



Nipun Kularathne

Follow

Jun 2, 2020 · 10 min read · Listen



Oracle Autonomous Database with REST API (ORDS) in action- Part II

This is the continuation of the [first section](#) of the same topic and you would want to go to that [article](#) and learn how we have done the server side configurations before going through this. We are going to have a simple introduction to OAuth authentication and how we can configure a client in Oracle Autonomous Database to authenticate the Rest Calls using token based OAuth authentication and finally going to test it. The scope we are going to cover in this section is

- Configure the previously created table-space and a user to use token-based OAuth authentication
- Registering a client and generating Client Id and Client Secret for OAuth

token based authentication

- Test the service using the Postman client.

Configure the OAD for OAuth token based authentication

In this section we are going to see how we can enable a client to connect to the rest endpoint of our database table and consume rest services after authentication.

We will be using OAuth2 standard authentication with client credentials grant type. What is OAuth2 and client credentials grant type?


There is a lot to learn if we talk about OAuth2 authentication. For now let's say it's a mechanism or a standard way of implementing authentication for clients or websites to get access to a different API services offered by different servers. In our case it's the Oracle Autonomous database that offers rest data services as rest endpoints for our created tables on OAD.

In this section we are going to look at a more simpler way of token based authentication which is OAuth 2.0 Client Credentials Grant type which we

[Sign up](#)

[Sign In](#)



 Search Medium

 Write



What is an access token?

A token or an access token is something like your boarding pass to a flight. When you are boarding to the flight you need to take it with you. Same way you need to send the access token to a server when you are sending whatever the request to get, update or delete data using rest api calls.

Also You need to provide your passport to get the boarding pass from the airport check in counter. Think of your passport as your client credentials. Exactly like you are issued a boarding pass when you provide a passport, when we use **Grant Type as Client Credentials** only when you provide your client credentials to the server you will be issued an access token from the server.

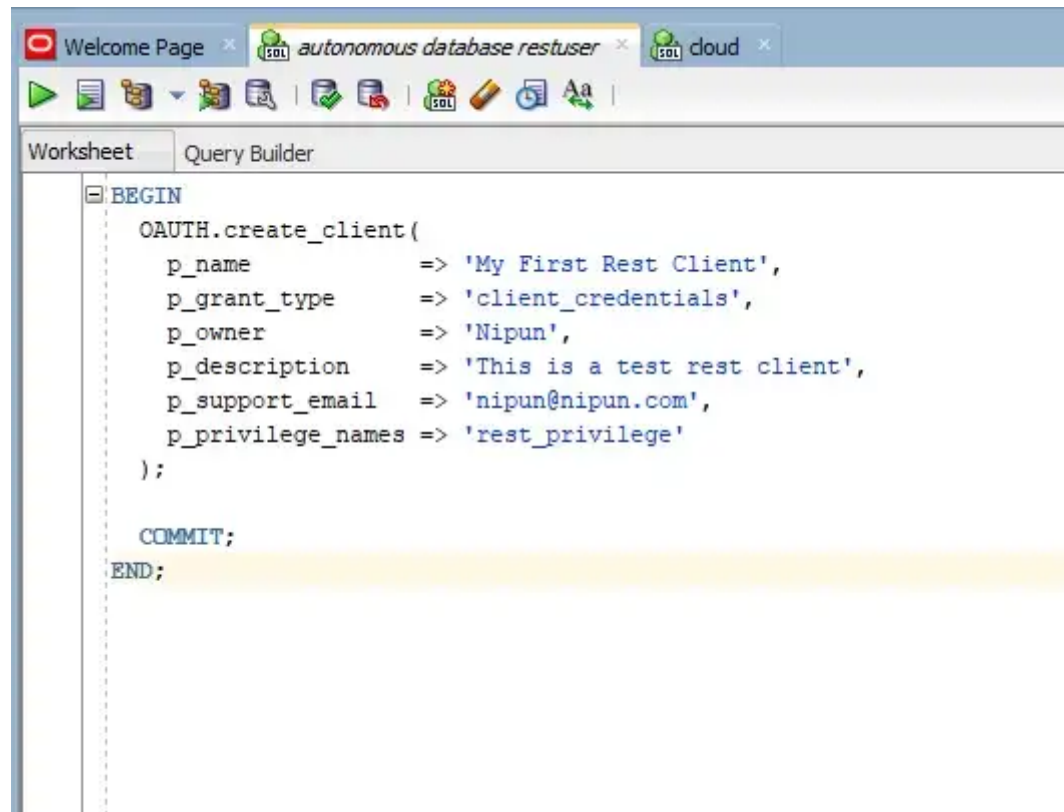
The boarding pass is only valid for some time; until your flight lands. Exactly in the same way access tokens also expires after some time. If you want to get an access token again you need to provide client credentials again to the server and get another access token for another limited amount of time.



Okay...In the above example i have mentioned about client credentials. What are those?

Think of them as a username and a password you use to log in to a website. Likewise client Credentials have two parts. A unique id that identifies the client, and a secret word that verifies the client. Those are called client id and client secret. How are we going to obtain these two for our Oracle Autonomous Database. For that we need to register a client in the OADB system. This is just like you sign up for a website with a user name and a password. Only difference is that Oracle asks the information such as client name, email etc from us and it generates a client id and client secret for us. Let's quickly see how we can register a client for ORDS access. Execute the below statement..

Registering a client and generating Client Id and Client Secret for OAuth token based authentication



The screenshot shows the Oracle SQL Developer interface. At the top, there are three tabs: 'Welcome Page', 'autonomous database restuser', and 'cloud'. Below the tabs is a toolbar with various icons. The main window is divided into two panes: 'Worksheet' and 'Query Builder'. The 'Query Builder' pane is active and displays a SQL query. The query is as follows:

```
BEGIN
  OAUTH.create_client(
    p_name          => 'My First Rest Client',
    p_grant_type     => 'client_credentials',
    p_owner         => 'Nipun',
    p_description    => 'This is a test rest client',
    p_support_email  => 'nipun@nipun.com',
    p_privilege_names => 'rest_privilege'
  );

  COMMIT;
END;
```

Note that you have to use `p_grant_type` as **client_credentials** in the above execution. There are several grant types used in OAuth authentication. This client credentials grant type simply tells Oracle system to grant an access token when we provide the client credentials; client id and client secret.

Most common and secure grant type is Authorization Code grant type where it uses an authorization code together with other credentials to obtain an access token. It has a more complex way of implementation and is more

secure therefore It is the recommended grant type to use with authentication of websites and native apps when connecting to remote servers using API calls. I have not tried this grant type with Oracle Autonomous database yet but hoping to try that soon and write my experience on that in a separate article. For now let's look at the one we previously mentioned; Grant Type = Client credentials. You can read more about grant types in OAuth website [here](#).

Now that we created the client and Oracle knows there's a client that would send a request to get or set data in the Autonomous Database. Still for that client to do any manipulation of data it needs to have a role assigned. The role for this is SQL Developer. One might think that we already have SQL Developer role assigned to the **restuser** we created. Yes, Still there is no user involvement when we send request from the client. Using this way you can have different clients created for a user and have different roles assigned to them.

To do this you need to execute the below statement.

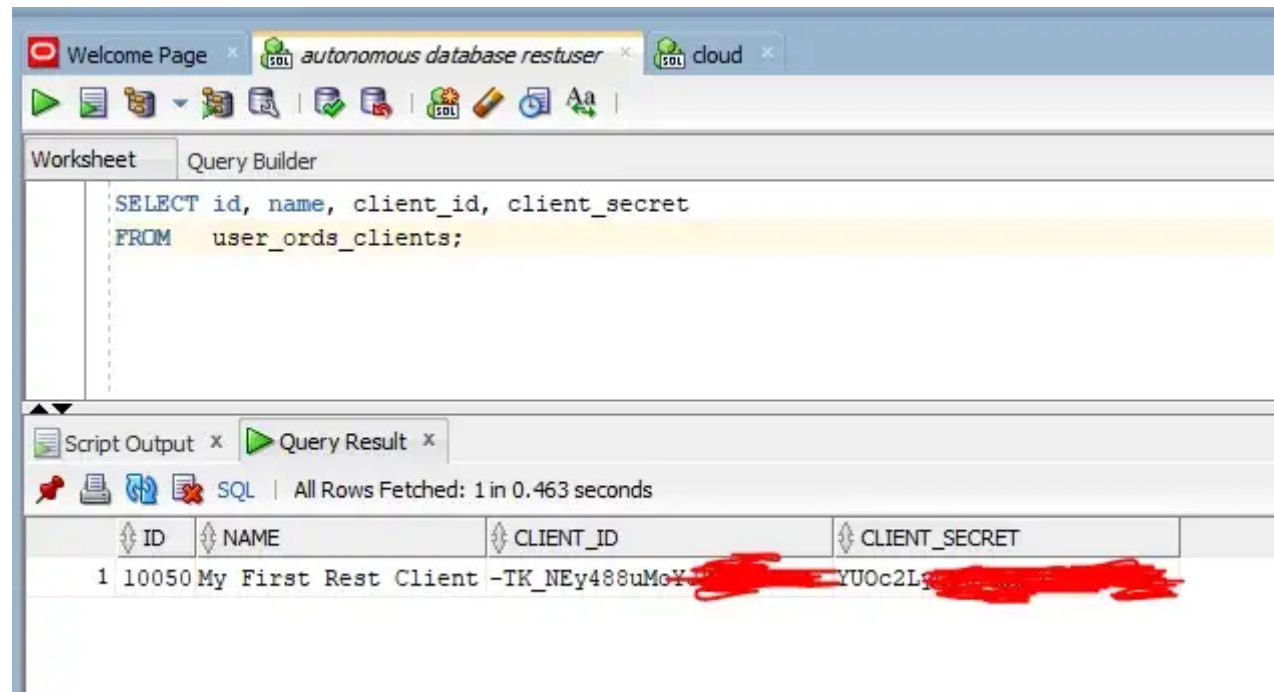


The screenshot shows the Oracle SQL Developer interface. The top window bar includes 'Welcome Page', 'autonomous database restuser', and 'cloud'. The toolbar contains various icons for file operations and execution. The 'Query Builder' tab is active, displaying a PL/SQL script. The script starts with 'BEGIN' and contains a call to 'OAUTH.grant_client_role' with parameters 'My First Rest Client' and 'SQL Developer'. It concludes with 'COMMIT;' and 'END;'.

```
BEGIN
  OAUTH.grant_client_role(
    p_client_name => 'My First Rest Client',
    p_role_name   => 'SQL Developer'
  );

  COMMIT;
END;
```


Now the client registration process is all completed. We need to obtain the client id and client secret to be sent with the request in order to obtain the access token. The clients that we register are written to a database table with their generated ids and secrets. If we query that table we can find the client id and client secret that corresponds to our newly created client. Let's do it.



You can copy the returned values and save it in a safe place to be used later when the authentication request is being made to the server..

Now that all the server side configurations are done we only need to know the Oracle ORDS endpoint (URL) for our autonomous database to do the authentication REST call. To find that we need to go to the Autonomous Database again and click on the service console button.

Autonomous Database » Autonomous Database Details



AVAILABLE

Oracle Database 19c is now available. [Learn how to upgrade](#) to Oracle Database 19c.

Test DB Always Free

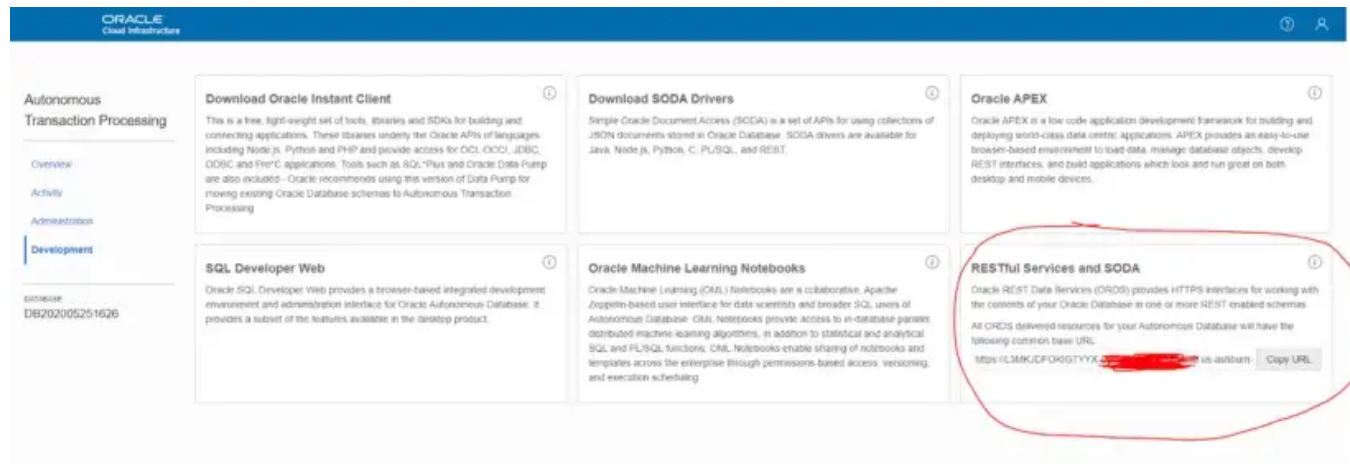
[DB Connection](#) [Performance Hub](#) [Service Console](#) [Scale Up/Down](#) [More Actions](#)

[Autonomous Database Information](#) [Tools](#) [Tags](#)

General Information

Database Name: DB202005251626
Workload Type: Transaction Processing
Compartment: nipunklk (root)
OCID: ...3d2msa [Show](#) [Copy](#)
Created: Mon, May 25, 2020, 11:04:23 UTC
OCPU Count: 1
Storage: 0.02
License Type: License included
Database Version: 18c [Upgrade to 19c](#)
Auto Scaling: Disabled [i](#)
Lifecycle State: Available
Instance Type: Free [Upgrade to Paid](#)

Then in the service console click on the development link on the right side pane to show the development related options. In the window you get you can see a section called **RESTful Services and SODA** inside in a rectangle. In that you have your common base url for the services REST endpoint. Click on the copy button and get the url copied to the clipboard.

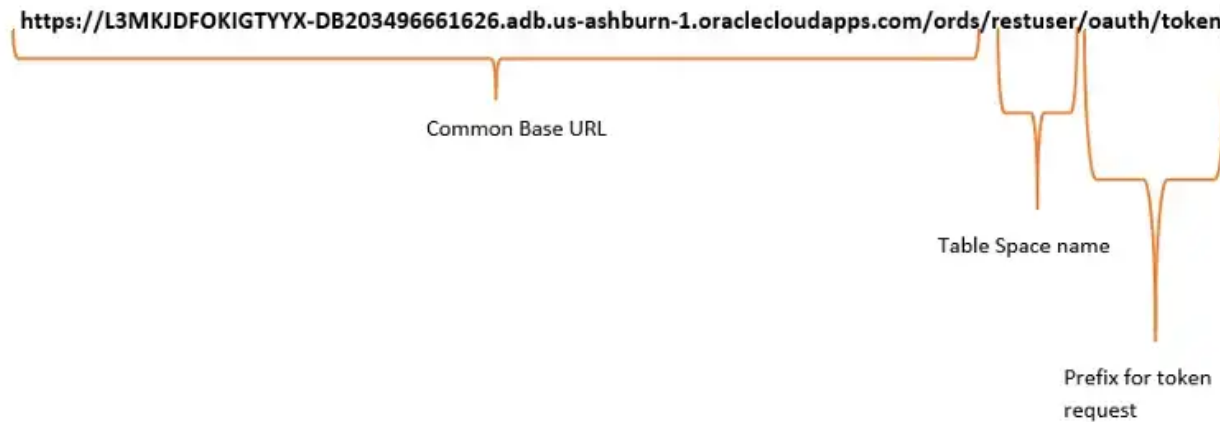


The common url looks like this.

<https://L3MKJDFORIGTYXX-DB203496661626.adb.us-ashburn-1.oraclecloudapps.com/ords/>

The url we need to use to get an authentication token is slightly different than the base url. It infact has the base url as the starting url. You need to append your **tablespace** name to the base url andat last you need to append **/oauth/token**

The the access token url would look like below.

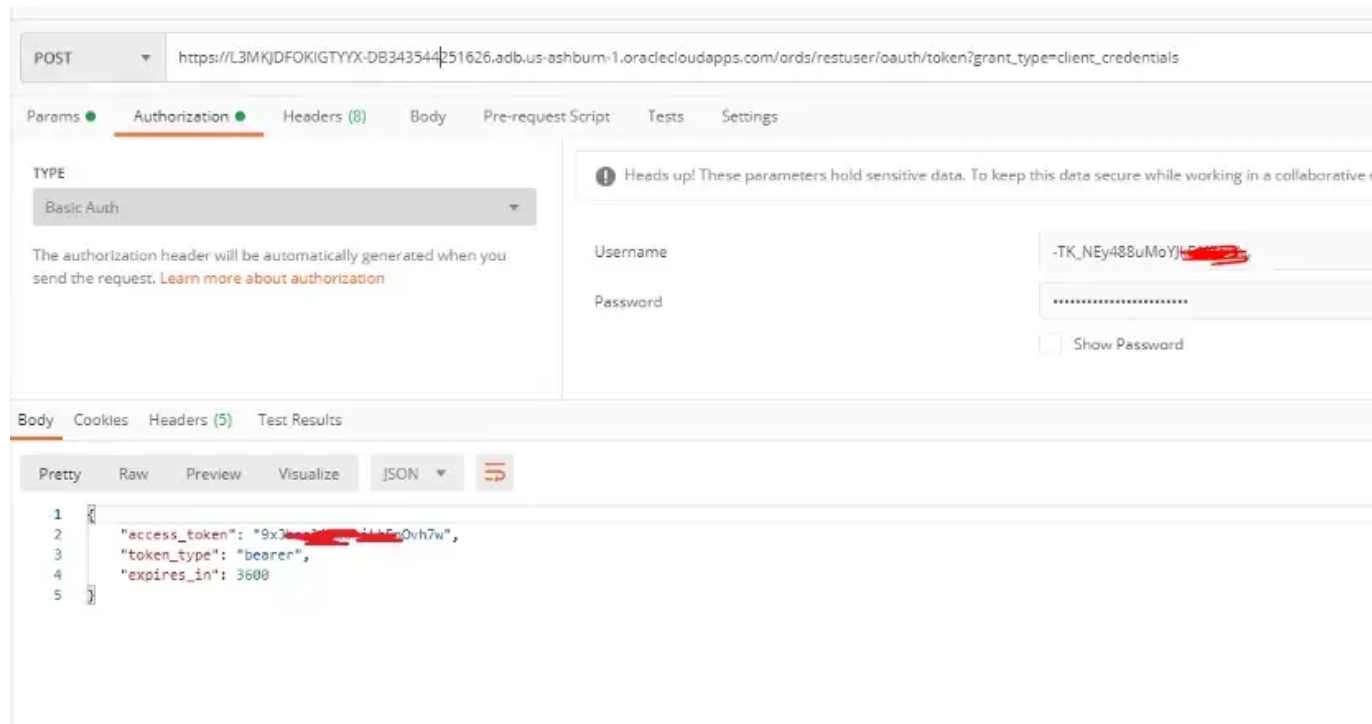


Additionally you need to send the grant type appended with a question mark at the end of the token url. The final url would look like below

https://L3MKJDFOKIGTYX-DB203496661626.adb.us-ashburn-1.oraclecloudapps.com/ords/restuser/oauth/token?grant_type=client_credentials

Test the service using the Postman client

Let's test if we can get an access token by sending a post request to the above url with our client id and client secret. For that we i am going to use a free http client **Postman**.



As you can see I have done sent request as a POST request and used Basic Auth as the authentication type. I have used my client id and client secret as the username and password of the request. As the result, in the response body, I am getting back an access token from the Autonomous Database Server. Now using this access token I can request a resource (Request data in the created table) to get the data as a Json response.

Note that the token has an expiry value which denotes that you will not be able to use this token to make any request to the server after that specified

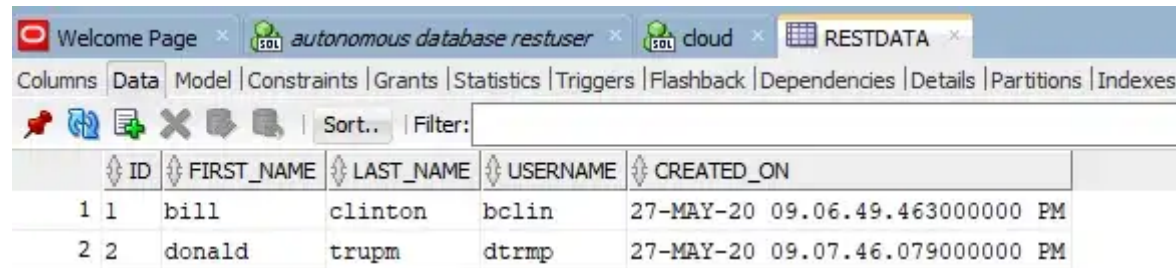
time, in this case 3600 milliseconds.

Another thing to consider is that the Postman client automatically encodes these client id and client secret in base64 encoding and sends them to the server. Server only accepts the base64 encoded values. When I tried to send the request using a simple mobile app which i created using flutter, the request failed with the server error 401 and the issue was in the http package i used to do the request. These values were not automatically encoded to base64 in that package. When I explicitly encoded the values for base64 the request was accepted and successful. I had to use **Fiddler** to check both the request headers to identify this problem. It's good to remember fiddler as a useful tool for examining the requests and troubleshooting the problem.

We will talk about creating a simple test mobile app using flutter to have Oracle Autonomous Database as the back end in an upcoming post which we will be using Fiddler to intercept the rest call. For now you can google on how to use Fiddler.

Now let's look at how we are going to get the records of our table using this access token.

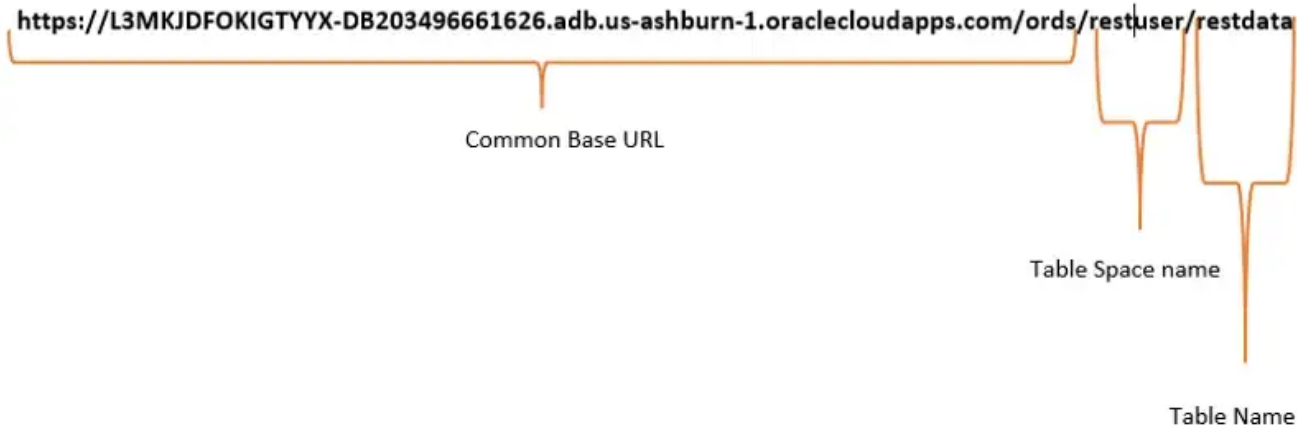
To do that first go to the Oracle Sql Developer client again and expand the tables section and add several rows to the created table. After adding two rows my table looked like this.



The screenshot shows the Oracle SQL Developer interface. The top bar includes tabs for 'Welcome Page', 'autonomous database restuser', 'cloud', and 'RESTDATA'. Below the tabs is a menu bar with 'Columns', 'Data', 'Model', 'Constraints', 'Grants', 'Statistics', 'Triggers', 'Flashback', 'Dependencies', 'Details', 'Partitions', and 'Indexes'. The 'Data' tab is selected, showing a table with two columns: 'ID' and 'FIRST_NAME'. The table has two rows of data: (1, 'bill') and (2, 'donald').

ID	FIRST_NAME
1	bill
2	donald

Now we need to know the endpoint to this table to do the request. How are we going to find the endpoint? Remember we got the common base url. We need to append the table space name and the table name at the end of it with a forward slash to the end. (forward slash it not needed when we query but it's always better to have it as we need it when writing data to the table using REST call)



url components

We need to use the GET method to fetch data from a rest endpoint. Below is my Postman request passing the received auth token to get the data of the `restdata` table. Note that I have used bearer token as the authentication type and passed the obtained auth token in the previous request.

The screenshot displays a REST client interface with a GET request to the URL `https://L3MKJDFOKIGITYX-DB343544251626.adb.us-ashburn-1.oraclecloudapps.com/ords/restuser/restdata/`. The 'Authorization' tab is active, showing a Bearer Token. The 'Body' tab is selected, displaying the JSON response in a 'Pretty' format. The response contains two items, each with fields for id, first_name, last_name, username, created_on, and links. The first item is for 'bill clinton' and the second is for 'donald trump'. The 'hasMore' field is set to false.

```
1 {
2   "items": [
3     {
4       "id": "1",
5       "first_name": "bill",
6       "last_name": "clinton",
7       "username": "belin",
8       "created_on": "2020-05-27T21:06:49.463Z",
9       "links": [
10        {
11          "rel": "self",
12          "href": "https://l3mkjdfokigityx-db202005251626.adb.us-ashburn-1.oraclecloudapps.com/ords/restuser/restdata/1"
13        }
14      ]
15    },
16    {
17      "id": "2",
18      "first_name": "donald",
19      "last_name": "trump",
20      "username": "dtrmp",
21      "created_on": "2020-05-27T21:07:46.079Z",
22      "links": [
23        {
24          "rel": "self",
25          "href": "https://l3mkjdfokigityx-db202005251626.adb.us-ashburn-1.oraclecloudapps.com/ords/restuser/restdata/2"
26        }
27      ]
28    }
29  ],
30  "hasMore": false,
31}
```

Response in JSON body

As the response I've got 2 rows which are in the database as a Json response. You can parse this response to be used in your client applications in any way accordingly.

Also like this way you can perform create, update and delete operations on the database table we made restenable. You need to use POST method in requests with a correct Json request body to create records and need to use the PUT method to update the records with the correct keys in the request. That's about it...

To summarize, in these 2 articles ([link to the first article](#))

we first provisioned an Oracle Autonomous Database in the Oracle free tier

Then we connected to that database using Oracle SQL developer app as the admin user

Then we created another user and gave privileges to that user and connected using the same user to the database

And using that new user then we created a table and made that table and table space **restenable**

We created a rest privilege to that table so that it can do rest call to that table endpoint,

Then we registered a **Rest Client** and assigned **Rest Privilege** to that client in the same way to be able to connect to the Rest endpoints.

Then we looked at obtaining the **client credentials** of that client to be used in the authentication rest call to get an access token

We looked at how to perform the authentication rest client call and obtain the token using Postman http client

Finally we used the obtained access token and did a GET request to the Oracle Autonomous Database table's Rest endpoint to get the data records in the table.

Now you can use any client which can consume Restful services (Mobile apps, Websites) to connect and get/ modify/ update data in any table that is configured to expose its data using Rest Endpoints securely.

Hope you have gained a good understanding about how we can simply use an Oracle Autonomous Database as a remote back end for our applications and to access data on the DB easily using Oracle Rest Data Services.

[Oracle Autonomous Db](#)

[Rest Api](#)

[Oauth](#)

[Ords](#)

[Rest Client](#)

[About](#)

[Help](#)

[Terms](#)

[Privacy](#)