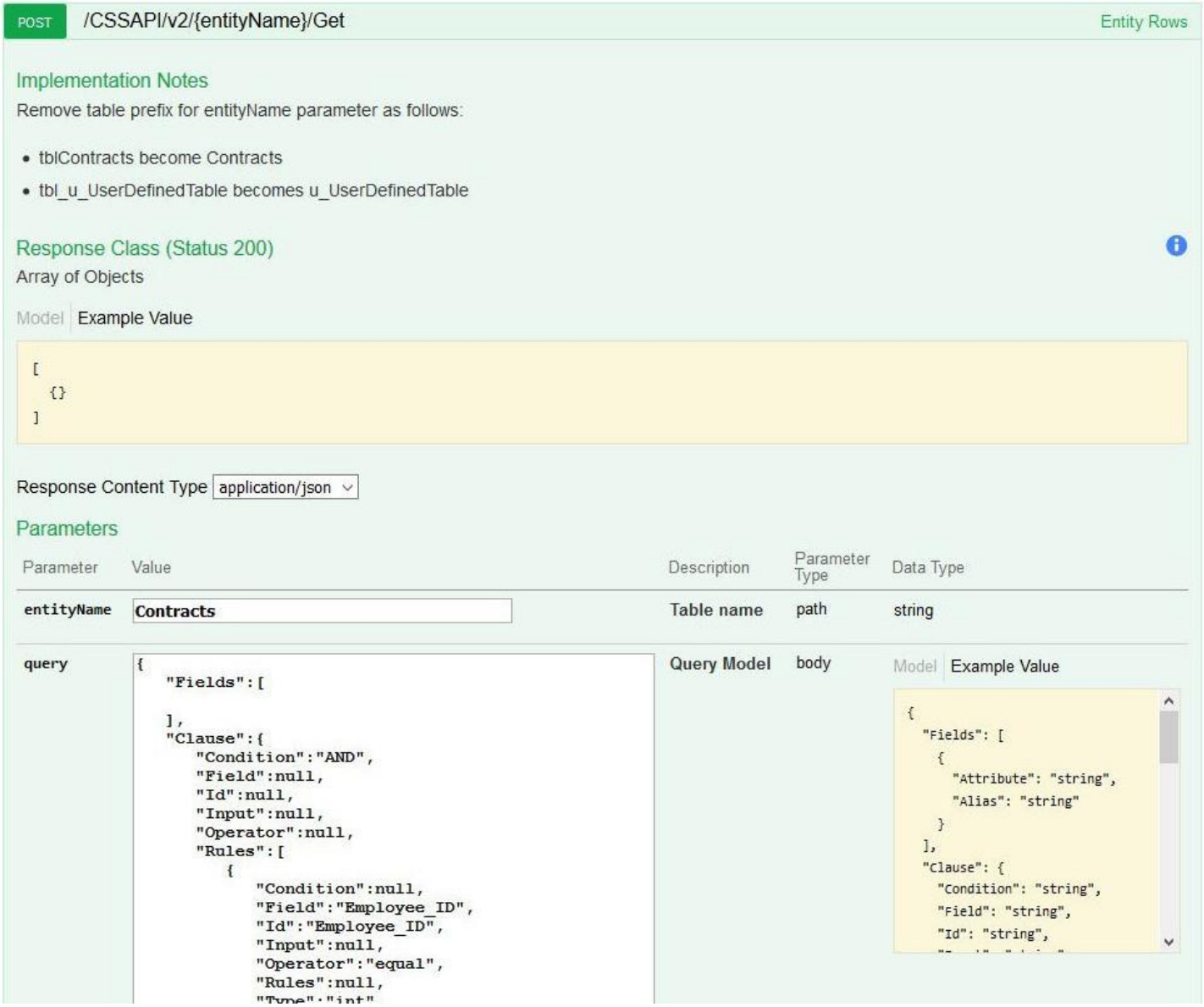
Schema Endpoint

- A call for single entity (table) specified by name from the consumer and it respond with metadata about all the exposed columns (i.e consumer allowed to query upon them which for example IMAGE and BLOB columns and excluded), response is basically a collection of column properties.



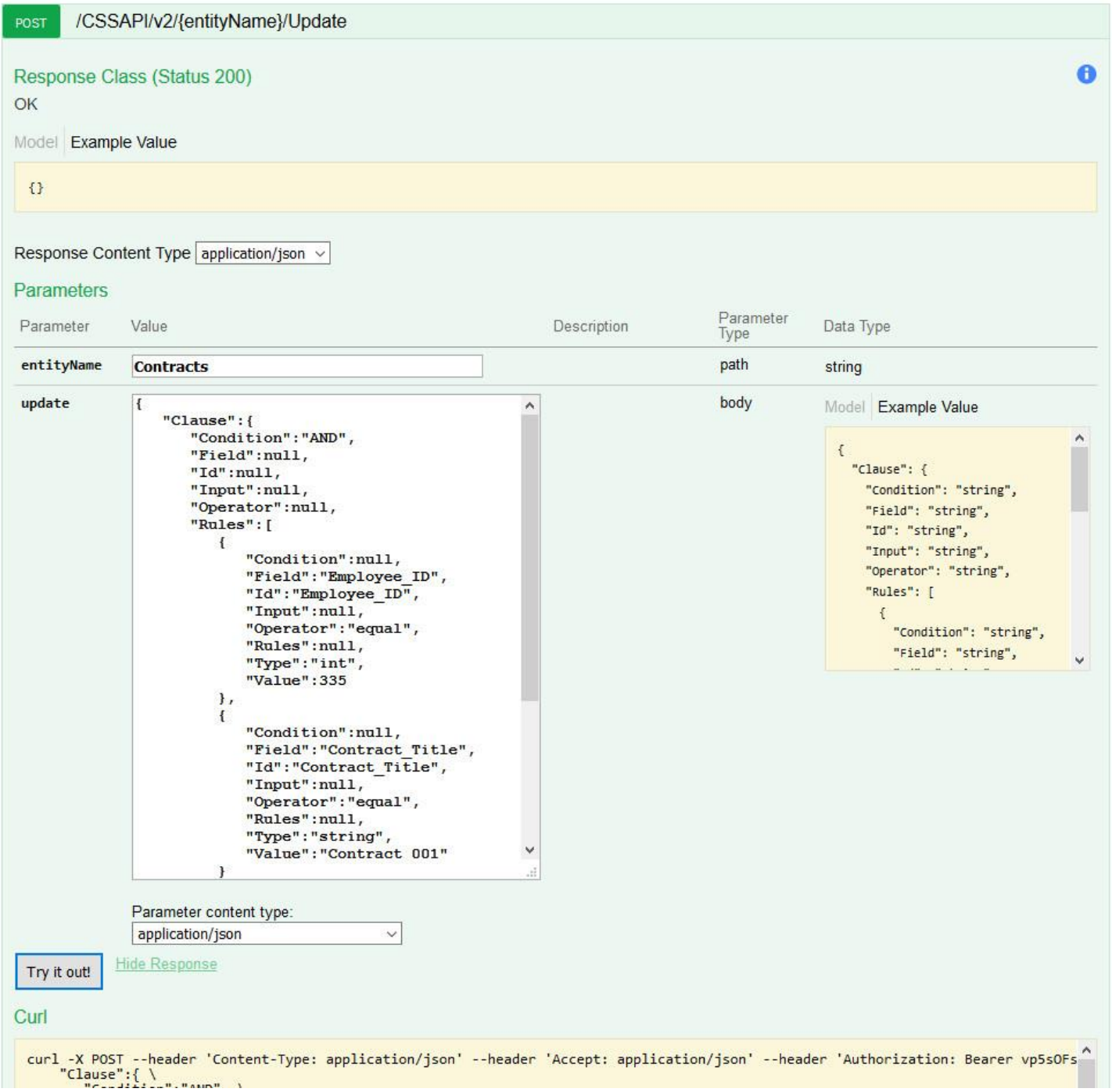
Query Endpoint

- Equivalent to Select statement in SQL against a single entity (table) but since this API, version do not support free SQL expression then most of SQL tags should be provided within specific nested data structure but and since it supports paging then it is limited to 1000 rows maximum per call and response is dynamic collection of entity's rows where columns mentioned in Schema End-Point above are excluded.



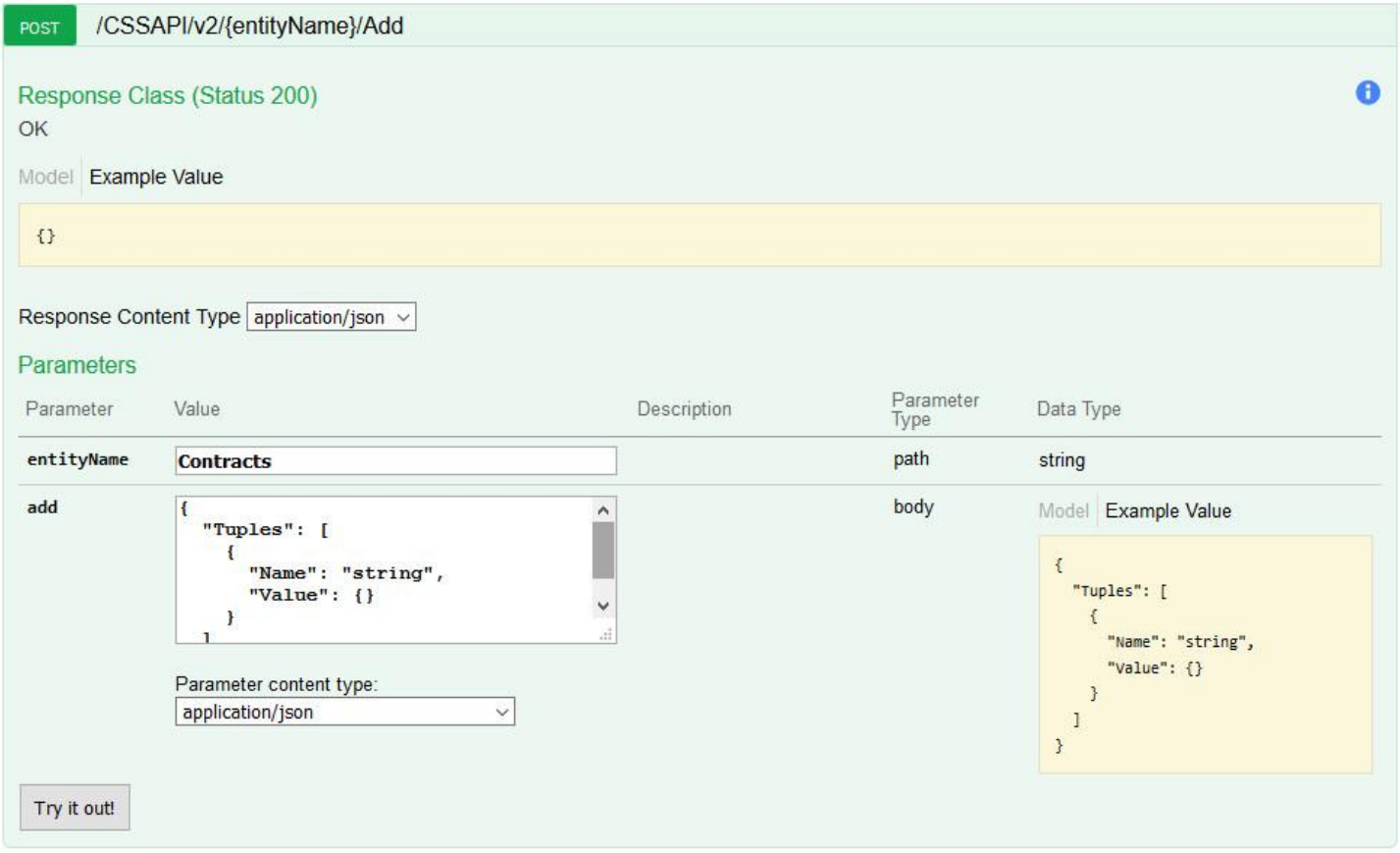
Record Update Endpoint

- Updates one or more rows in a single entity (table) base on criteria if exists but limited only to 500 rows maximum per call, SQL tags should be provided within specific nested data structure and consumer have the option to have CI workflows triggered after each individual row update or not, update only allowed on columns that are not either IMAGE or BLOB as mentioned above, response is number of affected rows.



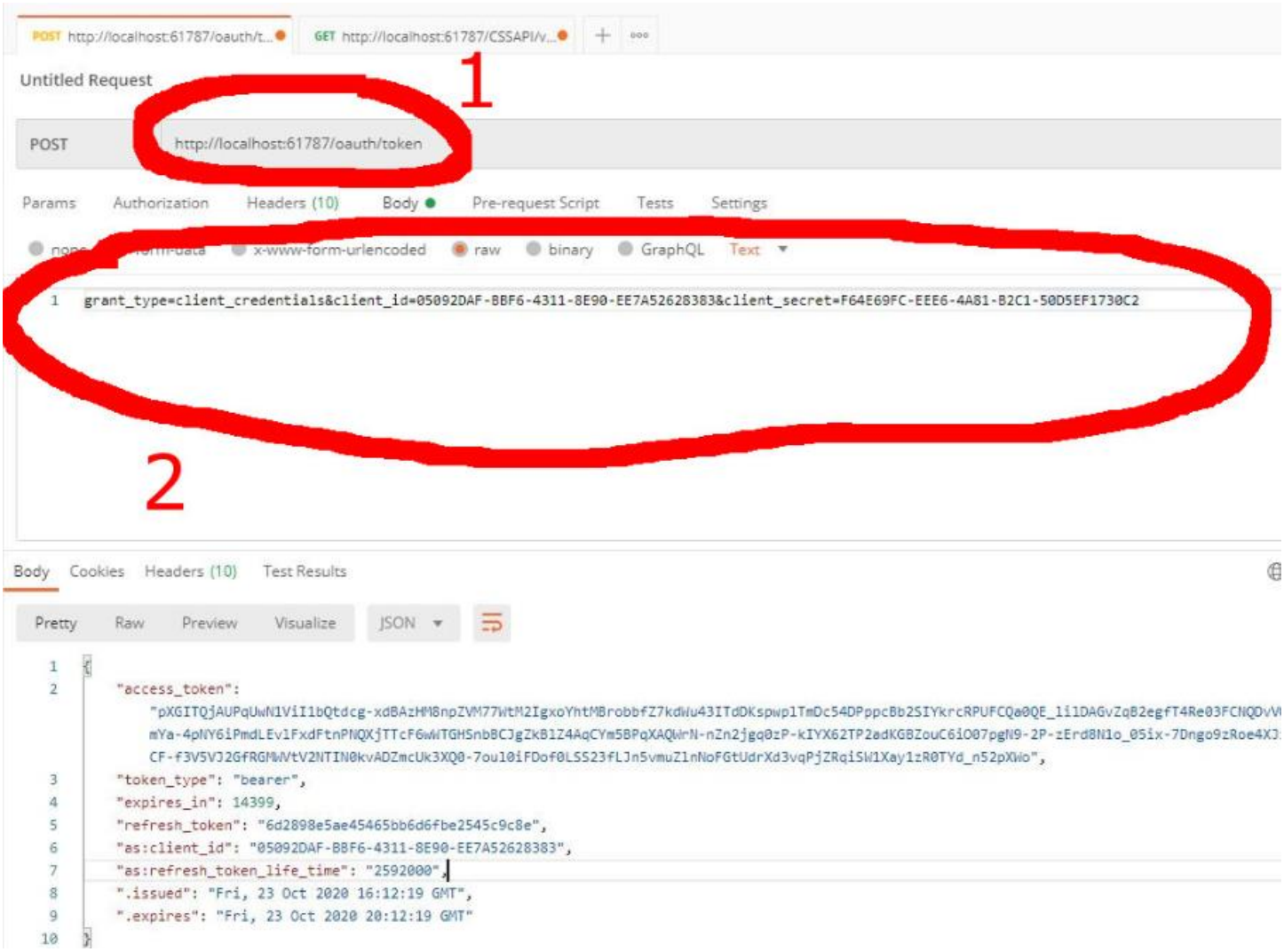
Record Adding/Creation Endpoint

- Insert one individual row in an entity provided from consumer within nested data structure and limited to column that not among IMAGE and BLOB as well as mentioned above and it responds with the newly generated row identifier.



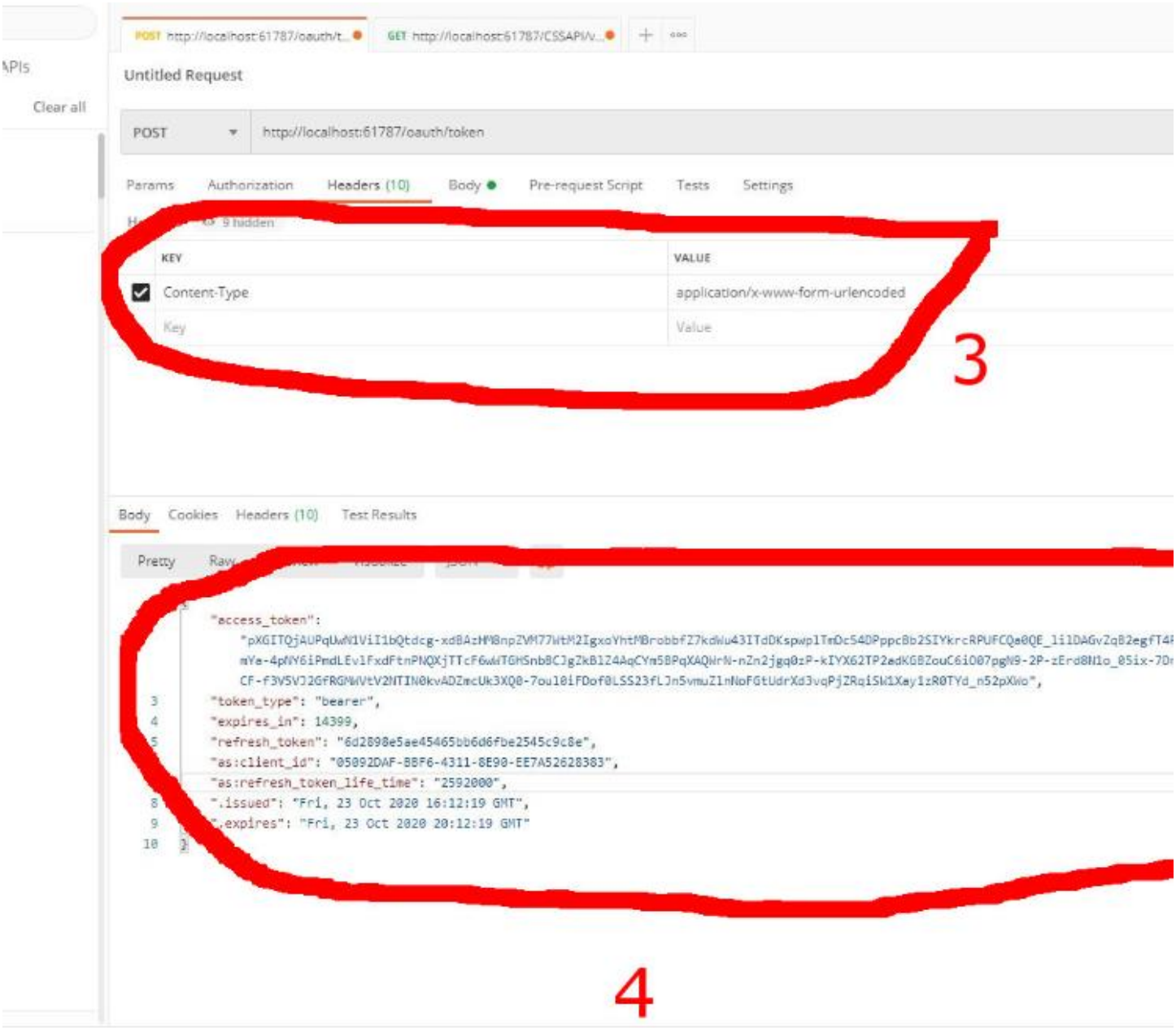
Sample Postman Connection

Section One – Initial Authentication



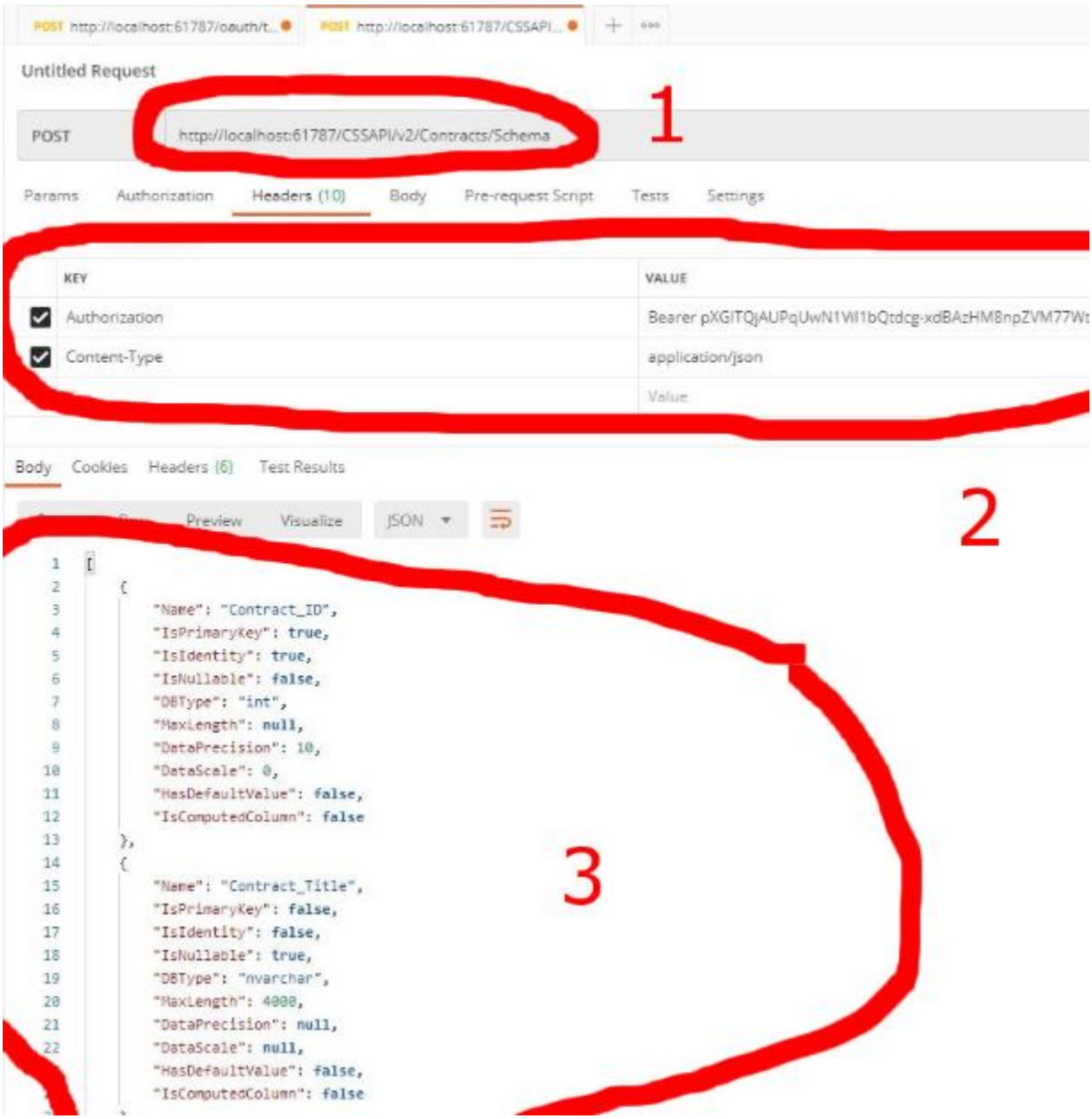
NOTE: in this example the test was against a local machine.

1) Make sure you change http://localhost/ to https://YOURCOMPANY.cobblestone.software/api2/ and keep it as a POST.
2) (under Body->Raw) change the values for the Client ID and Client Secret to one of those you generated in the UI of your system.



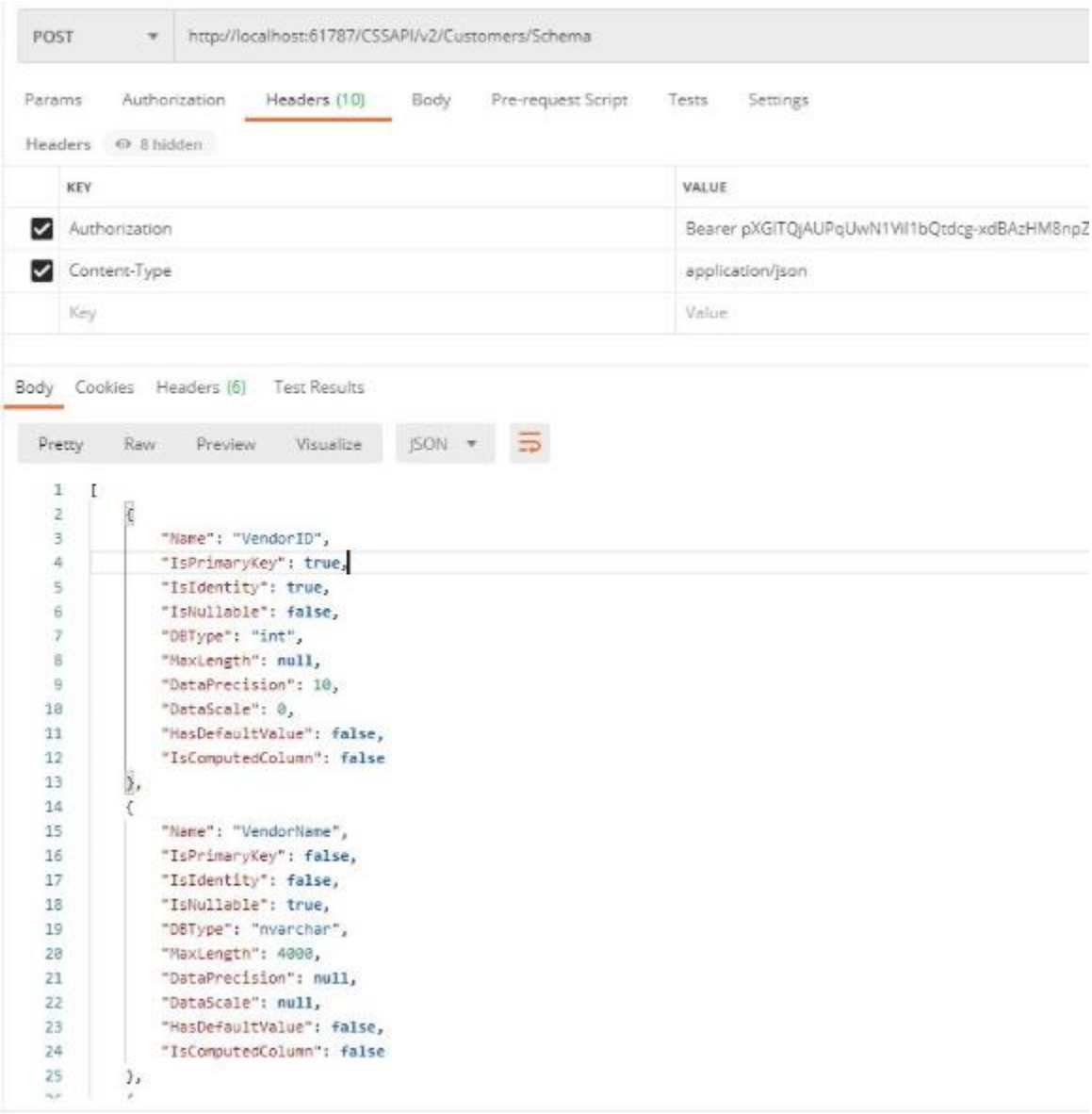
3) Add Content-Type as shown in picture inside the headers.
4) This shows the JSON that will be returned including the value for “access_token”. This token is going to be needed every time a hit is made against an entity in the rest of the following examples.

Section Two – Schema Endpoint (Contracts example)



1) Make sure you change http://localhost: 61787/ to https://YOURCOMPANY.cobblestone.software/api2/ and keep it as POST. As you can see in the picture "Contracts" was used instead of tblContracts, that is what the CobbleStone REST API is looking for. In addition, make sure that you keep /CSSAPI/v2/ after https://YOURCOMPANY.cobblestone.software/api2/".
2) Add content-type: application/json and Authorization: Bearer "Access Token" that was returned in the second screenshot in Section One.
3) The response will be the list of fields inside the Contracts table and their meta-data as in Field Explorer inside Contract Insight.

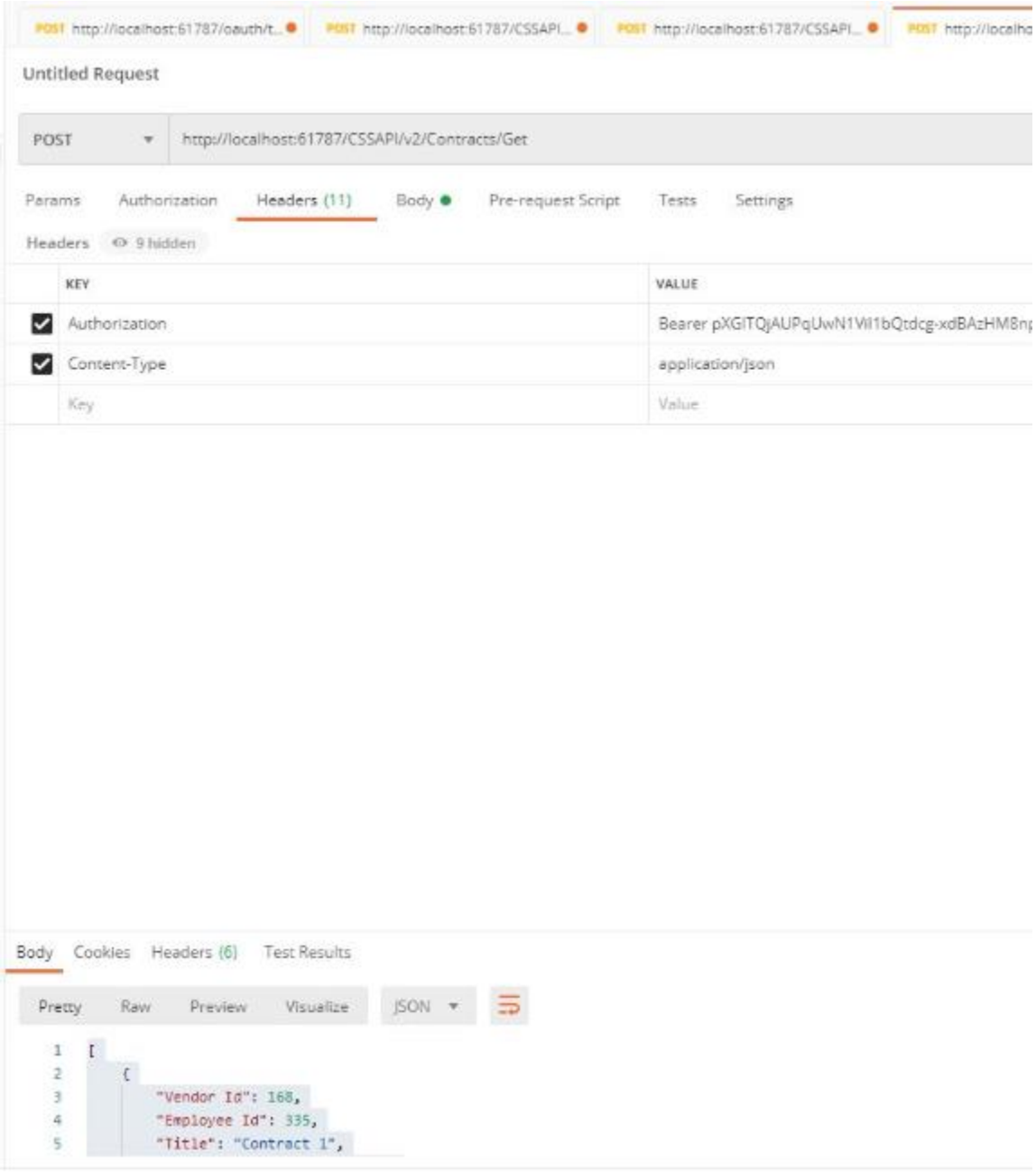
Section Three – Schema Endpoint (Customers example)

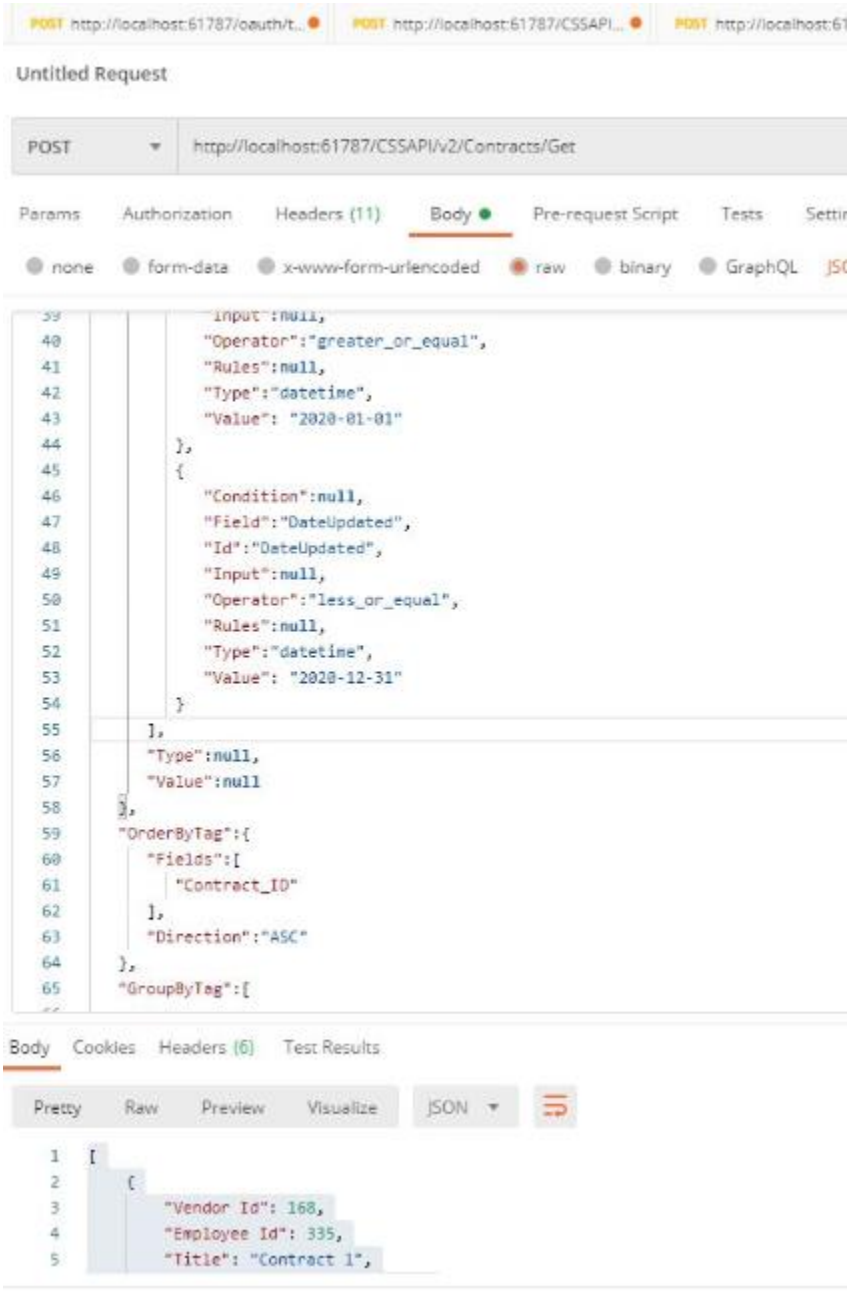


In this example, everything is identical to “Section Two” above, however this time the call is going against “Customers” instead of “Contracts”.

Section Four – Get/Select Contracts

1) URL will stay same as **Section Two and Three** but last Segment change from *Schema* to *Get*.
2) The same 2 headers must be added as in **Section Two and Three.**
3) In Body->Raw add the following JSON.
   NOTE: If you would like to have all the fields selected change fields property to [] “empty array”, Leave the GroupByTag empty unless there is a need to use any aggregate functions.
   Same goes to **Customers** table by changing *Contracts* to *Customers* in URL and changing the fields collection to the appropriate fields needed.





Sample JSON for Get/Select Contracts:

```
{
   "Fields":[
       {
           "Attribute": "Vendor_ID",
           "Alias": "'Vendor Id'" // Optional
       },
       {
           "Attribute": "Employee_ID",
           "Alias": "'Employee Id'" // Optional
       },
       {
           "Attribute": "Contract_Title",
```

```
            "Alias": "Title" // Optional
        },
        {
            "Attribute": "Contract_Start_Date"
        },
        {
            "Attribute": "Contract_End_Date"
        },
        {
            "Attribute": "u_ContractAmount"
        },
        {
            "Attribute": "u_Currency"
        }
    ],
    "Clause":{
       "Condition":"AND",
       "Field":null,
       "Id":null,
       "Input":null,
       "Operator":null,
       "Rules":[
          {
             "Condition":null,
             "Field":"DateUpdated",
             "Id":"DateUpdated",
             "Input":null,
             "Operator":"greater_or_equal",
             "Rules":null,
             "Type":"datetime",
             "Value": "2020-01-01"
          },
          {
             "Condition":null,
             "Field":"DateUpdated",
             "Id":"DateUpdated",
             "Input":null,
             "Operator":"less_or_equal",
             "Rules":null,
             "Type":"datetime",
             "Value": "2020-12-31"
          }
       ],
       "Type":null,
       "Value":null
    },
    "OrderByTag":{ // Optional
       "Fields":[
          "Contract_ID"
       ],
       "Direction":"ASC"
```

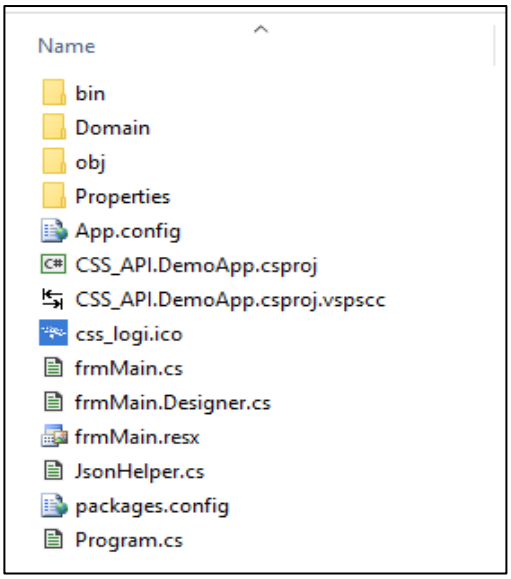
```
    },
    "GroupByTag":[

    ],
    "StartIndex":0,
    "Length":100 // By default this is 100, make sure it never exceeds 1000
}
```

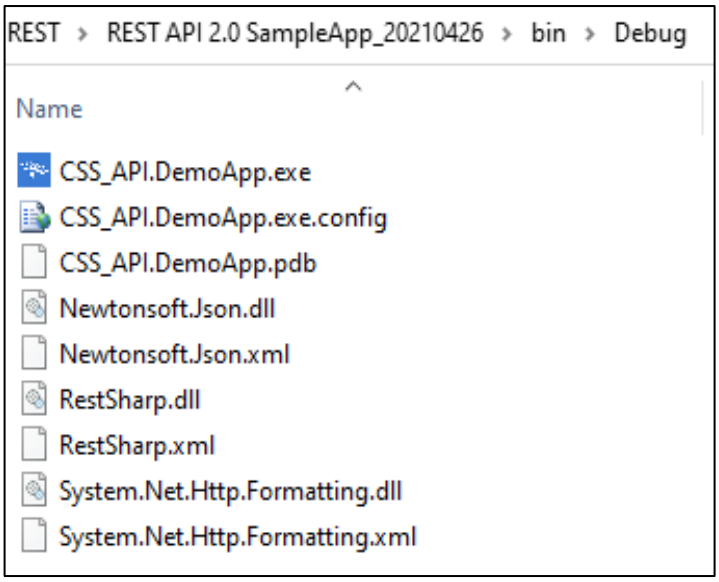
REST API Demo/Sample Application

CobbleStone's REST API Demo/Sample Application consists of two components, the first is the source for the application, the second is the compiled application for testing.

Application Files



Source Files



Compiled Files

Key Notes when Developing

NOTE: When building/modeling your own code off of this sample project please ensure that the following package is downloaded from NuGet: **Microsoft.AspNet.WebApi.Client.** Also note that this package can/will silently conflict with **Microsoft.Net.Http** when using the **ReadAsMultipartAsync** extension (the **Microsoft.Net.Http** package/reference will have to be removed first).

For detailed information on this please review the following link:

https://stackoverflow.com/questions/15088390/how-to-read-multipartcontent-from-httpresponsemessage

Revision History

Version	Date	Summary of Changes	Author
1.0	10-21-19	New document	Matthew Friebis, Fred Yuldashev
1.1	10-22-19	New format	Jennifer McGuigan
1.2	12-27-19	Updated document to include screenshots and examples of the various endpoints associated with the API	Matthew Friebis, Nizar Handal
1.3	04-09-21	Updated document to include section for "Sample Postman Connection" as well as section for "REST API Demo/Sample Application"	Matthew Friebis, Nizar Handal