TGH
Making Integrations Simpler

boomi Partner

Fetching Specific Fields from Database Table

Author
Vishnu R

Fetching Specific Fields from Database Table

In this blog, let us discuss how to capture and fetch those fields that are coming in the request profile from a Database table.

- ➢ We would be exposing an API that would be getting a request that contains some fields.
- ➢ Here, to capture the fields from the request we would be using scripting.
- ➢ For fetching the records from the database, we have used a connector Database V2 which allows a query to be passed dynamically.

First, let us see the payload that would be the request:

```
{
  "requestCount": 2,
  "requests":[
    {
      "studentId": "abc",
      "fields": ["emailId","percentage","grade","firstName","lastName","studentId","branch","YearOfJoining"]
    },
    {
      "studentId": "xyz",
      "fields": ["emailId","percentage","firstName"]
    }
  ]
}
```

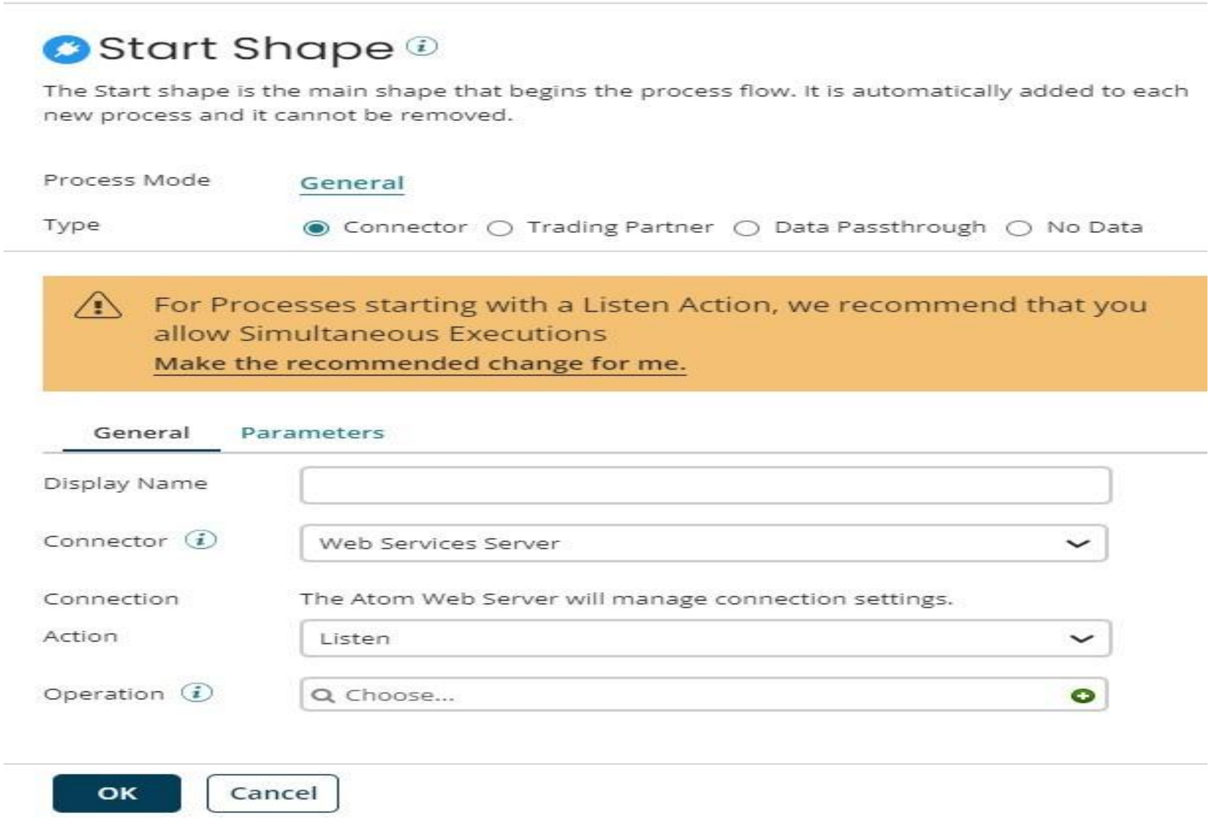
From this payload, we need to fetch the records according to the Student ID from the database and only those fields should be coming in the output which is present in the fields.

Sample Response:

```
{
   "Result": [
      {
         "emailId": "ad.2@gmail.com",
         "percentage": "90",
         "grade": "A",
         "firstName": "Adam",
         "lastName": "Levine",
         "studentId": "abc",
         "branch": "Science",
         "YearOfJoining": "2019"
      },
      {
         "emailId": "rick.77@gmail.com",
         "percentage": "71",
         "firstName": "Rick"
      }
   ]
}
```

Now, let us see the steps to achieve this.

Step 1: First, create a process with a start shape with the Web Services Server connector.



Start Shape

The Start shape is the main shape that begins the process flow. It is automatically added to each new process and it cannot be removed.

Process Mode: General

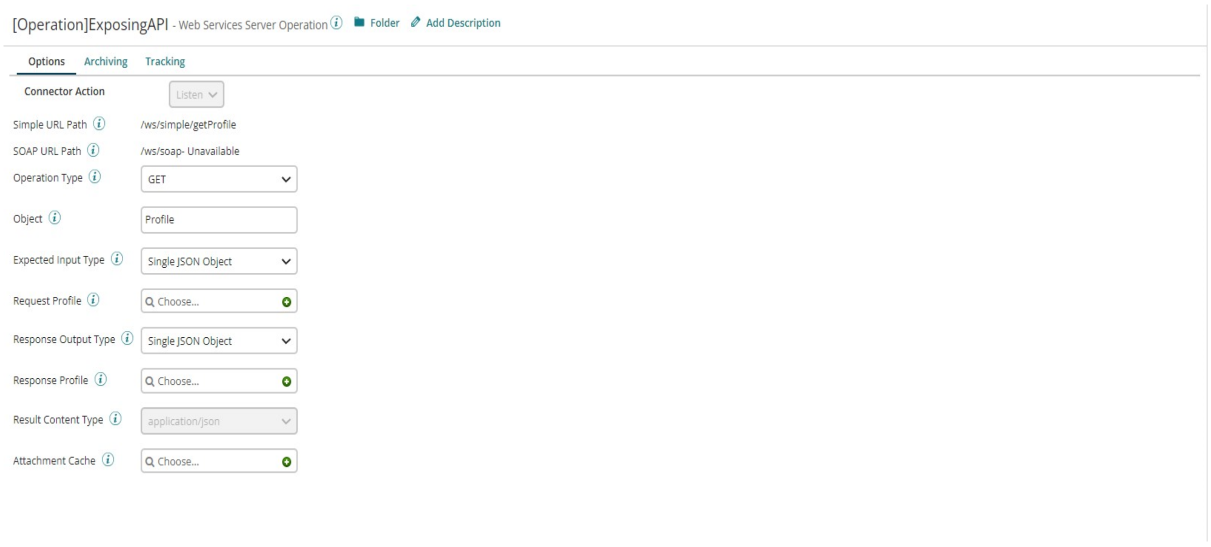
Type: [✓ Connector] Trading Partner Data Passthrough No Data

For Processes starting with a Listen Action, we recommend that you allow Simultaneous Executions
Make the recommended change for me.

Display Name	
Connector	Web Services Server
Connection	The Atom Web Server will manage connection settings.
Action	Listen
Operation	Choose...

Step 2: Configure the Operation related to the Web Service Server Connector.

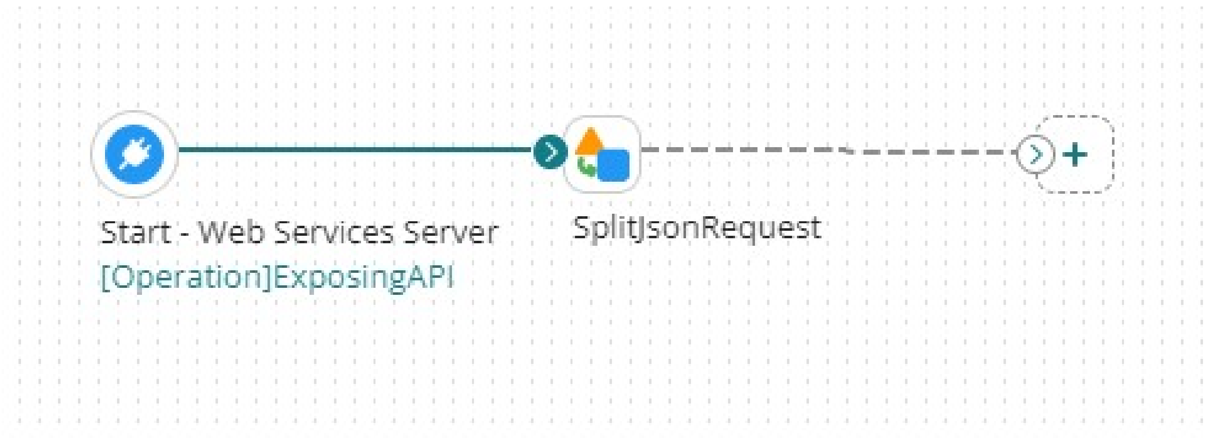
Note: Here the request profile and Response Profile are optional.



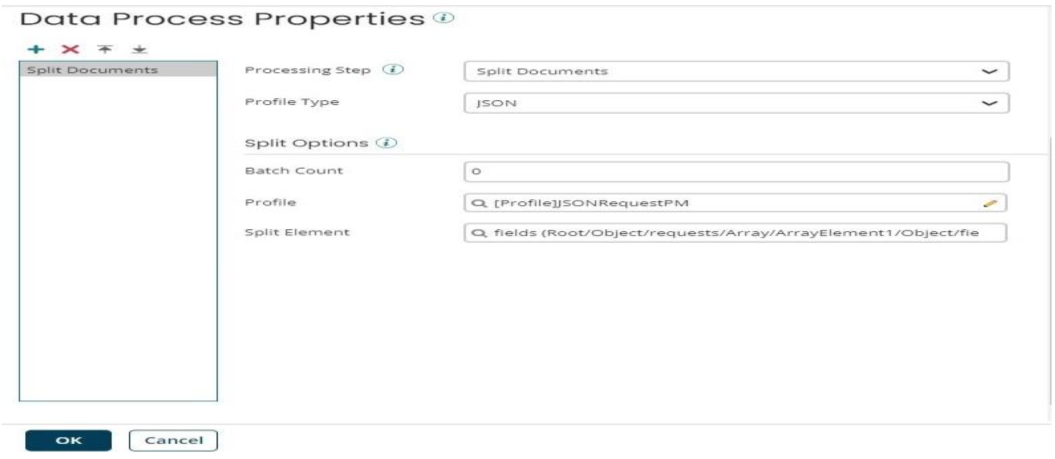
[Operation]ExposingAPI - Web Services Server Operation

Connector Action	Listen
Simple URL Path	/ws/simple/getProfile
SOAP URL Path	/ws/soap- Unavailable
Operation Type	GET
Object	Profile
Expected Input Type	Single JSON Object
Request Profile	Choose...
Response Output Type	Single JSON Object
Response Profile	Choose...
Result Content Type	application/json
Attachment Cache	Choose...

Step 3: Now, we need to drag and drop a Data Process shape for splitting the incoming request.



Step 4: We need to configure the Data Process for Splitting, first we have to select the profile type as JSON.



Data Process Properties

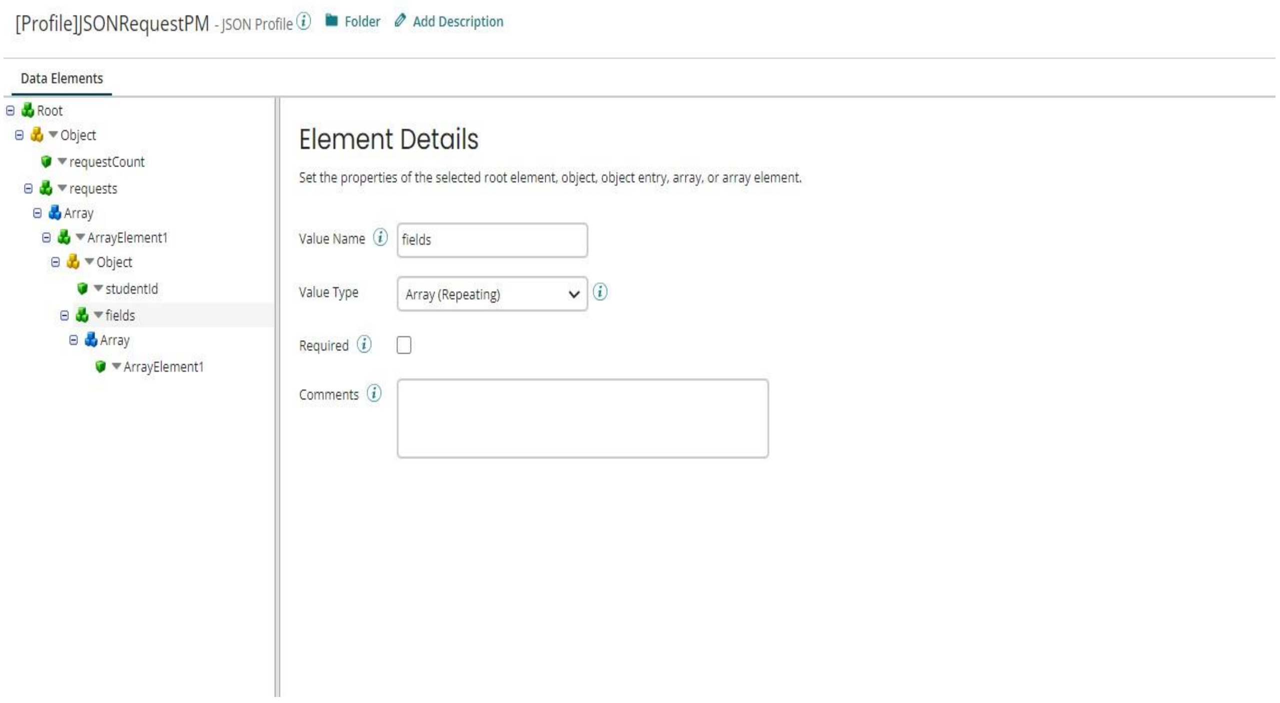
Split Documents

Processing Step	Split Documents
Profile Type	JSON

Split Options

Batch Count	0
Profile	[Profile]JSONRequestPM
Split Element	fields (Root/Object/requests/Array/ArrayElement1/Object/fie

Step 5: We need to select the JSON Profile that needs to be split.


[Profile]JSONRequestPM - JSON Profile Folder Add Description

Data Elements

- Root
  - Object
    - requestCount
    - requests
      - Array
        - ArrayElement1
          - Object
            - studentId
            - fields
              - Array
                - ArrayElement1

Element Details

Set the properties of the selected root element, object, object entry, array, or array element.

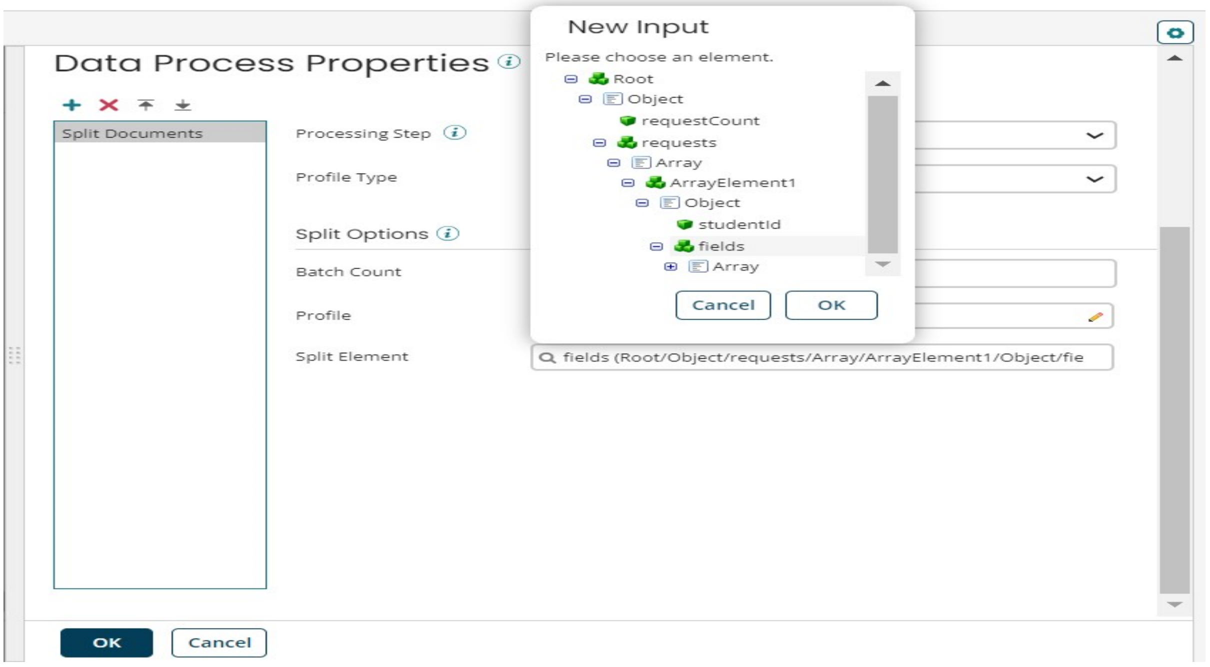
Value Name: fields

Value Type: Array (Repeating)

Required: ☐

Comments:

Step 6: Now, we need to select the split element with which you need to split and that element should be repeating.


Data Process Properties

Split Documents

Processing Step

Profile Type

Split Options

Batch Count

Profile

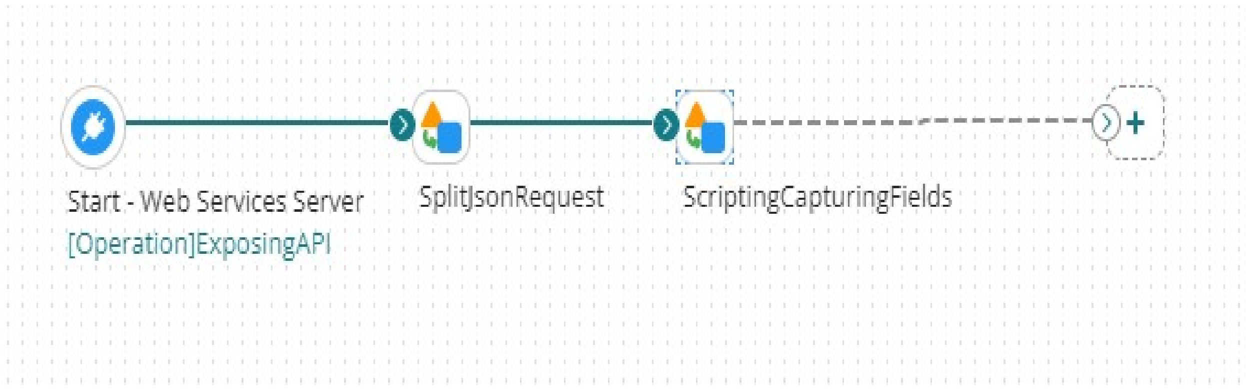
Split Element: fields (Root/Object/requests/Array/ArrayElement1/Object/fie

New Input

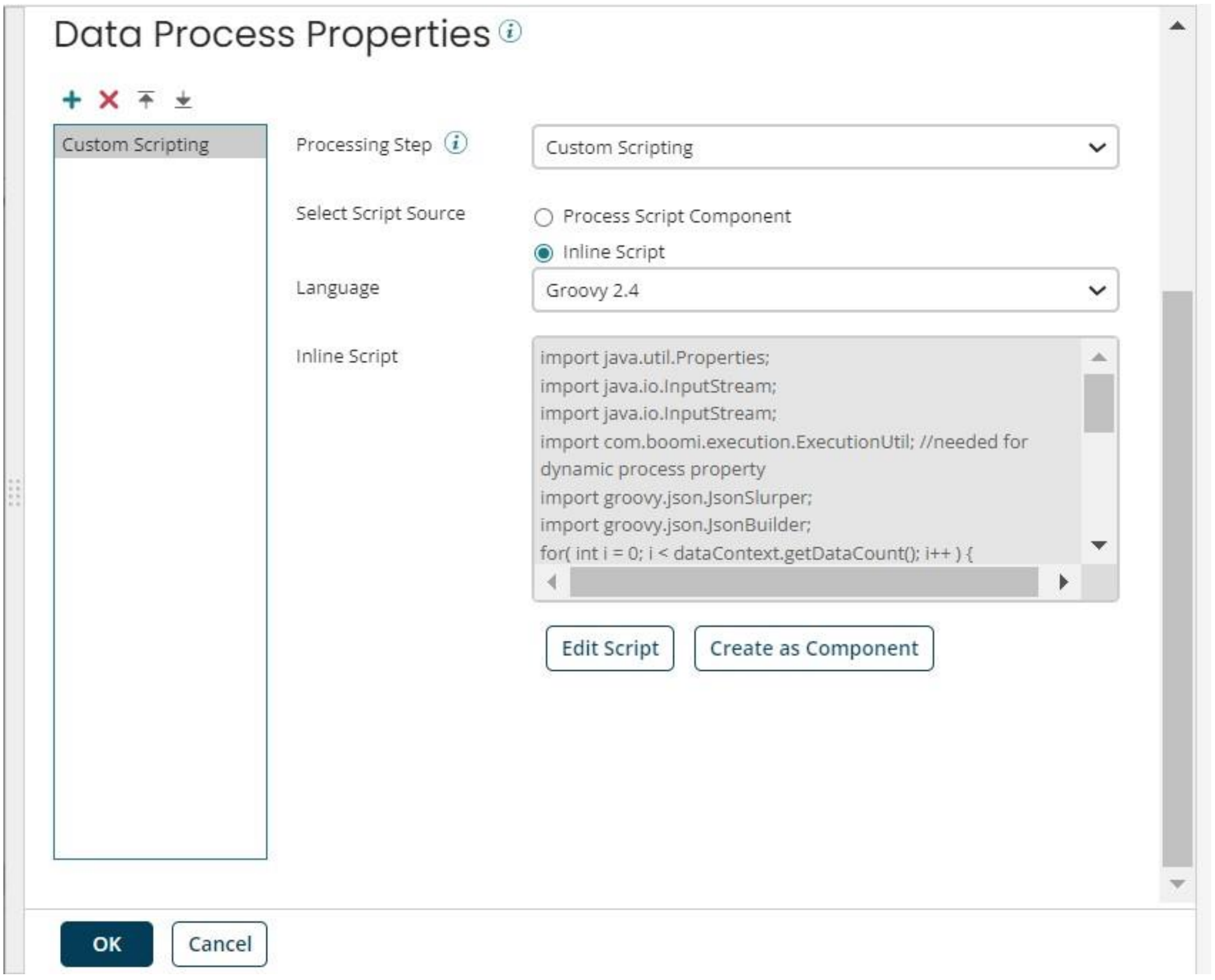
Please choose an element.

- Root
  - Object
    - requestCount
    - requests
      - Array
        - ArrayElement1
          - Object
            - studentId
            - fields
              - Array

Step 7: Now, we need to drag and drop the Data Process Shape for scripting.



Step 8: In Data Process, you need to choose custom scripting with inline script or process script component if you need it to be reused.



Data Process Properties

Custom Scripting

Processing Step: Custom Scripting

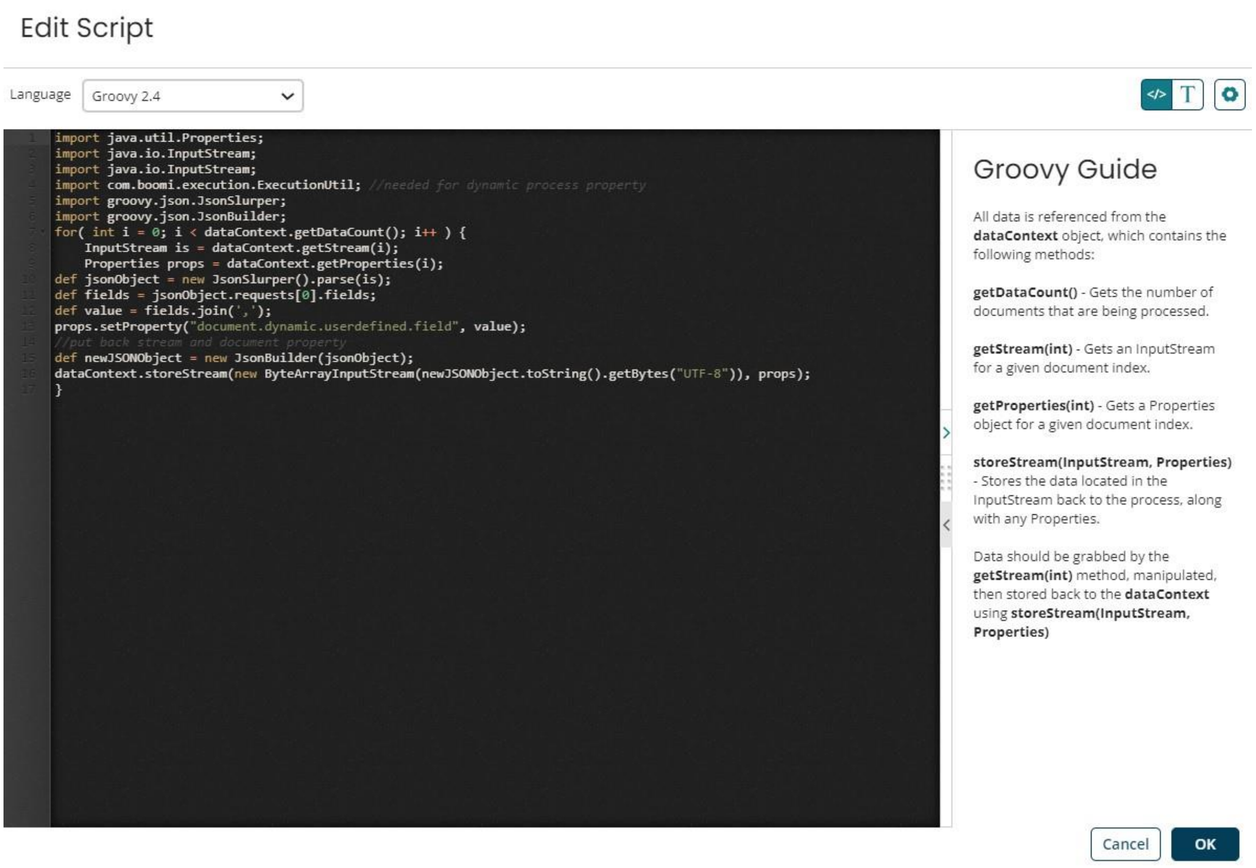
Select Script Source: Process Script Component [✓ Inline Script]

Language: Groovy 2.4

Inline Script:

```
import java.util.Properties;
import java.io.InputStream;
import java.io.InputStream;
import com.boomi.execution.ExecutionUtil; //needed for
dynamic process property
import groovy.json.JsonSlurper;
import groovy.json.JsonBuilder;
for( int i = 0; i < dataContext.getDataCount(); i++ ) {
```

Step 9: The script that should be written is:


Edit Script

Language: Groovy 2.4

```
import java.util.Properties;
import java.io.InputStream;
import java.io.InputStream;
import com.boomi.execution.ExecutionUtil; //needed for dynamic process property
import groovy.json.JsonSlurper;
import groovy.json.JsonBuilder;
for( int i = 0; i < dataContext.getDataCount(); i++ ) {
    InputStream is = dataContext.getStream(i);
    Properties props = dataContext.getProperties(i);
def jsonObject = new JsonSlurper().parse(is);
def fields = jsonObject.requests[0].fields;
def value = fields.join(',');
props.setProperty("document.dynamic.userdefined.field", value);
//put back stream and document property
def newJSONObject = new JsonBuilder(jsonObject);
dataContext.storeStream(new ByteArrayInputStream(newJSONObject.toString().getBytes("UTF-8")), props);
}
```

Groovy Guide

All data is referenced from the **dataContext** object, which contains the following methods:

getDataCount() - Gets the number of documents that are being processed.

getStream(int) - Gets an InputStream for a given document index.

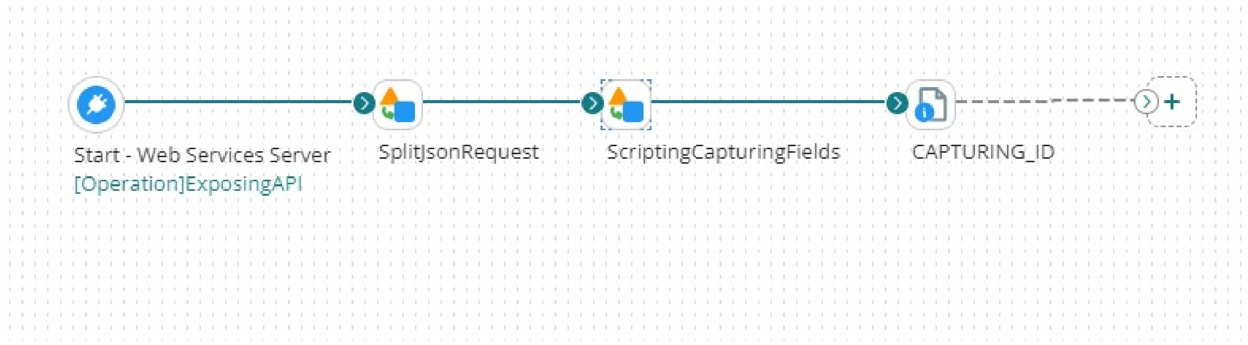
getProperties(int) - Gets a Properties object for a given document index.

storeStream(InputStream, Properties) - Stores the data located in the InputStream back to the process, along with any Properties.

Data should be grabbed by the **getStream(int)** method, manipulated, then stored back to the **dataContext** using **storeStream(InputStream, Properties)**

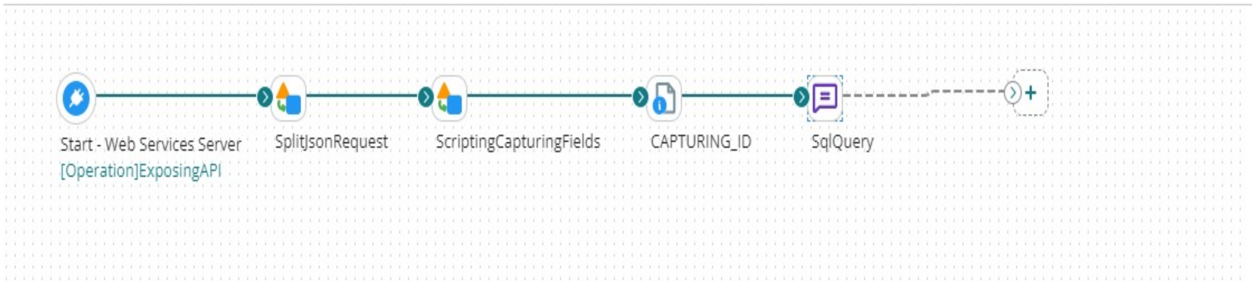
- Here, in the script we are using a JSON Slurper for parsing the JSON and we will take the JSON Object, and using this JSON Object we would be capturing the Fields that are coming in the request.
- Once it is captured, we would separate each field with a comma and it is stored in dynamic document property named fields.

Step 10: Drag and drop a Set Properties for capturing the ID.

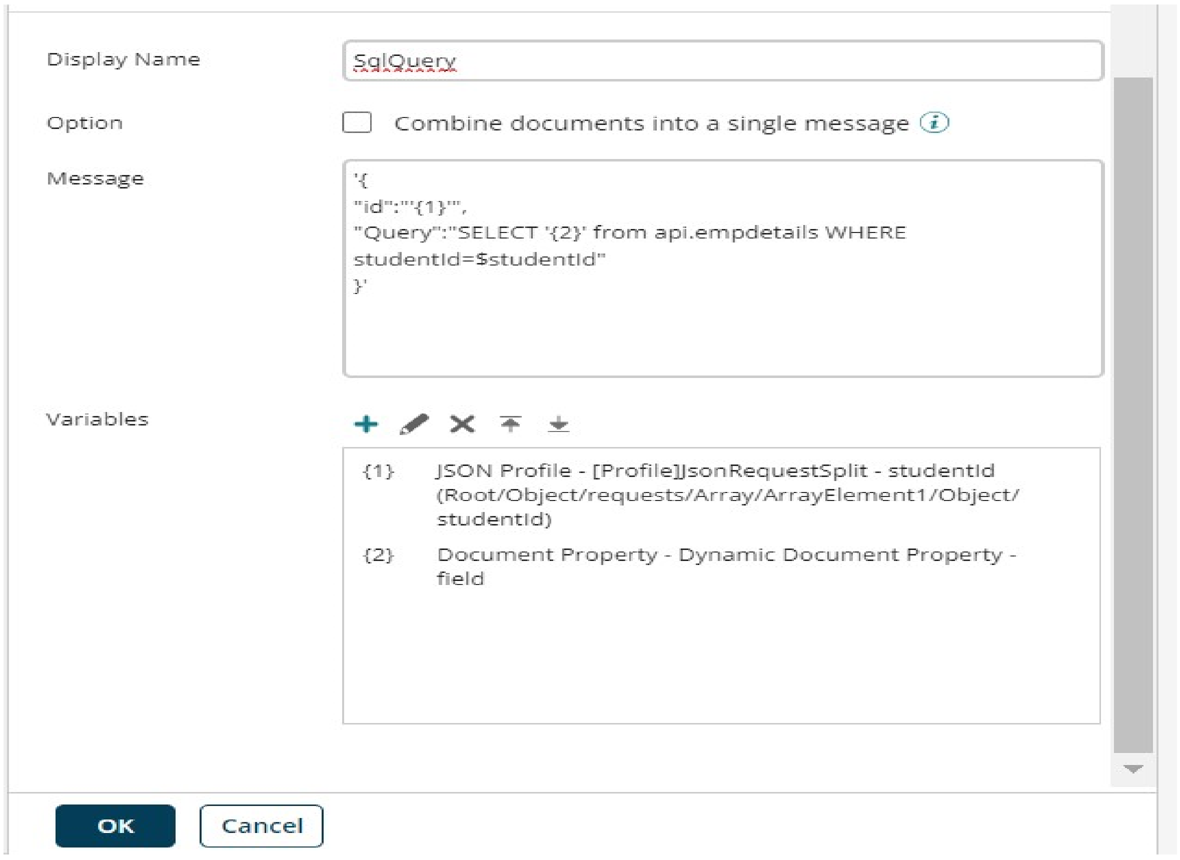


- You would be able to capture the ID with a dynamic document property from the output coming from the data process.

Step 11: Drag and drop a message shape for capturing the Dynamic Query which needs to be passed to the Database Connector.



Step 12: You need to use placeholders for passing the ID and SQL Query to the Database connector in a JSON Format.



Display Name: SqlQuery

Option: Combine documents into a single message

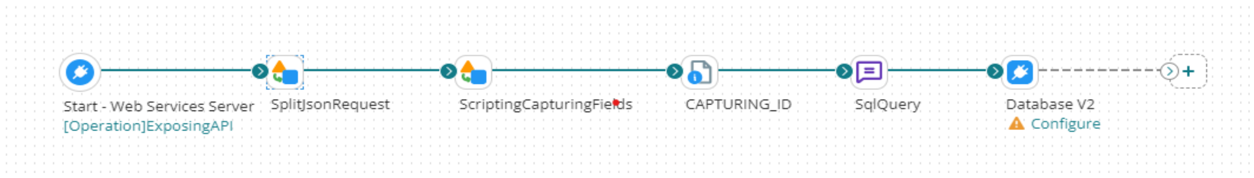
Message:

```
'{
"id":"'{1}'",
"Query":"SELECT '{2}' from api.empdetails WHERE
studentId=$studentId"
}'
```

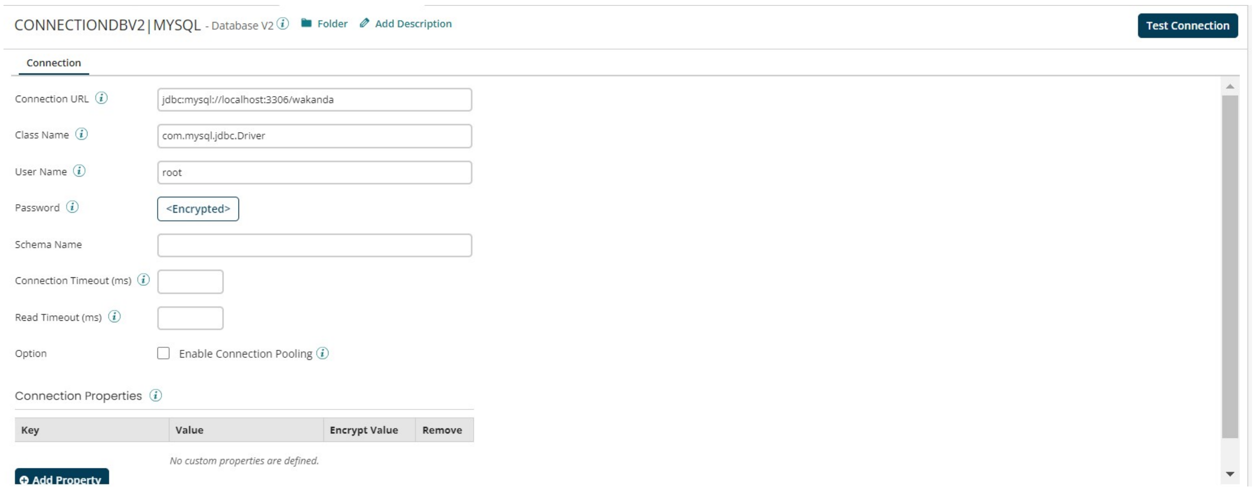
Variables:

{1}	JSON Profile - [Profile]JsonRequestSplit - studentId (Root/Object/requests/Array/ArrayElement1/Object/studentId)
{2}	Document Property - Dynamic Document Property - field

Step 13: We need to drag and drop a Database V2 connector for fetching the records from the Database.



Step 14: Now, we need to configure the connection related to Database V2 and the action as get.



Connection URL	jdbc:mysql://localhost:3306/wakanda
Class Name	com.mysql.jdbc.Driver
User Name	root
Password	<Encrypted>
Schema Name	
Connection Timeout (ms)	
Read Timeout (ms)	
Option	☐ Enable Connection Pooling

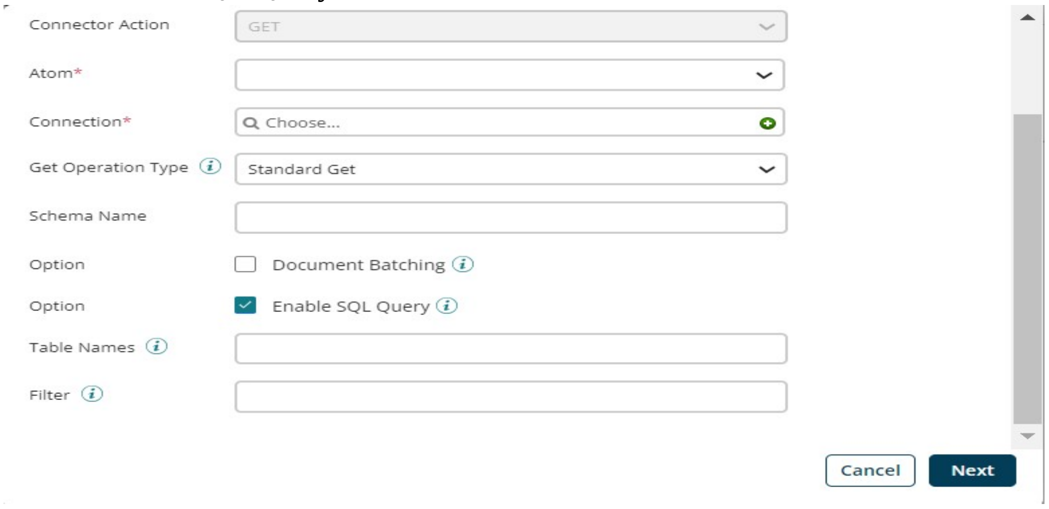
Connection Properties

Key	Value	Encrypt Value	Remove

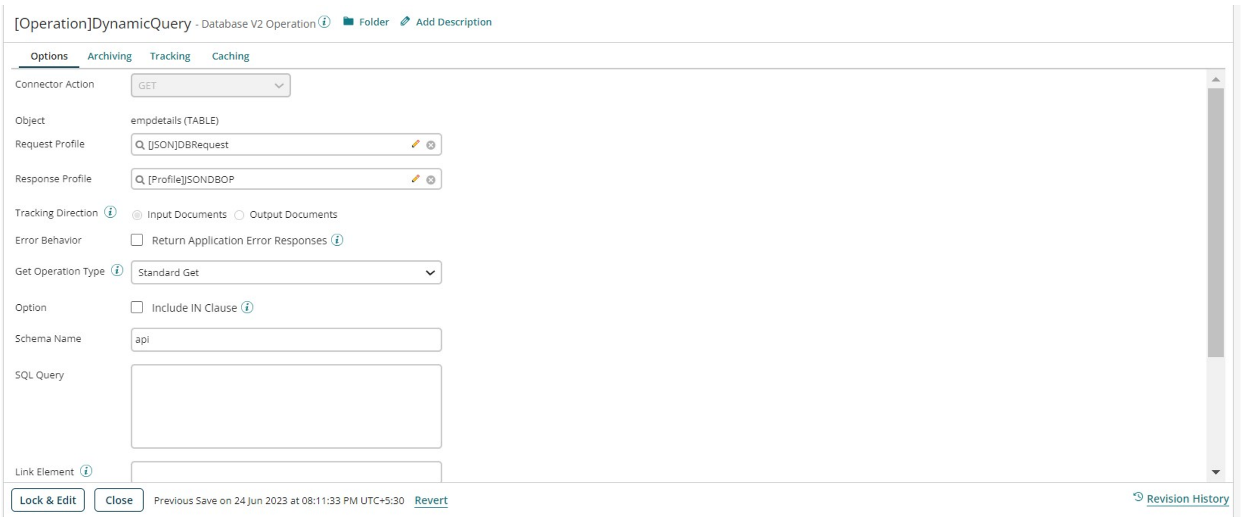
No custom properties are defined.

Step 15: Now, we need to configure the operation related to Database V2.

- First, we need to perform import, and while importing
  - ➢ The get operation type should be Standard Get.
  - ➢ Enable SQL Query should be checked.



Connector Action	GET
Atom*	
Connection*	Choose...
Get Operation Type	Standard Get
Schema Name	
Option	☐ Document Batching
Option	☑ Enable SQL Query
Table Names	
Filter	

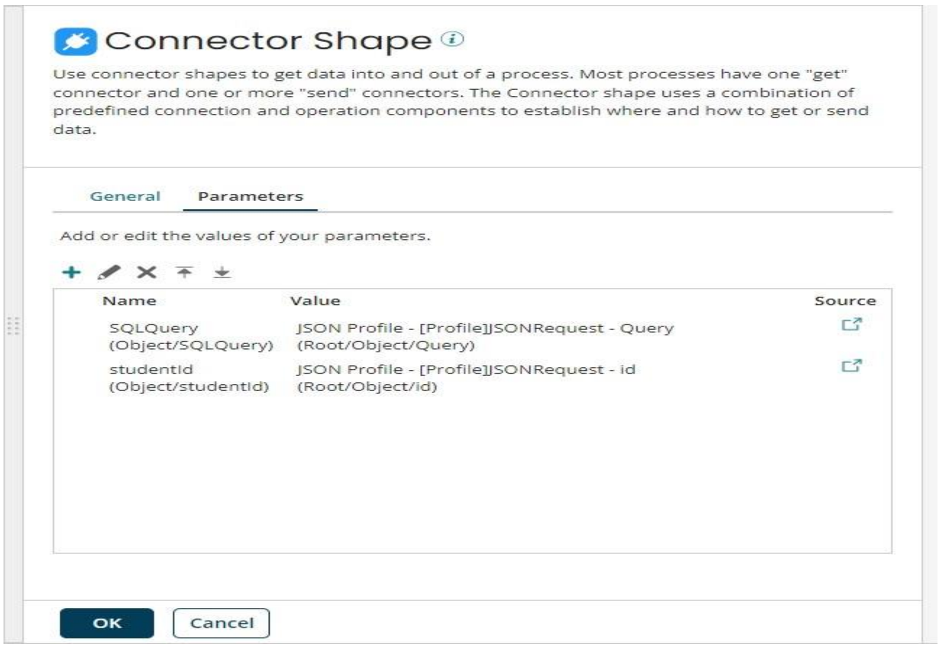


[Operation]DynamicQuery - Database V2 Operation

Connector Action	GET
Object	empdetails (TABLE)
Request Profile	[JSON]DBRequest
Response Profile	[Profile]JSONDBOP
Tracking Direction	● Input Documents ○ Output Documents
Error Behavior	☐ Return Application Error Responses
Get Operation Type	Standard Get
Option	☐ Include IN Clause
Schema Name	api
SQL Query	
Link Element	

Previous Save on 24 Jun 2023 at 08:11:33 PM UTC+5:30

Step 16: Now, we need to Add 2 parameters in the Database V2 Connector.



Connector Shape

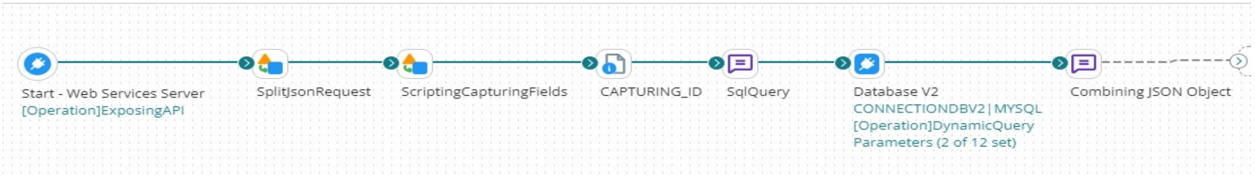
Use connector shapes to get data into and out of a process. Most processes have one "get" connector and one or more "send" connectors. The Connector shape uses a combination of predefined connection and operation components to establish where and how to get or send data.

Add or edit the values of your parameters.

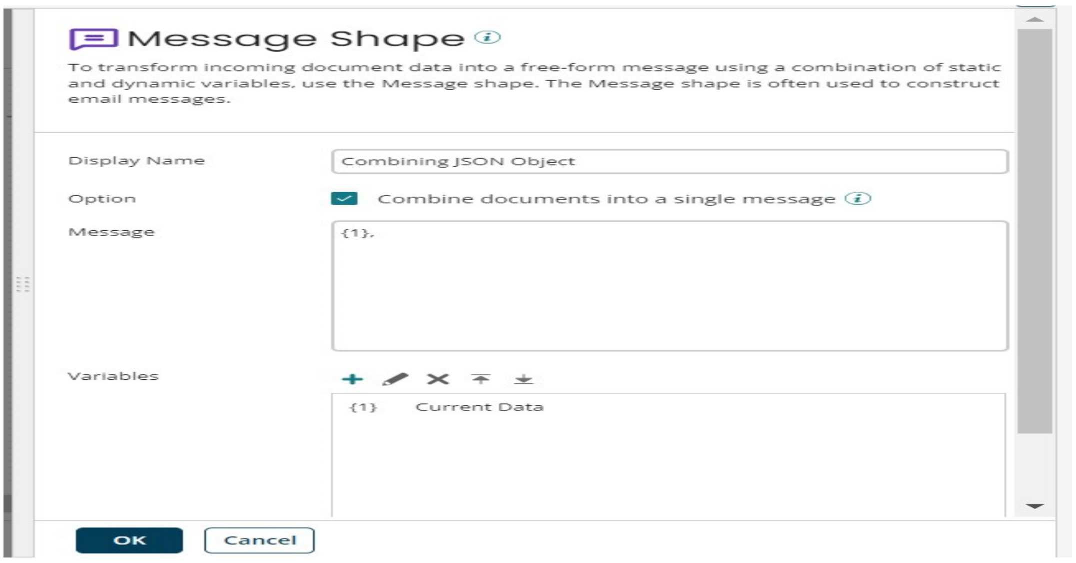
Name	Value	Source
SQLQuery (Object/SQLQuery)	JSON Profile - [Profile]JSONRequest - Query (Root/Object/Query)	
studentId (Object/studentId)	JSON Profile - [Profile]JSONRequest - id (Root/Object/id)	

- The first parameter would be the SQL Query and the Second parameter would be the Student ID based on which we need to fetch the records.

Step 17: We need to drag and drop a Message shape for combining the JSON Records that are coming as the output of the Database V2 connector.



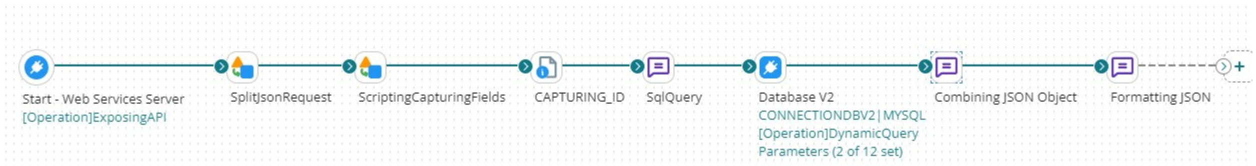
Step 18: Inside the message shape we need to add a placeholder and a comma, combine documents into a single message checkbox should be ticked.


Message Shape

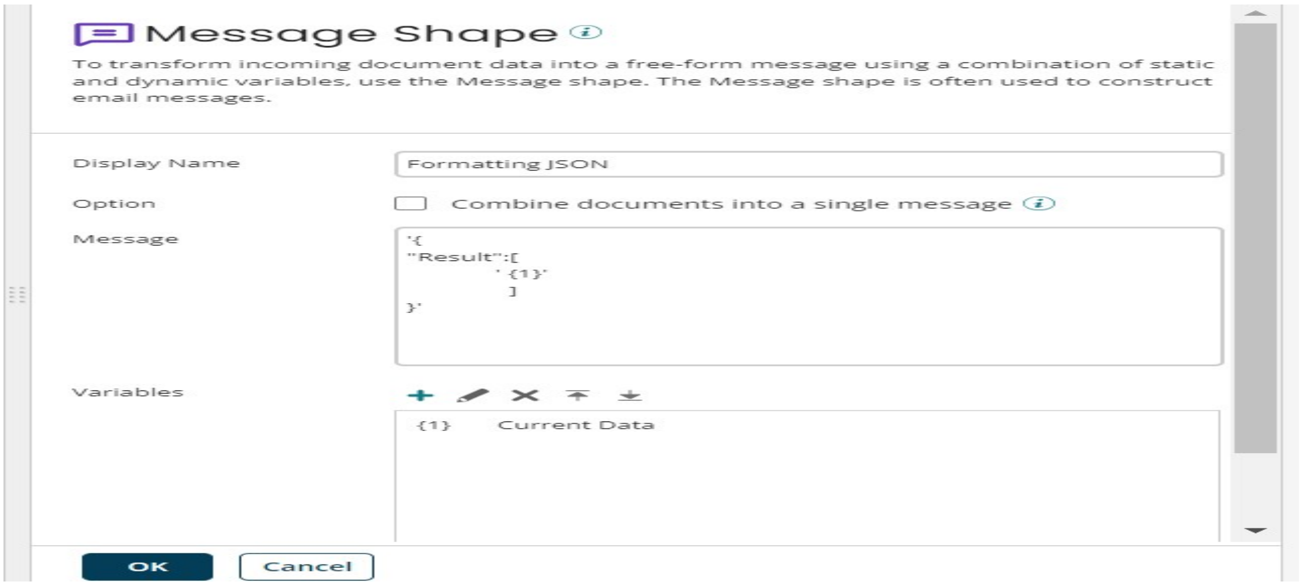
To transform incoming document data into a free-form message using a combination of static and dynamic variables, use the Message shape. The Message shape is often used to construct email messages.

Display Name	Combining JSON Object
Option	[✓] Combine documents into a single message
Message	{1},
Variables	{1} Current Data

Step 19: Drag and drop another message shape to format it into a JSON.



Step 20: Drag and drop another message shape to format it into a JSON.


Message Shape

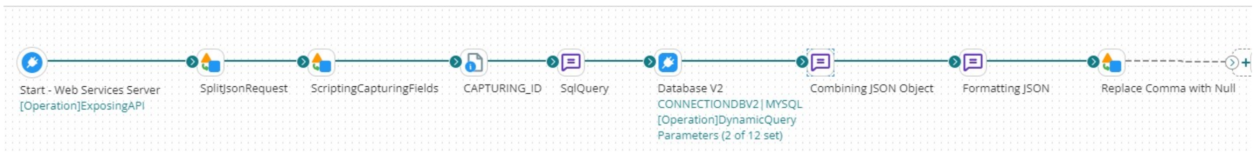
To transform incoming document data into a free-form message using a combination of static and dynamic variables, use the Message shape. The Message shape is often used to construct email messages.

Display Name	Formatting JSON
Option	[ ] Combine documents into a single message
Variables	{1} Current Data

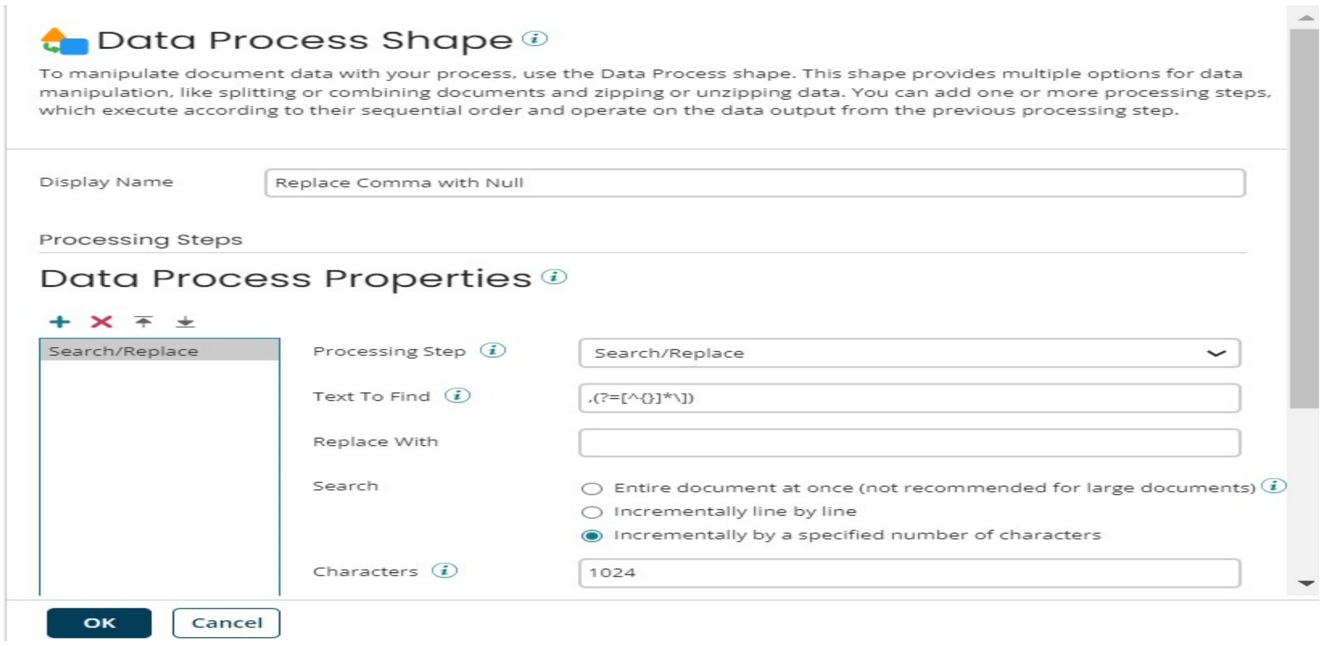
Message:

```
'{
"Result":[
           ' {1}'
            ]
}'
```

Step 21: Drag and drop a Data Process shape to make the payload a well-formed JSON.



Step 22: In the Data Process, you need to choose the search and replace option where you can specify,(?=[^{}]*\]) this regex in the text to find section and Null in the replace with section.



Data Process Shape

To manipulate document data with your process, use the Data Process shape. This shape provides multiple options for data manipulation, like splitting or combining documents and zipping or unzipping data. You can add one or more processing steps, which execute according to their sequential order and operate on the data output from the previous processing step.

Display Name: Replace Comma with Null

Processing Steps

Data Process Properties

Search/Replace

Processing Step: Search/Replace

Text To Find: ,(?=[^{}]*\])

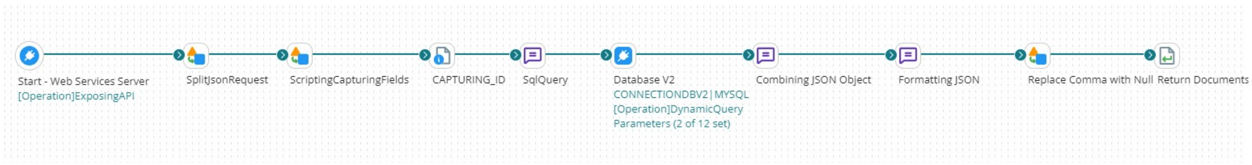
Replace With:

Search:
- ○ Entire document at once (not recommended for large documents)
- ○ Incrementally line by line
- ● Incrementally by a specified number of characters

Characters: 1024

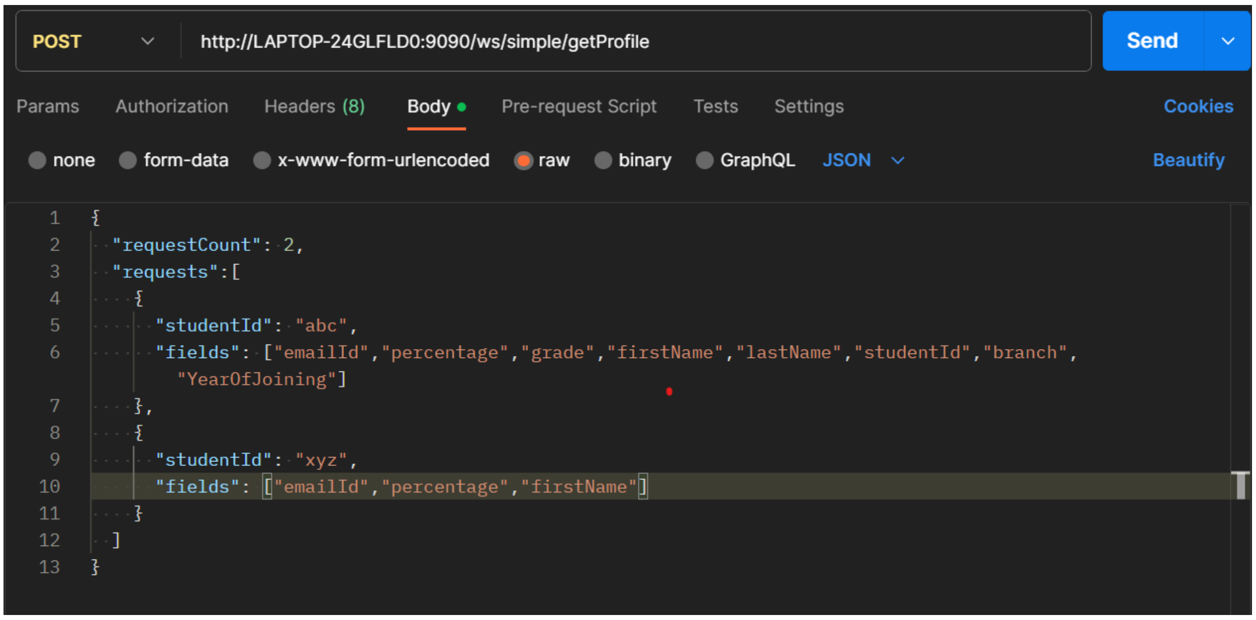
OK Cancel

Step 23: Finally, you need to add a Return document shape for sending the response back.



Step 24: Now, the process is done and you need to create a package component and deploy the process to an environment. Once deployed, you can test it using any testing tool like Postman by using a POST request with the input payload in the body.

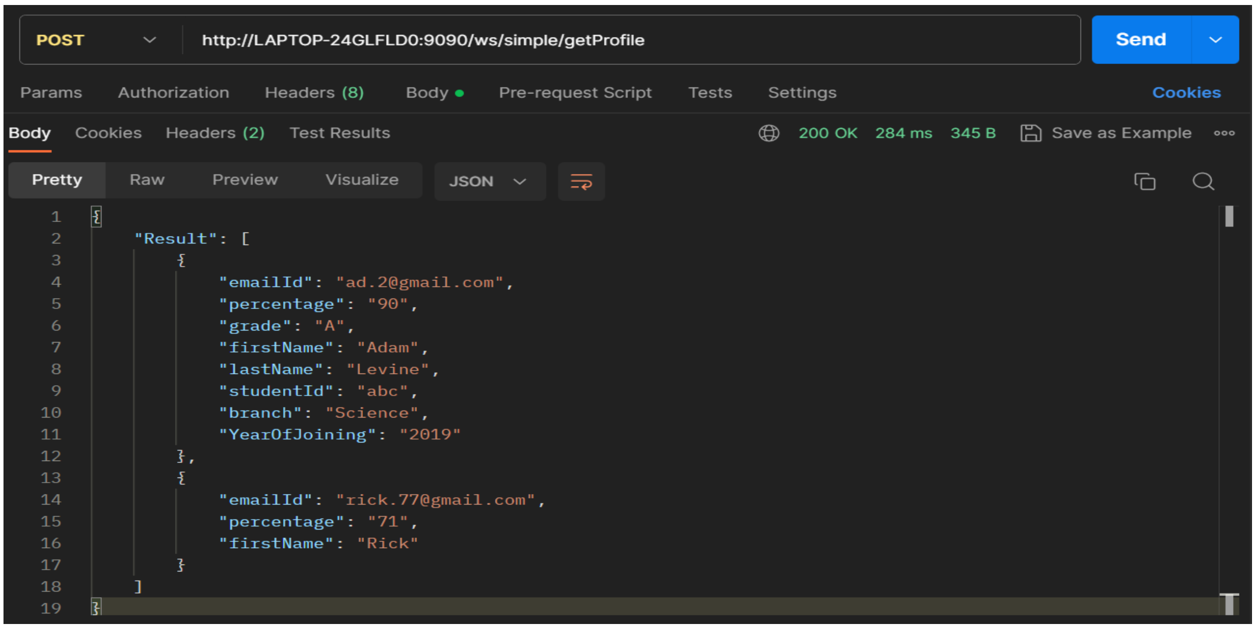
Request from Postman



http://LAPTOP-24GLFLD0:9090/ws/simple/getProfile

```
{
  "requestCount": 2,
  "requests":[
    {
      "studentId": "abc",
      "fields": ["emailId","percentage","grade","firstName","lastName","studentId","branch",
        "YearOfJoining"]
    },
    {
      "studentId": "xyz",
      "fields": ["emailId","percentage","firstName"]
    }
  ]
}
```

Response



http://LAPTOP-24GLFLD0:9090/ws/simple/getProfile

200 OK 284 ms 345 B

```
{
    "Result": [
        {
            "emailId": "ad.2@gmail.com",
            "percentage": "90",
            "grade": "A",
            "firstName": "Adam",
            "lastName": "Levine",
            "studentId": "abc",
            "branch": "Science",
            "YearOfJoining": "2019"
        },
        {
            "emailId": "rick.77@gmail.com",
            "percentage": "71",
            "firstName": "Rick"
        }
    ]
}
```

TGH

Making Integrations Simpler

boomi Partner

TGH Software Solutions Pvt. Ltd.

www.techygeekhub.com

At TGH, we specialize in driving digital transformation through seamless Integration Technologies.

Operating as an INTEGRATION FACTORY, we serve as a one-stop shop for all your integration needs. Our expert team is well-versed in enterprise software and legacy system integration, along with leading iPaaS technologies like Boomi, MuleSoft, Workato, OIC, and more.

We're committed to enhancing business processes and solving problems through our integration expertise.

Email address

connect@techygeekhub.com

Phone number

+ 011-40071137
+ 91-8810610395

Our offices

Noida Office
iThum
Plot No -40, Tower A,
Office No: 712,
Sector-62, Noida,
Uttar Pradesh, 201301

Hyderabad Office
Plot no: 6/3, 5th Floor,
Techno Pearl Building,
HUDA Techno Enclave,
HITEC City, Hyderabad,
Telangana 500081