GraphQL Integration with Mulesoft

Billennium

IT for people. People for IT.

What is GraphQL?

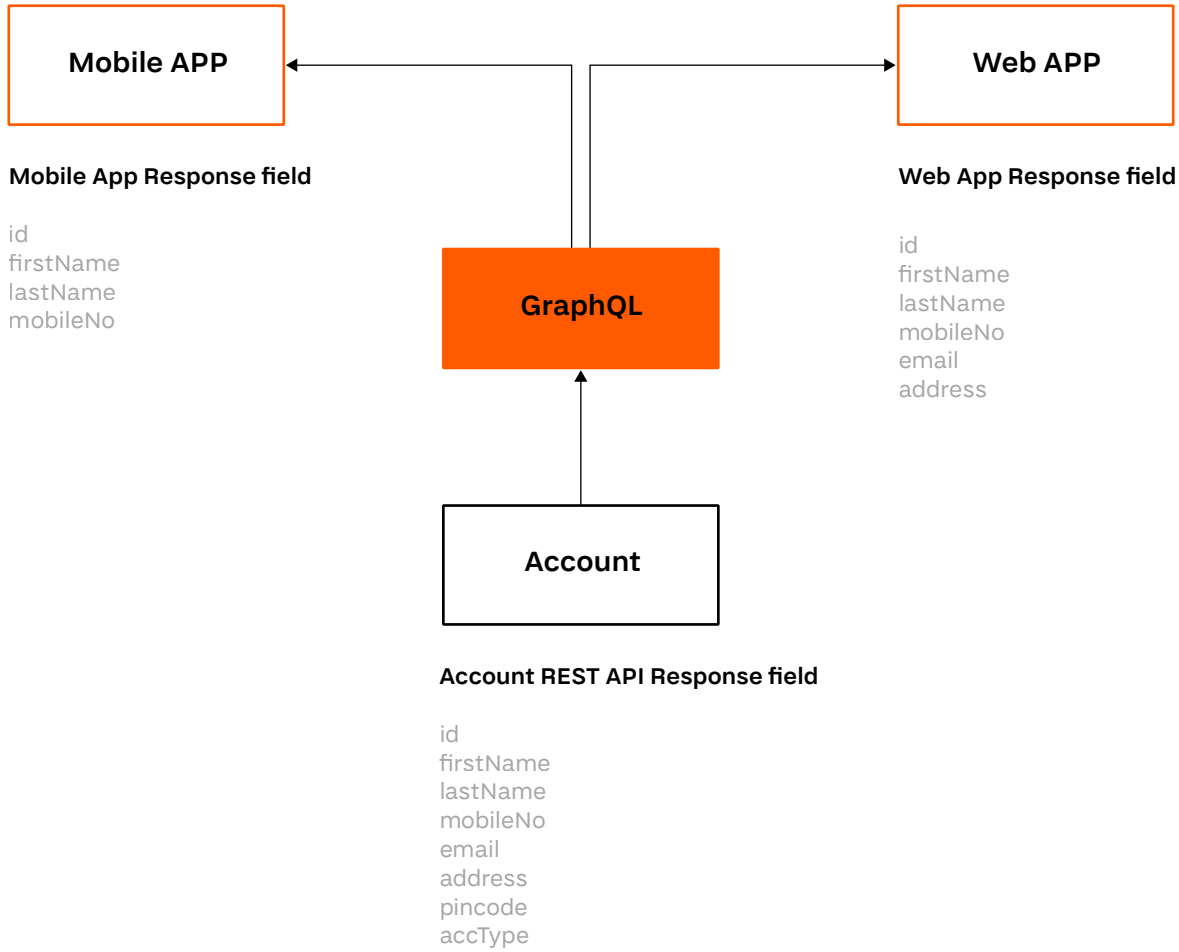
GraphQL for APIs (Application Programming Interfaces) and runtime is an open-source query language allowing clients to choose the fields they are interested in, thus cut down on the amount of data they receive when doing a query, only concentrating on what they need. GraphQL queries always return predictable results. Using GraphQL may also reduce the number of API's you need to create to meet your client's needs.

Why GraphQL?

- Clients can handle logic as per their requirement. By using the GraphQL query they will get an expected response.
- By using GraphQL, Apps will be fast and stable because they control the data they get, not the server.
- Forget about versioning APIs. Evolve your API without versions meaning that adding more fields to an existing endpoint will not break your API.
- The frontend and backend teams can work independently. The frontend team can easily generate a schema from the backend without knowing the code. The schema generated can directly be used to create queries.

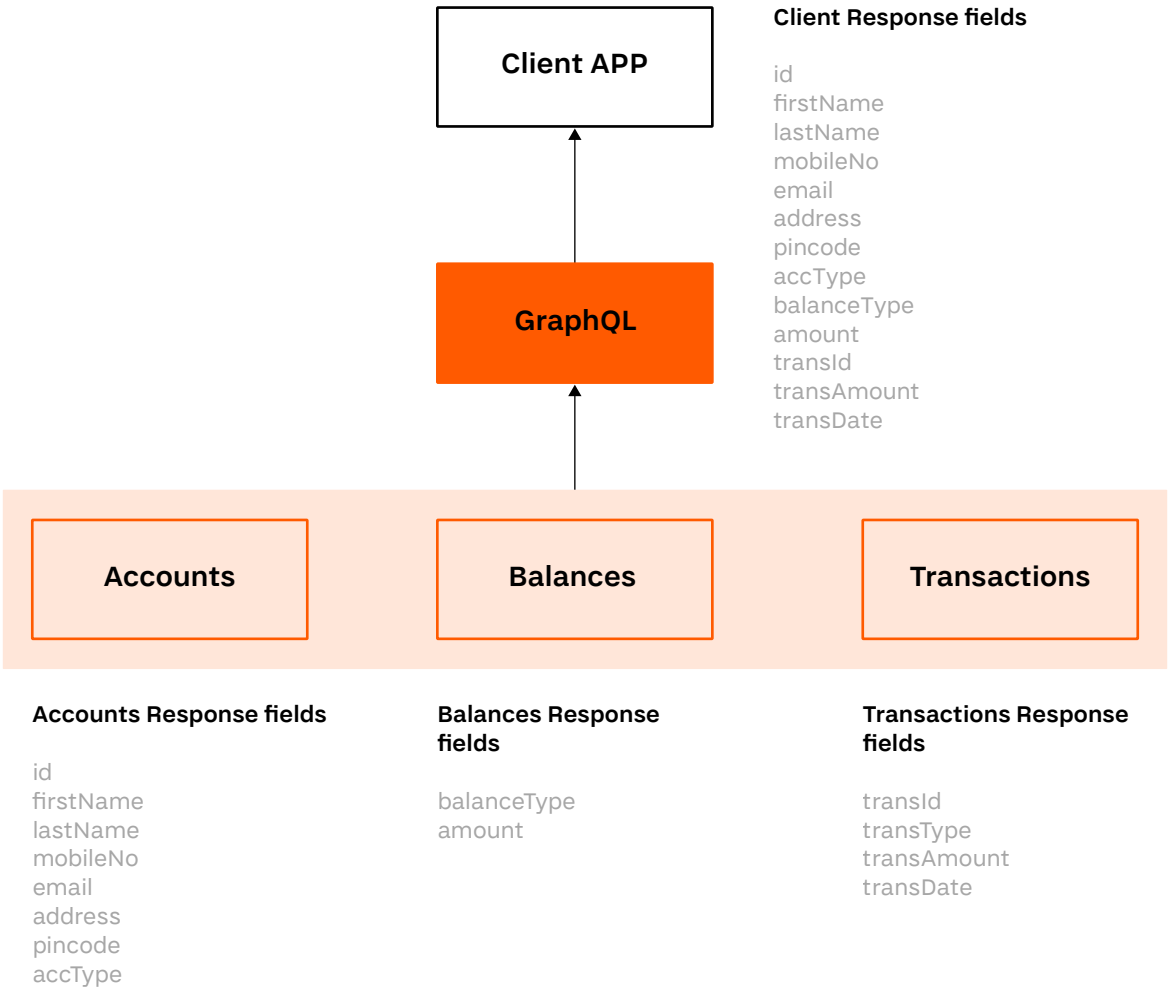
User Case 1:

Suppose you have a REST API that returns 8 fields in response, and you have a mobile application and a web application. As per the requirement, the web application expects only 6 fields, whereas the mobile application expects only 4 fields in response. Here, according to the REST API architecture we need to create two endpoints, but with GraphQL we can achieve the above scenario with a single endpoint without any code change. Below is the illustration of User Case 1.



User Case 2:

Suppose you must call multiple REST APIs and each API will respond with a different JSON (JavaScript Object Notation) payload. GraphQL will route multiple requests from REST API and aggregate all the responses and send back the required response to the client. Below is the illustration of User Case 2.



The above user cases can be implemented using two solutions: Anypoint Datagraph framework or GrahpQL connector. We will explore each of them below.

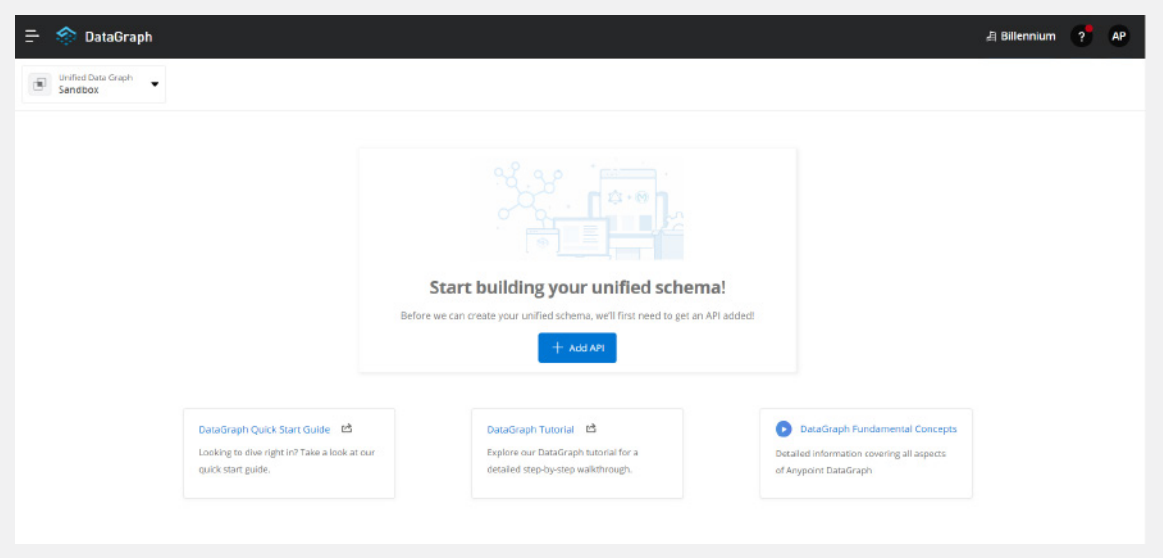
Anypoint Platform Datagraph tool:

Anypoint Datagraph enables you to connect with these graphs into one unified schema that runs as a single SaaS (Software as a Service) GraphQL endpoint. It contains all the links and fields defined in your APIs. As a result, you can query across the underlying APIs without needing to understand all the relationships or specific capabilities that exist within them.

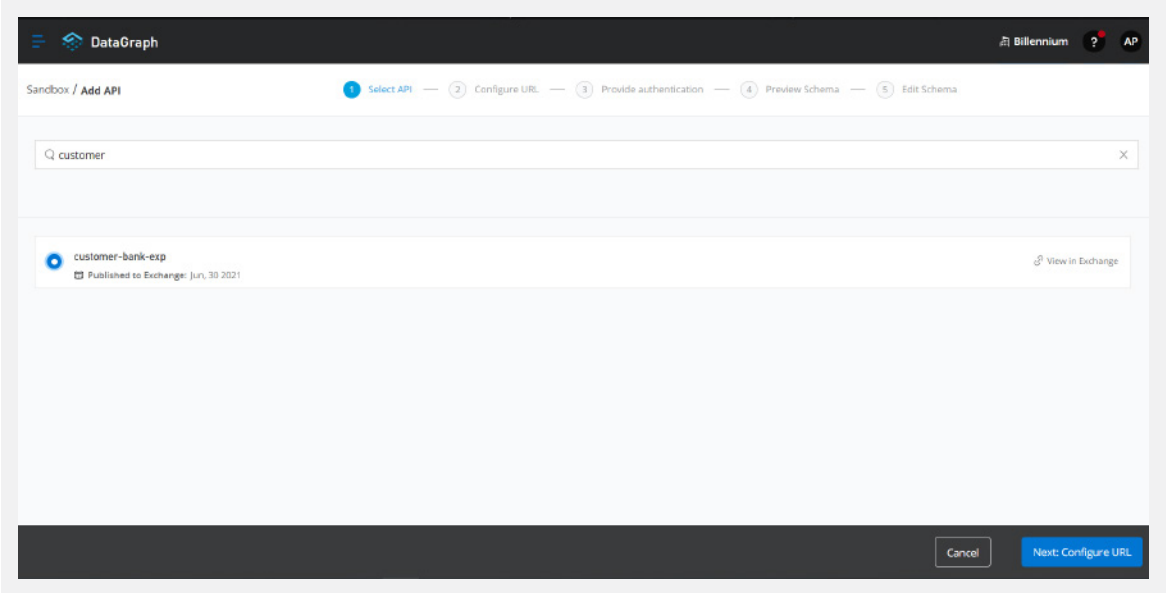
This article assumes that you are familiar with Anypoint Platform and Anypoint Studio.

Firstly, you must create an API specification in the design centre and publish RAML into exchange.

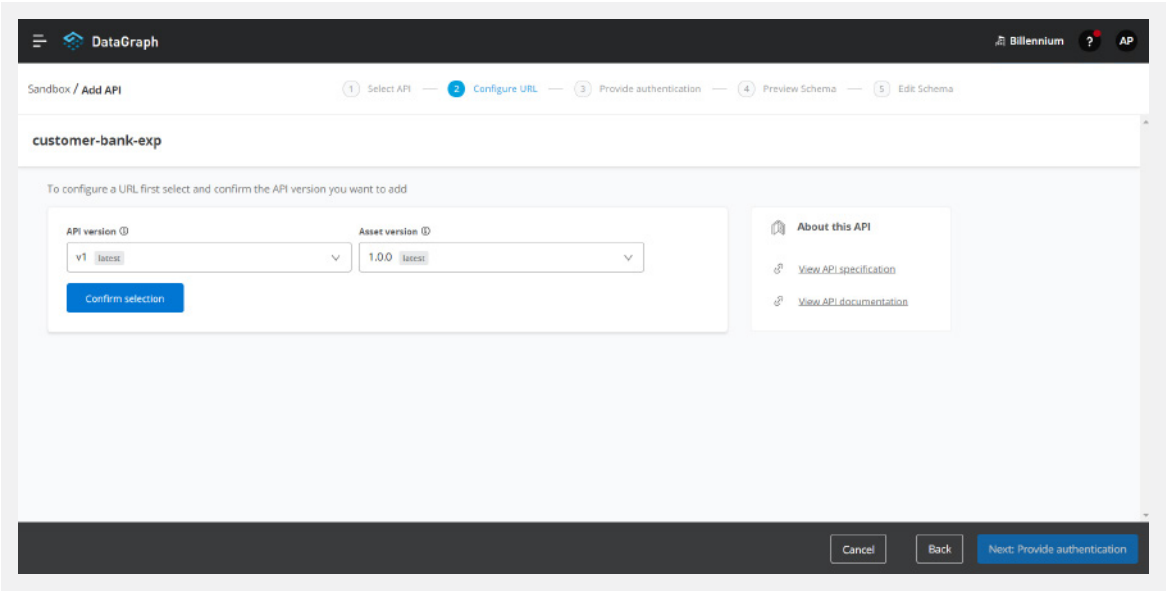
→ **Step 1:**
Go to the Anypoint Datagraph tool and select Add API.



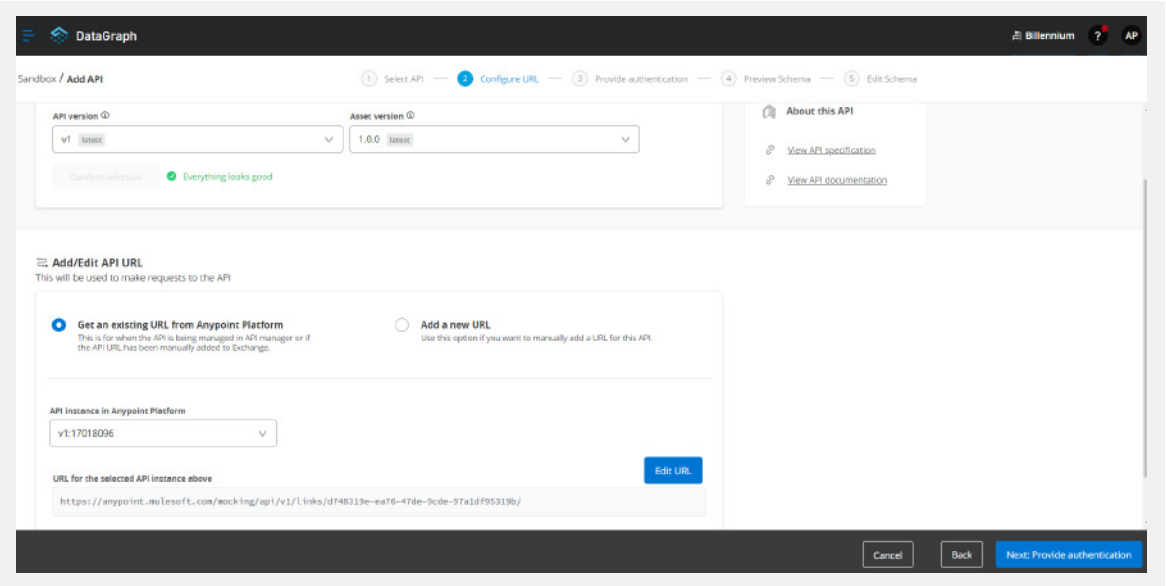
→ **Step 2:**
Search your application in the search bar and after 'select application', click on Next: Configure URL.



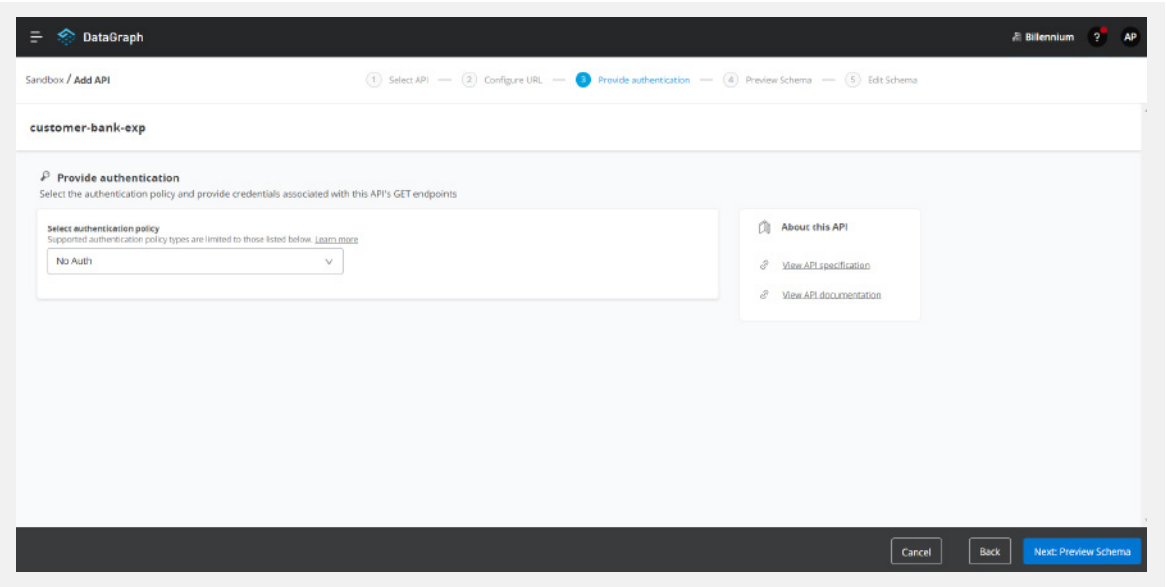
→ **Step 3:**
Confirm the API and Asset version and click on Confirm selection.



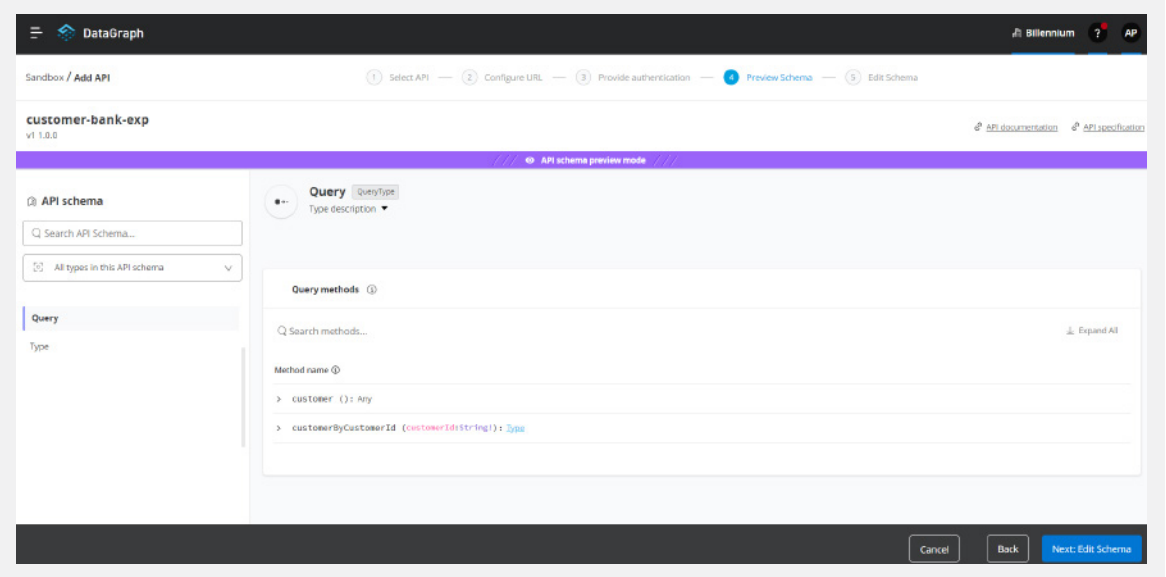
→ **Step 4:**
After confirming the selection, provide the base URL of your application and click on Next: Provide Authentication. (In my case I am using a mock URL from the Design Centre)



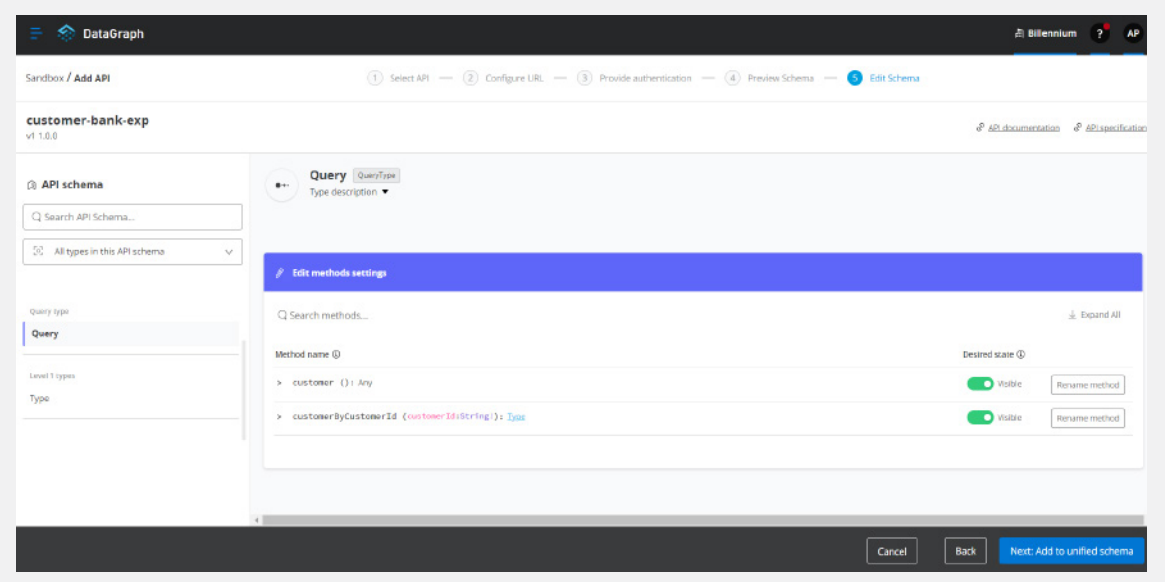
→ **Step 5:**
If you have any authentication enabled for your API, please select the authentication method accordingly and click on Next: Preview Schema (In my case I am using No Auth).



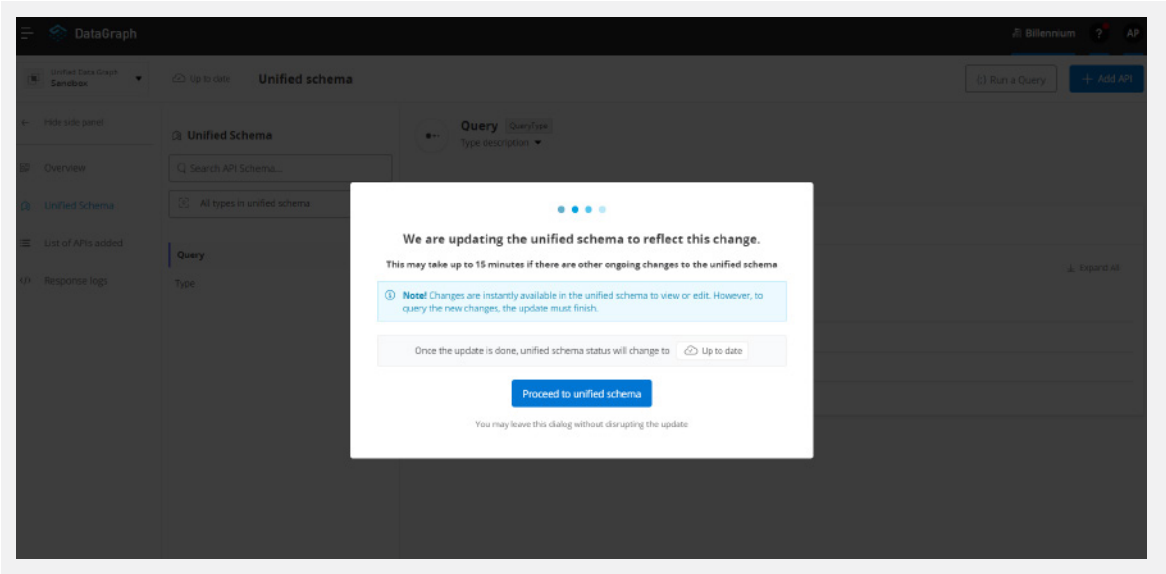
→ **Step 6:**
In the preview schema section you can see all the listed methods in your RAML and data types. Click on Next: Edit Schema.



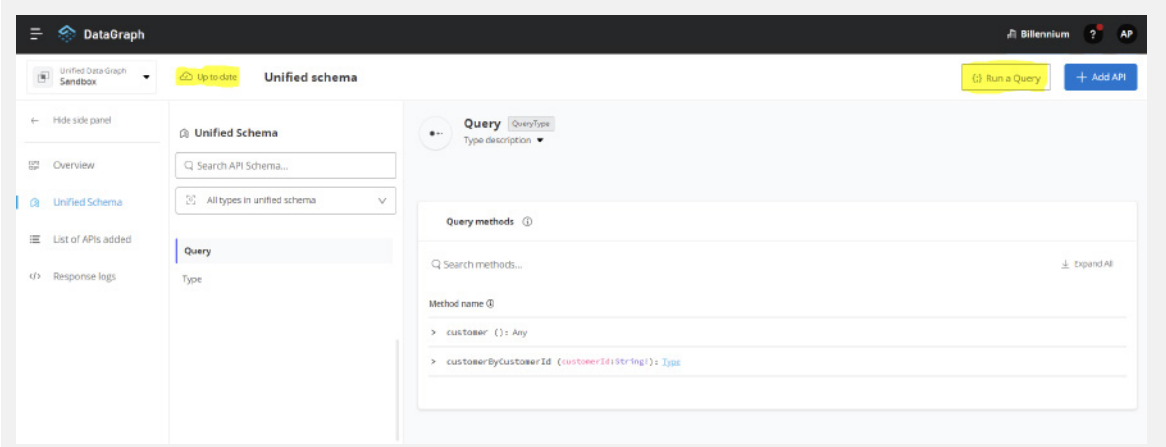
→ **Step 7:**
In the edit schema section you can hide and enable methods as per your requirement. Click on Next: Add to unified schema.



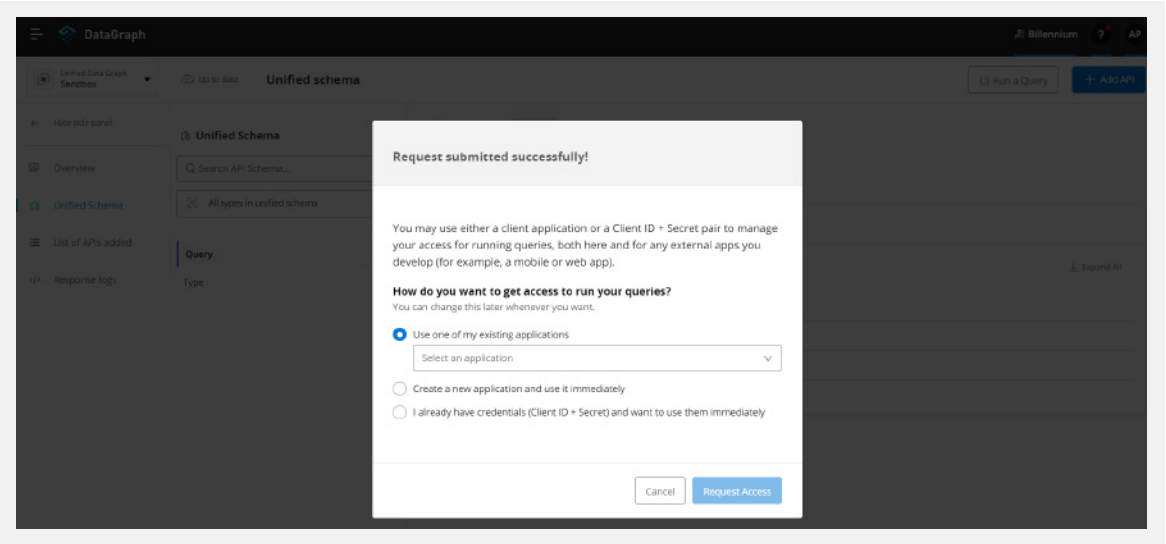
→ **Step 8:**
It may take up to 15 minutes to update the unified schema.



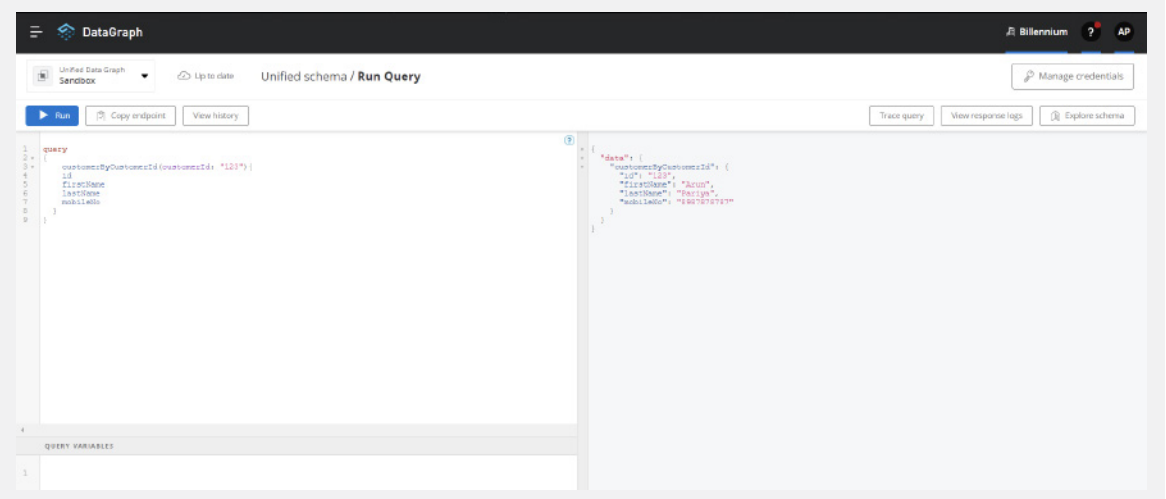
→ **Step 9:**
Once it is completed, you can see the up to date status as shown below and click on Run a Query.



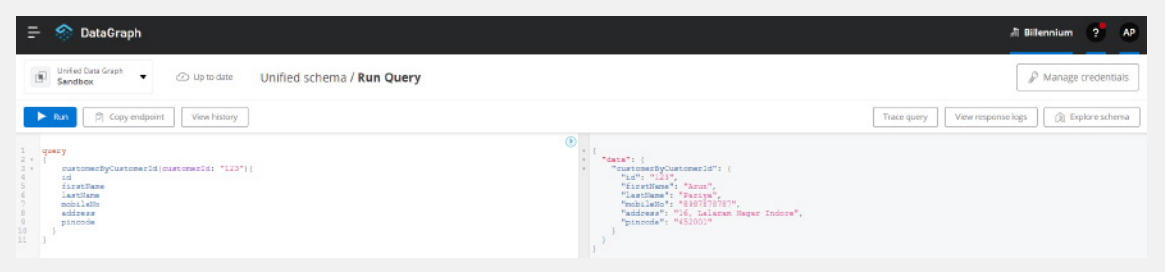
→ **Step 10:**
You can select an existing application or create a new application and click on Request Access.



→ **Step 11:**
After application request access, your graph query console will open, and you can start to write and validate your query here.



According to your requirement, you can add more fields.



GraphQL Implementation with Mulesoft:

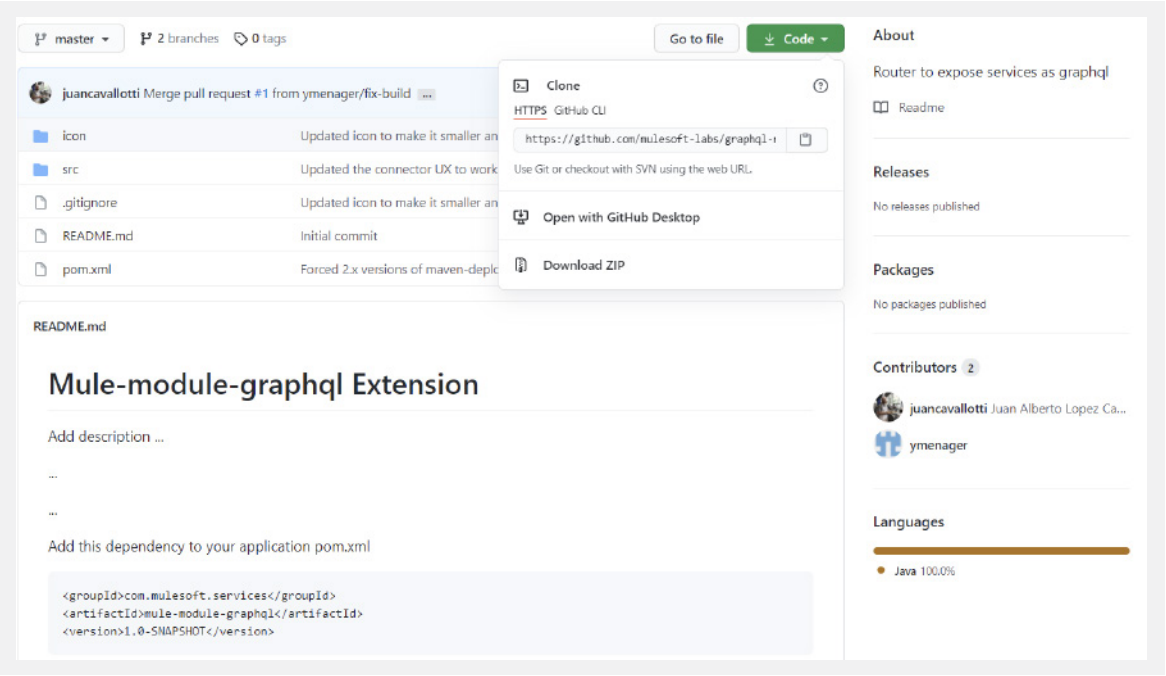
Prerequisites:

- Anypoint Studio 7
- JDK 1.8
- Postman

Firstly, you must download the graphql router that is available on GitHub as part of open source.

Go to: https://github.com/mulesoft-labs/graphql-router

Copy the clone URL or download a .zip file into your local system.



After cloning or downloading the project, go to the pom.xml file and change the mule-modules-parent version from 1.0.0 to 1.1.3.

Add below plugin repository into your pom.xml file.

```
<pluginRepositories>
        <pluginRepository>
            <id>mulesoft-releases</id>
            <name>MuleSoft Releases Repository</name>
            <layout>default</layout>
            <url>https://repository.mulesoft.org/releases/</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </pluginRepository>
    </pluginRepositories>
```

Go to the location where the project is cloned or downloaded and then go to command prompt and run command:

```
<pluginRepositories>
```

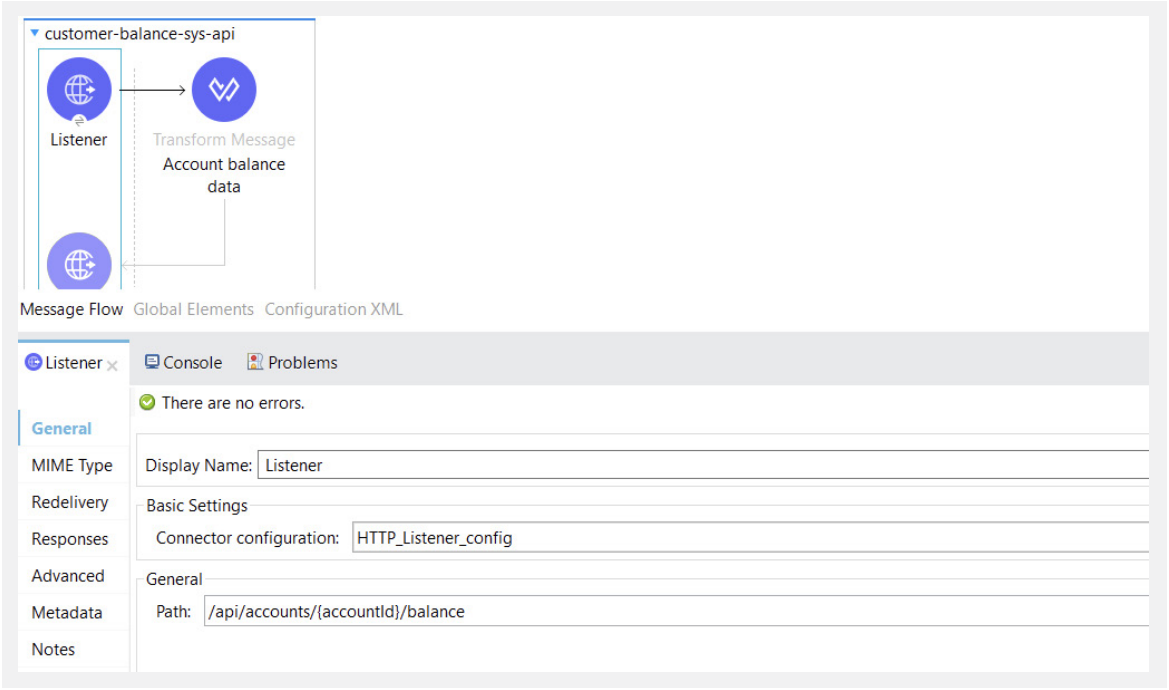
After Building Success, let's get started...!

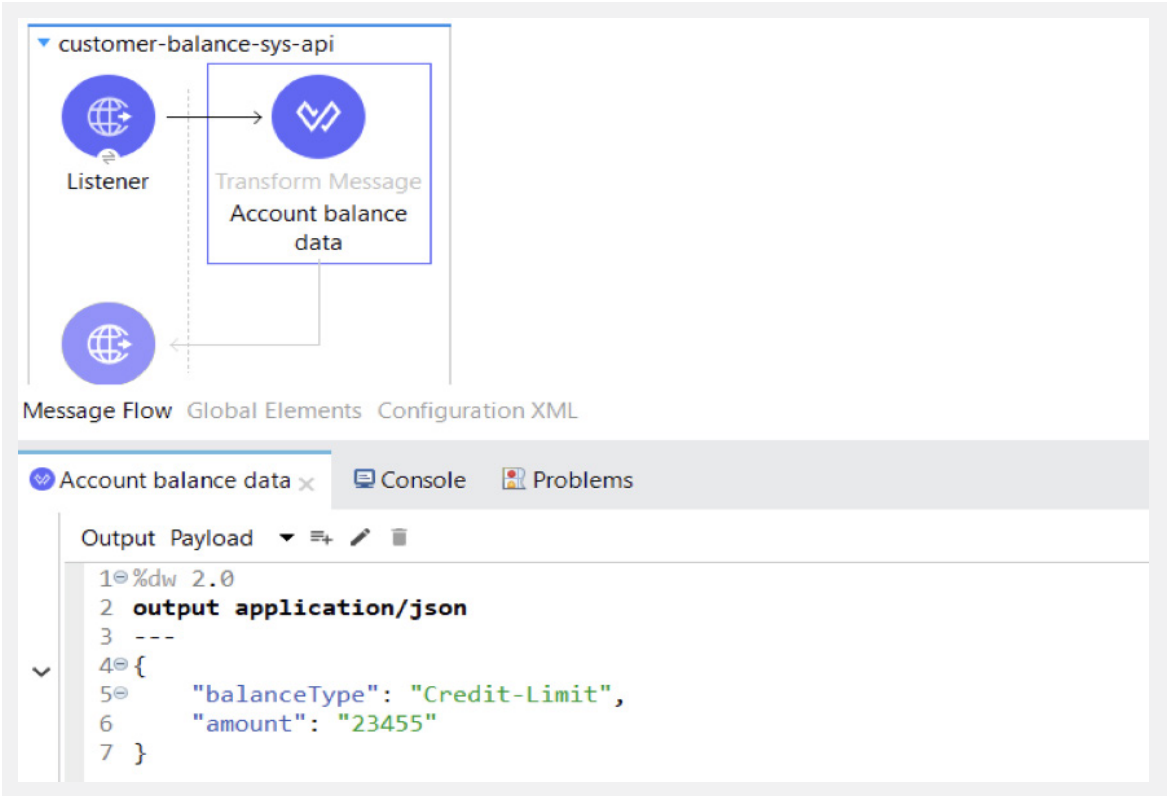
→ **Step 1:**
Create a sample Mule project in the Anypoint studio and add the below dependency in the pom.xml file for the GraphQL module.

```
<dependency>
  <groupId>com.mulesoft.services</groupId>
  <artifactId>mule-module-graphql</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <classifier>mule-plugin</classifier>
</dependency>
```

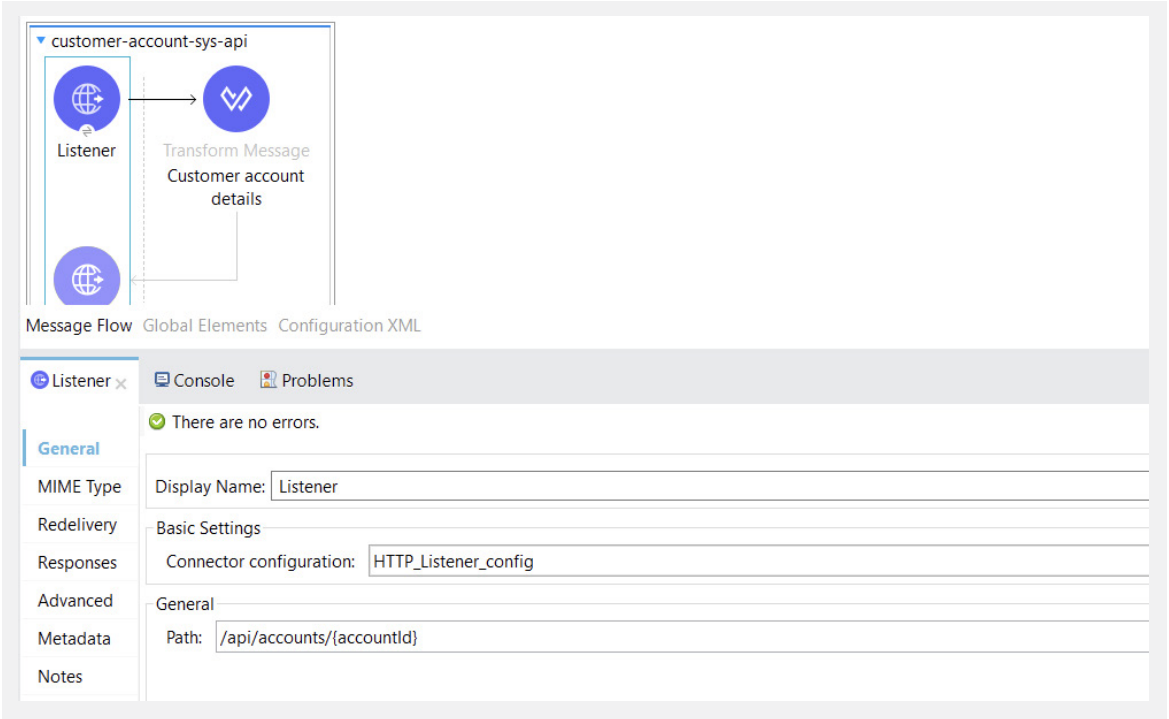
→ **Step 2:**
Create a sample REST API for account, balance, and transaction details.

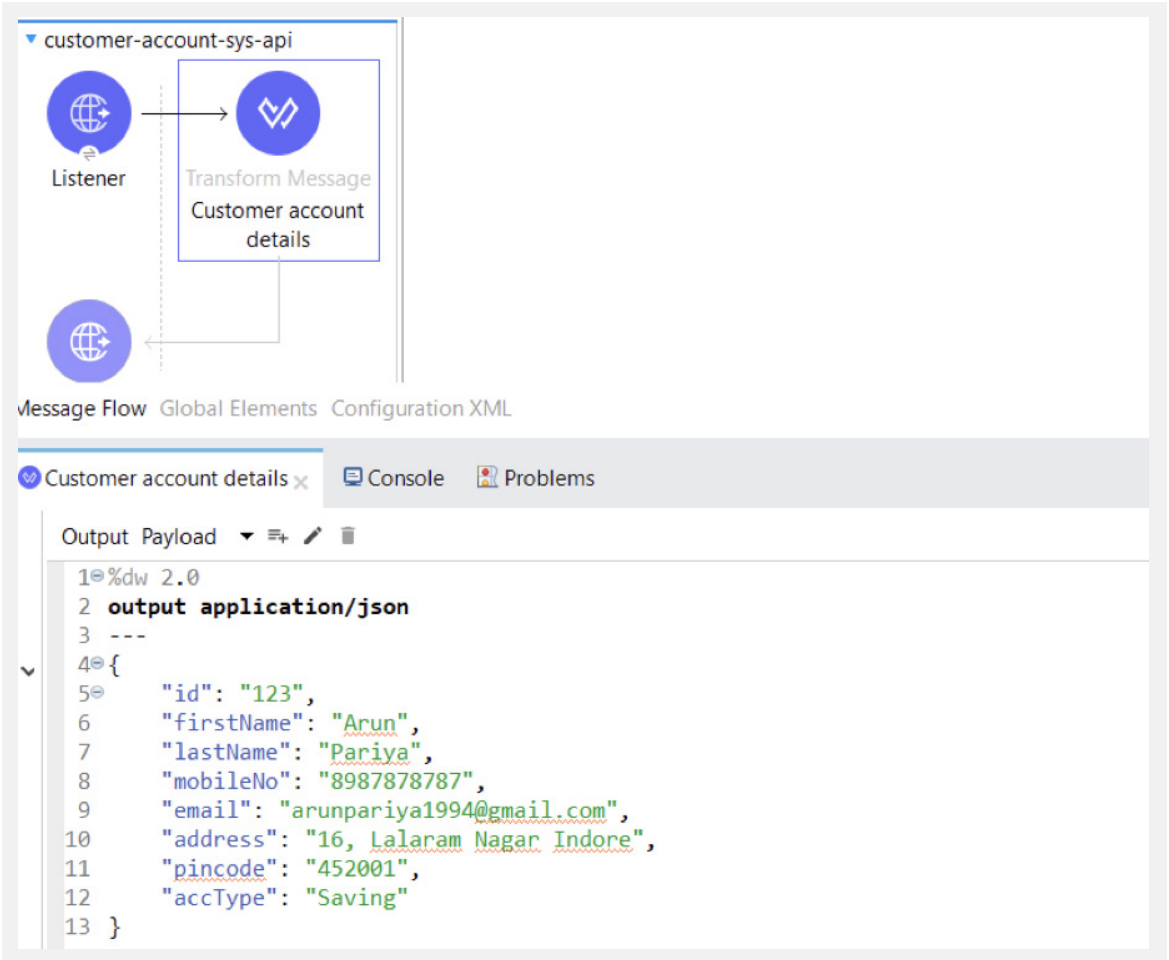
- customer-balance-sys API



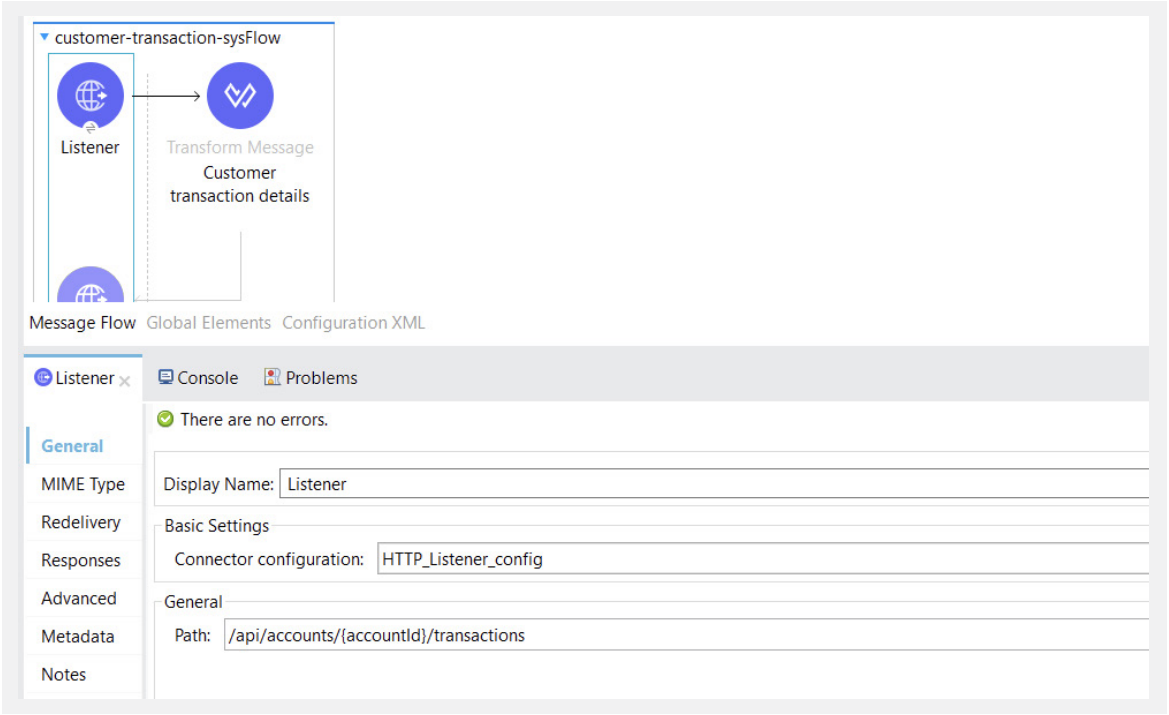


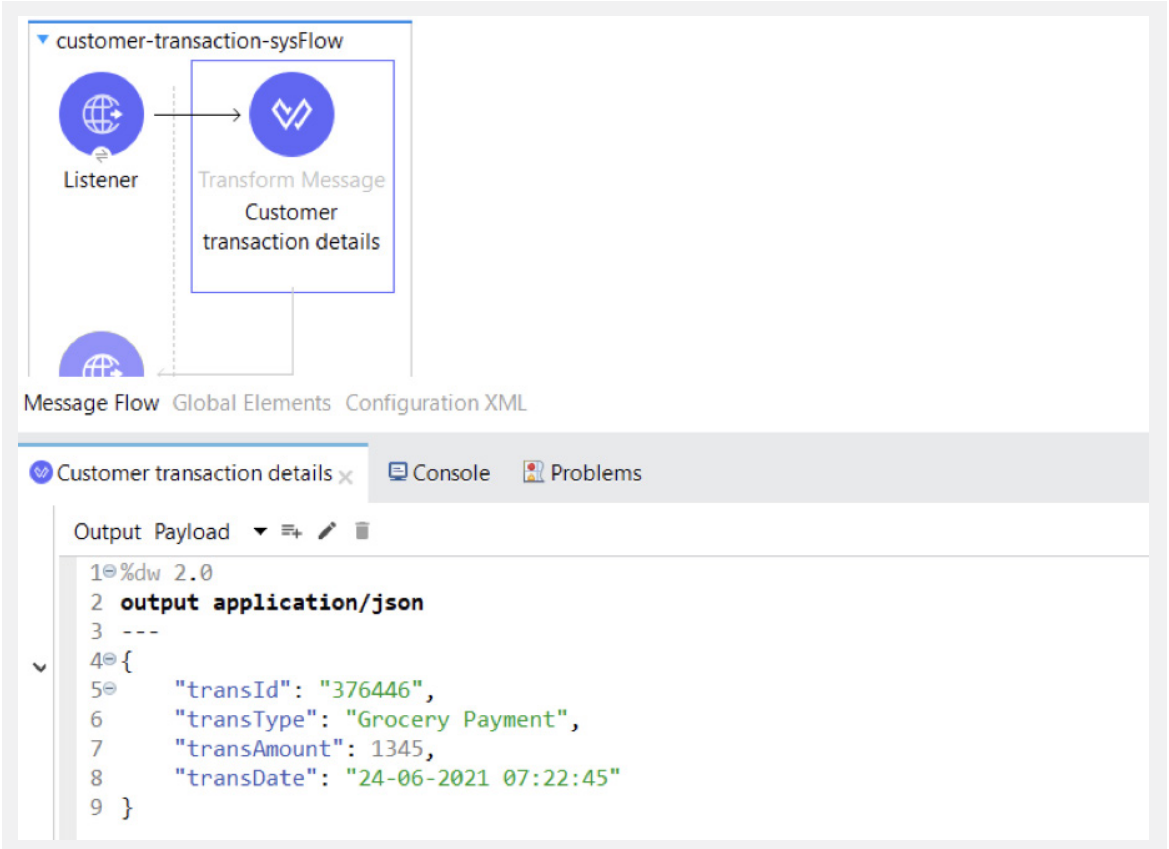
- customer-account-sys API





- customer-transaction-sys API

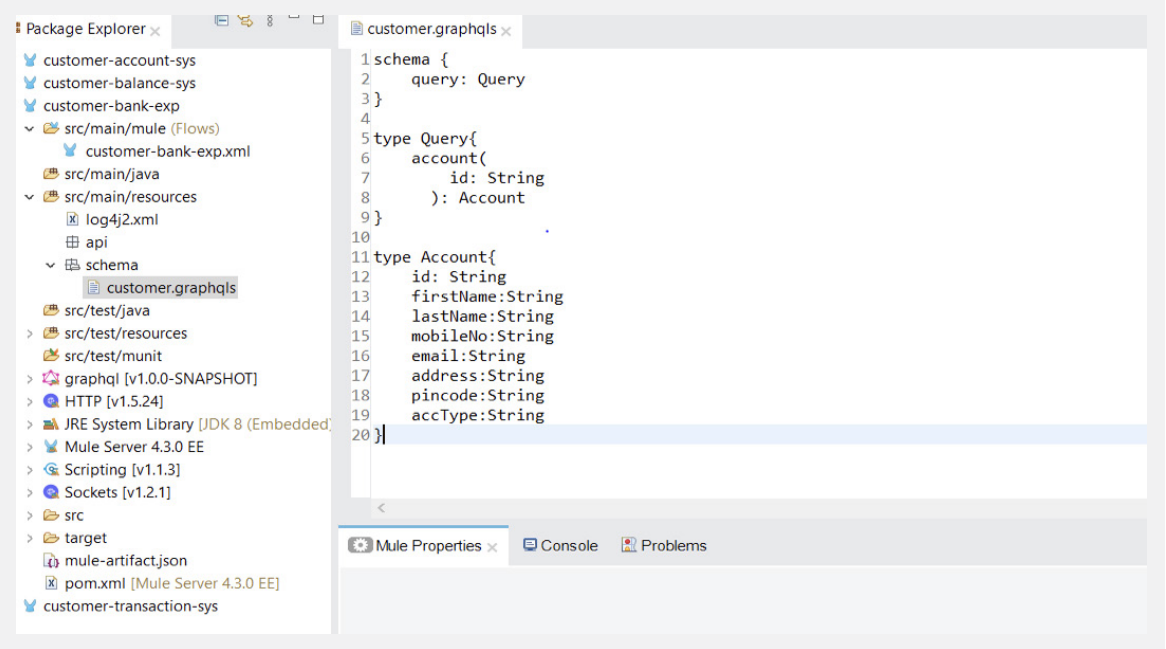




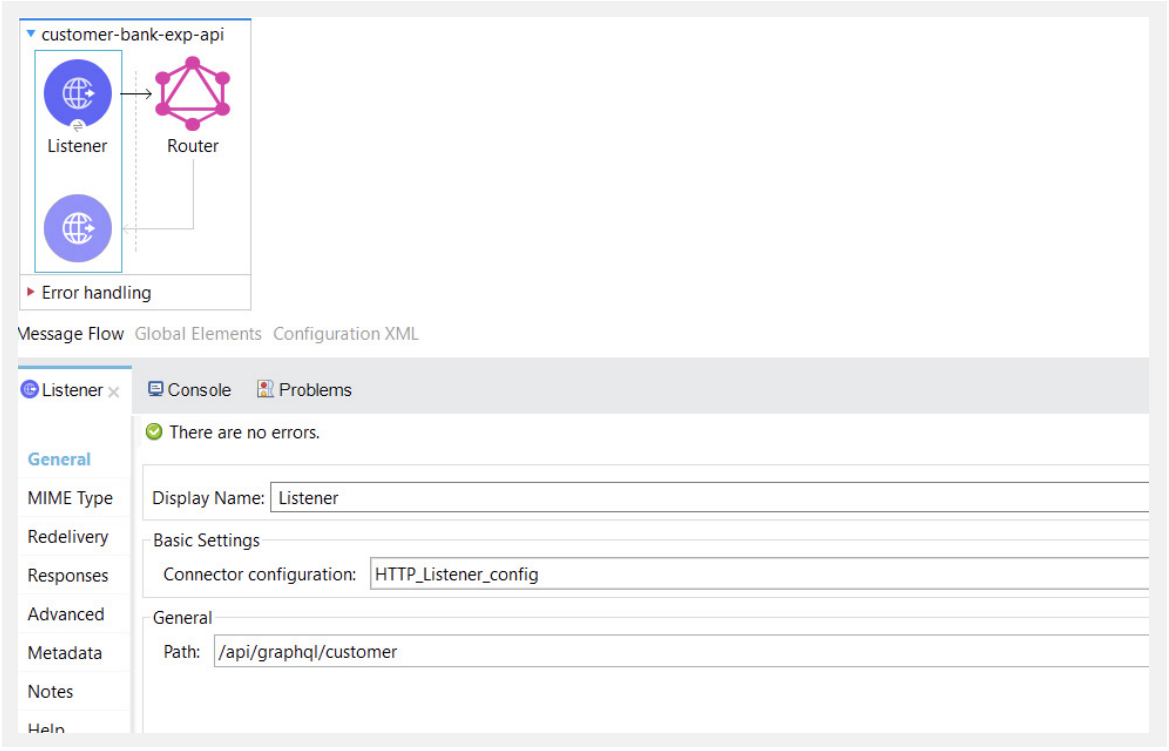
→ **Step 3:**
Now we will create a customer-bank-exp API.

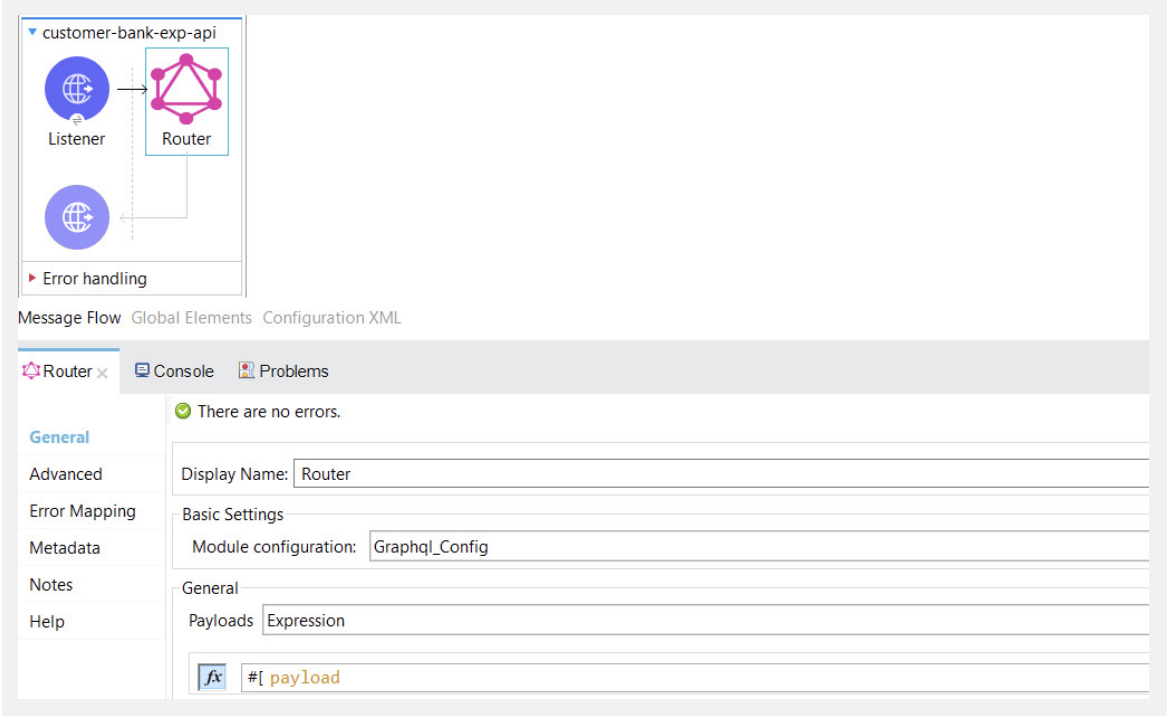
→ **Step 4:**
Go to /src/main/resource, create a new folder and name it schema.

→ **Step 5:**
Right click on the schema Folder under /src/main/resources New → File.

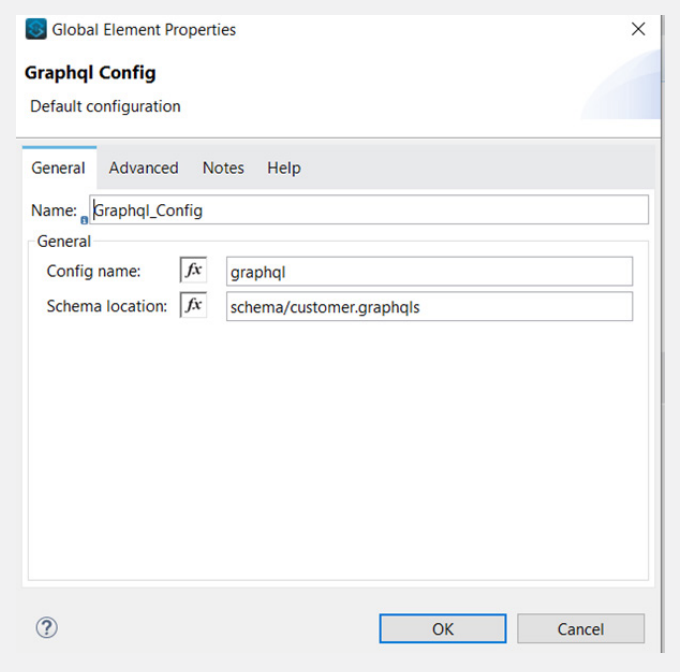


→ **Step 6:**
Create customer-bank-exp-api flow as shown below.

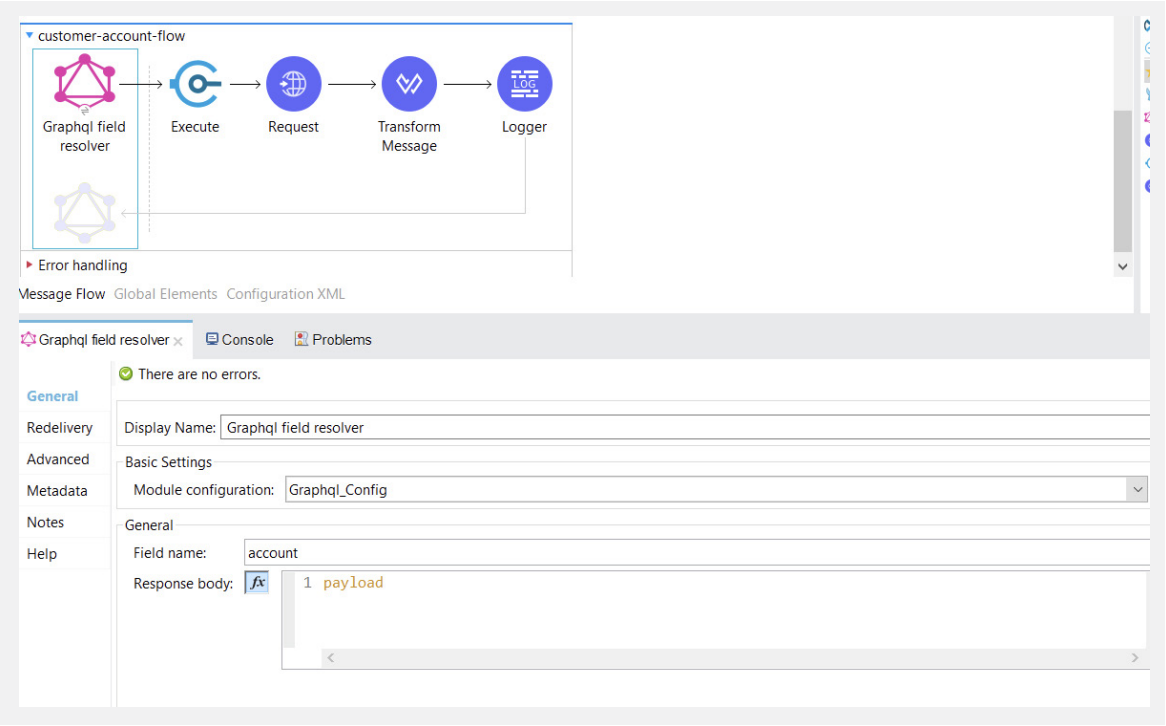




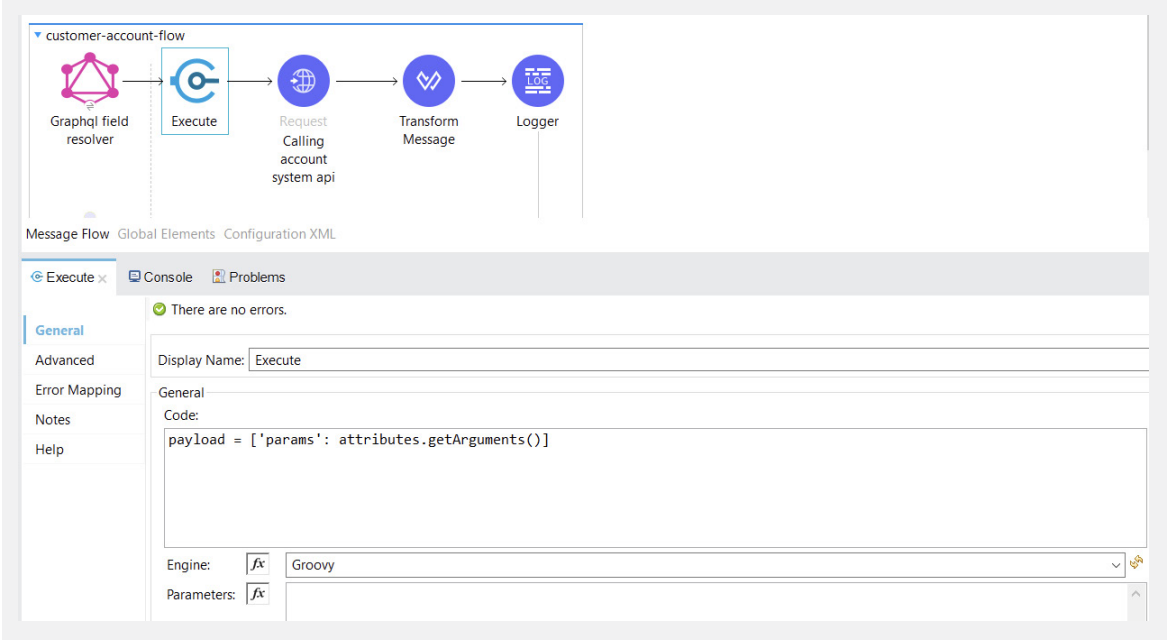
→ **Step 7:**
Configure GraphQL in global configuration as shown below.



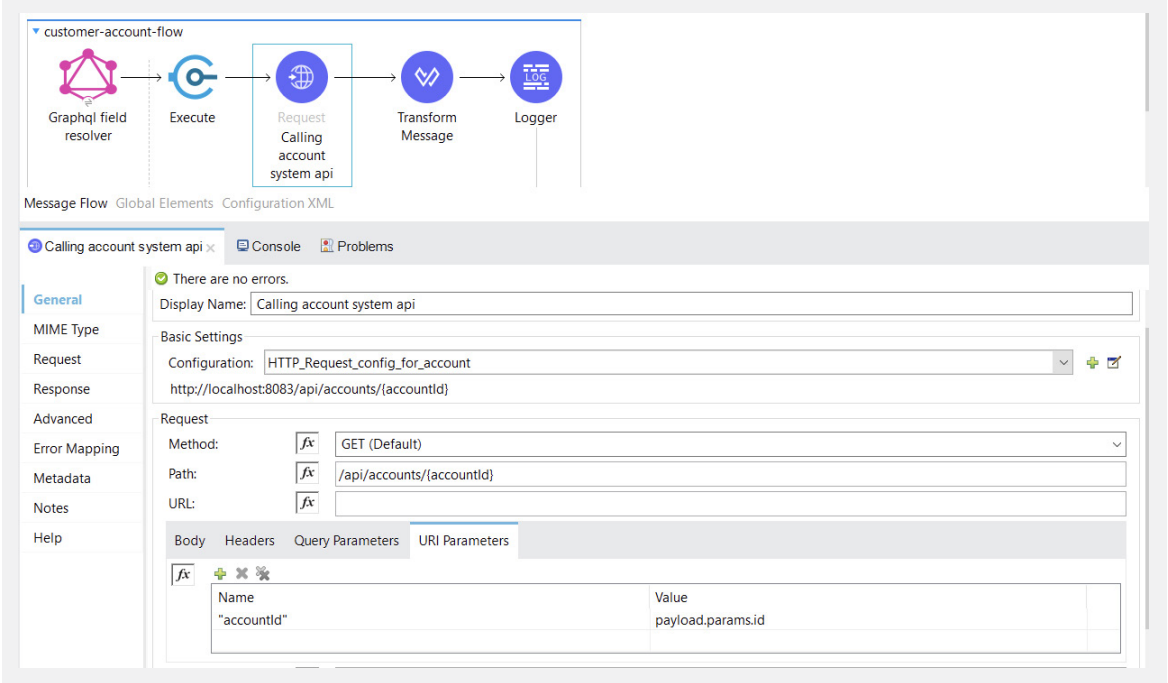
→ **Step 8:**
Create customer account flow as show below and configure GraphQL field resolver with the field name: account.

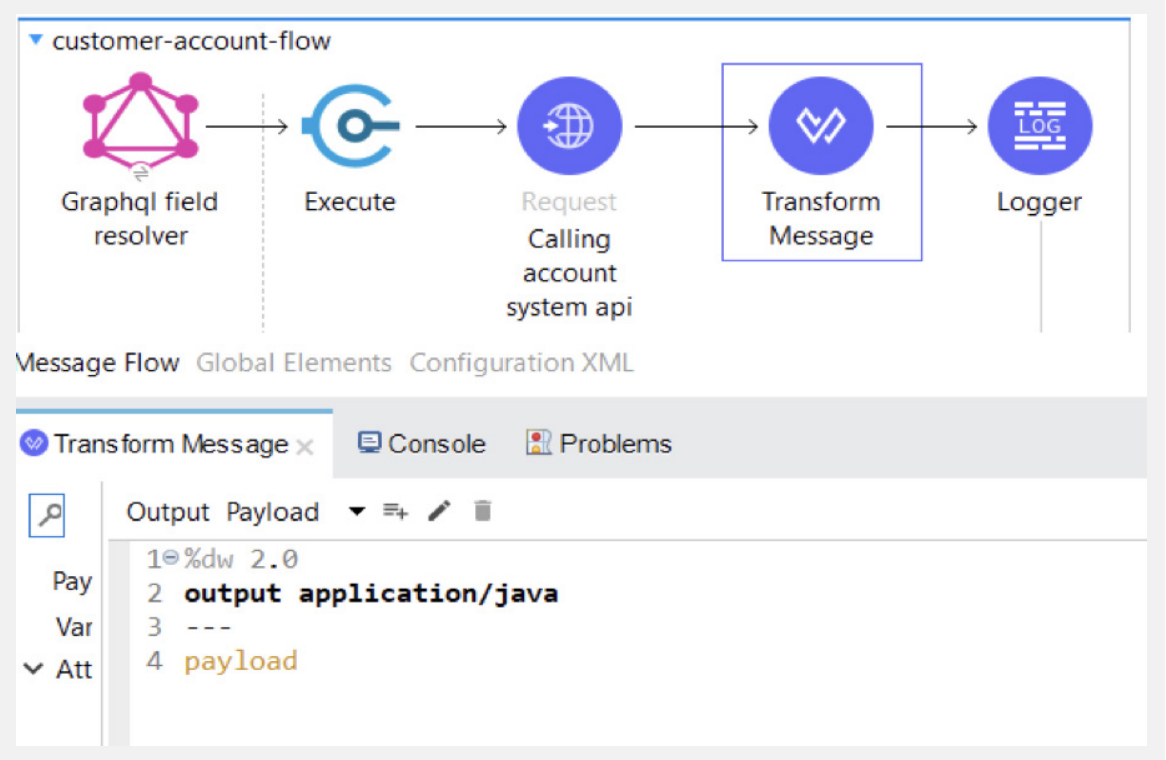


→ **Step 9:**
Configure the execute component. If you are not able to find the Groovy engine inside the connector then change the scripting version 1.1.3 in the pom.xml file.

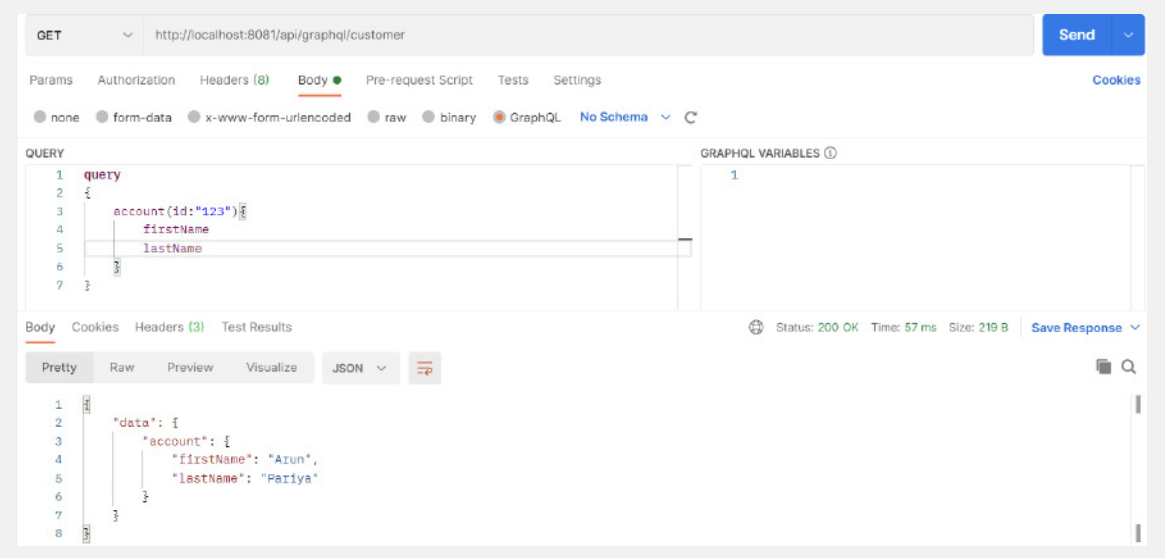


→ **Step 10:**
Configure the request component.

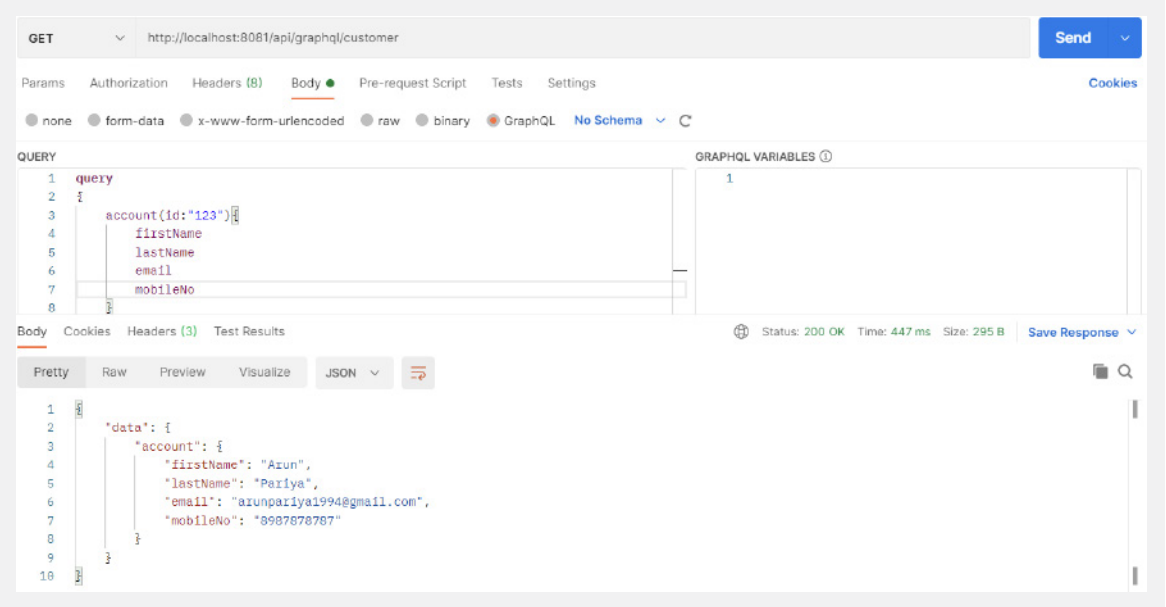




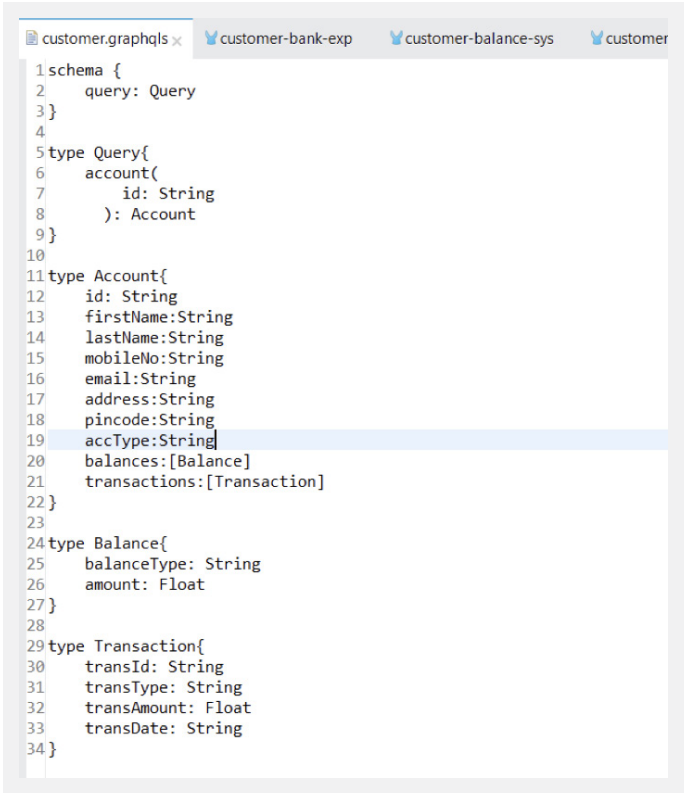
For Use Case 1:
Go to the postman and hit graphql endpoint. Select body → GraphQL.



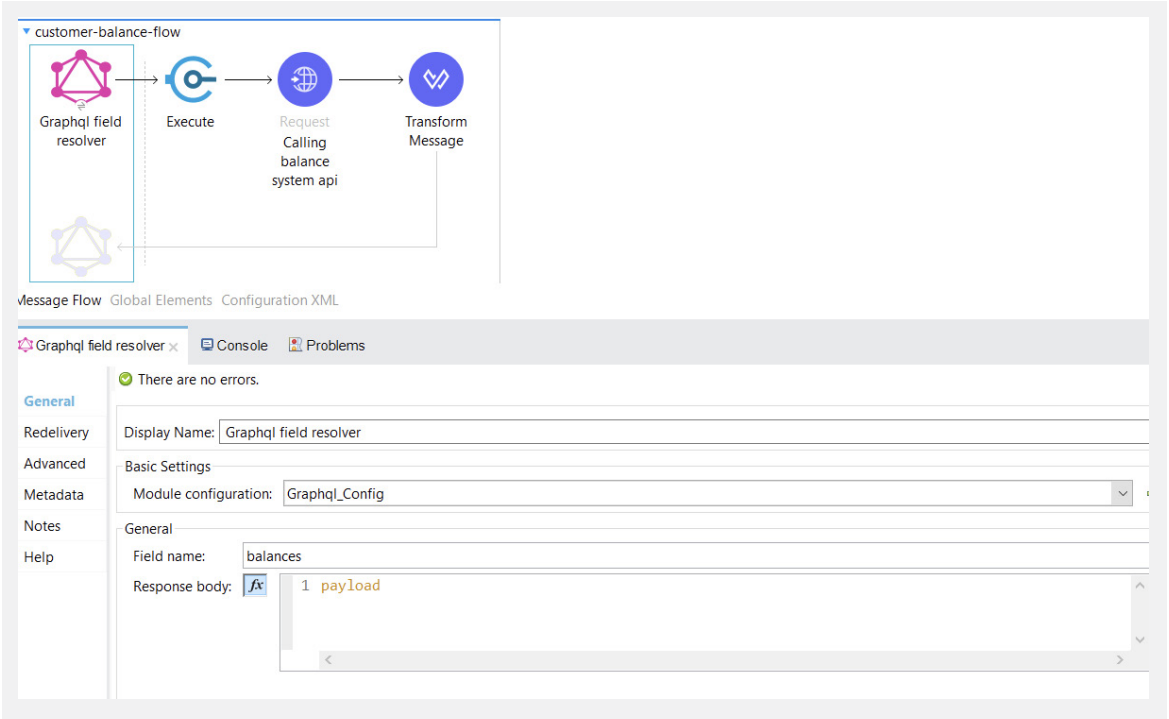
Now Change the fields according to the requirement.



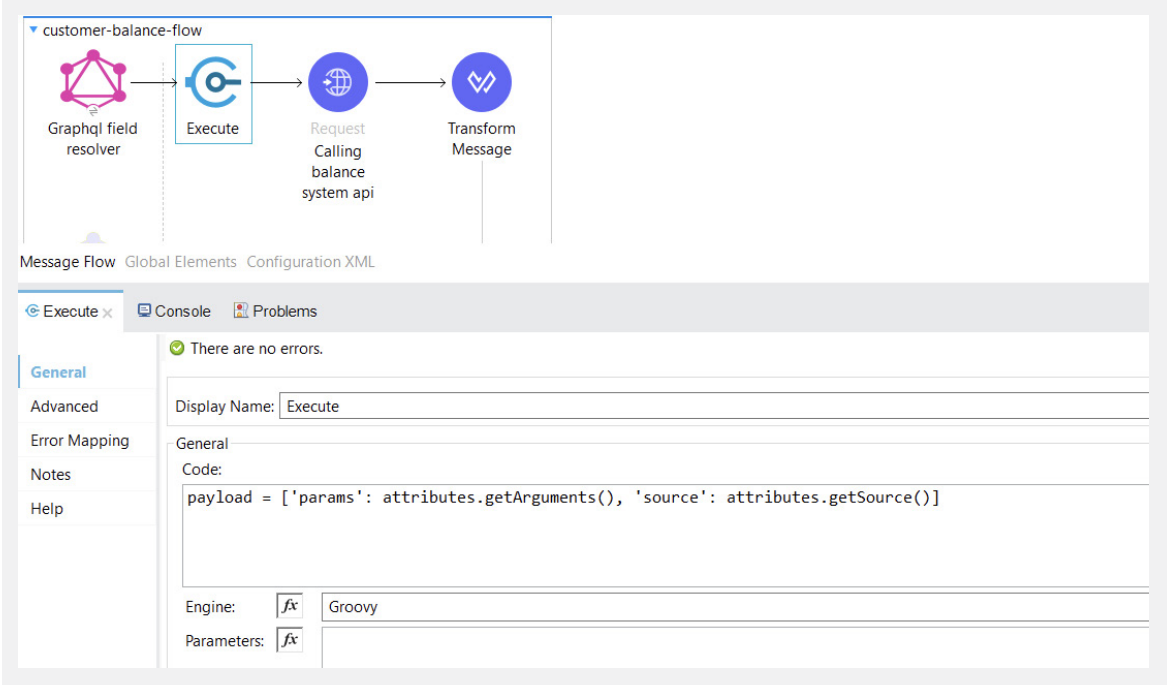
For Use Case 2:
Add the fields inside the schema under src/main/resources/schema folder.



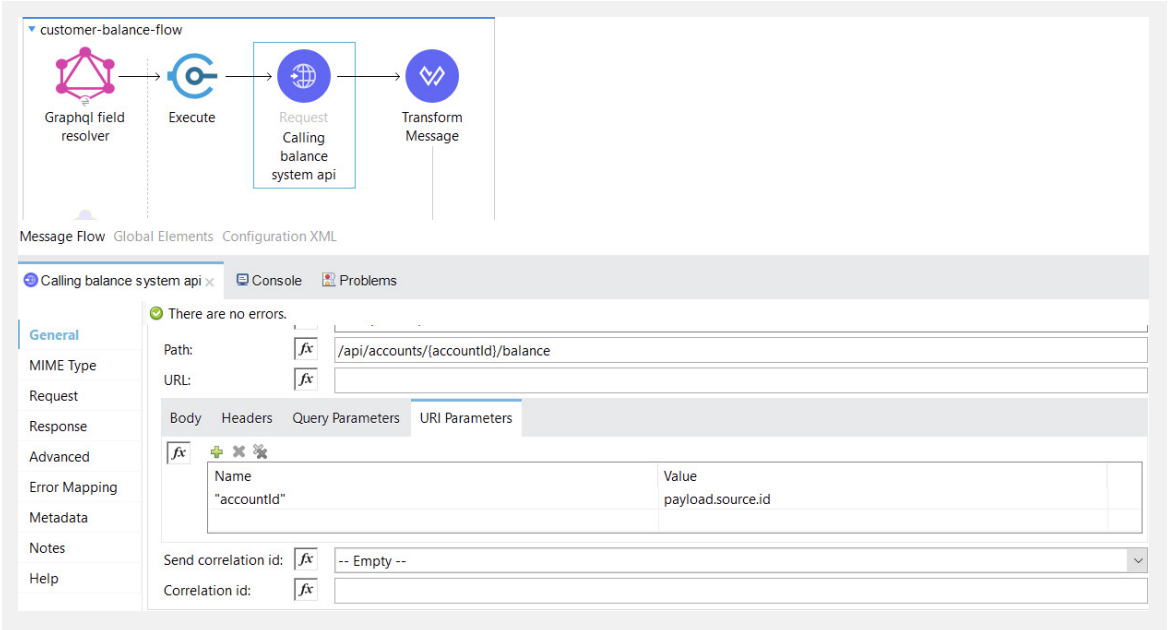
- Add the balance flow as shown below.



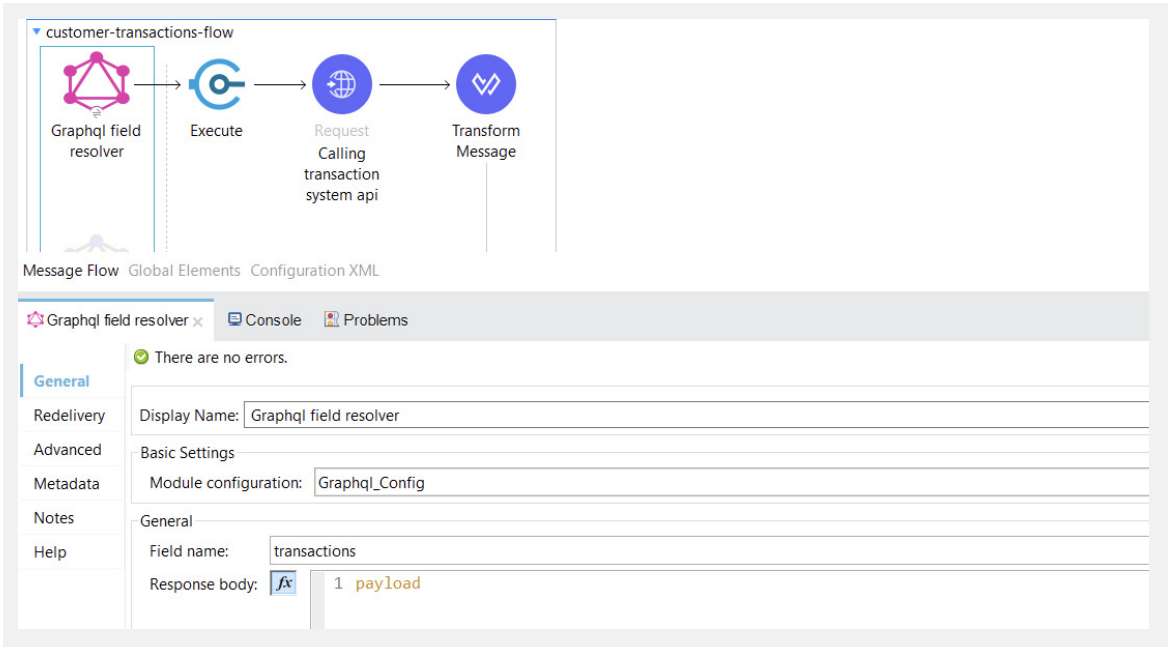
- Add the following script.



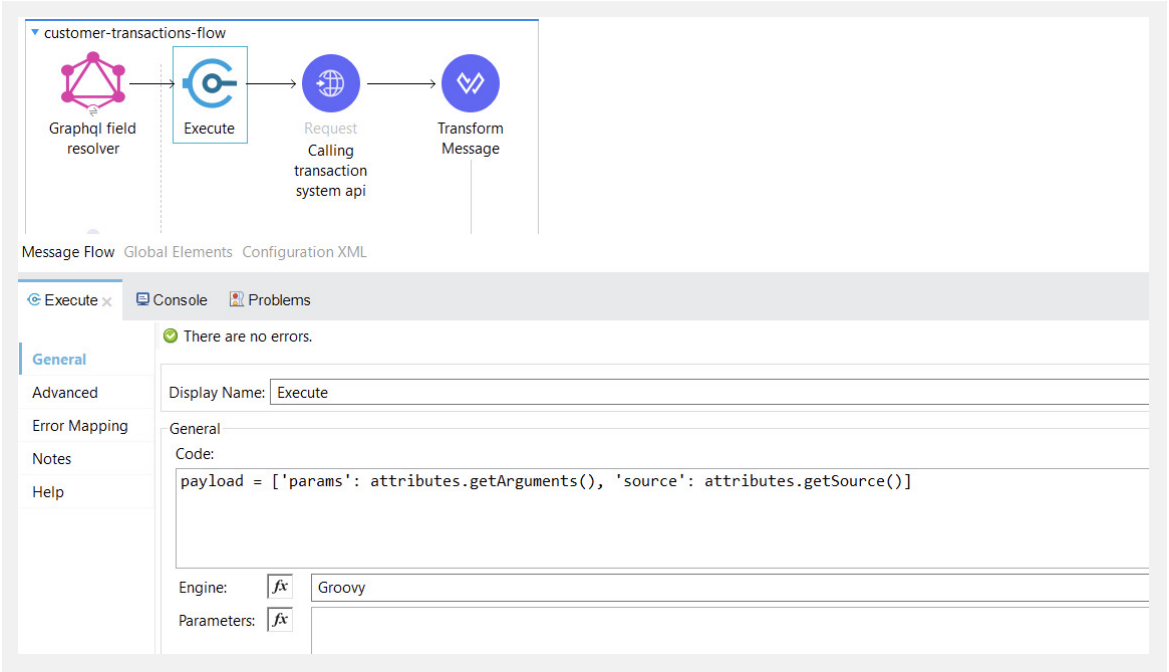
- Configure the request component as below.



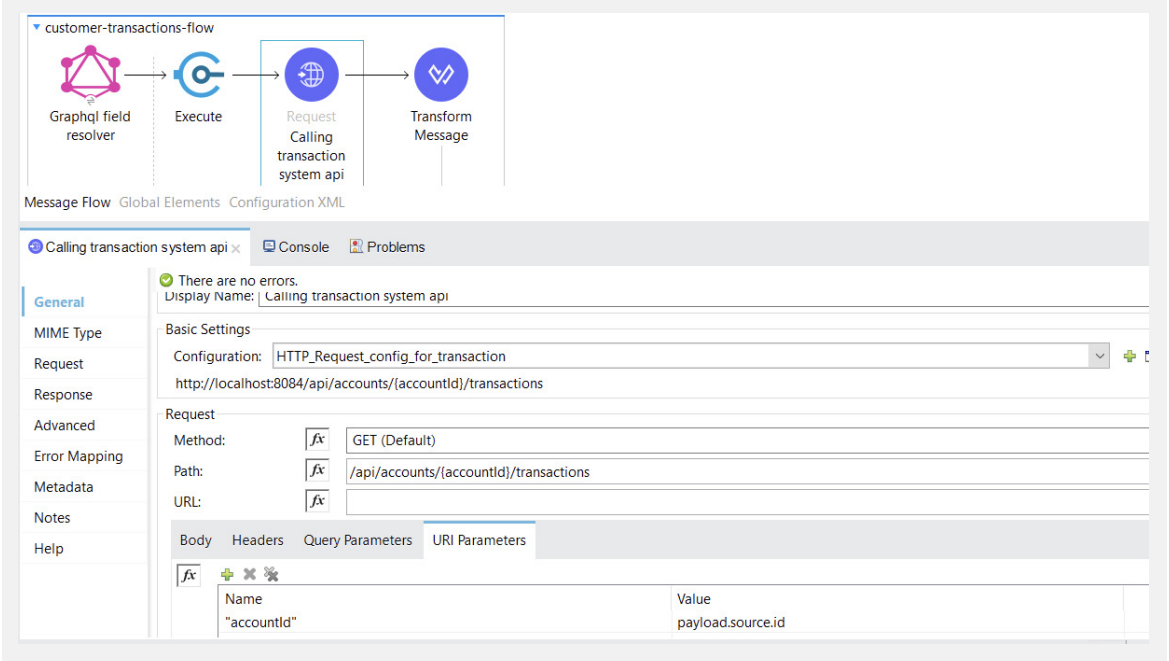
- Add the transaction flow as show below.



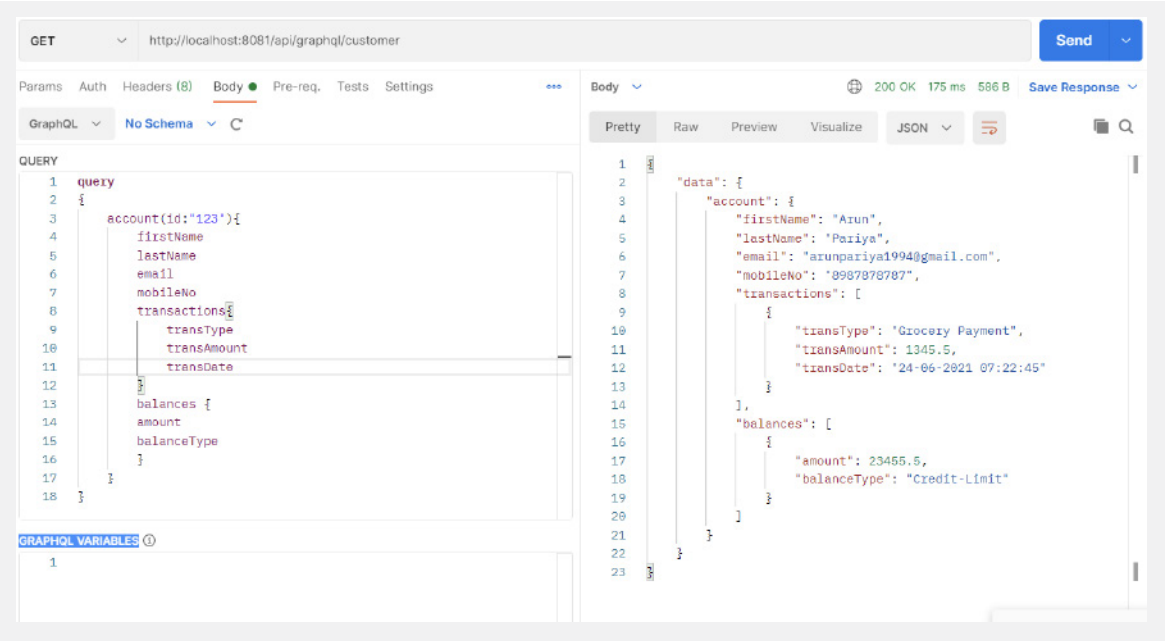
- Add the following script as shown below.



- Configure the request component as shown below.



- Go to the postman and hit graphql endpoint.



This API source code is available ➔ **here**.

References:

GraphQL: https://graphql.org/
Mulesoft: https://anypoint.mulesoft.com/datagraph

Summary:

GraphQL queries are fast and stable because they control the data and get exactly what you need, nothing more, nothing less. GraphQL queries always return predictable results.