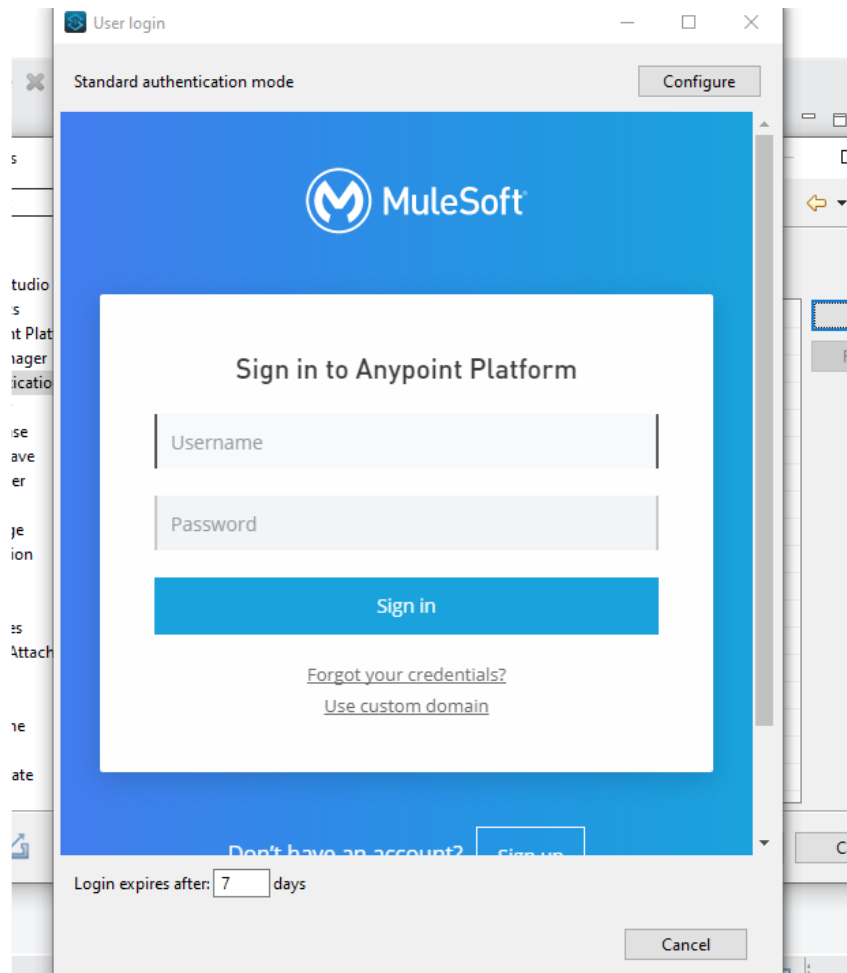


Sync Anypoint Platform account with Anypoint Studio

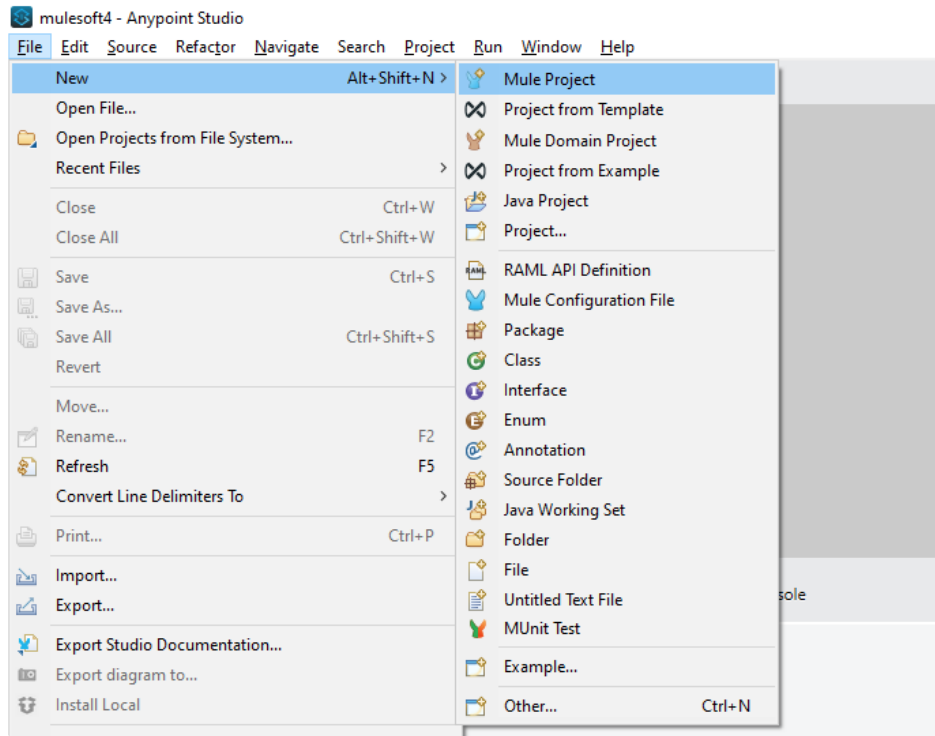
Under Anypoint Studio section, click on Authentication item





Create new Mulesoft API project in Anypoint Studio

Go to File → New → Mule Project



Input the project name

New Mule Project

Project Settings
Create a Mule project in the workspace or in an external location.

Project Name:

Runtime
Mule Server 4.3.0 EE

[Install Runtimes](#)

API Implementation
Add an API implementation to your project to automatically set up an APIkit router and create placeholder flows for each resource method

Import a published API Import RAML from local file Import from Design Center

ⓘ Import both OAS and RAML specifications here [Learn more](#)

+ ✕ ↗

Name	Version
Please select or add a dependency to see more information.	

☒ Scaffold flows from these API specifications

Project location
☒ Use default location

Under Import a published API tab, click green plus button, choose from Exchange

[Install Runtimes](#)

API Implementation
Add an API implementation to your project to automatically set up an APIkit router and create placeholder flows for each resource method

Import a published API Import RAML from local file Import from Design Center

ⓘ Import both OAS and RAML specifications here [Learn more](#)

+ ✕ ✓

	Version
from Exchange	
from Maven	

Please select or add a dependency to see more information.

☒ Scaffold flows from these API specifications

Project location

☒ Use default location

Location: D:\workspace\microsoft\SecondApp

In search box, type the project name that you published to the Exchange in previous article

✕ Add Dependencies to Project

Add Dependencies to Project

Search for dependencies in Exchange to add them to the project

Username: sonle4 Add Account

Search: SecondApp ✕

Name	Publisher
SecondApp	Isobar

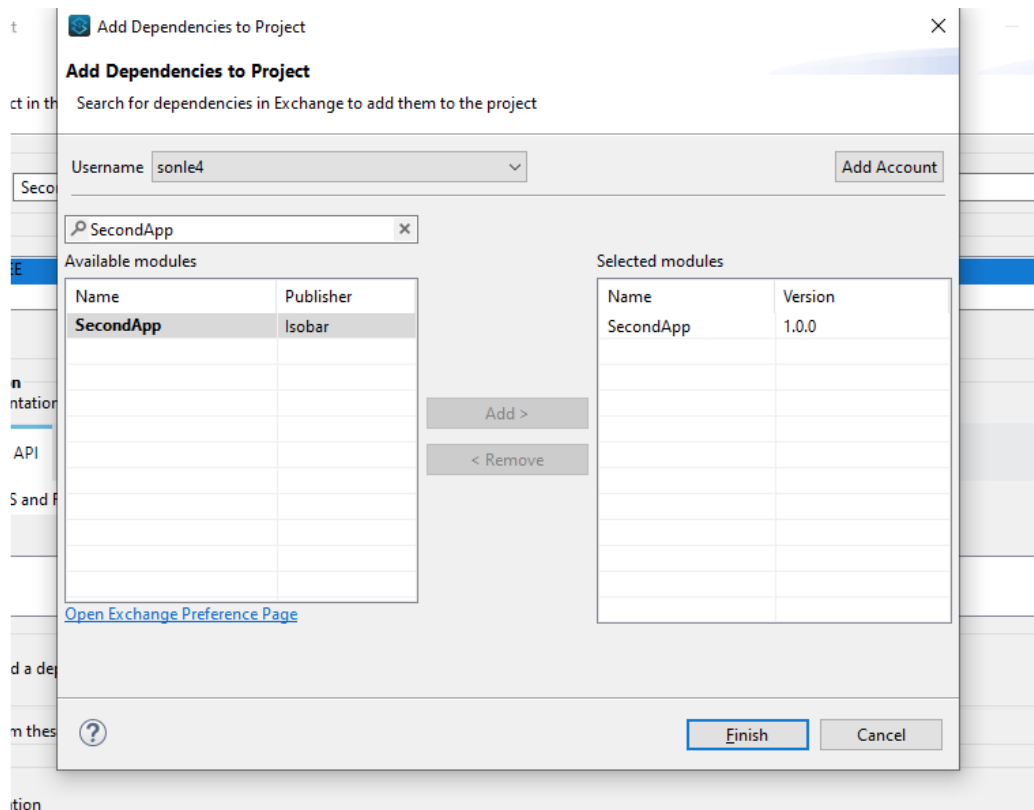
Add > < Remove

Name	Version

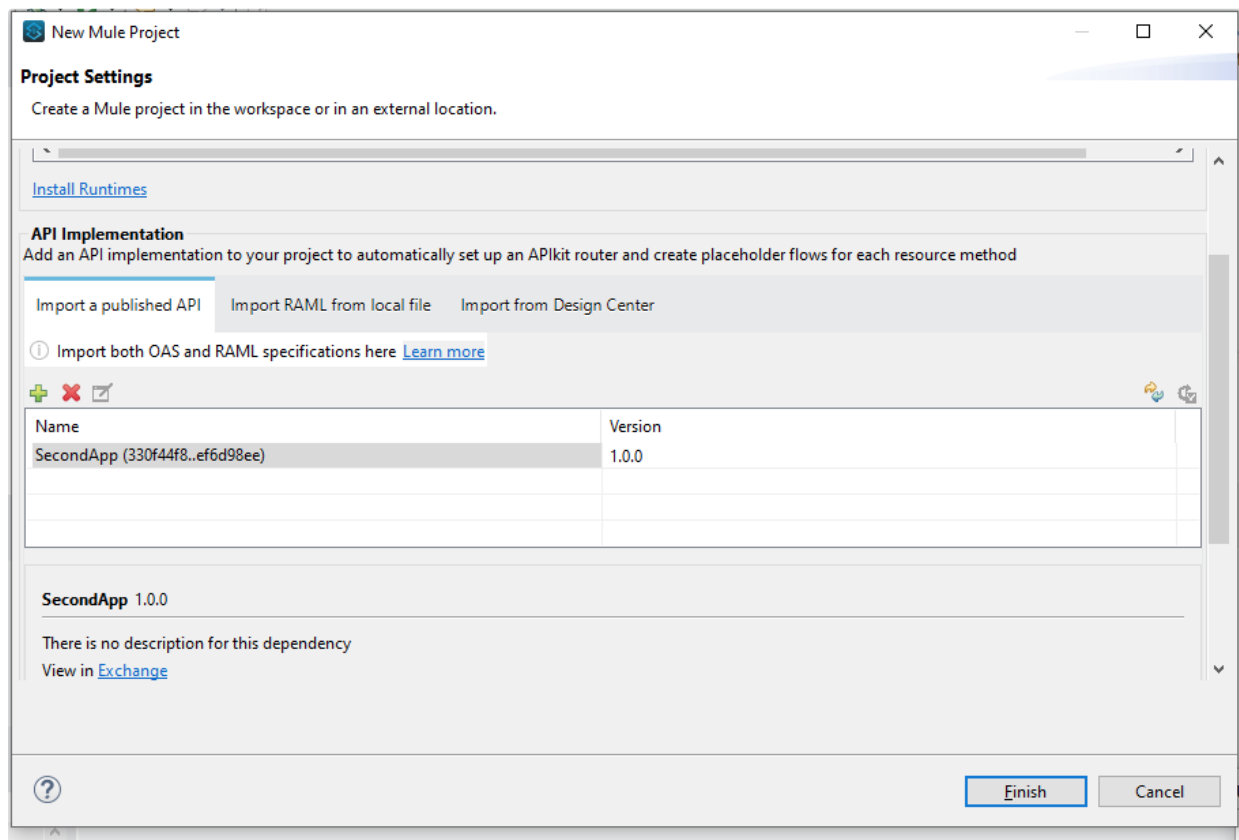
[Open Exchange Preference Page](#)

ⓘ Finish Cancel

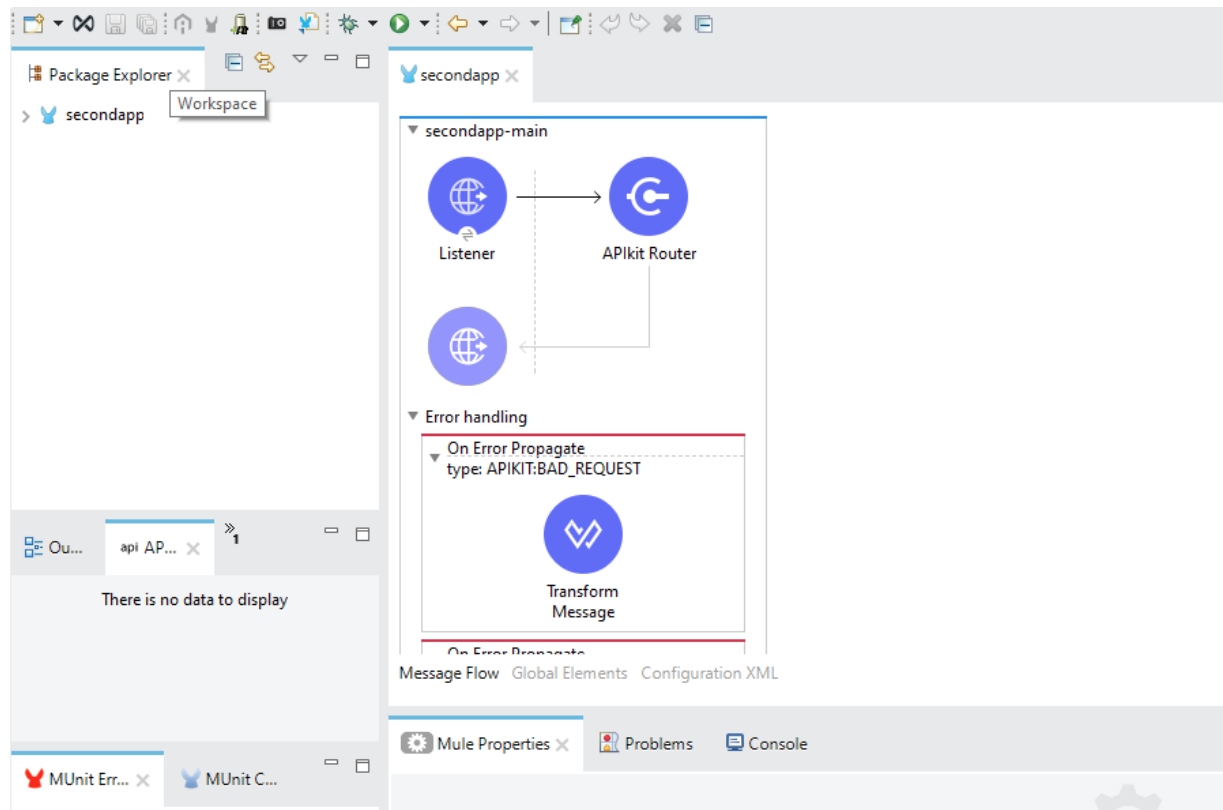
You will see the app under the list, click on the app then click Add button, then click Finish button



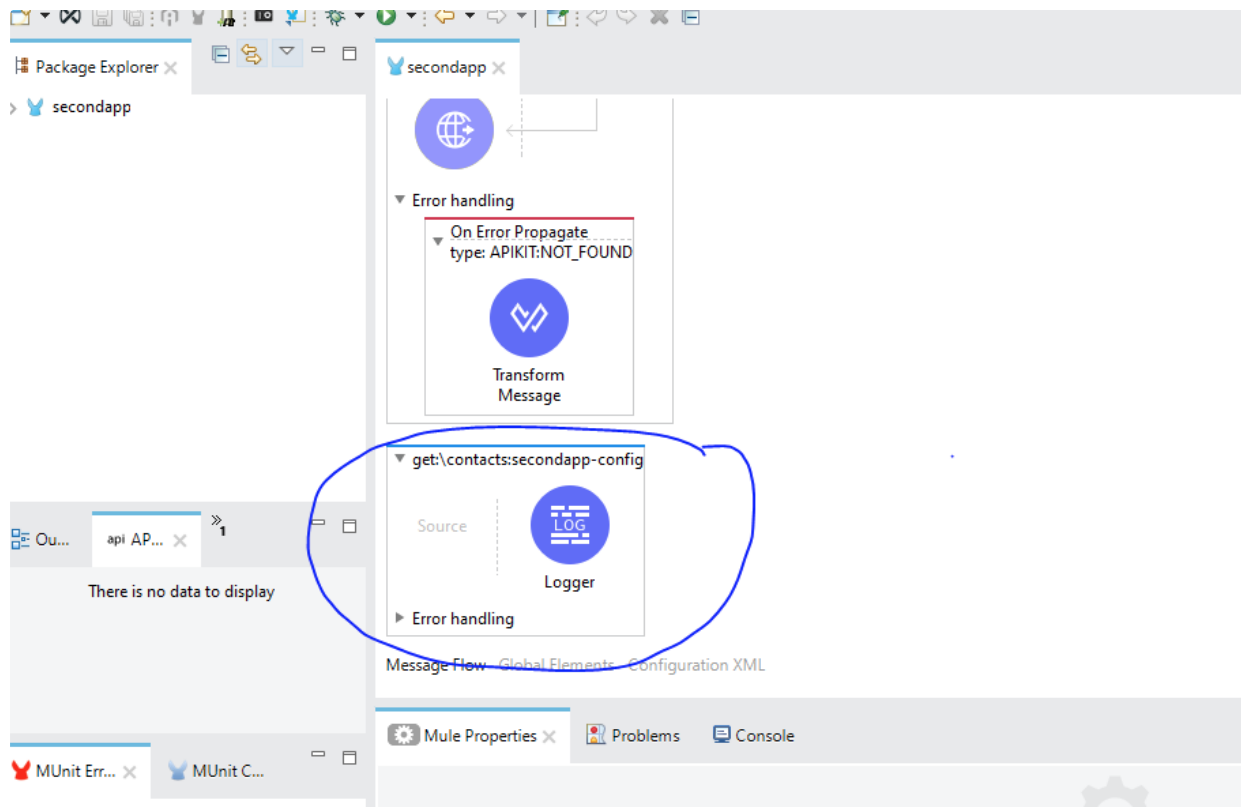
Click Finish button on the Mule Project panel to create the project



The project will be created in while



Now, go to the entry point that we want to implement

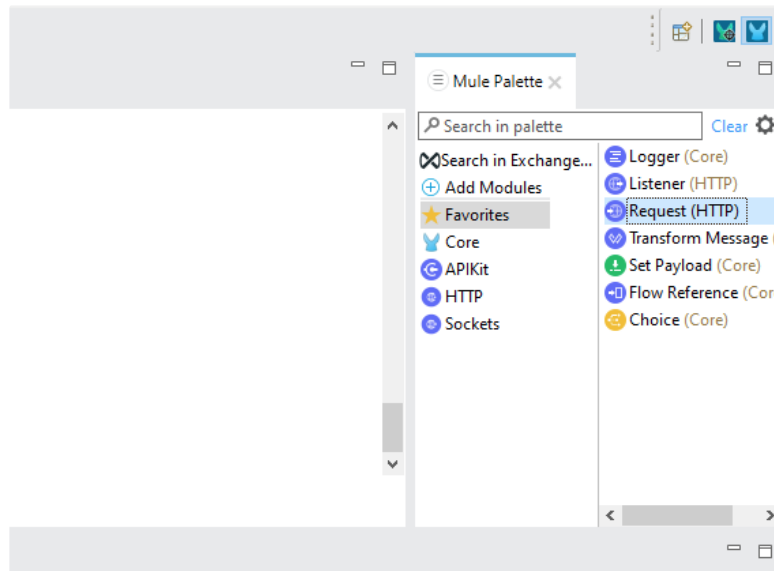


We must prepare an external data, we can use many different kinds of the data from a database, from SOAP webservice, from Restful API, etc. In this article, we just use a simple Restful API

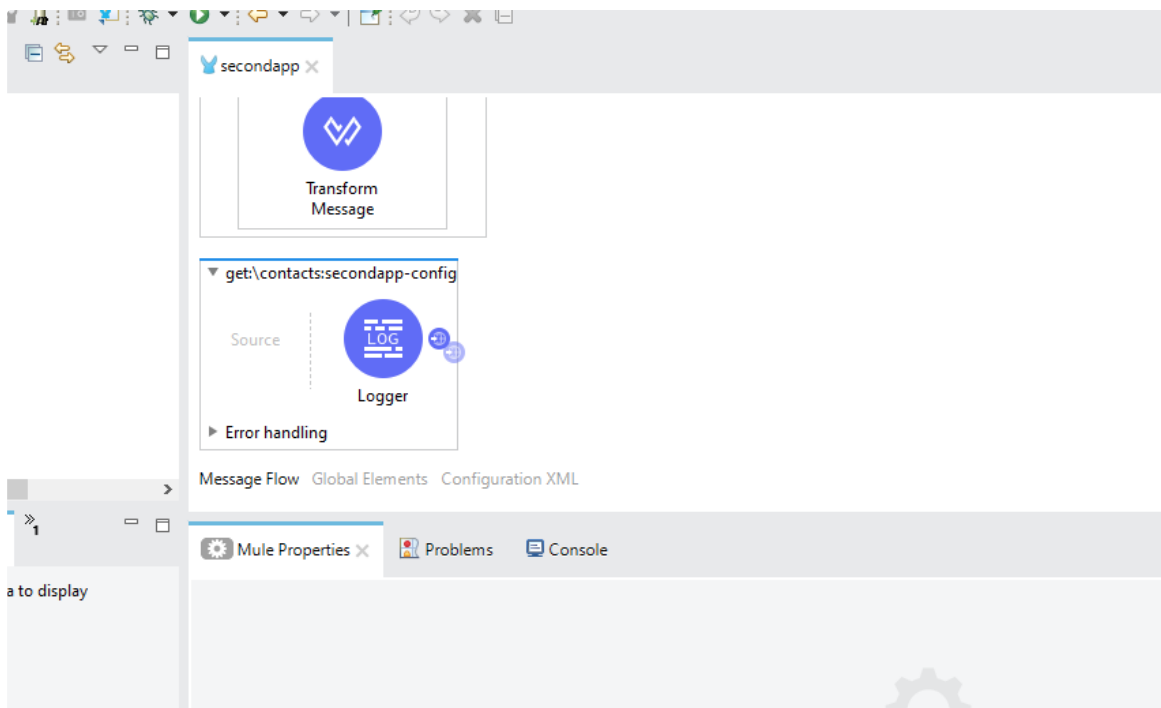
I have prepared an API <https://60499478fb5dcc001796a453.mockapi.io/secondapp/contact>

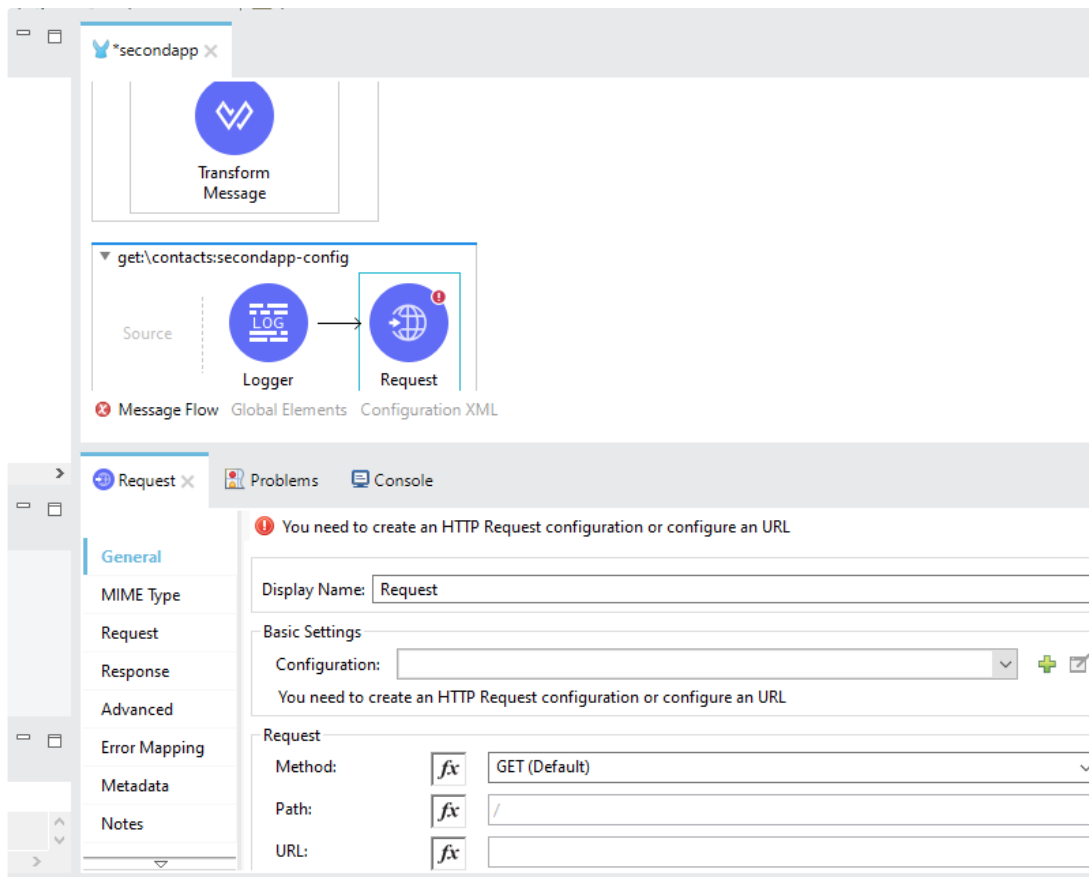
We will use an HTTP Request node to get the data from the API and use a Transform Message node to transform the data to expected data

Under the Mule Palette perspective, search Request, then choose Request (HTTP) node



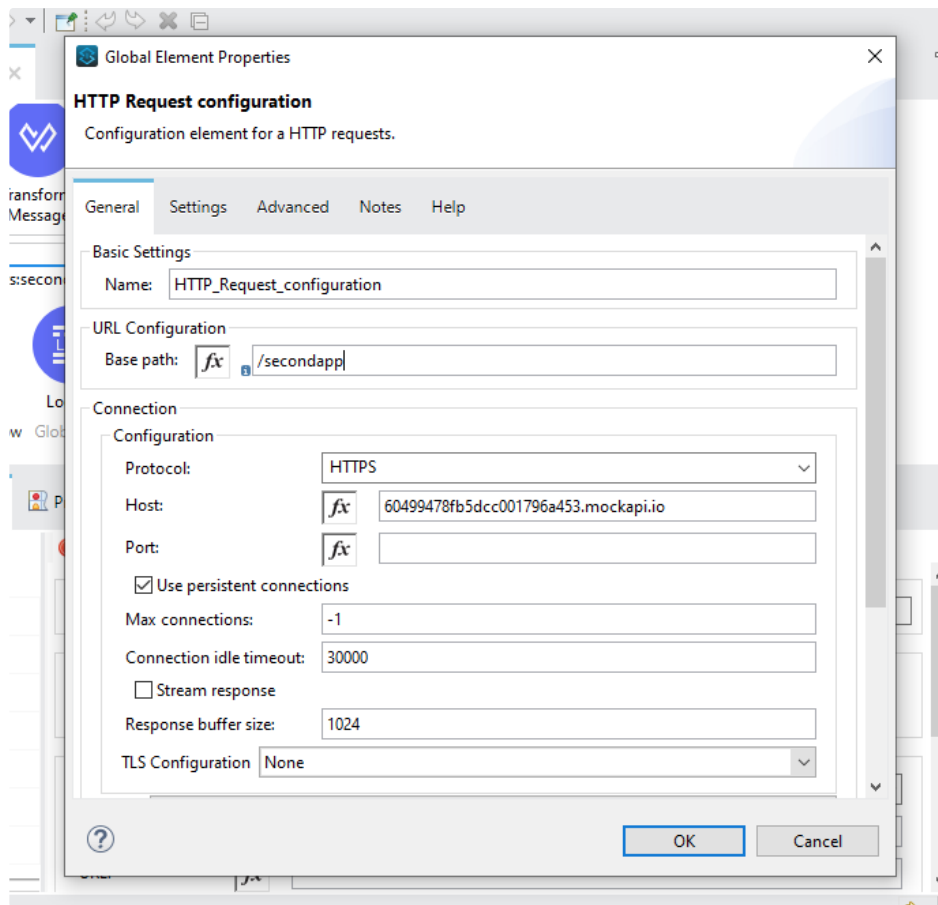
Drag the node to the main panel and connect with the Logger node



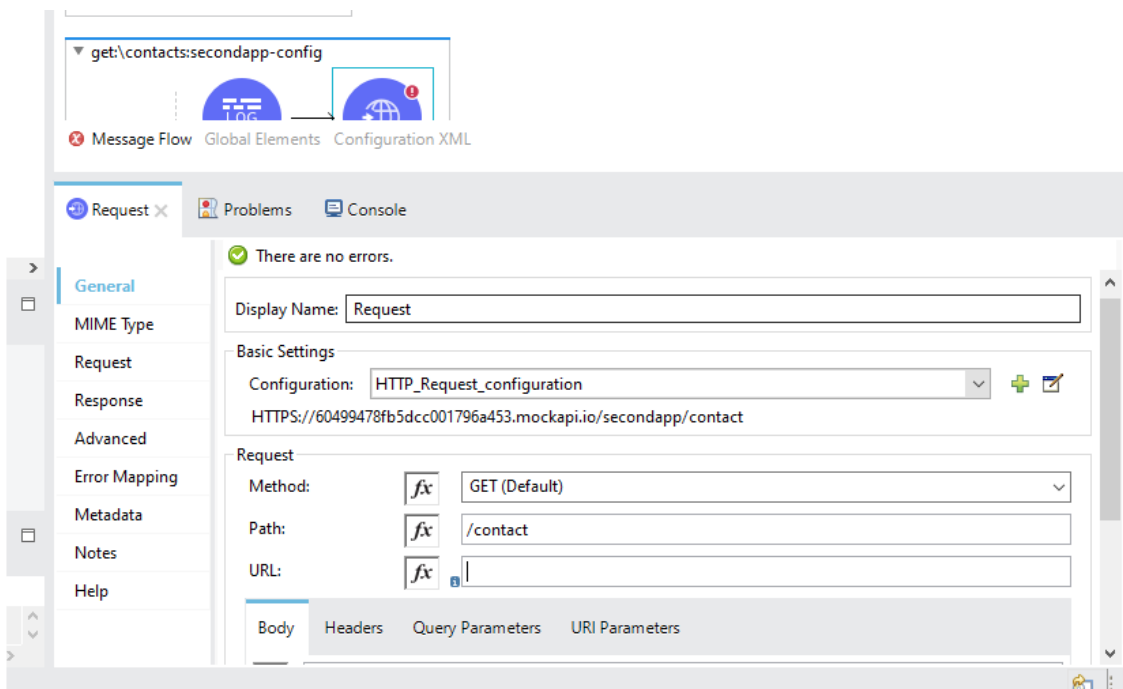


Click on the Request node, the Anypoint Studio will open new Tab allow us to fill the necessary information for this node.

Under the Basic Settings, click on green plus button beside the Configuration input



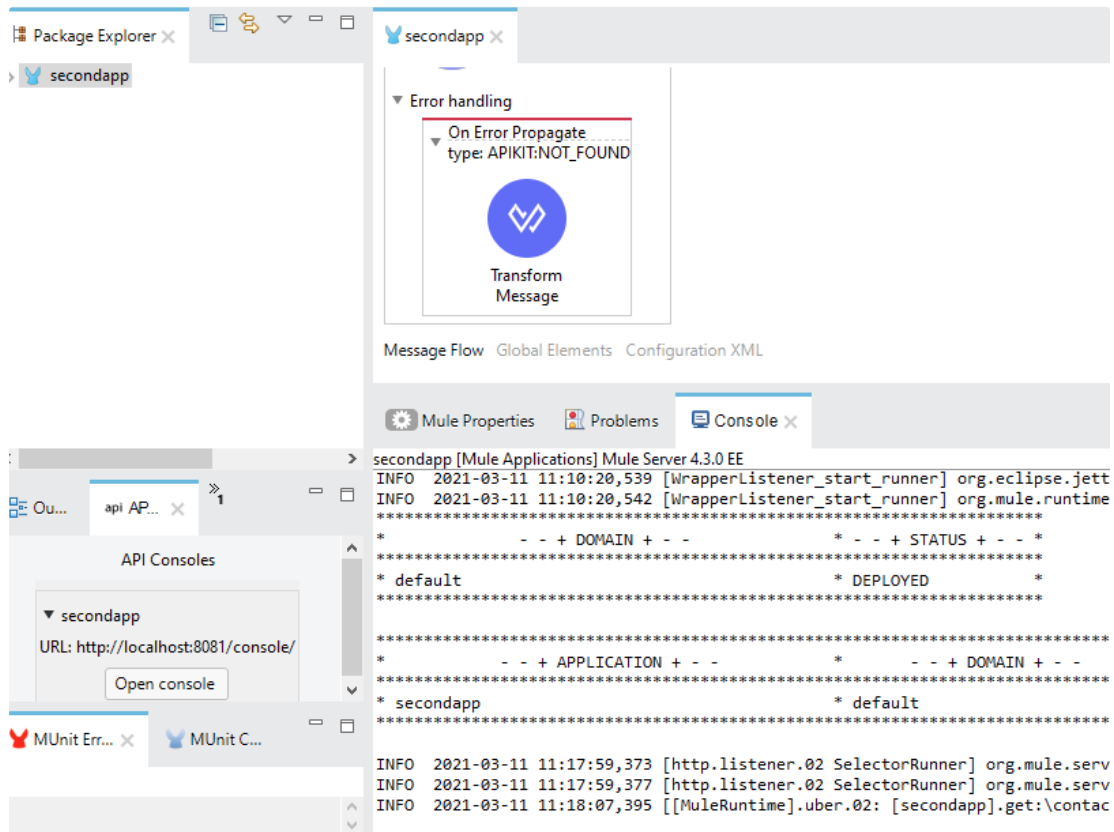
Input the Host, select HTTPS as protocol, input the base path then click OK button



We are using the GET method for this request and input "contact" as the path for the request

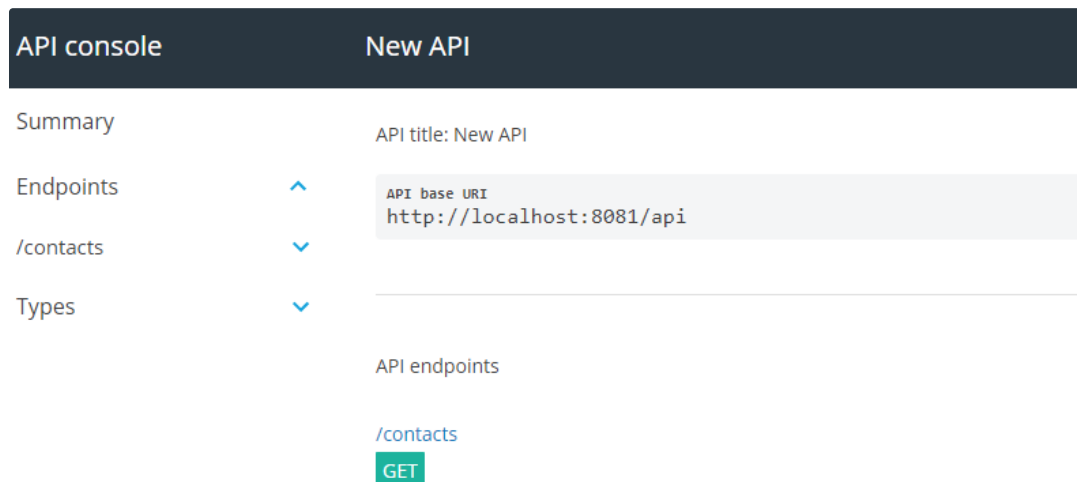
Save the project, right now, you can try to run the app on your local by go to Run → Run as → Mule Application

The Anypoint Studio will build the Application and run the API on <http://localhost:8081/console>



Open your browser and go to <http://localhost:8081/console>, you can see all the information about the API

Let try to run the API by click on the GET method of the contacts endpoint



Then click on the Try it button

New API

Get Try it

GET http://localhost:8081/api/contacts

Code examples Show ▾

Responses

200

Body Hide ▲

Media type: application/json

Then click on the Send button

New API

Get

Request URL
http://localhost:8081/api/contacts

Query parameters

⊕ ADD PARAMETER

Send

You will get the data from the external API with 200 status code

200 OK 1586.53 ms



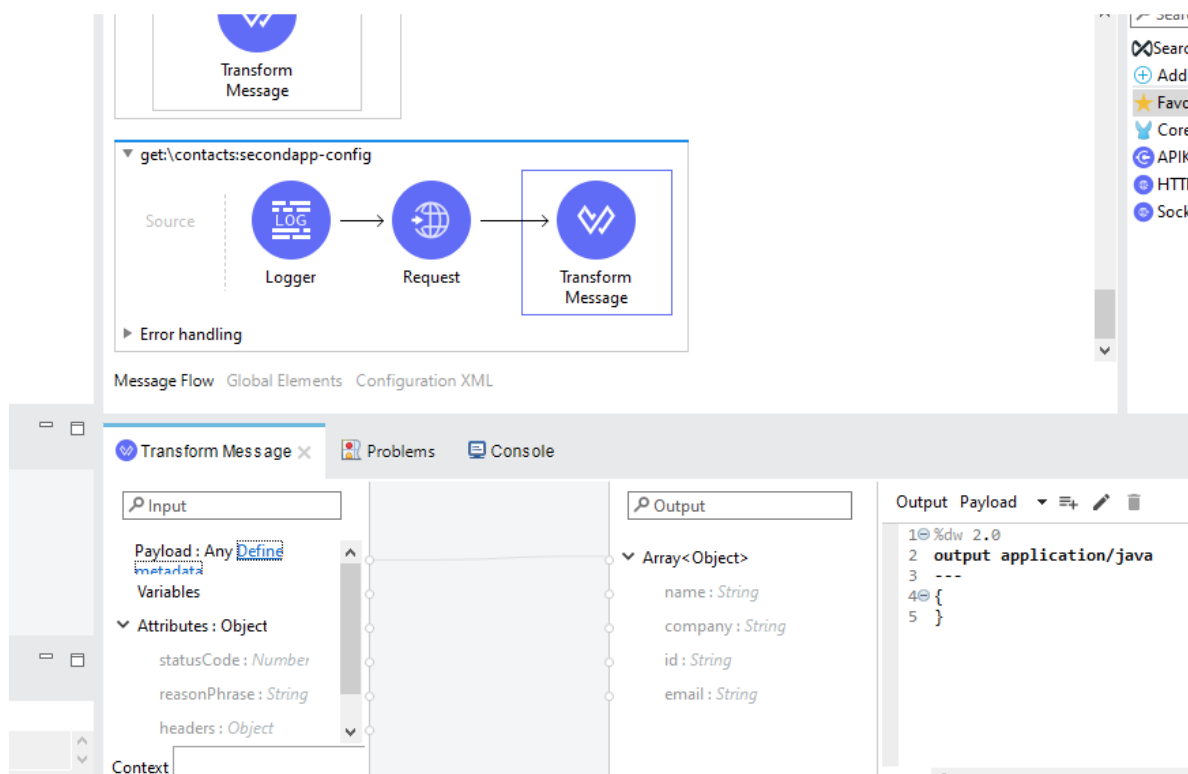
Copy Save Source view Data table

```
[Array[50]
  -0: {
    "Id": "1",
    "Name": "Name 1",
    "Email": "Email 1",
    "Company": "Company 1"
  },
  -1: {
    "Id": "2",
    "Name": "Name 2",
    "Email": "Email 2",
    "Company": "Company 2"
  },
  -2: {
    "Id": "3",
    "Name": "Name 3"
```

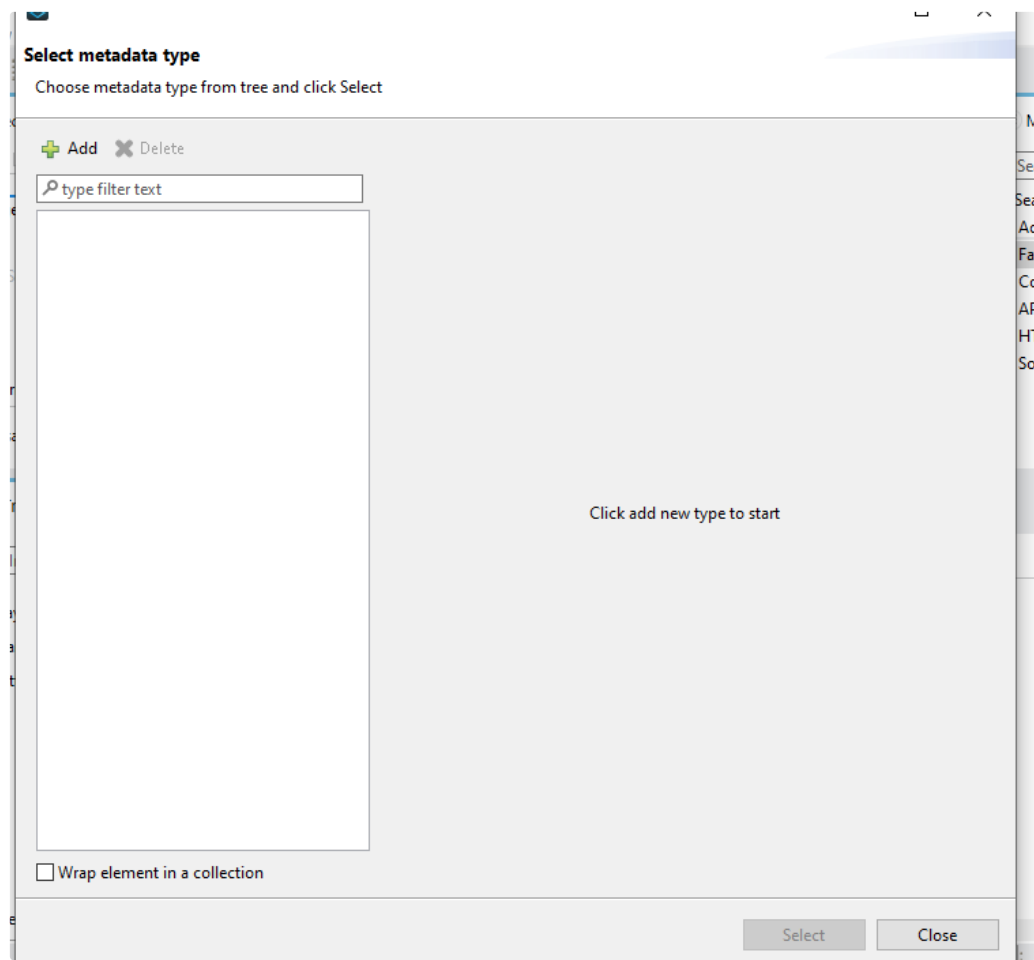
As you can see, the data is same the external API data but this isn't our expected data. Our data should has the lowercase for the object property as we defined in previous article.

We need the Transform Message node to transform the data to our expected data

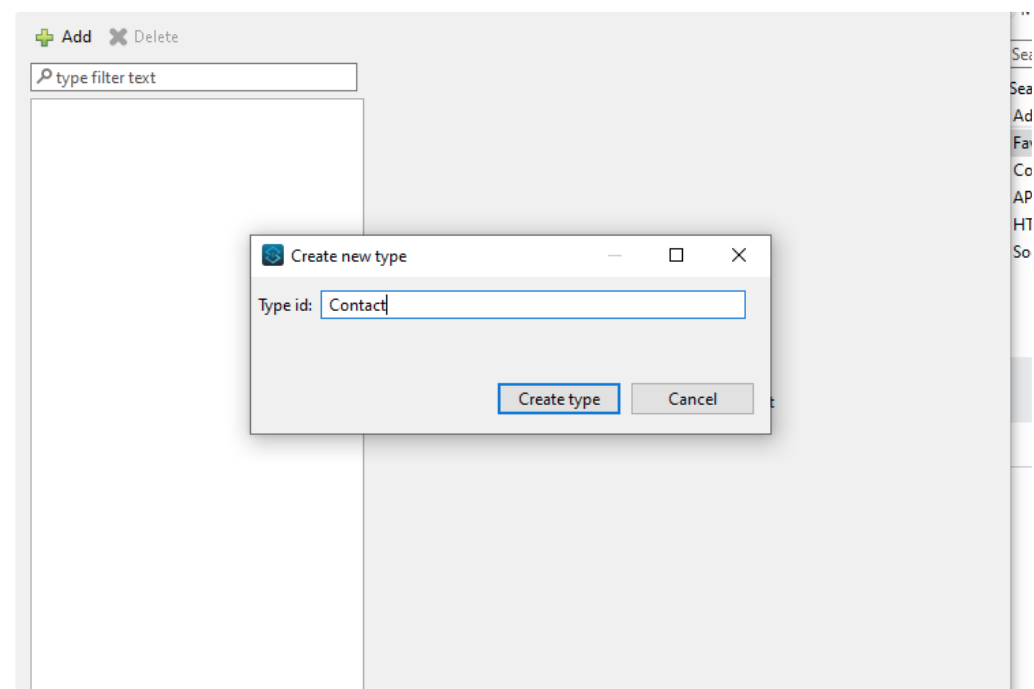
Under the Mule Palette perspective, search Transform, then select the Transform Message node, drag it to main panel and connect with the Request node



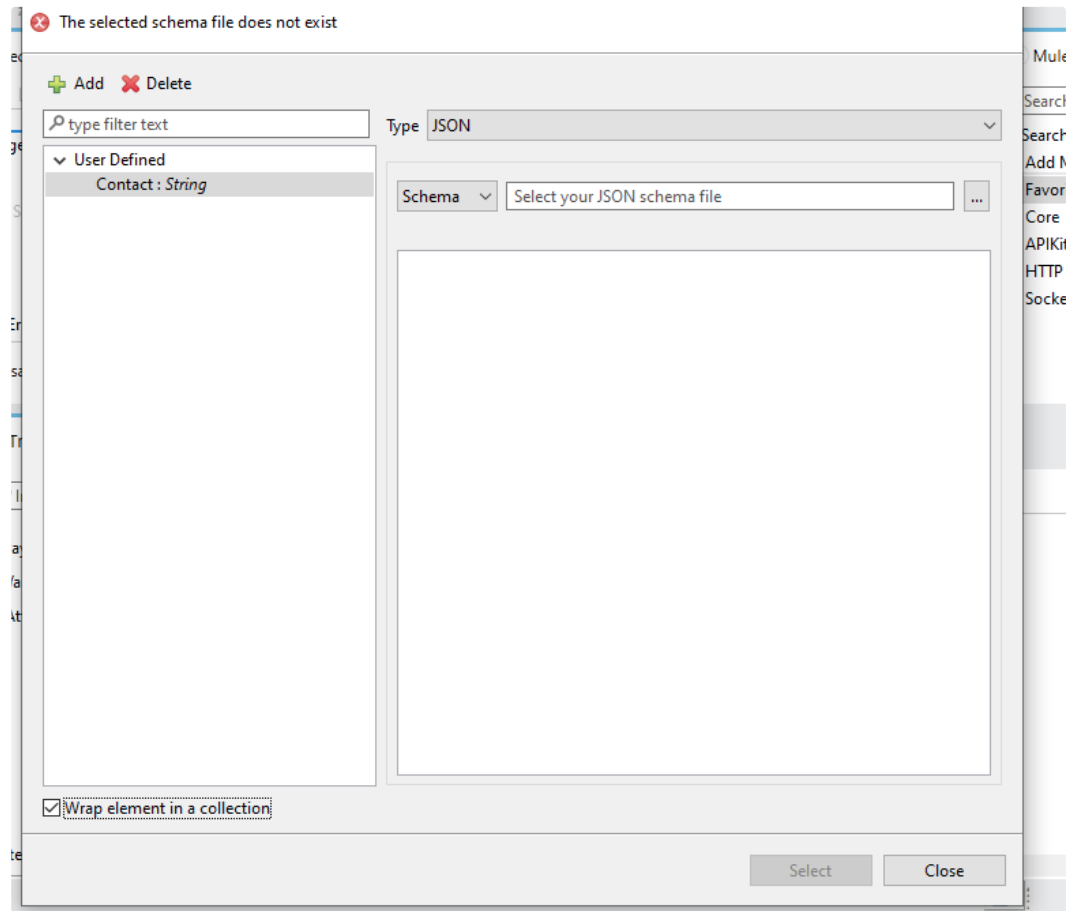
Under the Transform Message tab, beside the Payload, click on the Define metadata text link



Click Add button, input Type ID then click Create type button



In the type dropdown box, select JSON



We need to prepare a JSON schema for the Contact object

Go to [Free Online JSON to JSON Schema Converter](#)

Input `{"Id": "1", "Name": "Name 1", "Email": "Email 1", "Company": "Company 1"}` as sample JSON data, then click Generate Schema button to get the JSON schema

Sample JSON Document

```
1 {"Id": "1", "Name": "Name 1", "Email": "Email 1", "Company": "Company 1"}
```

Options

Generate Schema

Liquid Technologies Web Site uses cookies. [Learn more.](#) [Close](#)

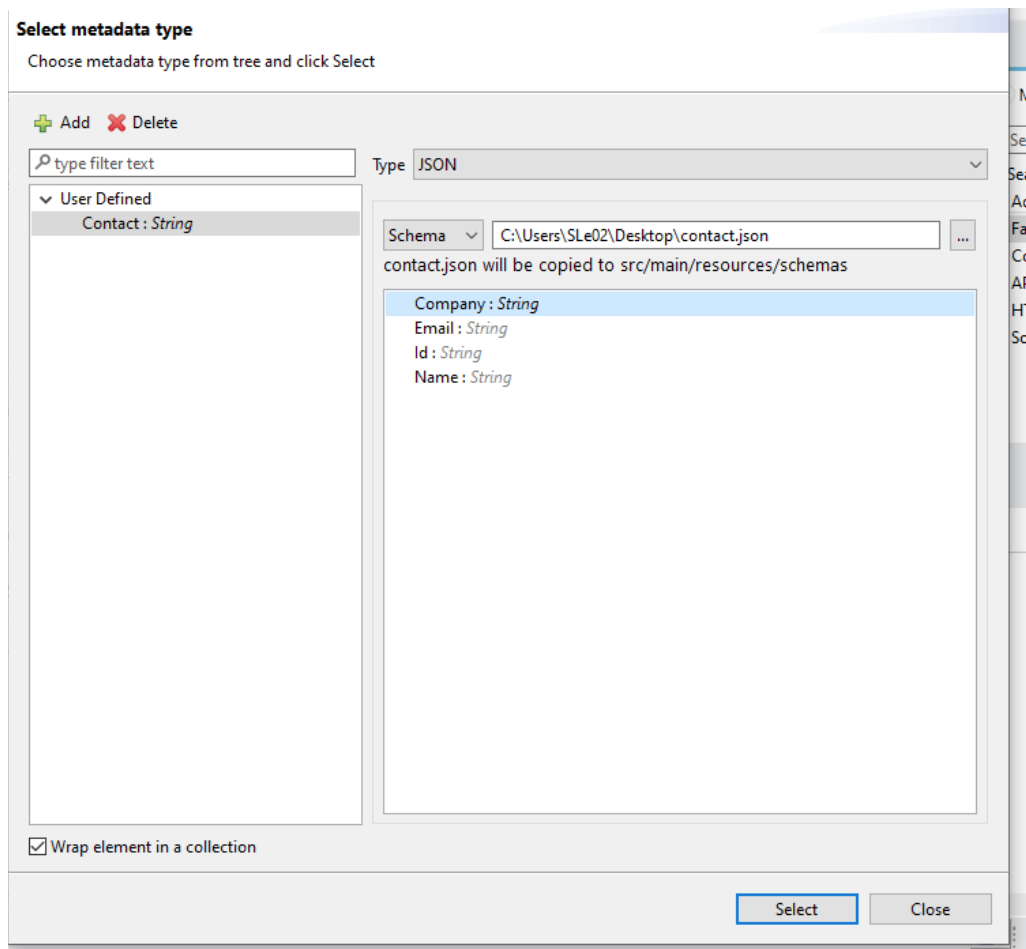
Inferred JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "Id": {
      "type": "string"
    },
    "Name": {
      "type": "string"
    },
    "Email": {
      "type": "string"
    },
    "Company": {
      "type": "string"
    }
  },
  "required": [
    "Id",
    "Name",
    "Email",
    "Company"
  ]
}
```

Save this string to a text file with extension .json

Back to Anypoint Studio, In the Schema, select the file that you saved before

Select **Wrap element in a collection** checkbox



Click Select button

Drag the properties on the left column to the right column with the corresponding property to connect them

The screenshot displays the MuleSoft IDE interface. At the top, a message flow diagram is visible, showing a sequence of components: **Source** (with a 'LOG' icon), **Logger**, **Request**, and **Transform Message**. The configuration is for the path `get:\contacts:secondapp-config`. Below the diagram, tabs for **Message Flow**, **Global Elements**, and **Configuration XML** are present.

The bottom section of the IDE is split into two main areas. On the left, the **Transform Message** configuration is shown, with tabs for **Input**, **Problems**, and **Console**. The **Input** tab shows a **Payload: Array<Object>** with fields: `Company: String`, `Email: String`, `Id: String`, and `Name: String`. The **Output** tab shows the transformed payload structure: `Array<Object>` with fields: `name: String`, `company: String`, `id: String`, and `email: String`.

On the right, the **Output Payload** tab displays the resulting JSON output:

```

1 %dw 2.0
2   output application/json
3   ---
4   payload map ( payload01 , indexOfPayload01 ) ->
5     name: payload01.Name,
6     company: payload01.Company,
7     id: payload01.Id,
8     email: payload01.Email
9   }

```

Now, run the Project again, try to run the API again and you can get the expected data

200 OK 1534.04 ms

Copy

Save

Source view

Data table

```

[Array[50]
-0: {
  "name": "Name 1",
  "company": "Company 1",
  "id": "1",
  "email": "Email 1"
},
-1: {
  "name": "Name 2",
  "company": "Company 2",
  "id": "2",
  "email": "Email 2"
},
-2: {
  "name": "Name 3",
  "company": "Company 3",
  "id": "3",
  "email": "Email 3"
}
]

```