[Using JSON with Data Services](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services)

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

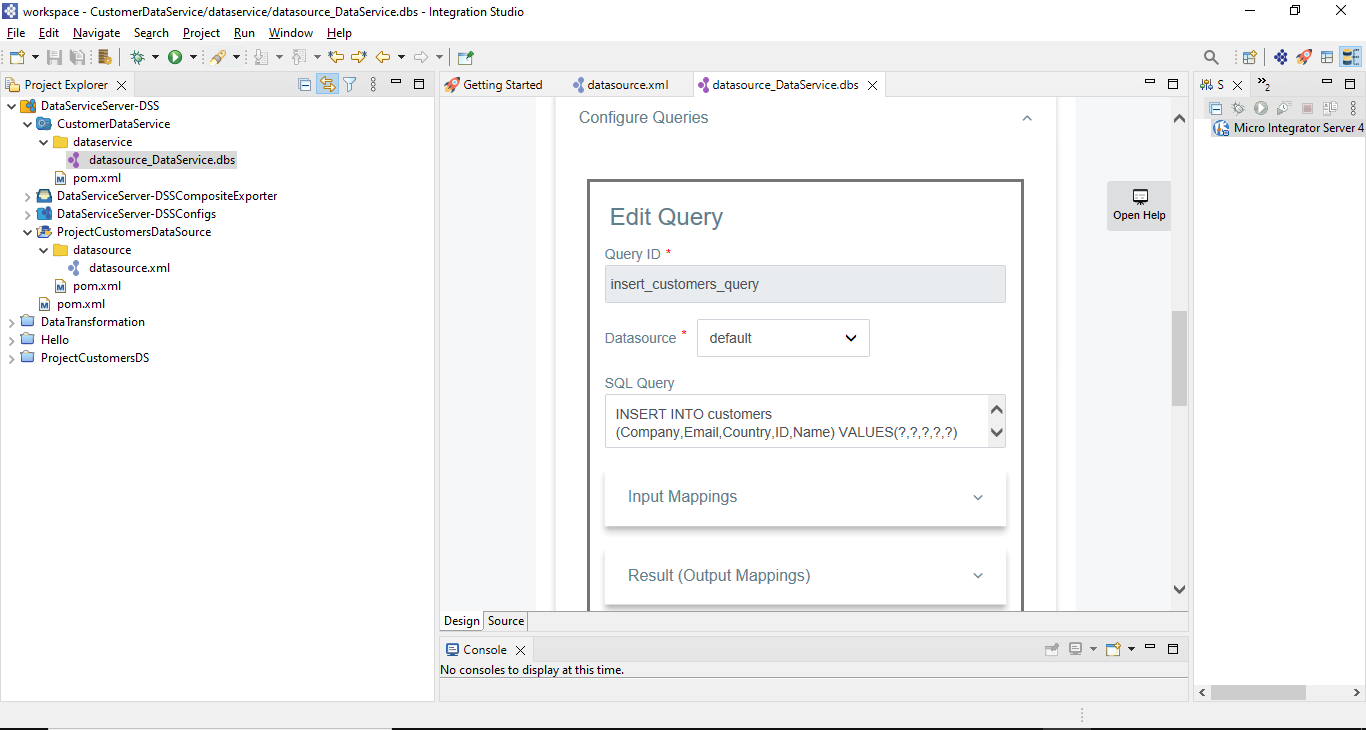
Description automatically generated

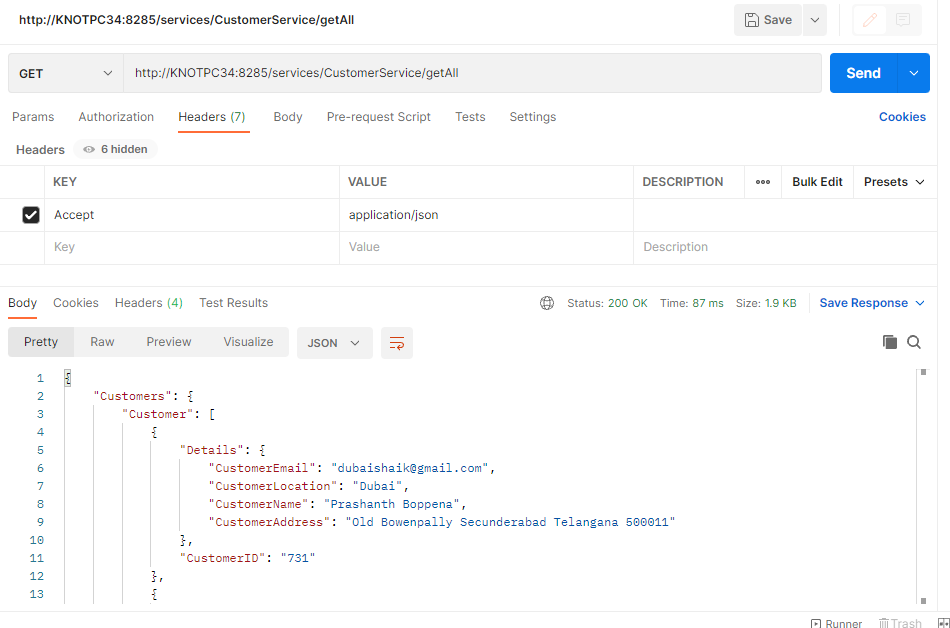
Graphical user interface, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated





You can send and receive JSON messages by default via WSO2 Enterprise Integrator's (WSO2 EI) ESB profile. See the topics given below to understand how data can be exposed in the JSON format, and how data can be changed by sending JSON payloads. In this tutorial, you will use a data service that exposes RDBMS data.

**Before you begin!**

If you have not tried the [Exposing a Datasource as a Data Service](https://docs.wso2.com/display/EI660/Exposing+a+Datasource+as+a+Data+Service) tutorial previously follow the steps given below:

Download the **RDBMSDataService** from [here](https://docs.wso2.com/download/attachments/141239586/RDBMSDataService.dbs?version=1&modificationDate=1576209927000&api=v2).

Download the product installer from [here](http://wso2.com/integration/), and run the installer.  
Let's call the installation location of your product the **<EI\_HOME>** directory. This is located in a place specific to your OS as shown below:

| **OS** | **Home directory** |
| --- | --- |
| Mac OS | /Library/WSO2/EnterpriseIntegrator/6.6.0 |
| Windows | C:\Program Files\WSO2\EnterpriseIntegrator\6.6.0\ |
| Ubuntu | /usr/lib/wso2/EnterpriseIntegrator/6.6.0 |
| CentOS | /usr/lib64/EnterpriseIntegrator/6.6.0 |

Download the JDBC driver for MySQL from [here](http://dev.mysql.com/downloads/connector/j/). Unzip it, get the <MySQL\_HOME>/mysql-connector-java-8.0.16.jar JAR, and place it in the <EI\_HOME>/lib directory.

If the driver class does not exist in the relevant folders when you create the datasource, you will get an exception, such as 'Cannot load JDBC driver class com.mysql.jdbc.Driver'.

Start the WSO2 ESB profile.

[On MacOS/Linux/CentOS](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#ba002576187447b8ae425b539a0f6095)

[On Windows](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#f15bcd27a1b64c7686e862428e7ca62d)

Open a terminal and execute the following command:

|  |
| --- |
| sudo wso2ei-6.6.0-integrator |

Go to product's management console: https://localhost:9443/carbon

Enter admin as the username and password.

Click **Add > Data Service > Upload**.

Browse and add the RDBMSDataService.dbs file you downloaded.

[Map the output type to JSON](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#UsingJSONwithDataServices-MaptheoutputtypetoJSON)

[GET data in JSON](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#UsingJSONwithDataServices-GETdatainJSON)

[POST/UPDATE data using JSON](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#UsingJSONwithDataServices-POST/UPDATEdatausingJSON)

[Post data](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#UsingJSONwithDataServices-Postdata)

[Post data in batches](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#UsingJSONwithDataServices-Postdatainbatches)

[Update data](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#UsingJSONwithDataServices-Updatedata)

[Post data using Request Box](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#UsingJSONwithDataServices-PostdatausingRequestBox)

A data service can expose data in one of the following formats: XML, RDF, or JSON. You can select the required format by specifying the output type for the data service query. To expose data in JSON, you need to select JSON as the output type, and map the output to a JSON template.

**Map the output type to JSON**

The data service we uploaded or created previously, maps the output type to XML. Follow the steps given below to change it to JSON.

Open the ESB profile's Management Console using <https://localhost:9443/carbon>, and log in using admin as the username and the password.

Click **List** under **Main > Services**. The **RDBMS** data service should be listed.

Click the data service to open the **Service Dashboard**.

Click **Edit Data Service (Wizard)** to open the data service using the **Create Data Service** wizard.

Click **Next** until you get to the **Queries** screen.

Edit the **GetEmployeeDetails** query.

Change the Output Type to **JSON**.

You can now start defining the JSON template for the output. Listed below are a few sample templates that you can use for this query.

[Simple JSON template](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#9c8c73943d1248a9befc8836efe80dfd)

[Define data types](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#f4e95495db3d4ac5841f36438598a543)

[Nested queries](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#258c9ae5d40645718946d4ac0ed398d9)

Shown below is a basic mapping.

**Sample JSON Mapping**

|  |
| --- |
| { "Employees":        {"Employee":[          {"EmployeeNumber":"$EmployeeNumber",           "Details": {            "FirstName":"$FirstName",            "LastName":"$LastName",            "Email":"$Email",            "Salary":"$Salary"           }          }        ]      }  } |

**Note the following:**

As shown in the sample given above, the column name values that are expected in the query result should be referred to by the column name with the "$" prefix. **E.g.** "$EmployeeNumber".

Also, the structure of the JSON template should follow some guidelines in order to be compatible with the result. These guidelines are:

The top most item should be a JSON object. It cannot be a JSON array.

For handling multiple records from the result set, the immediate child of the top most object can be a JSON array, and the array should contain only a single object.

If only a single result is returned, the immediate child of the top most object can be a single JSON object.

After the immediate child of the top most object, there cannot be other JSON arrays in the mapping.

All JSON responses are returned as an array.

Save the query.

**GET data in JSON**

The RDBMSDataService that you are using already contains the following resource:

|  |  |
| --- | --- |
| **Resource Path** | Employee/{EmployeeNumber} |
| **Resource Method** | GET |
| **Query ID** | GetEmployeeDetails |

You can now RESTfully invoke the above resource. To send a JSON message to a RESTful resource, you can simply add the “Accept:Application/json” to the request header when you send the request. The service can be invoked in REST-style via [curl](http://curl.haxx.se/).   
Shown below is the curl command to invoke the GET resource:

|  |
| --- |
| curl -X GET -H "Accept: application/json" http://localhost:8280/services/RDBMSDataService/Employee/{EmployeeNumber} |

Example:

|  |
| --- |
| curl -X GET -H "Accept: application/json" http://localhost:8280/services/RDBMSDataService/Employee/1 |

As a result, you receive the response in JSON format as shown below.

|  |
| --- |
| {"Employees":{"Employee":[{"EmployeeNumber":"1","FirstName":"John","LastName":"Doe",  "Email":"JohnDoe@gmail.com","Salary":"10000"},{"EmployeeNumber":"1","FirstName":"John",  "LastName":"Doe","Email":"JohnDoe@gmail.com","Salary":"20000"}]} |

**POST/UPDATE data using JSON**

When a client sends a request to change data (POST/PUT/DELETE) in the datasource, the HTTP header Accept should be set to application/json.  Also, if the data is sent as a JSON payload, the HTTP header Content-Type should be set to application/json.

The RDBMSDataService that you are using already contains the following resources for adding and updating data.

Resource for adding employee information:

|  |  |
| --- | --- |
| **Resource Path** | Employee |
| **Resource Method** | POST |
| **Query ID** | AddEmployeeDetails |

Resource for updating employee information:

|  |  |
| --- | --- |
| **Resource Path** | Employee |
| **Resource Method** | PUT |
| **Query ID** | UpdateEmployeeDetails |

You can RESTfully invoke the above resource by sending HTTP requests as explained below.

**Post data**

To post new employee information, you need to invoke the resource with the POST method.

First, create a file named employee-payload.json, and define the JSON payload for posting new data as shown below.

|  |
| --- |
| {    "user\_defined\_value": {      "EmployeeNumber" : "14001",      "LastName": "Smith",      "FirstName": "Will",      "Email": "will@google.com",      "Salary": "15500.0"    }  } |

On the terminal, navigate to the location where the **employee-payload.json**file is stored, and execute the following HTTP request:

|  |
| --- |
| curl -X POST -H 'Accept: application/json'  -H 'Content-Type: application/json'   --data "@employee-payload.json" -k -v  http://localhost:8280/services/RDBMSDataService/Employee |

**Post data in batches**

You are able to post JSON data in batches using the RDBMSDataService that you created or uploaded.

To verify that batch requesting is enabled:

Log in to the EI Management Console.

Click **List** under **Main > Services**and select the**RDBMSDataService**.

Click the data service to open the **Service Dashboard**.

Click **Edit Data Service (Wizard)** to open the data service using the **Create Data Service** wizard.

See that the **Batch Requesting** check box is selected.

First, create a file named **employee-batch-payload.json**, and define the JSON payload for posting multiple employee records (batch) as shown below.

|  |
| --- |
| {      "user\_defined\_value": {          "user\_defined\_value": [              {                  "EmployeeNumber": "5012",                  "FirstName": "Will",                  "LastName": "Smith",                  "Email": "will@smith.com",                  "Salary": "13500.0"              },              {                  "EmployeeNumber": "5013",                  "FirstName": "Parker",                  "LastName": "Peter",                  "Email": "peter@parker.com",                  "Salary": "15500.0"              }          ]      }  } |

On the terminal, navigate to the location where the**employee-batch-payload.json** file is stored, and execute the following HTTP request:

|  |
| --- |
| curl -X POST -H 'Accept: application/json'  -H 'Content-Type: application/json'   --data "@employee-batch-payload.json" -k -v  http://localhost:8280/services/RDBMSDataService/Employee\_batch\_req |

**Update data**

To update the existing employee records, you need to invoke the resource with the PUT method.

First, create a file named **employee-upload-update.json**, and define the JSON payload for updating an existing employee record as shown below.   
For example, change the salary amount. Make sure that the employee number already exists in the database.

|  |
| --- |
| {    "user\_defined\_value": {      "EmployeeNumber" : "1",      "FirstName": "Will",      "LastName": "Smith",      "Email": "will@smith.com",      "Salary": "78500.0"    }  } |

On the terminal, navigate to the location where the **employee-upload-update.json**file is stored, and execute the following HTTP request:

|  |
| --- |
| curl -X PUT -H 'Accept: application/json'  -H 'Content-Type: application/json'   --data "@employee-upload-update.json" -k -v  http://localhost:8280/services/RDBMSDataService/Employee |

**Post data using Request Box**

When the Request Box feature is enabled, you can invoke multiple operations (consecutively) using one single operation. The process of posting a JSON payload through a request box transaction is explained below.

To verify that batch requesting is enabled:

Log in to the EI Management Console.

Click **List** under **Main > Services**and select the**RDBMSDataService**.

Click the data service to open the **Service Dashboard**.

Click **Edit Data Service (Wizard)** to open the data service using the **Create Data Service** wizard.

See that the **Enable Boxcarring** check box is selected.

First, create a file named **employee\_request\_box\_payload.json**, and define the JSON payload for posting multiple employee records (batch) as shown below.

The following payload works for this use case. When you create payloads for different use cases, be mindful of the tips [given here](https://docs.wso2.com/display/EI660/Using+JSON+with+Data+Services#UsingJSONwithDataServices-JSON_payloads).

|  |
| --- |
| {   "request\_box"  : {        "\_postemployee" : {                  "EmployeeNumber"  : "14005",                  "LastName" :  "Smith" ,                  "FirstName" :  "Will" ,                  "Email" :  "will@google.com" ,                  "Salary" : "15500.0"                          },        "\_getemployee\_employeenumber":{                  "EmployeeNumber"  : "14005"             }      }  } |

On the terminal, navigate to the location where the**employee\_request\_box\_payload.json** file is stored, and execute the following HTTP request:

|  |
| --- |
| curl -X POST -H 'Accept: application/json'  -H 'Content-Type: application/json' --data "@employee\_request\_box\_payload.json" http://localhost:8280/services/RDBMSDataService/request\_box |

**Creating JSON payloads for Request Box transactions**

Note the following when you define a JSON payload for a request box transaction: The object name specified in the payload must be in the following format: "\_<HTTP\_METHOD><RESOURCE\_PATH>" where RESOURCE\_PATH represents the path value specified in the data service resource. For example, if the RESOURCE\_PATH is "employee", the payload object name should be as follows:

For HTTP POST requests: \_postemployee

For HTTP PUT requests: \_putemployee

The child name/values of the child fields in the payload should be the names and values of the input parameters in the target query.

**Handling a resource path with the "/" symbol**

If the RESOURCE\_PATH specified in the data service contains the "/" symbol, be sure to replace the "/" symbol with the underscore symbol ("\_") in the payload object name.

**Important!** In this scenario, the RESOURCE\_PATH value should only contain simple letters. For example, the value can be "/employee/add" but not "/Employee/Add".

For example, if the  RESOURCE\_PATH  is /employee/add, the payload object name should be as follows:

For HTTP POST requests: \_post\_employee\_add

For HTTP PUT requests: \_put\_employee\_add