SAP BTP Integration Suite

Contents

About:

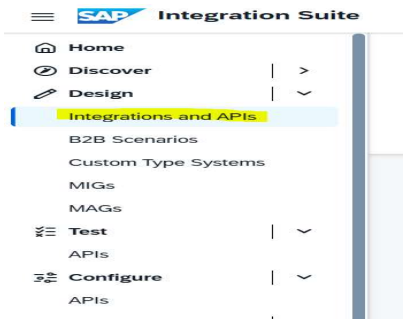
This document provides a step-by-step guide on creating iFlows using the SAP BTP Integration Suite. Due to confidentiality and security restrictions, details from the live project cannot be shared. Therefore, this Proof of Concept (POC) document demonstrates similar scenarios and examples for reference purposes.

Scenario 1: Encoder or Decoder

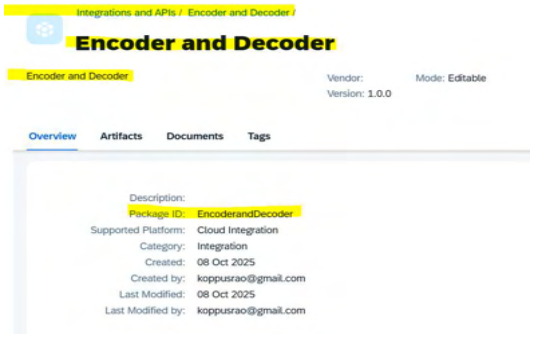
Description: Use the iFlow which will convert normal input to Encode format

Login integration suite

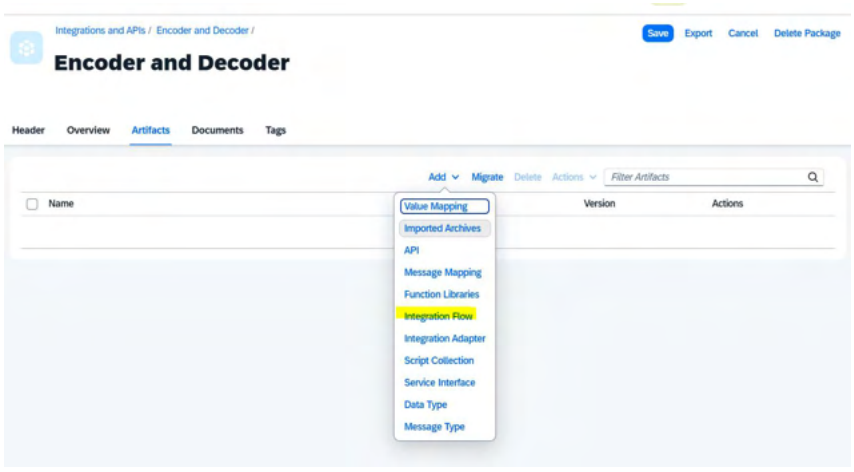
Select Design → Integration and APIs → Create Package

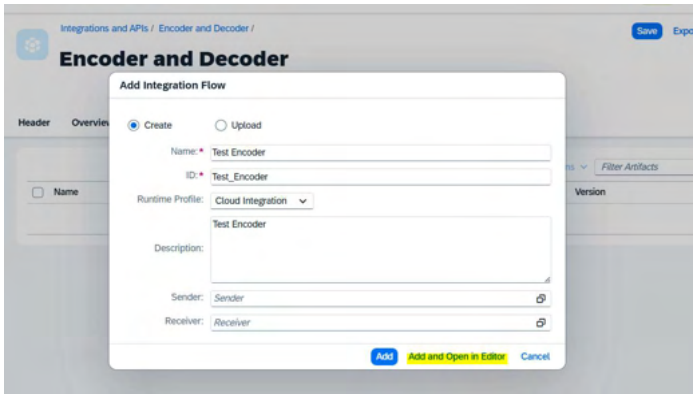


Save the Package after filling the mandatory details. Output will be like below:

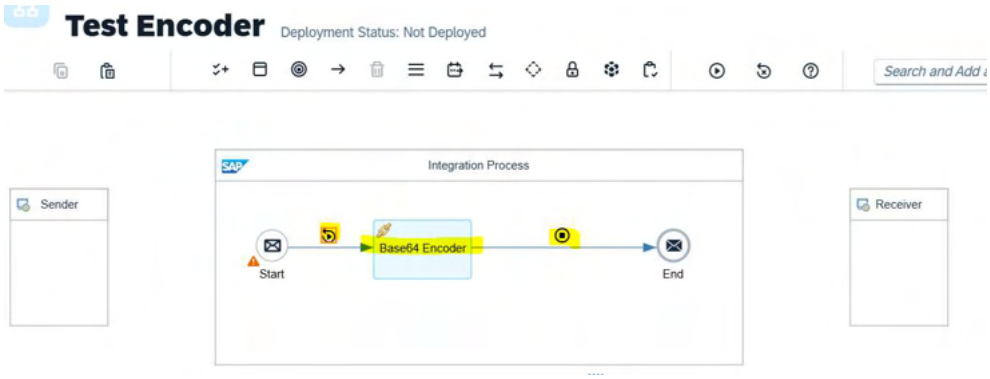


Create the I flow by clicking Artifacts → Add → Integration Flow

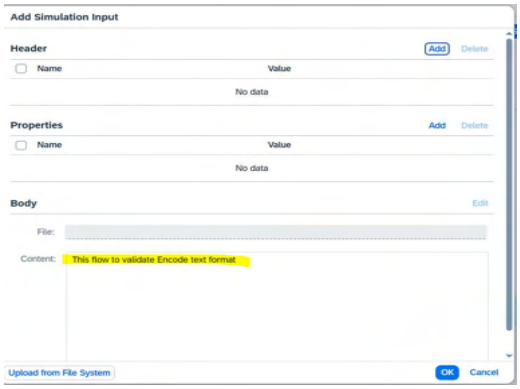




Add from Plotter i.e., **“Bae64 Encoder”** and add “simulator start” and Simulator End” in flow

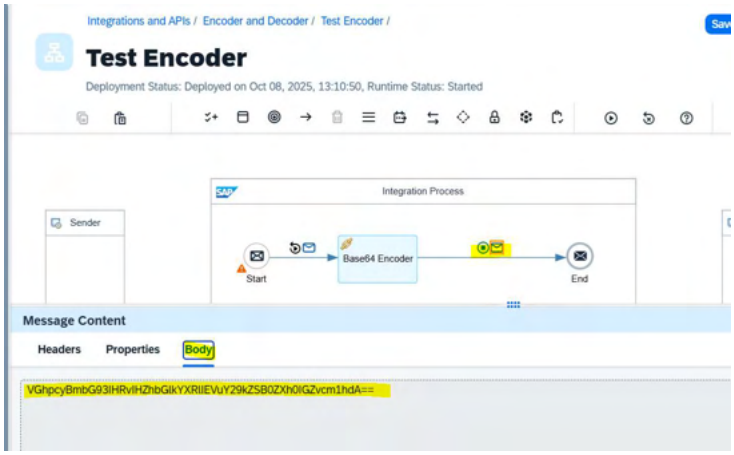


Add the information / input in “Simulator start body”

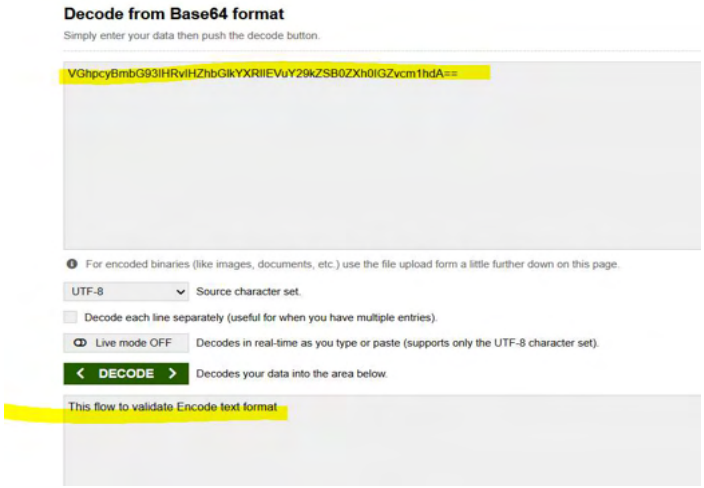


Save and deploy. Click “Run Simulator” from plotter (from the top menu)

Output:



Note: Cross check the encode data from URL (https://www.base64decode.org/)



Scenario 2: “Combine” and “Enrich” options by using Odata Protocal (iFlow)

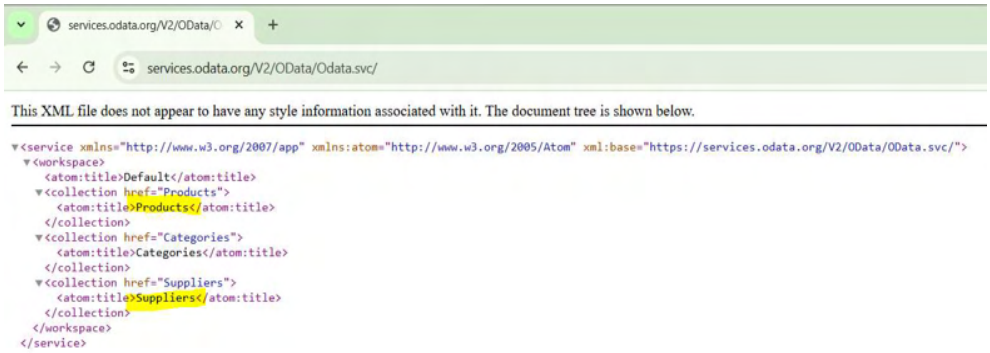
(SAP_BTP_CPI | OData With Content Enricher)

Real-Time Scenario:
The input data is received from multiple sources or two different files, such as “Product” and “Supplier.”

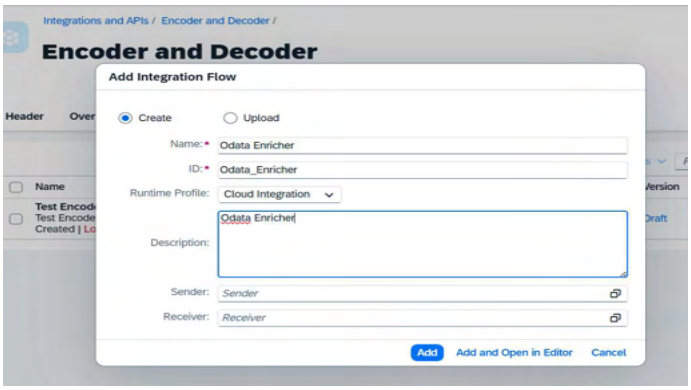
- Scenario 1: Merge both datasets sequentially — first the Product data, followed by the Supplier data — using the “Combine” option in the iFlow.
- Scenario 2: Merge the data within the same tag based on a common field (e.g., ID) using the “Enrich” option in the iFlow.

Data source:

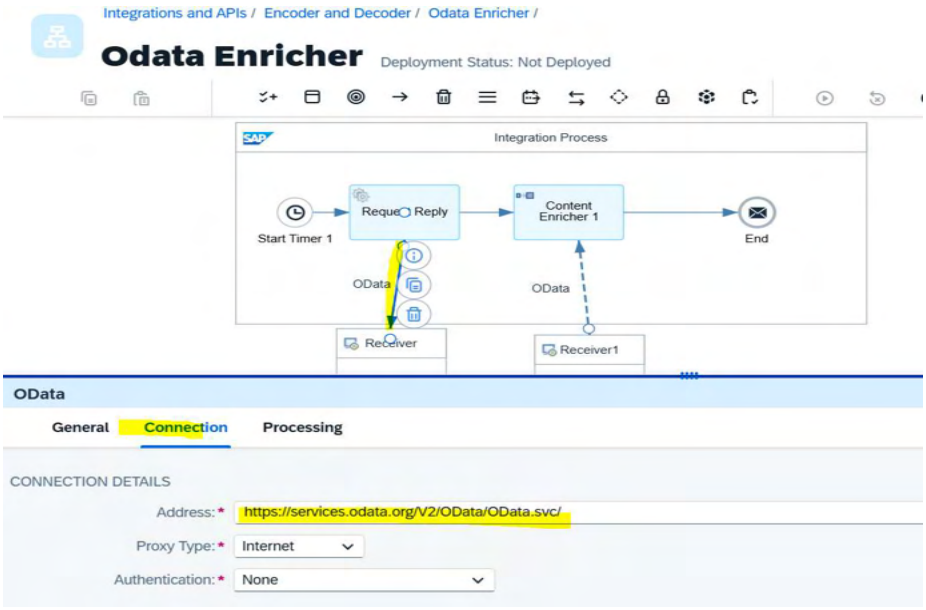
https://services.odata.org/V2/OData/Odata.svc/



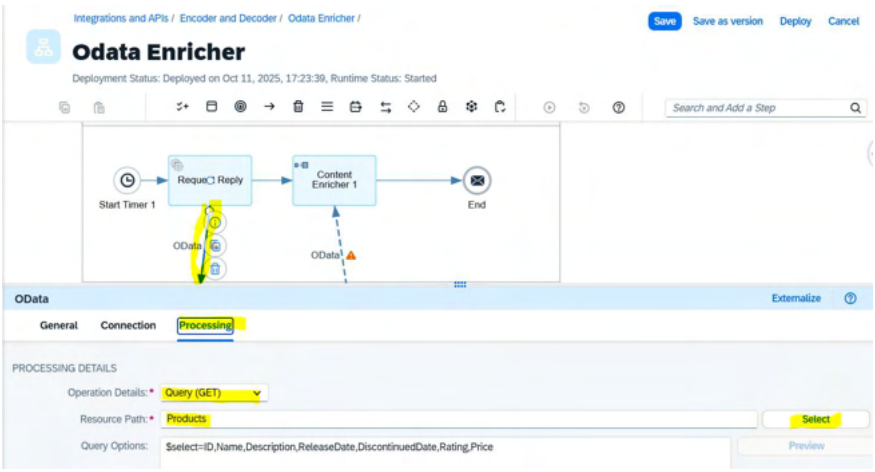
Go to “integration and API” -→ Add new integration flow like below.



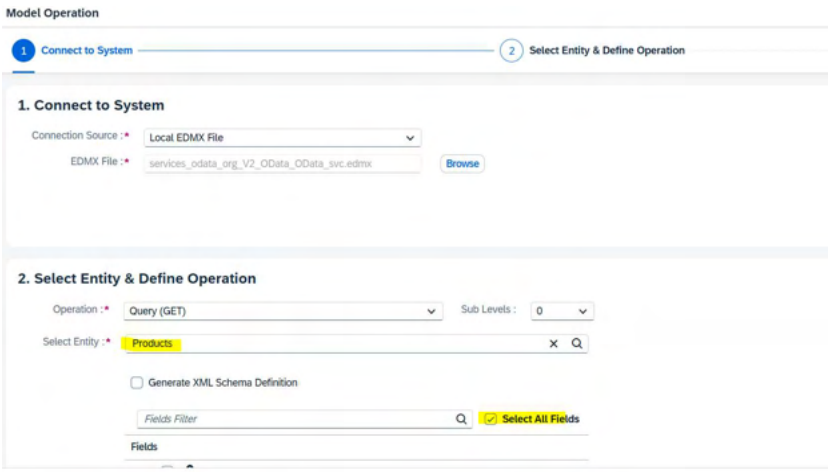
Remove “sender” and Receiver flows and add “timer” in “integration Process”



Click on “Processing” → “select”

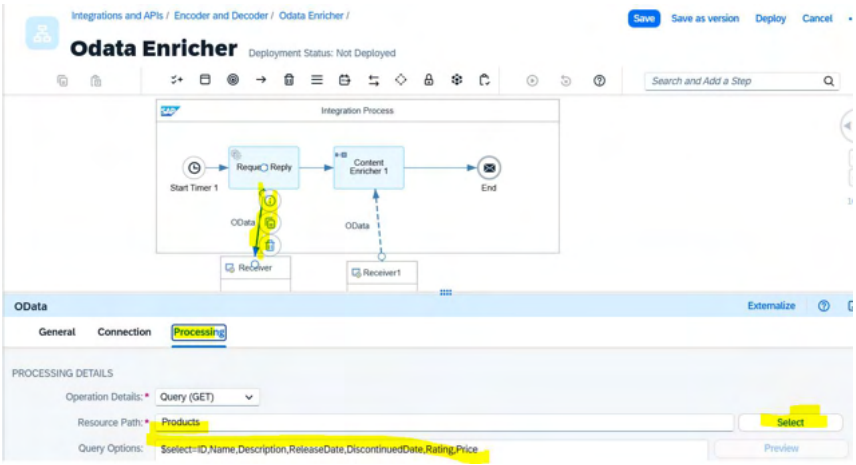


select “product” from drop down → “select all field”

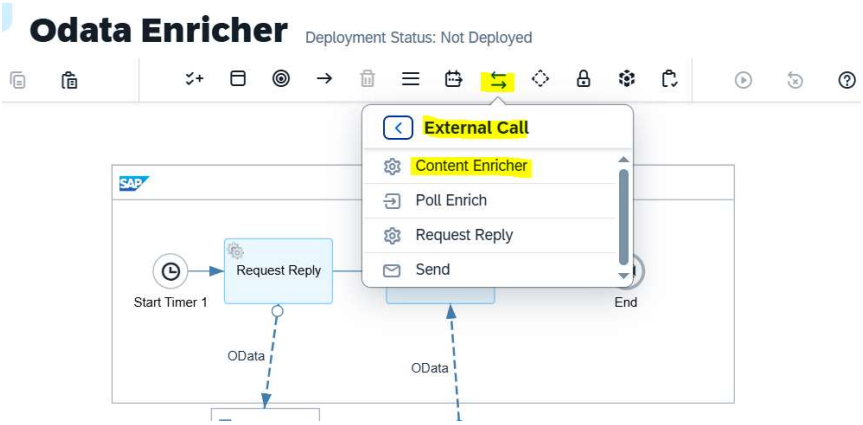


Click finish

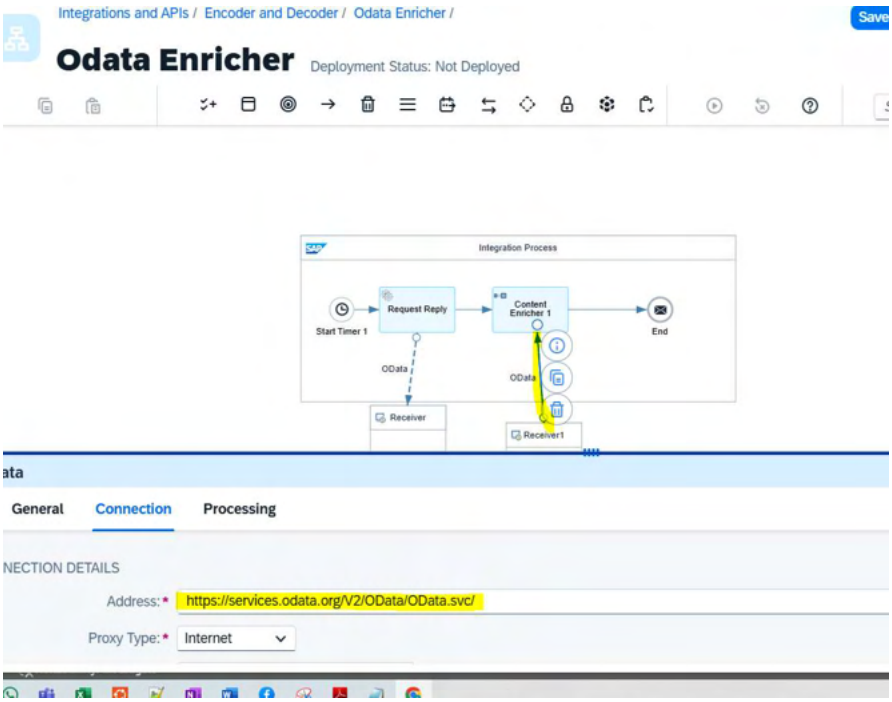
After “finish” iFlow as below:



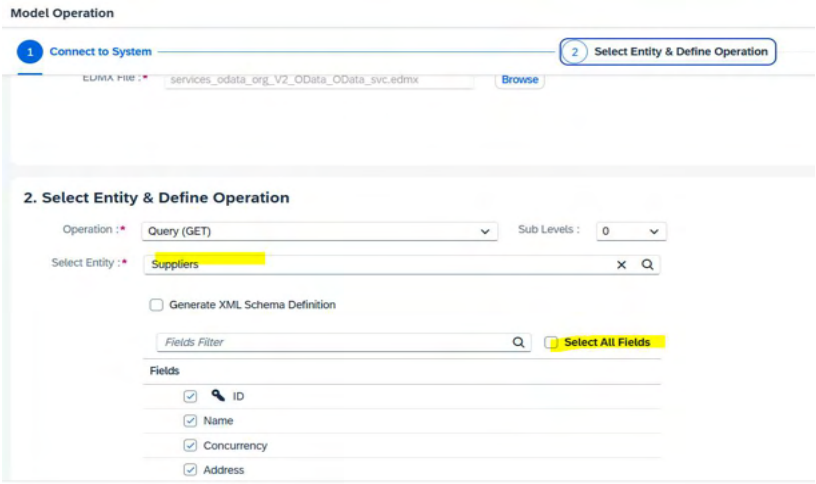
Add the “External call → select “Content Enricher”



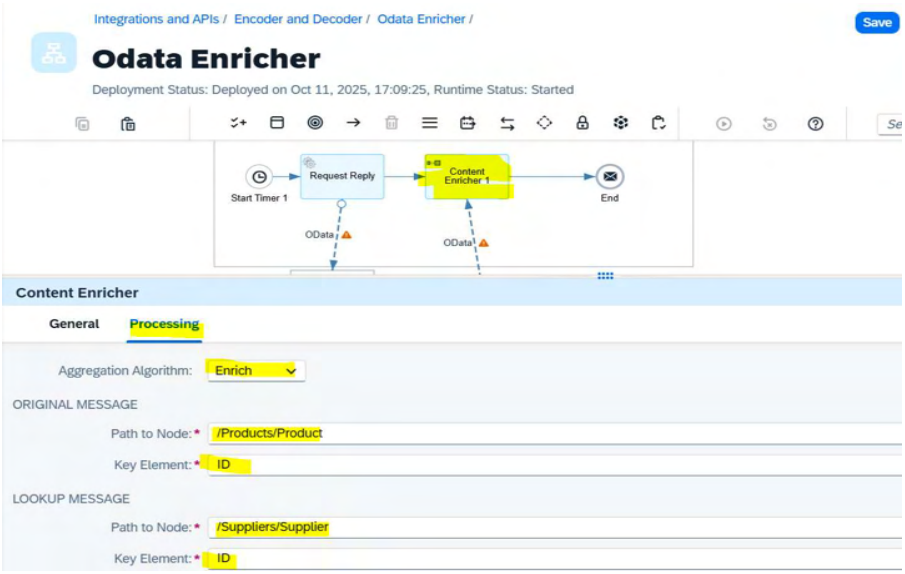
Add “receiver” and link from “enricher” to “receiver” with reverse arrow i.e, Arrow from receiver to “Contnct Enricher”, choose the Odata Protocal.



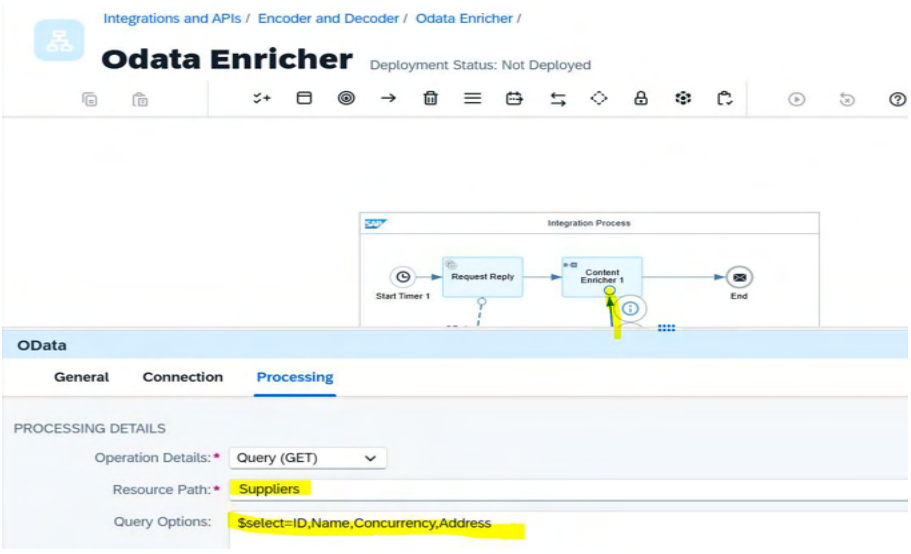
Select the “Supplier” option from Processing



Select “Content Enricher 1” and configure as below:



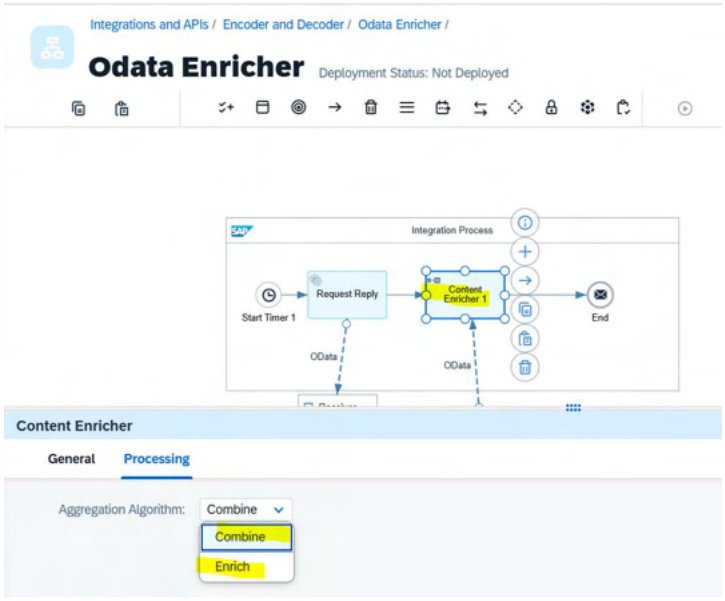
Select “reverse arrow” which is connected to “content modifier 1” user protocol as “Odata”



Note: Content Enrich having 2 options

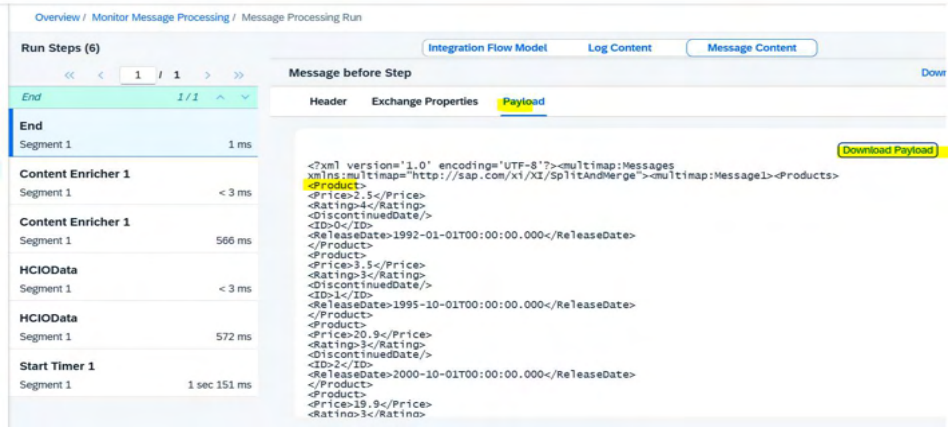
1. ***Combined (it will combine "product" and "Supplier" data in the end/output)***
2. ***Enrich (it will combine matched data of "product" and "Supplier data" based on the common filed i.e, "ID" will nest within XML and non-matched data at the end of output)***

- **Combined:**



Save → Deploy → trace log → deploy

Go to monitoring → download the log



Fist it will shows "Product" data at the end it will shows "Supplier" data.

Data from Downloaded Log file :

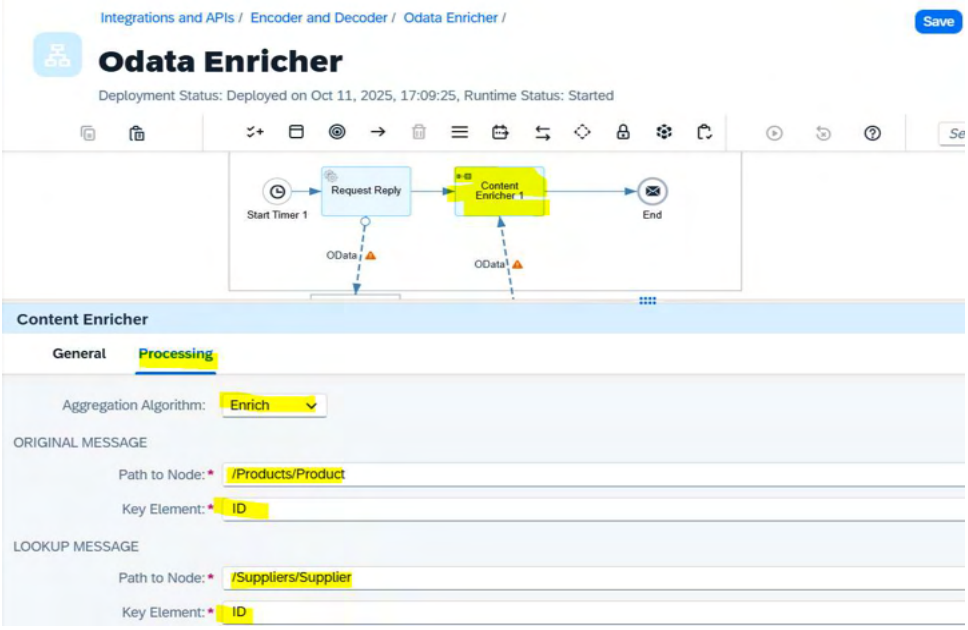
```
<?xml version='1.0' encoding='UTF-8'?>
<multimap:Messages xmlns:multimap="http://sap.com/xi/XI/SplitAndMerge">
<multimap:Message1>
<Products>
    <Product>
      <Price>2.5</Price>
      <Rating>4</Rating>
      <DiscontinuedDate/>
      <ID>0</ID>
      <ReleaseDate>1992-01-01T00:00:00.000</ReleaseDate>
    </Product>
    <Product>
    <Product>
    <Product>
    <Product>
    <Product>
      <Price>22.8</Price>
      <Rating>3</Rating>
      <DiscontinuedDate/>
      <ID>5</ID>
      <ReleaseDate>2006-08-04T00:00:00.000</ReleaseDate>
    </Product>
    <Product>
    <Product>
    <Product>
  </Products>
  </multimap:Message1>
  <multimap:Message2>
  <Suppliers>
    <Supplier>
      <Address>
        <State>WA</State>
        <ZipCode>98074</ZipCode>
        <Street>NE 228th</Street>
        <Country>USA</Country>
        <City>Sammamish</City>
      </Address>
      <Concurrency>0</Concurrency>
      <ID>0</ID>
      <Name>Exotic Liquids</Name>
    </Supplier>
    <Supplier>
```

- Output from “**Enriched**”:

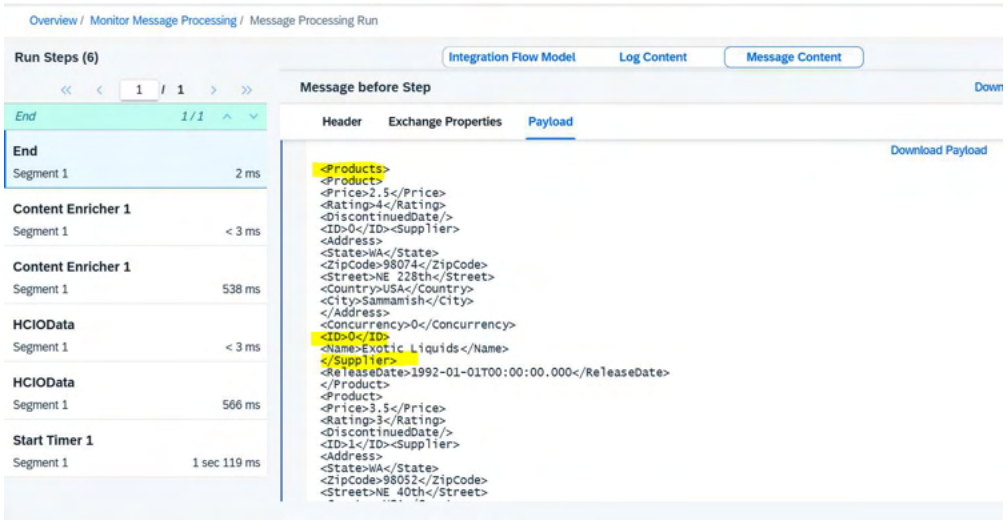
Changes to be made below:

Original Message → means in “request reply” we used “Products”, hence that tag tobe given

Lookup message → In enrich configuration we used “Suppliers”, hence that tag to be given



Note: Product and Supplier information in one block where ID is matching Ex: ID=0

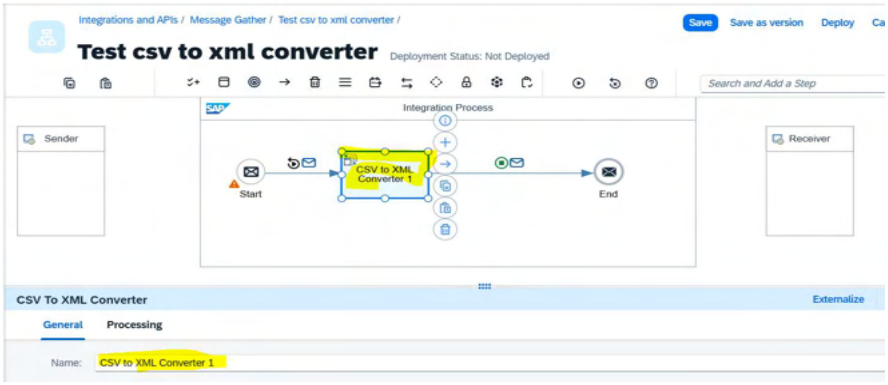


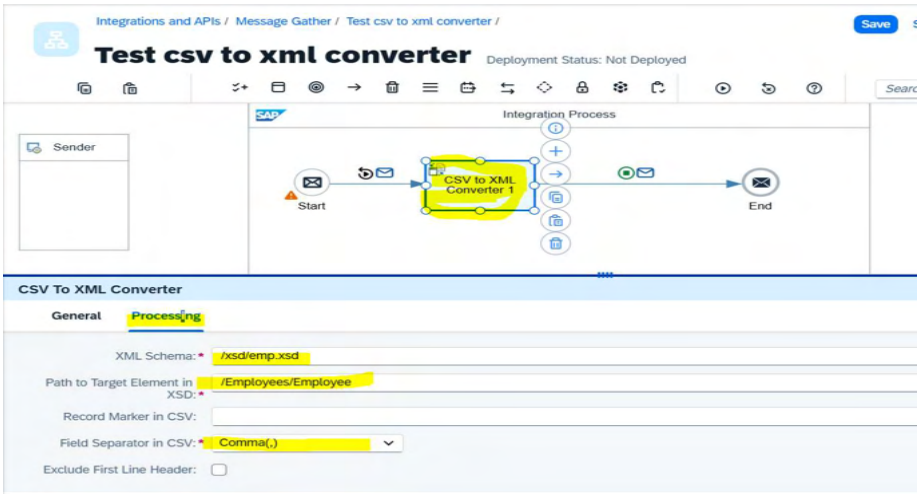

```
<Products>
    <Product>
      <Price>2.5</Price>
      <Rating>4</Rating>
      <DiscontinuedDate/>
      <ID>0</ID>
      <Supplier>
      <Address>
        <State>WA</State>
        <ZipCode>98074</ZipCode>
        <Street>NE 228th</Street>
        <Country>USA</Country>
        <City>Sammamish</City>
      </Address>
      <Concurrency>0</Concurrency>
      <ID>0</ID>
      <Name>Exotic Liquids</Name>
    </Supplier>
      <ReleaseDate>1992-01-01T00:00:00.000</ReleaseDate>
    </Product>
    <Product>
      <Price>3.5</Price>
      <Rating>3</Rating>
      <DiscontinuedDate/>
      <ID>1</ID>
      <Supplier>
      <Address>
        <State>WA</State>
        <ZipCode>98052</ZipCode>
        <Street>NE 40th</Street>
        <Country>USA</Country>
        <City>Redmond</City>
      </Address>
      <Concurrency>0</Concurrency>
      <ID>1</ID>
      <Name>Tokyo Traders</Name>
    </Supplier>
      <ReleaseDate>1995-10-01T00:00:00.000</ReleaseDate>
    </Product>
    <Product>
      <Price>20.9</Price>
      <Rating>3</Rating>
```

Scenario 3: CSV to XML conversion

Real-Time Scenario:
As per the requirement, when an input file is received in **CSV** format, it needs to be **converted into XML** format before being sent to the destination system.

- After logging in to the **SAP BTP Integration Suite**, create a **new package**, and then create a **new artifact (integration flow)** within it
- From transform select the “CSV to XML converter” or search from search bar and place in the iflow as below and configure as below:





Sample xsd file used for this test :

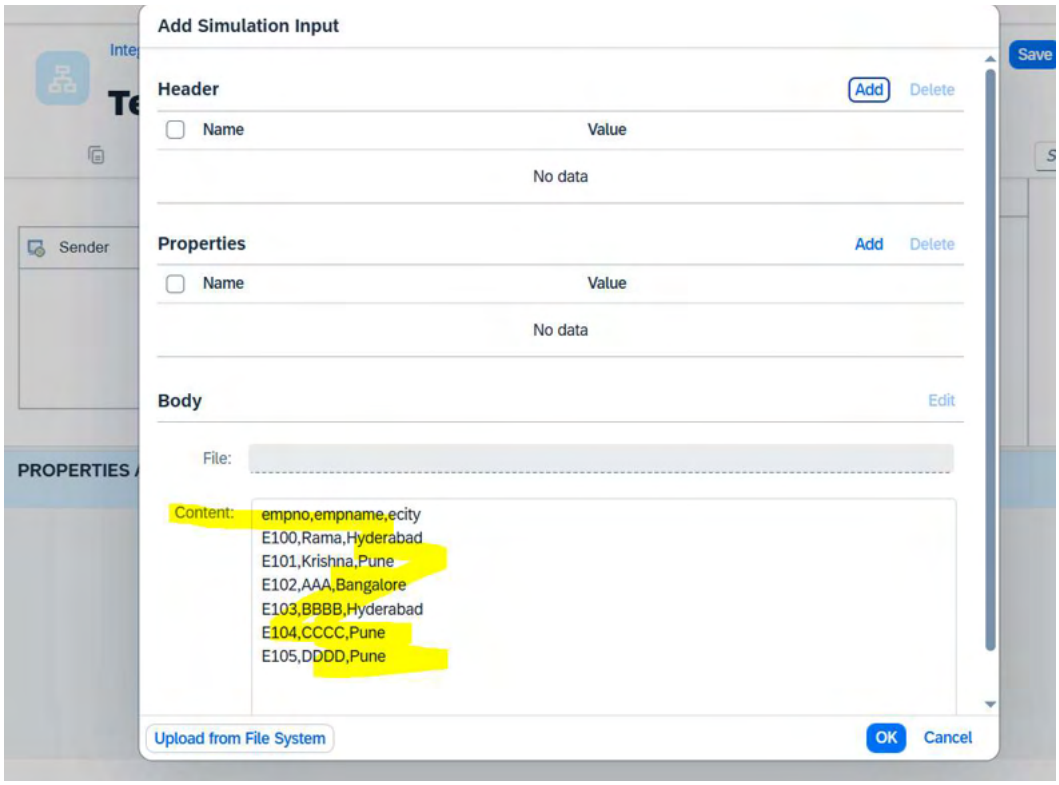
Note: this xsd file based on the csv data information. Use public site to generate the xsd file based on the CSV data.


emp.xsd

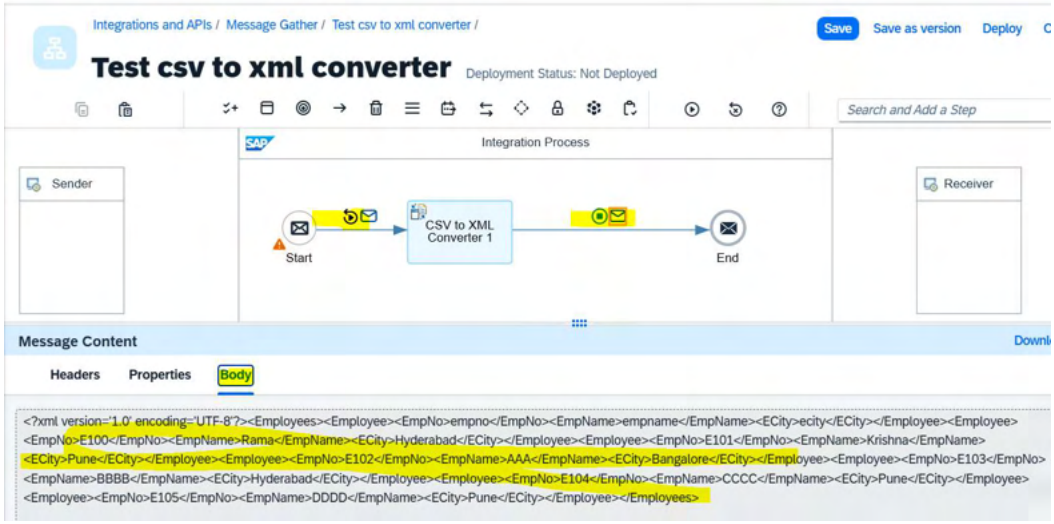
Sample data in CSV file:

empno,empname,ecity
E100,Rama,Hyderabad
E101,Krishan,Pun
E102,AAA,Bangalore
E1-3,BBBB,Hyderabad
E104,CCCC,Pune
E105,DDDD,Pune

Add “Simulation start” and Simulation End” and add the CSV data in the “simulation start” body as below:



Execute the Simulation and see the output in the “simulation end” tool as below (which is converting csv data to xml format).



Scenario 4: Filtering

This will support and filter based on the filter condition and provide the output.

In POC covered Boolean, Node, Nodelist options

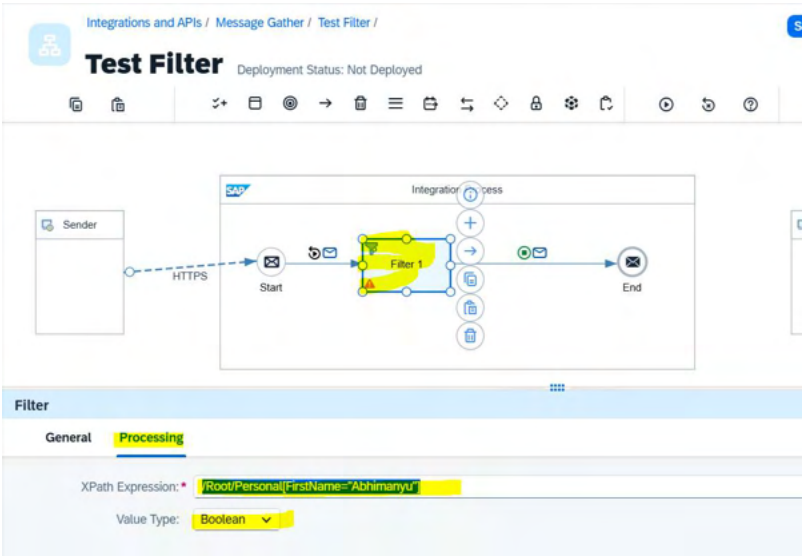
The below sample data used for this POC:

Filter_sample.xml

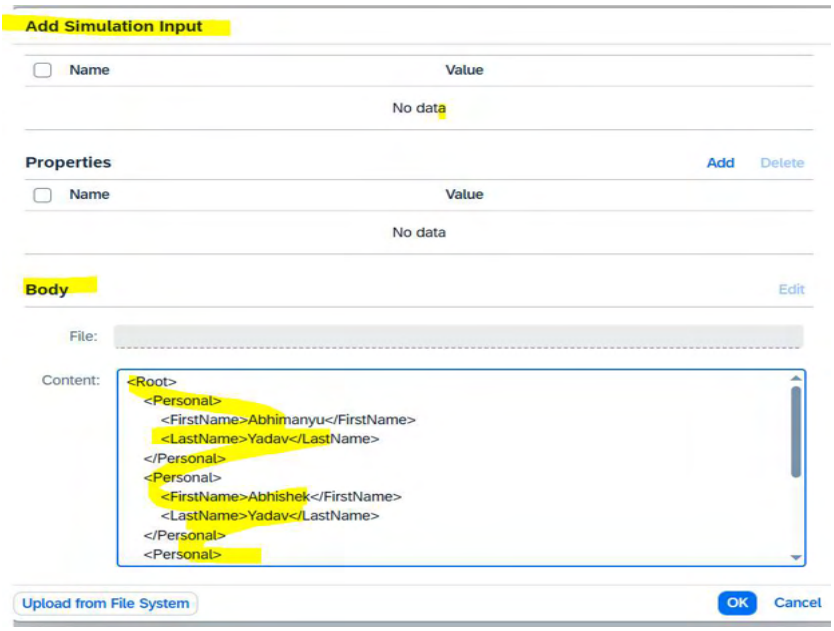
- **Boolean**: based on the condition if match it will give “true” or “false”

  In the example we confiurated to search the “/Root/Personal[FirstName=**"Abhimanyu**"]”

  If this match, “simulation output body” will give the result “true (if found) or “false” (if not found).

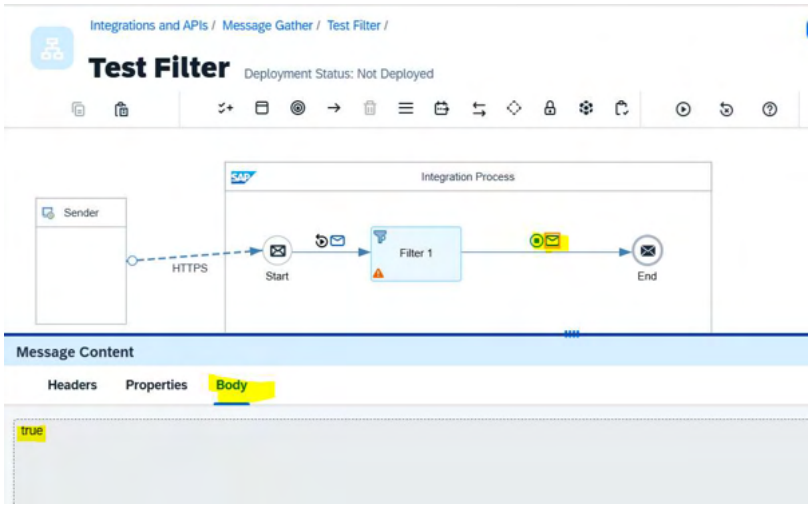


Input provided in the “simulation start body”



Note: Since the input condition matches (i.e., **firstname = "Abhimanyu"**), the output is displayed as **“true”**, because the **Value Type** option was set to **Boolean**.

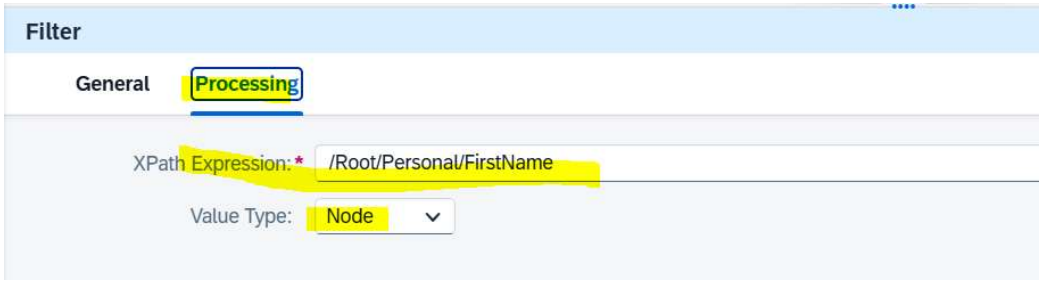
Seen the below output:



The below Filter Output is based on ***Value type*** “Node” or Nodelist”

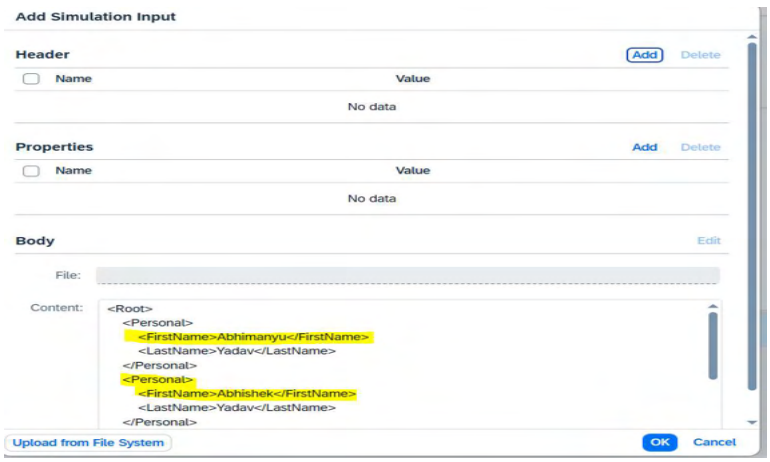
(Node vs Nodelist)

- **Node** option from *Value Type*:
When the **Value Type** is set to **Node**, the output shows the details of the **first record** that matches the filter condition. This means only the **first matching entry** from the input will be displayed as the **single output**.

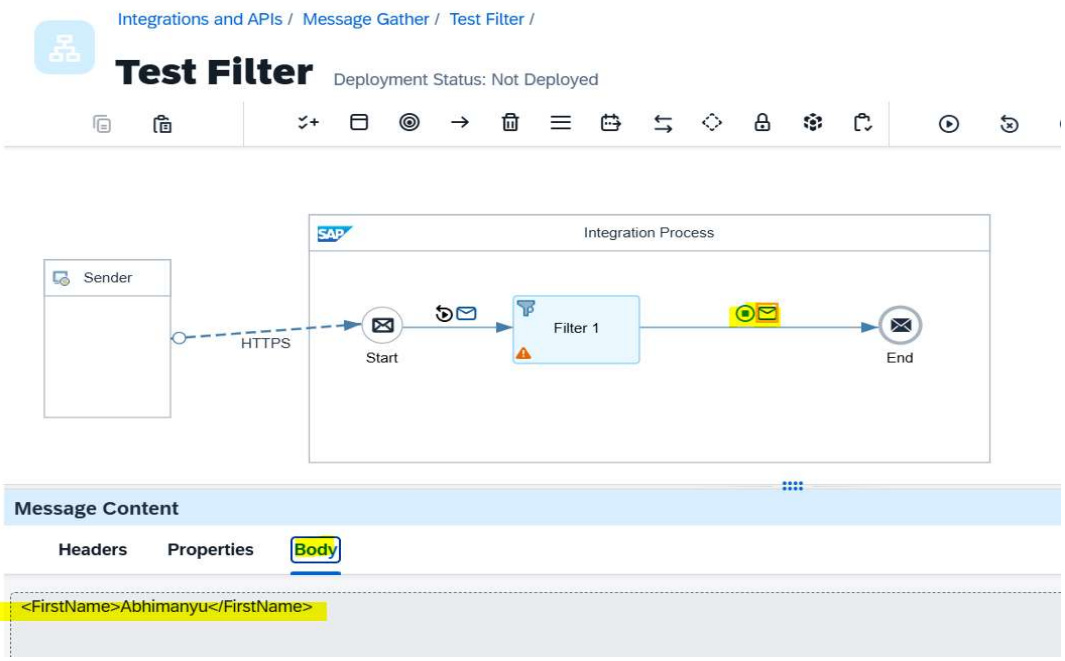


Note: in the XML file / input file having the filed “FirstName” and found, then that value will return in output file.
The below is input from “simulation start body” having multile “Firaname” values

Input in the simulation body:



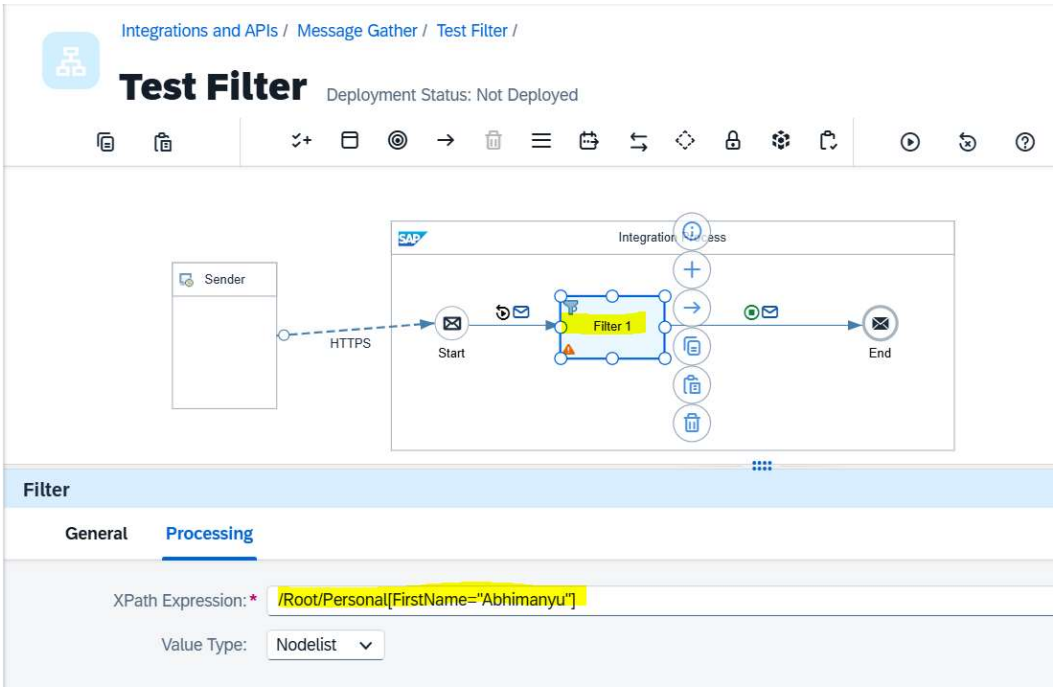
Ouput:
Eventhough “Firstanme” having mulitiple times in the input (approx.. 3 times) output will **consider First Occurance** and display that value:



- **Nodelist** option from *Value Type* :

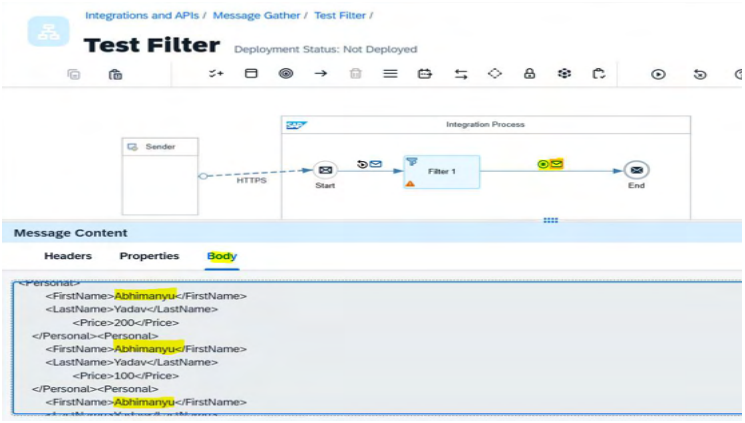
  Outwill display all matched list / details (Single/multiple output)

  Configuration:



Output → will consider and display all data matched with filed (FirstName)—" /Root/Personal[FirstName=**"Abhimanyu"**]”

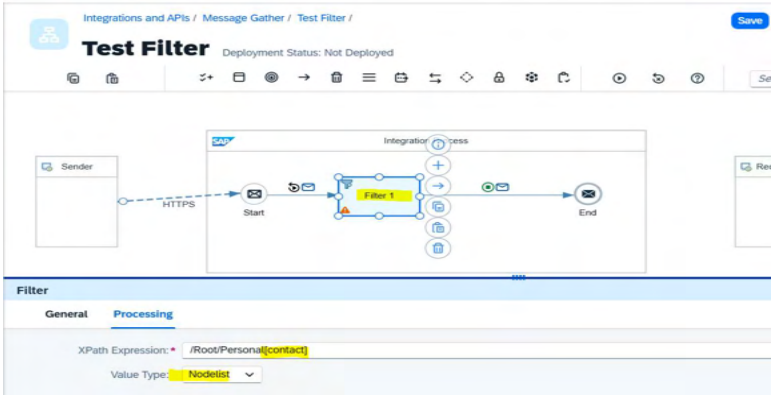
Matched 3 members data and showing the same in the output.



Scenario 5: Filter and display the output for records that contain the “contact” field or a specific tag/field in the input data.

If the input file contains the **“contact”** tag (in the XML), only the records with this tag will appear in the output, and all other records will be ignored.
Example: Out of 10 records, if only 3 have the **“contact”** tag, only those 3 records will be displayed in the output, and the remaining 7 will be ignored.

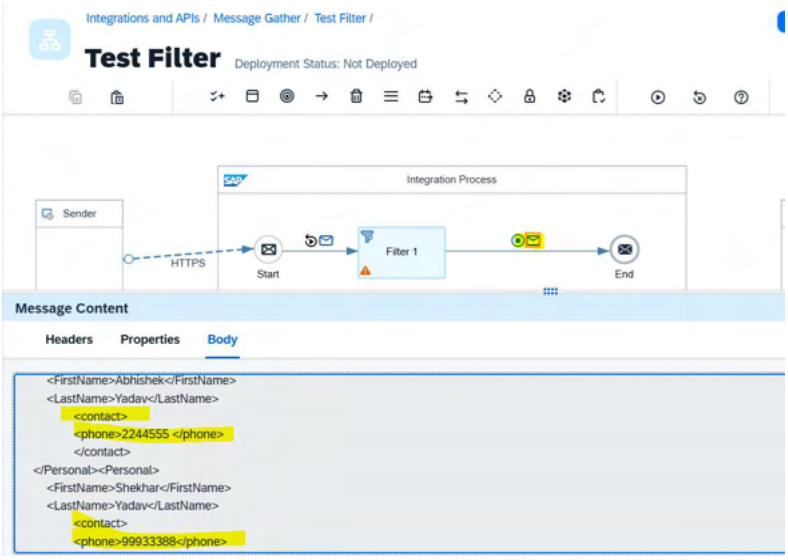
Configuration:



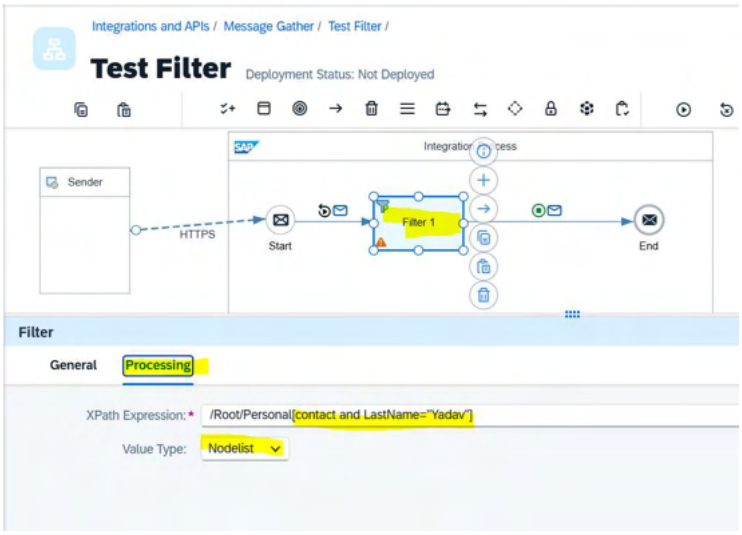
Input data having the below information:

```
<Root>
    <Personal>
        <FirstName>Abhimanyu</FirstName>
        <LastName>Yadav</LastName>
        <Price>200</Price>
    </Personal>
    <Personal>
        <FirstName>Abhishek</FirstName>
        <LastName>Yadav</LastName>
        <contact>
        <phone>2244555 </phone>
        </contact>
    </Personal>
    <Personal>
        <FirstName>Shekhar</FirstName>
        <LastName>Yadav</LastName>
        <Price>300</Price>
        <contact>
        <phone>99933388</phone>
        </contact>
    </Personal>
    <Personal>
        <FirstName>Abhimanyu</FirstName>
        <LastName>Yadav</LastName>
        <Price>100</Price>
    </Personal>
   <Personal>
        <FirstName>Shekhar</FirstName>
        <LastName>Yadav</LastName>
        <Price>400</Price>
        <contact>
        <phone>99933388</phone>
        </contact>
    </Personal>
```

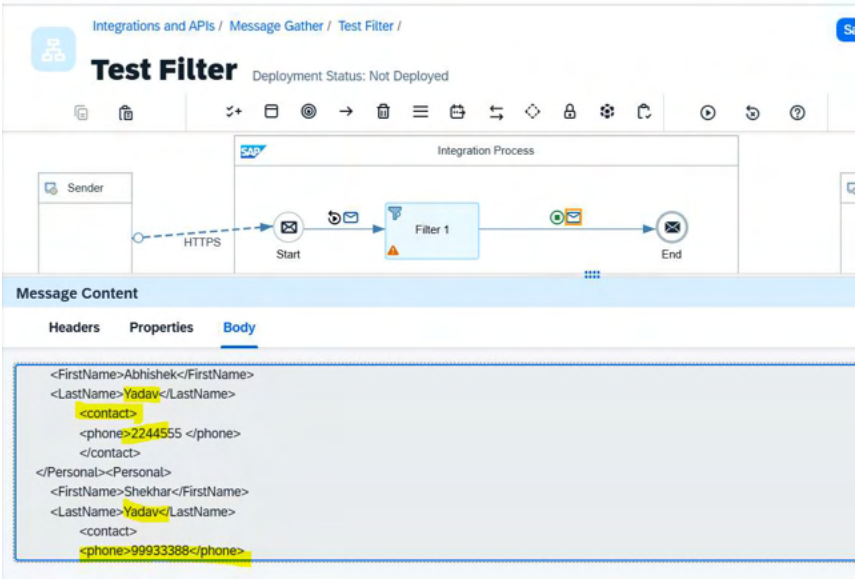
Output:



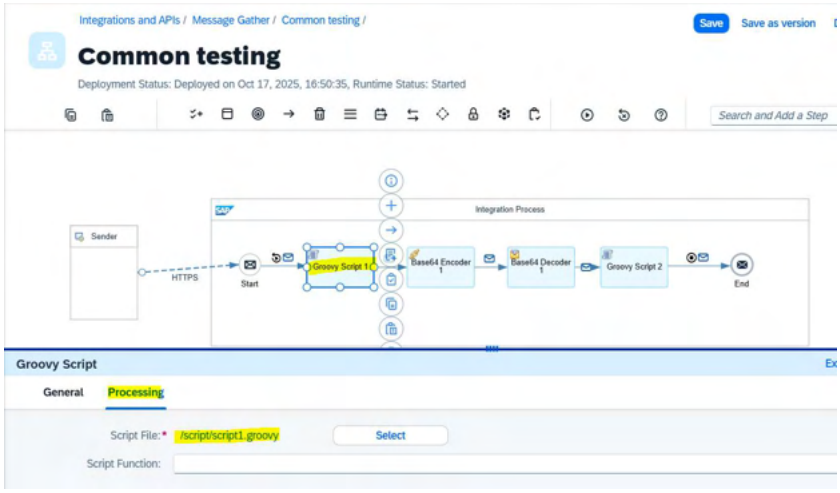
Double filter: Scenario → Filter condition is …. Contact should be there along with Last Name as “Yadav”



Output: Output is only Last name is “**Yadav**” along with that, that person having “**contact**”.



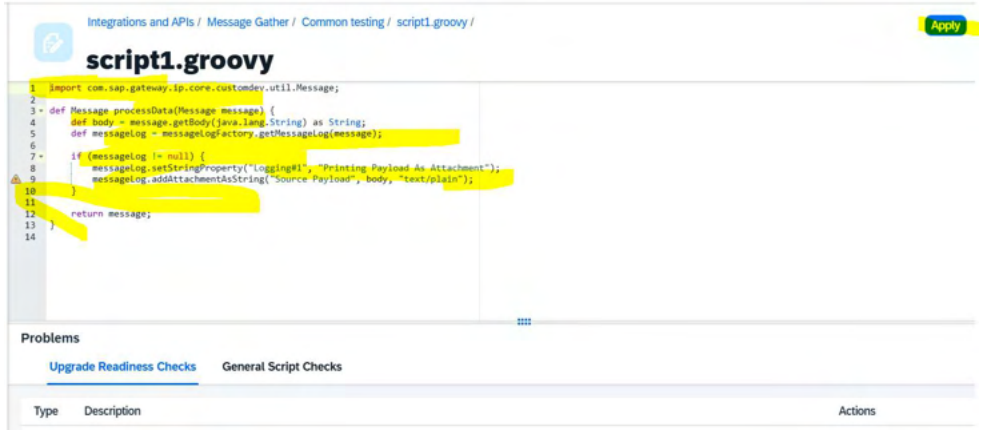
Scenario 6: How to Capture the Source and Target logs by using Groovy Script.



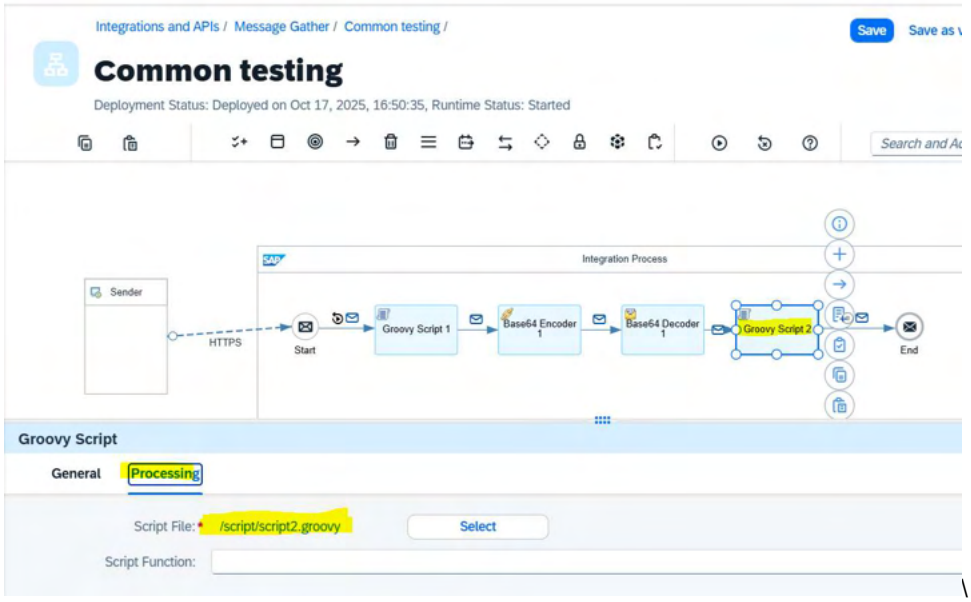
See the Groovy Script in side "/script/script1.groovy"

After adding / writing the Groovy script click on "Apply" to validate and fix the script.

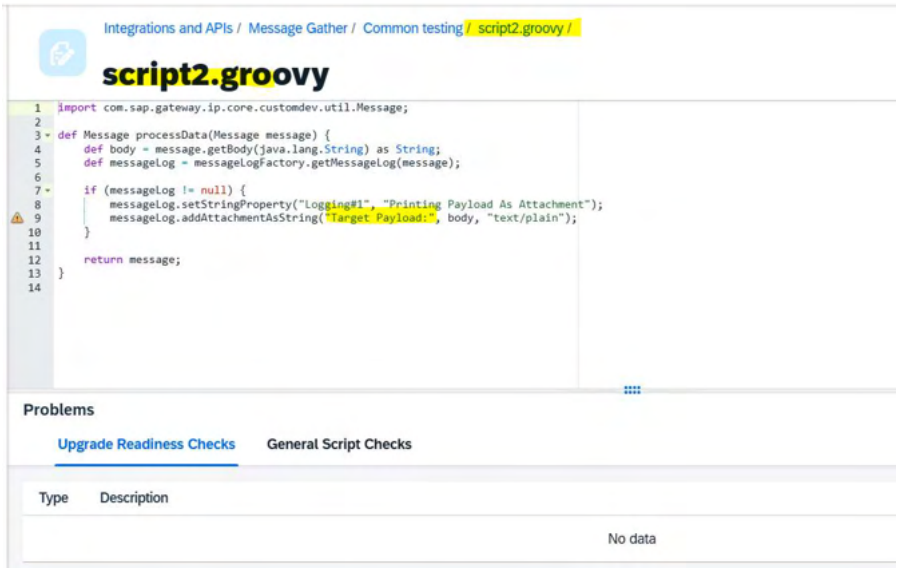
Groovy Script at Source:



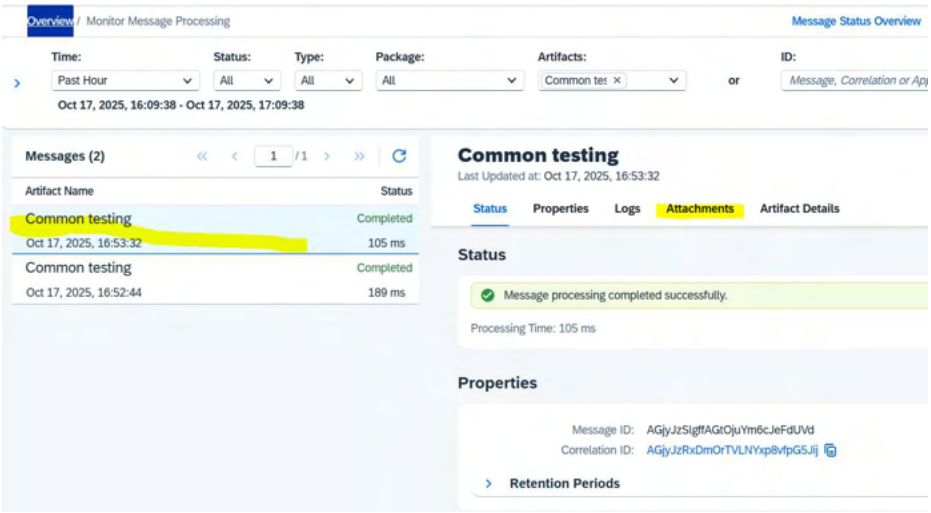
Groovy Script at Target:

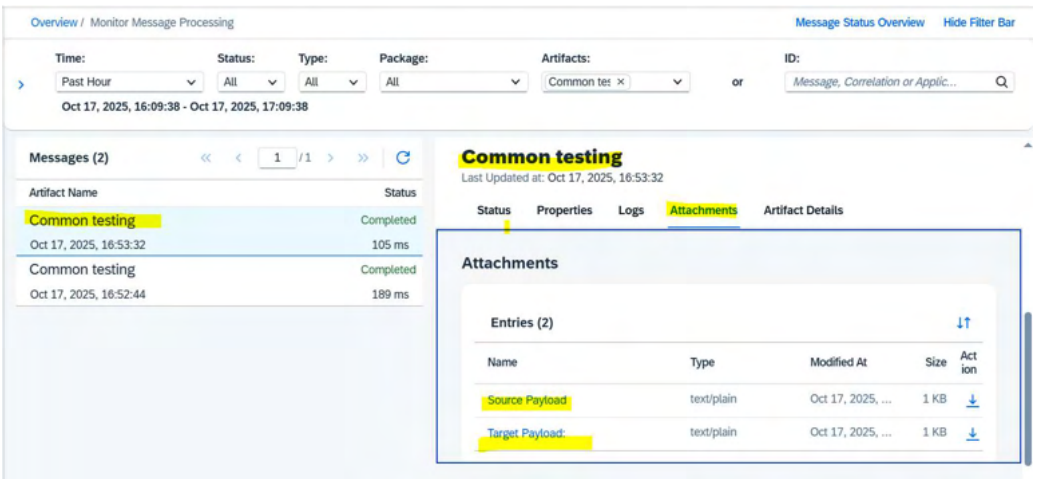


Groovy script at Target:

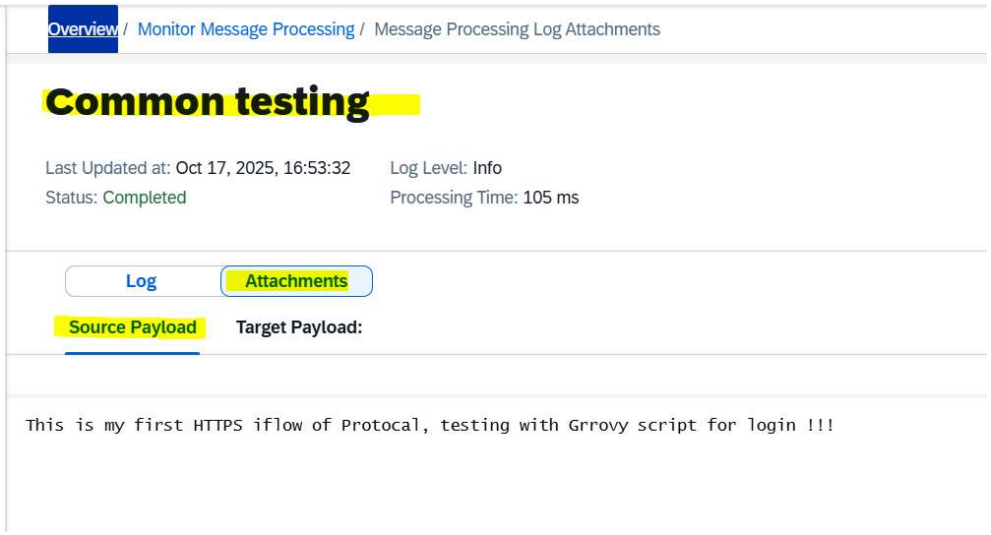


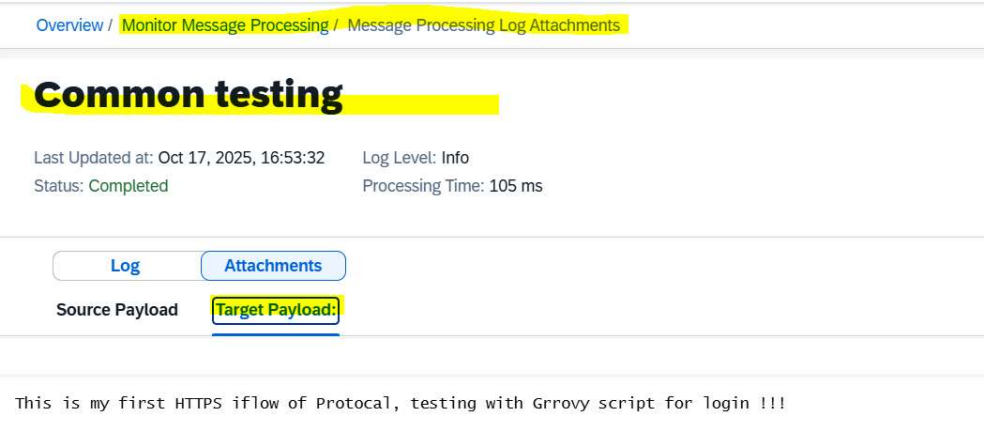
Save and Deploy the Iflow and check the log in “monitoring”



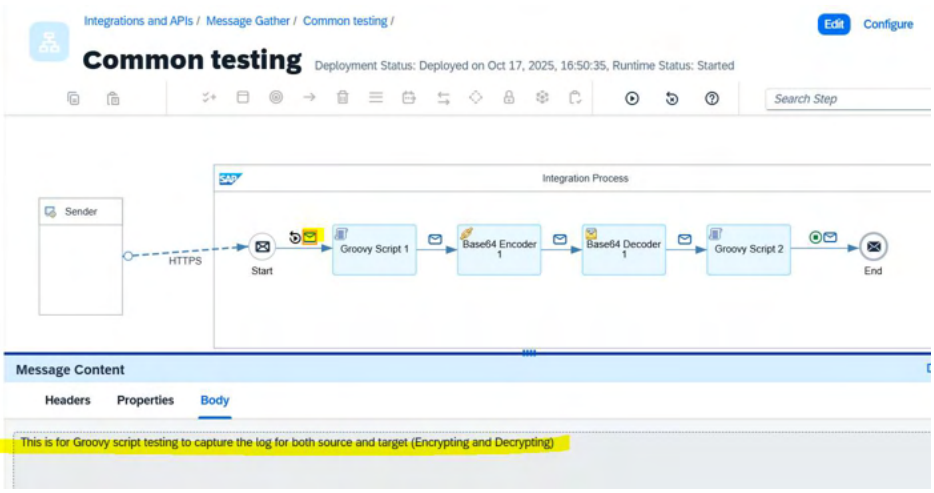


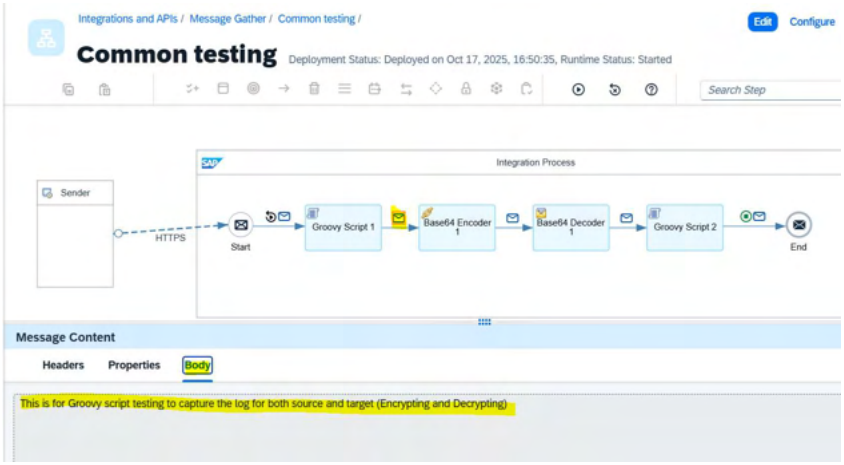
Source Payload / Log:

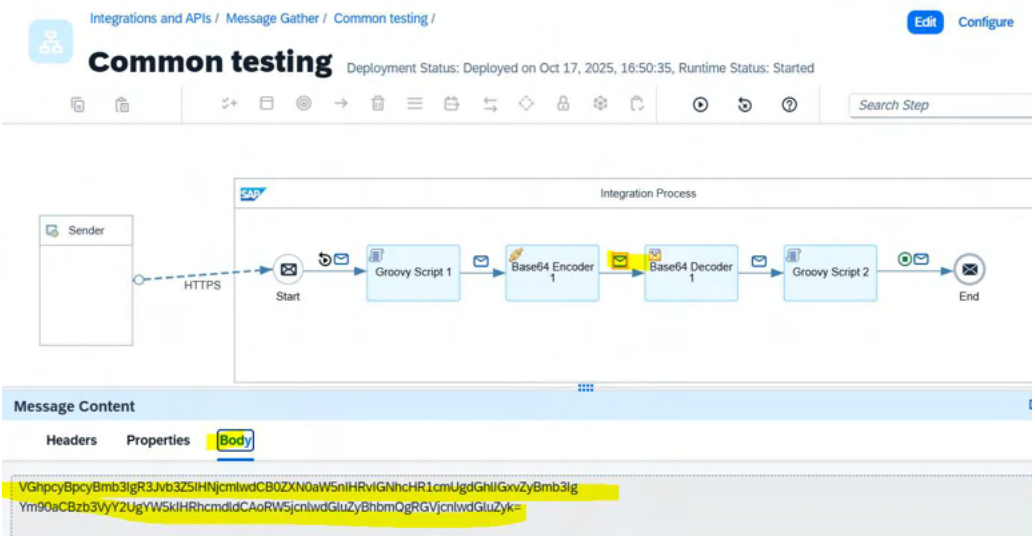


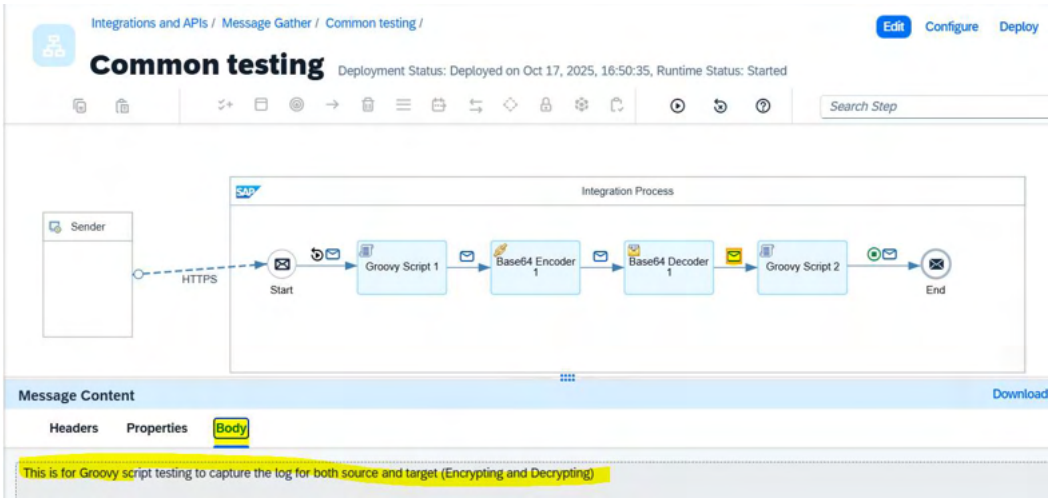


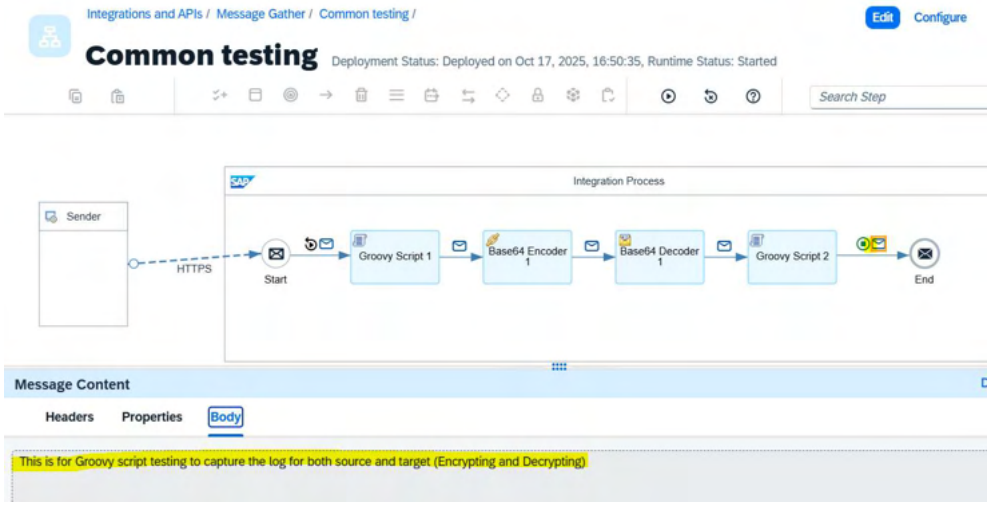
Logs for each step through simulation mode:



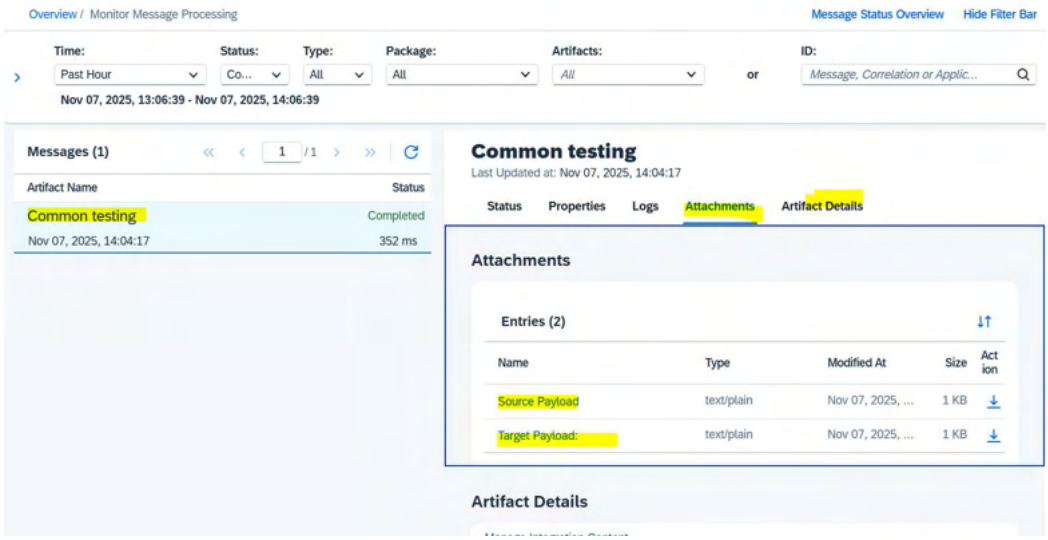

Integrations and APIs / Message Gather / Common testing /
Edit
Configure
Common testing
Deployment Status: Deployed on Oct 17, 2025, 16:50:35, Runtime Status: Started
Search Step
Integration Process
Sender
HTTPS
Start
Groovy Script 1
Base64 Encoder 1
Base64 Decoder 1
Groovy Script 2
End
Message Content
Headers
Properties
Body
This is for Groovy script testing to capture the log for both source and target (Encrypting and Decrypting)


Integrations and APIs / Message Gather / Common testing /
Edit
Configure
Common testing
Deployment Status: Deployed on Oct 17, 2025, 16:50:35, Runtime Status: Started
Search Step
Integration Process
Sender
HTTPS
Start
Groovy Script 1
Base64 Encoder 1
Base64 Decoder 1
Groovy Script 2
End
Message Content
Headers
Properties
Body
VGhpcyBpcyBmb3lgR3Jvb3Z5IHNjcmlwdCB0ZXN0aW5nIHRvIGNhcHR1cmUgdGhlIGxvZyBmb3lg
Ym90aCBzb3VyY2UgYW5kIHRhcmdldCAoRW5jcnlwdGluZyBhbmQgRGVjcnlwdGluZyk=

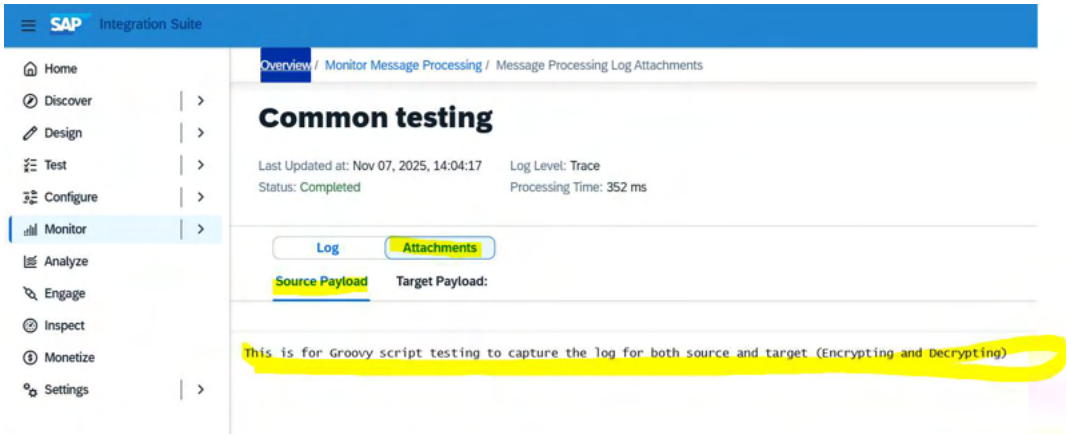

Integrations and APIs / Message Gather / Common testing /
Edit
Configure
Deploy
Common testing
Deployment Status: Deployed on Oct 17, 2025, 16:50:35, Runtime Status: Started
Search Step
Integration Process
Sender
HTTPS
Start
Groovy Script 1
Base64 Encoder 1
Base64 Decoder 1
Groovy Script 2
End
Message Content
Download
Headers
Properties
Body
This is for Groovy script testing to capture the log for both source and target (Encrypting and Decrypting)



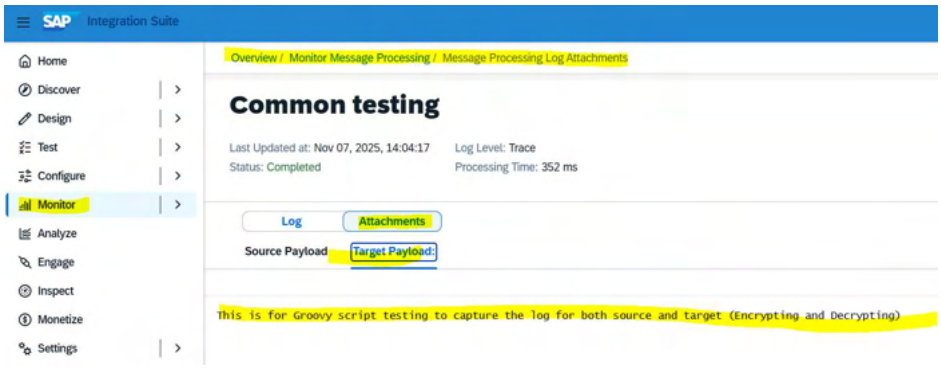
Logs from the iflow:



Source payload log:



Target payload log:



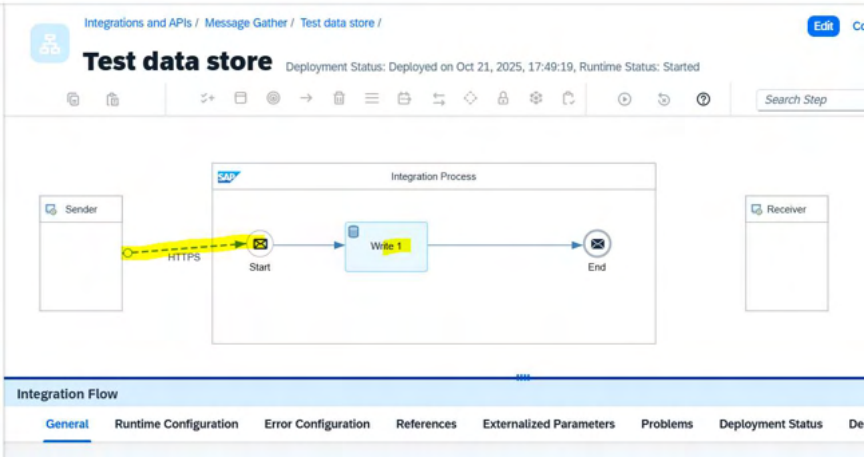
= = = =

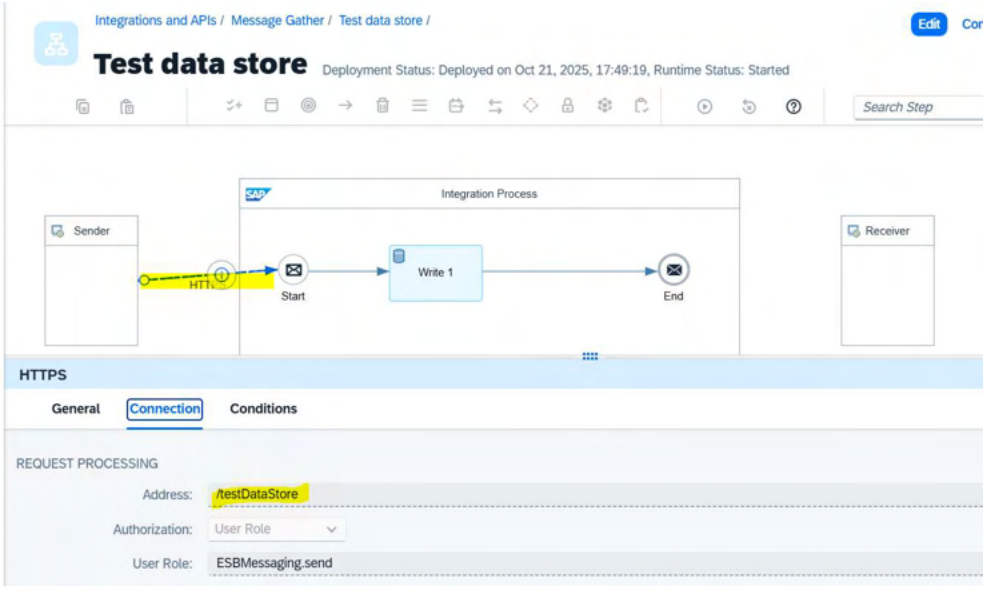
Scenario 7: Data Store and Fetch

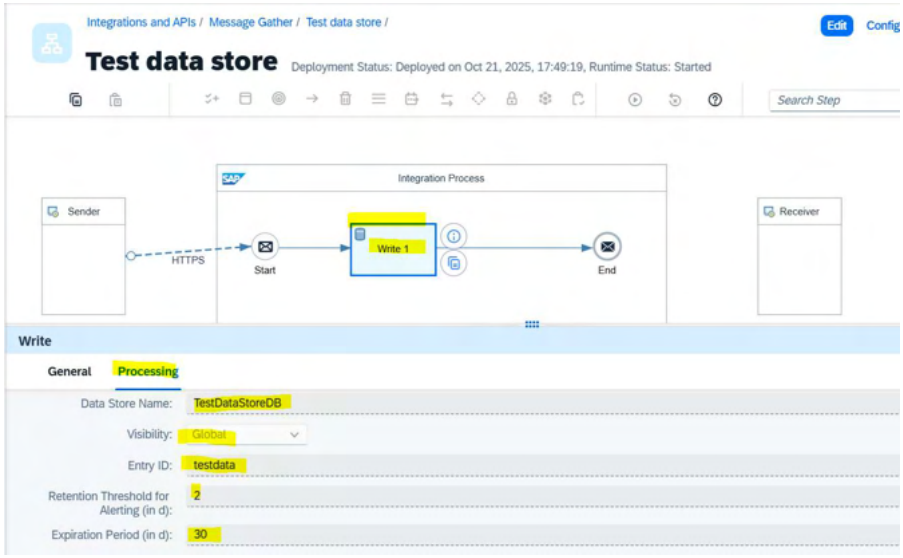
Realtime scenario:

We can use an iFlow to **store data into a database** (with the DB and table created during the iFlow) and then **retrieve that data from the same database and table** using another iFlow.

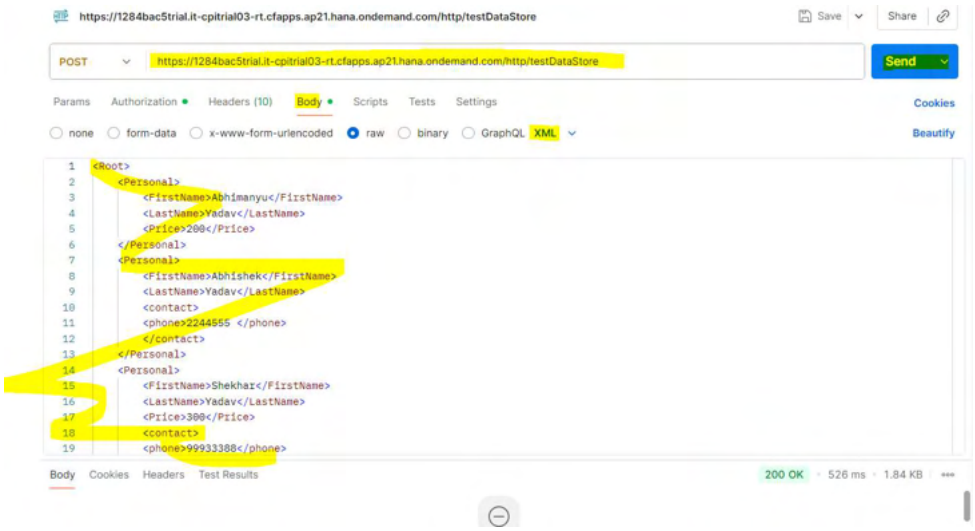
- **Data Storing (1st iFlow to store the data in DB and table):**



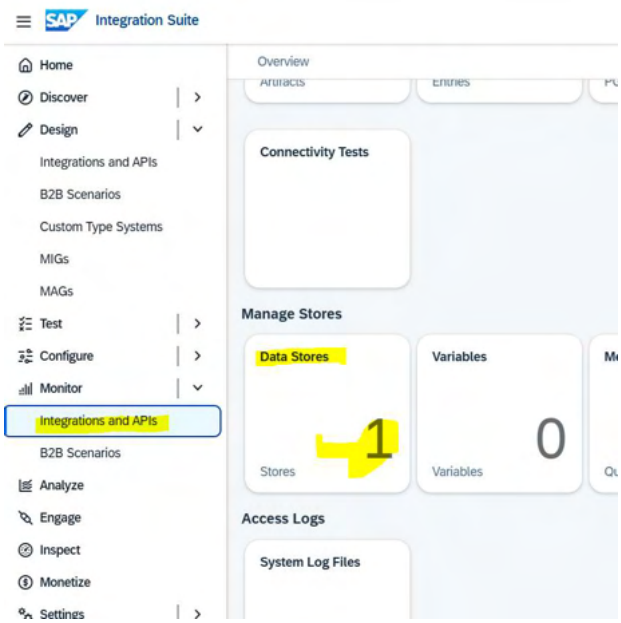


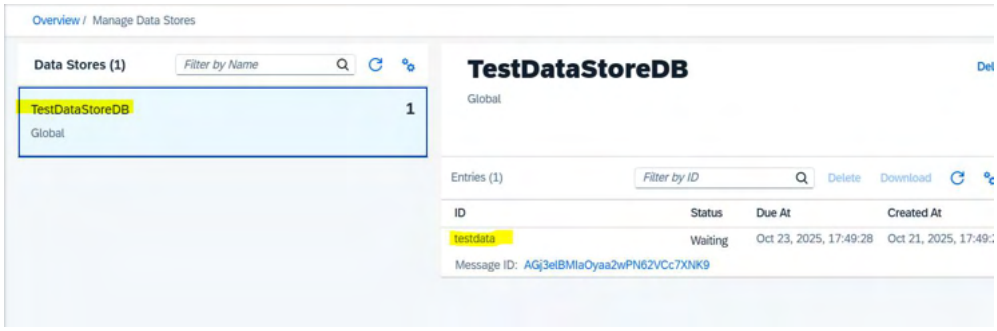


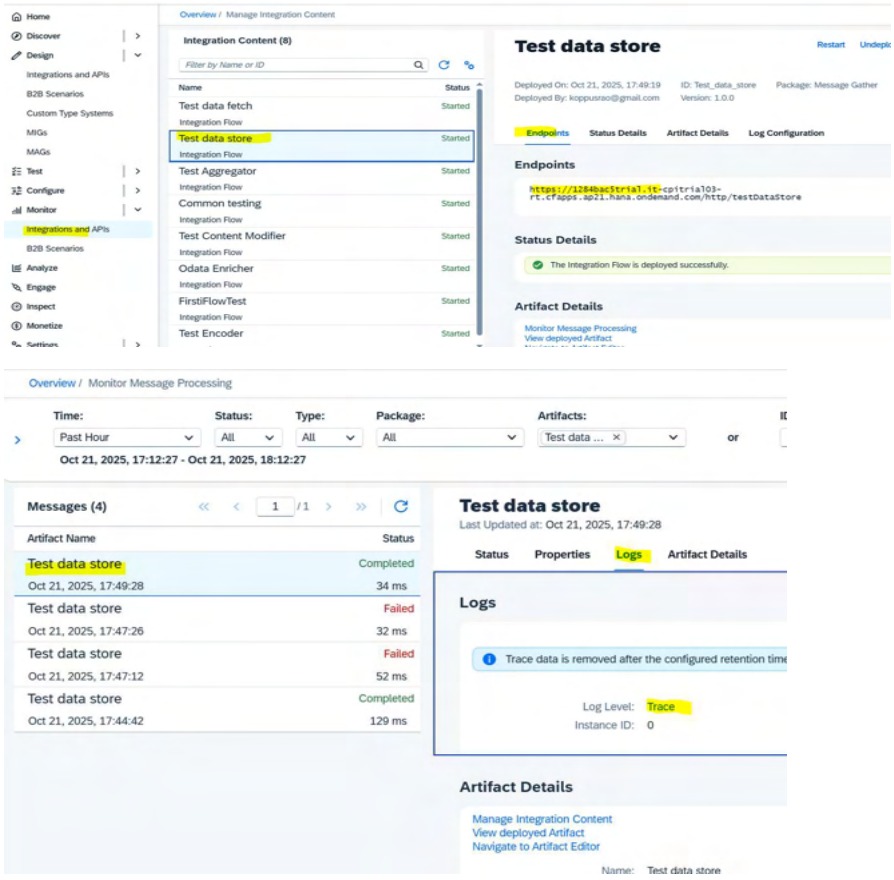
Execute the input data from POSTMAN (use <u>POST</u> method option to store in the DB and table)



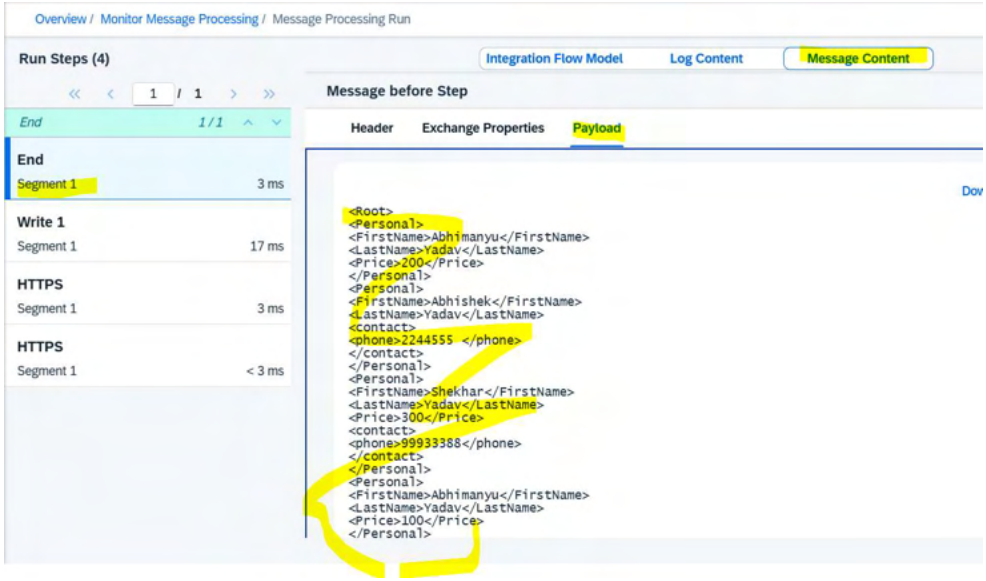
Data stored in the DB (check from the iFlow messaging) :





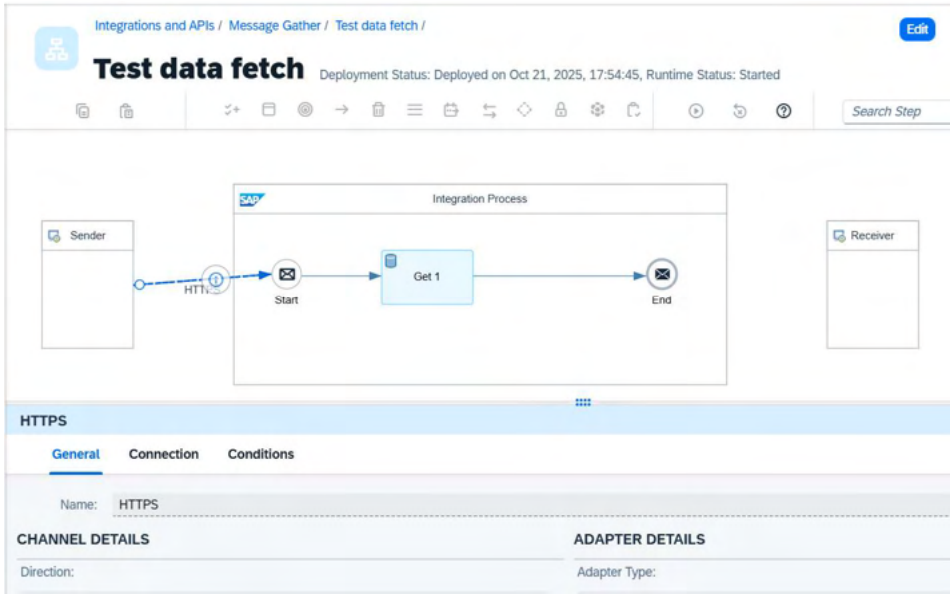


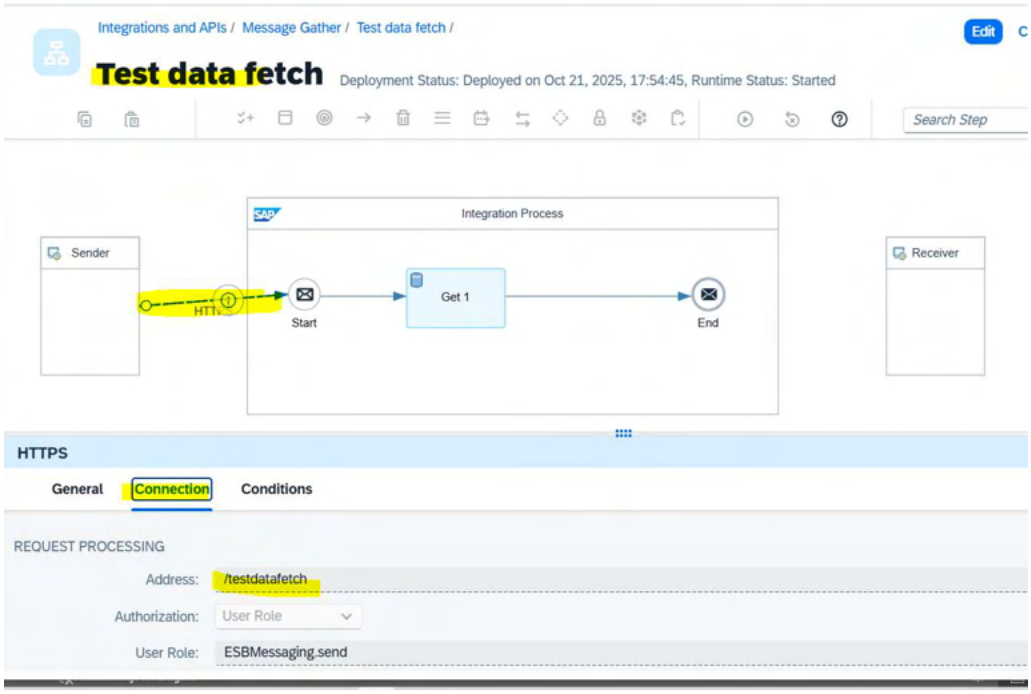
The below data stored in the “TestDataStoreDB” → testdata (table)

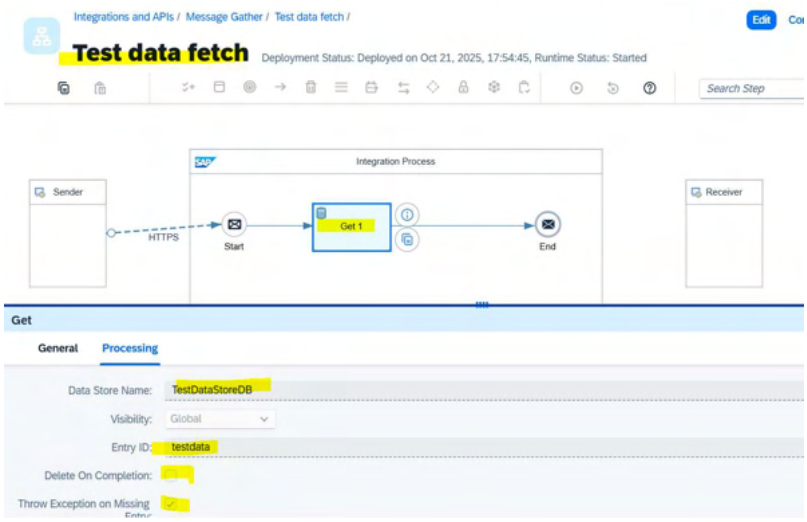


- **Retrieve / Fetch / Get the data from the “TestDataStoreDB” → testdata (table)**

  Note: This DB is configured as “Global”

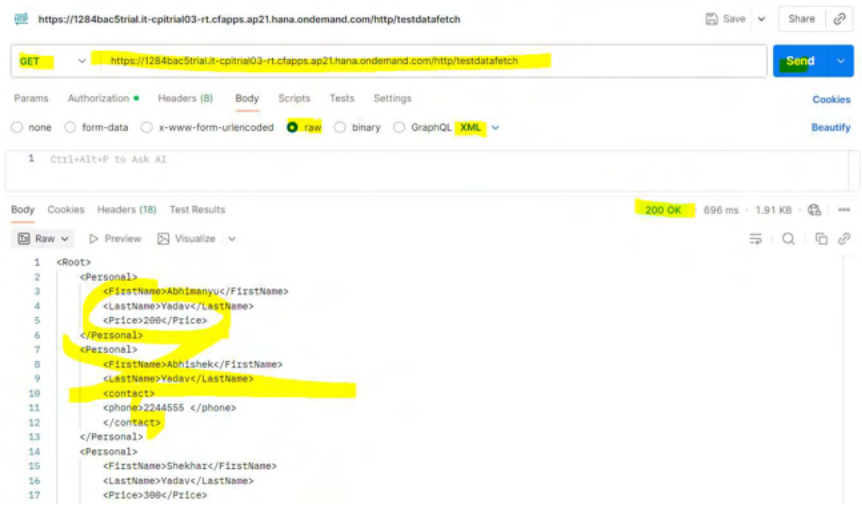




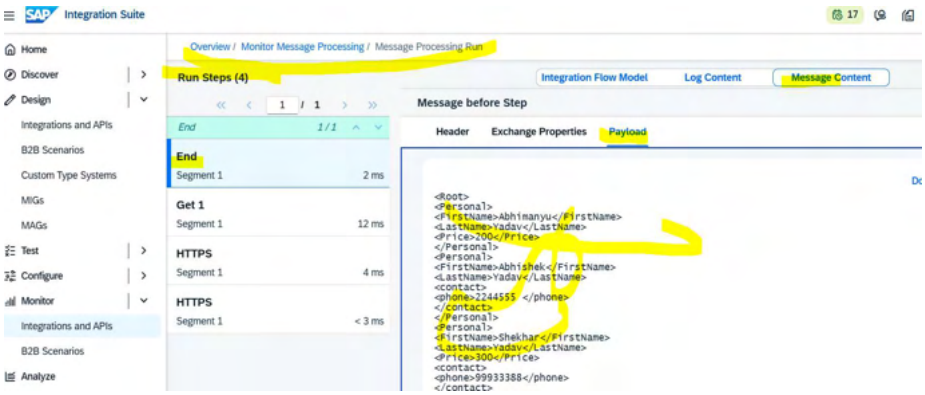


Note: Provide the same name in “data Store Name” as DB name (TestDataStoreDB) and table name (testdata) because while creatation / writing time we used the same names.

- Save and Deploy
- Execute from POSTPAN by calling GET URL to fetch the data from DB/Table

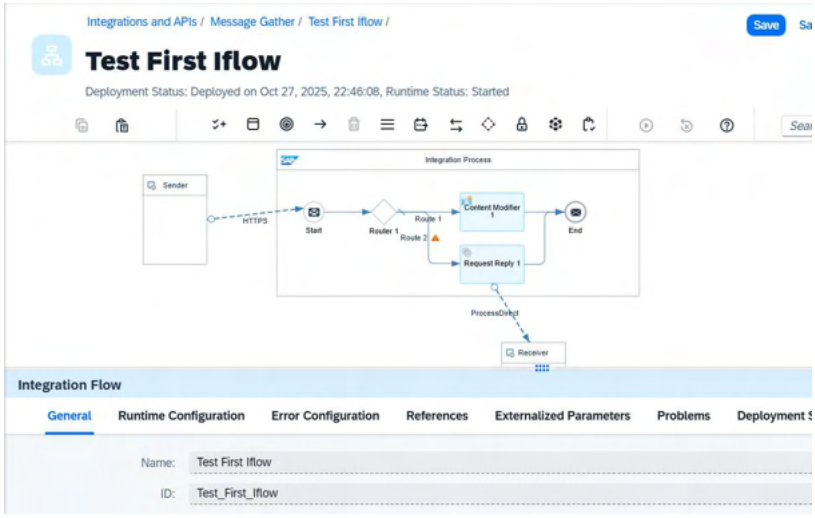


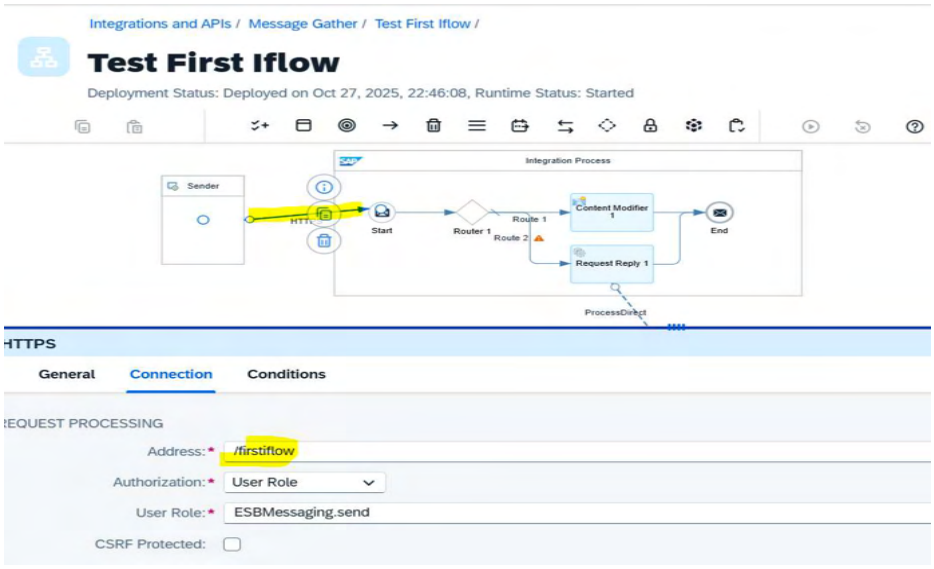
The same information can be observed form iFlow Message /Monitoring section:

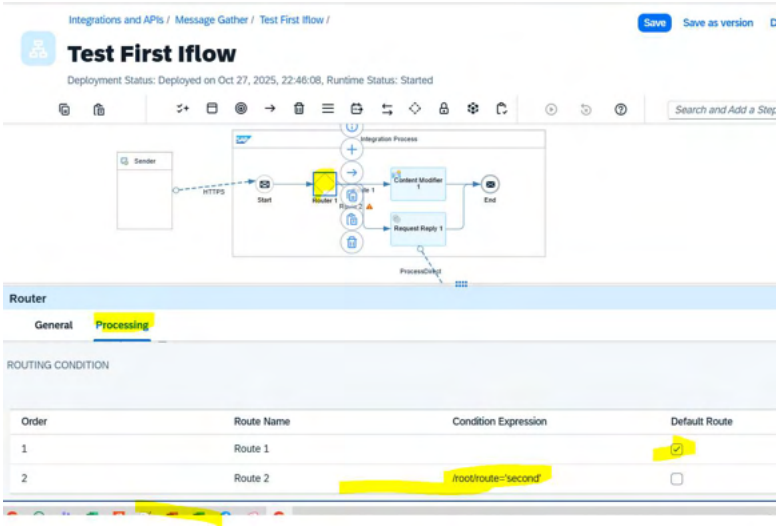


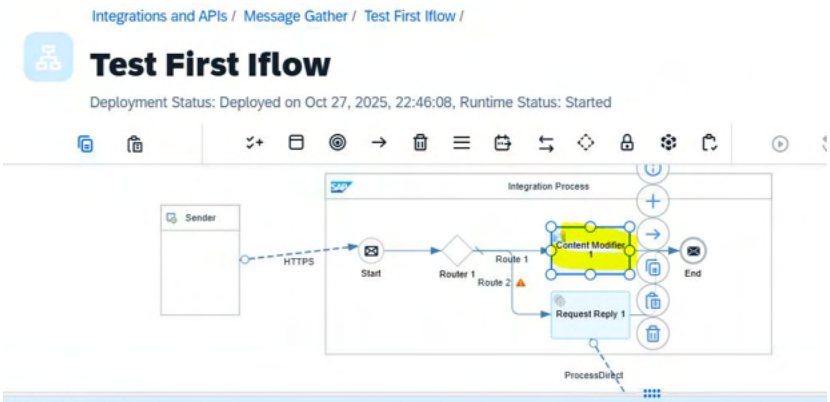
Scenerio 8: Create two iFlows and configure one iFlow to call the other.

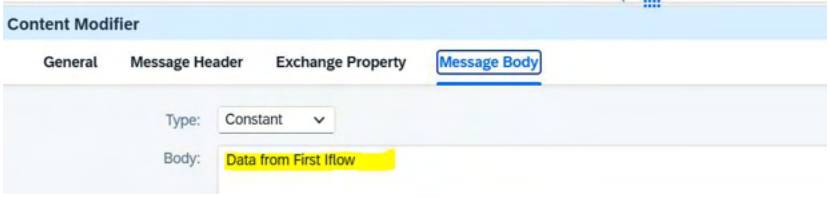
- Create first IFlow:

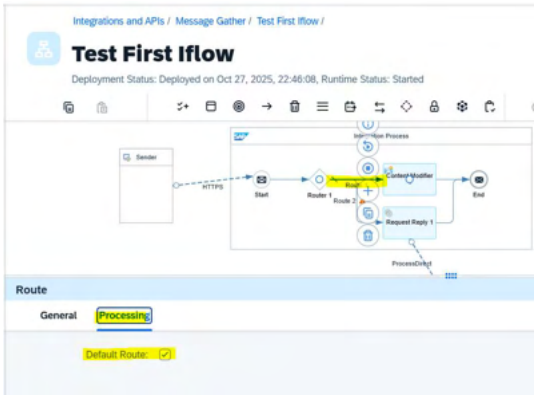


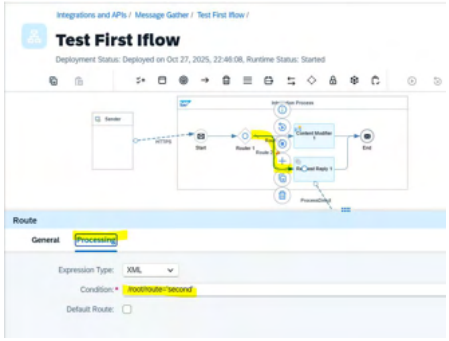


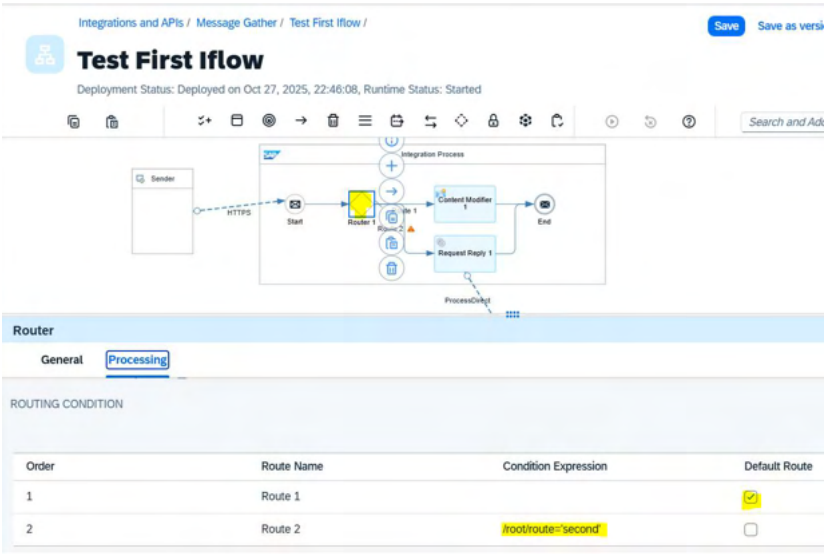


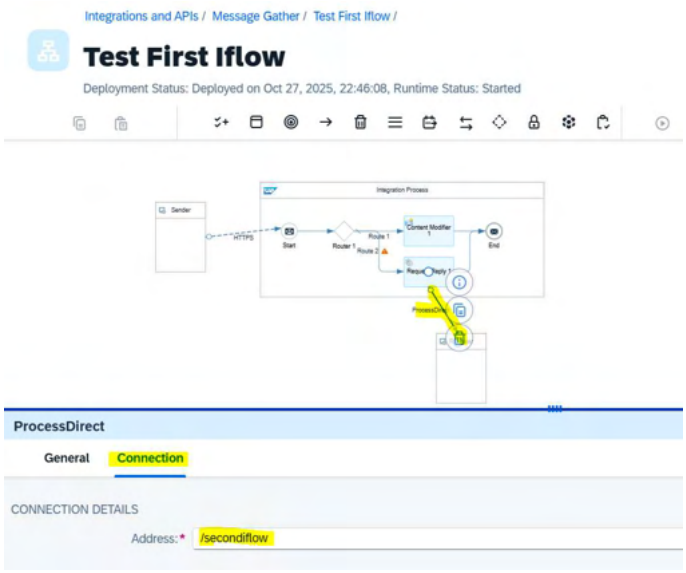

Integrations and APIs / Message Gather / Test First Iflow /
Test First Iflow
Deployment Status: Deployed on Oct 27, 2025, 22:46:08, Runtime Status: Started
Integration Process
Sender
HTTPS
Start
Router 1
Route 1
Route 2
Content Modifier 1
End
Request Reply 1
ProcessDirect


Content Modifier
General
Message Header
Exchange Property
Message Body
Type:
Constant
Body:
Data from First Iflow

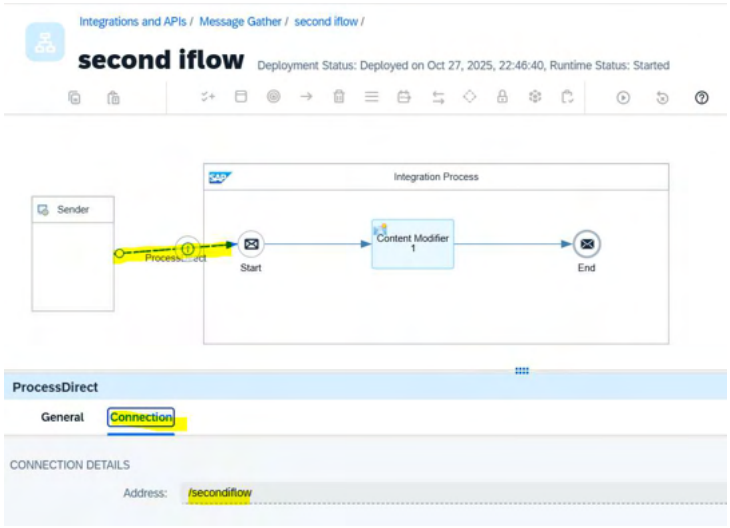

Integrations and APIs / Message Gather / Test First Iflow /
Test First Iflow
Deployment Status: Deployed on Oct 27, 2025, 22:46:08, Runtime Status: Started
Route
General
Processing
Default Route:

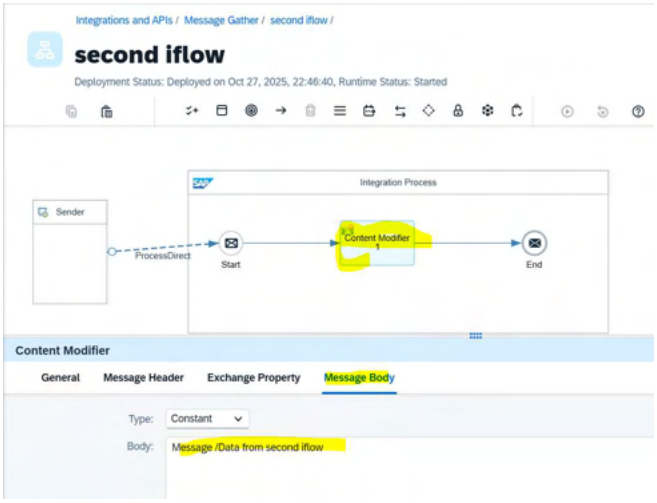

Test First Iflow
Route
General
Processing


Integrations and APIs / Message Gather / Test First Iflow /
Save
Test First Iflow
Deployment Status: Deployed on Oct 27, 2025, 22:46:08, Runtime Status: Started
Router
General
Processing
ROUTING CONDITION
Order
Route Name
Condition Expression
Default Route
1
Route 1
2
Route 2
/root/route='second'



- Create 2nd Iflow:

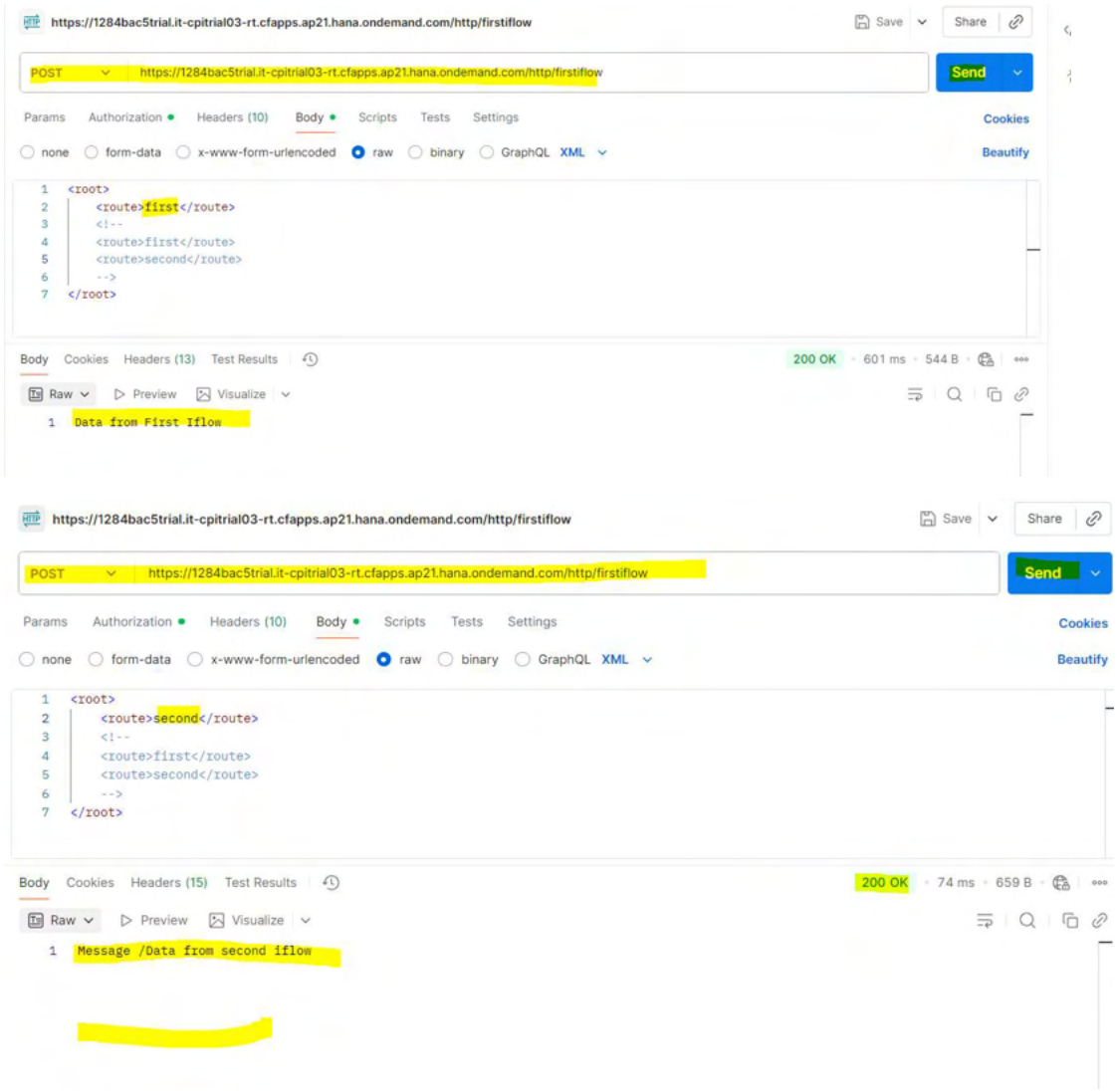




Validation using POSTMAN:

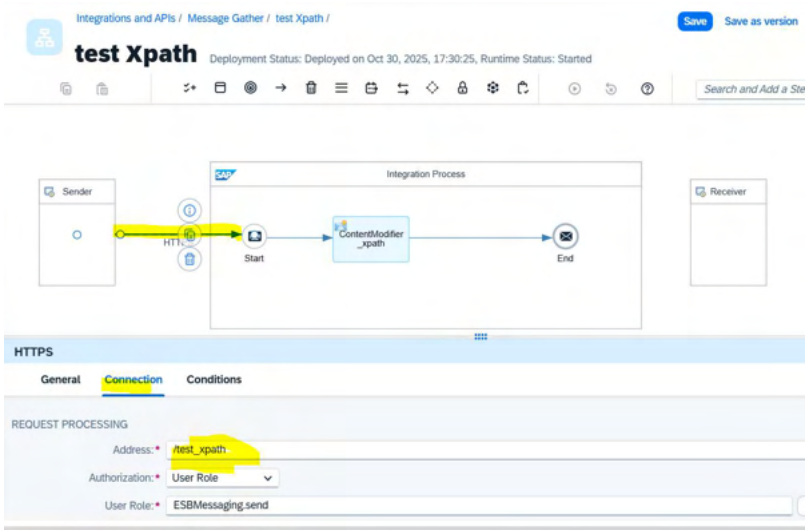
If the input in the <route> tag is **anything other than "second"**, the response will come from the **first iFlow** based on its configured body.

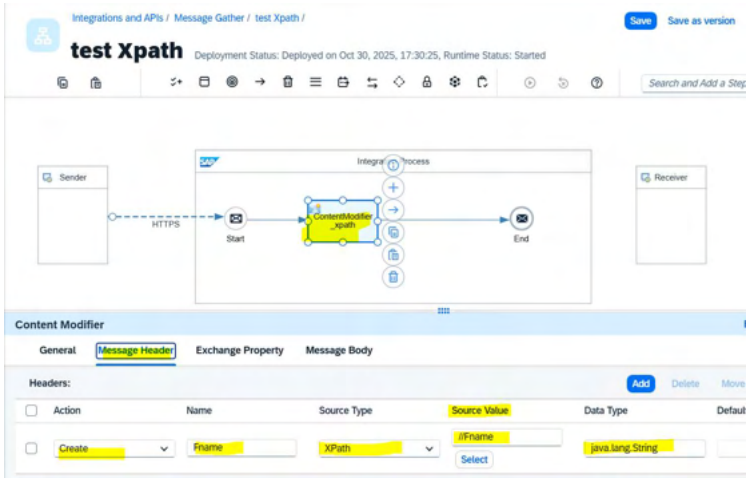
If the <route> tag value is **"second"**, the response will come from the **second iFlow** as per its configured body.



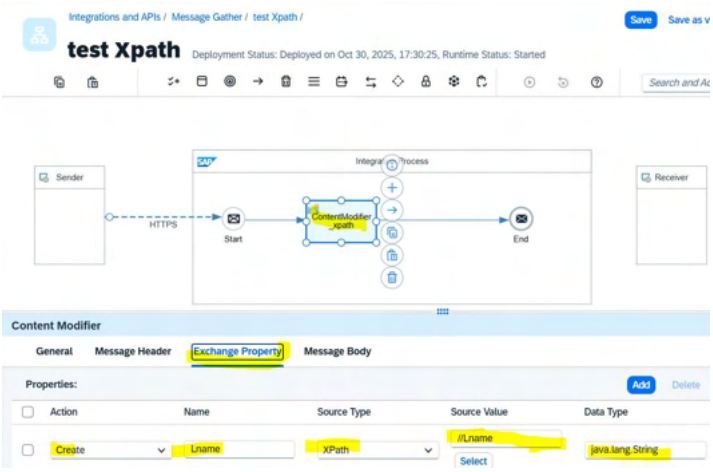
Scenario 9: XPath in Cloud Integration

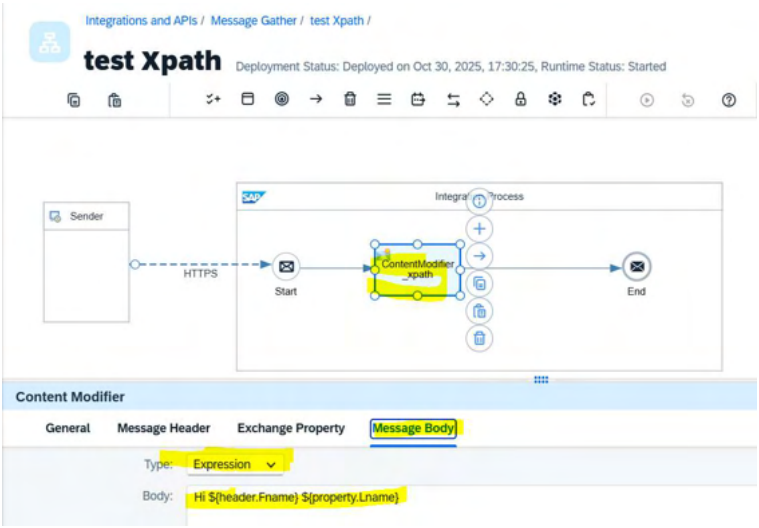
XPath is used to identify the exact **tag path or hierarchy** in the input/source XML file. For example, to access <Fname> or <Lname>, the precise path from the input XML must be configured in the iFlow.



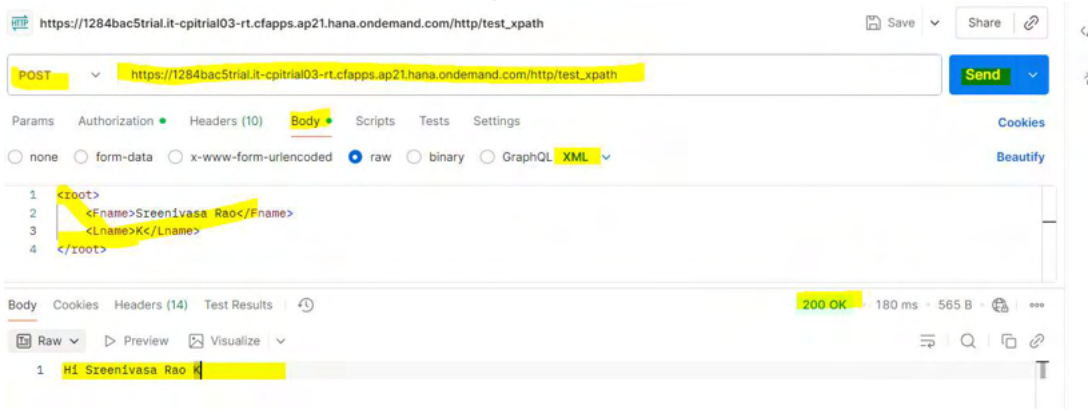


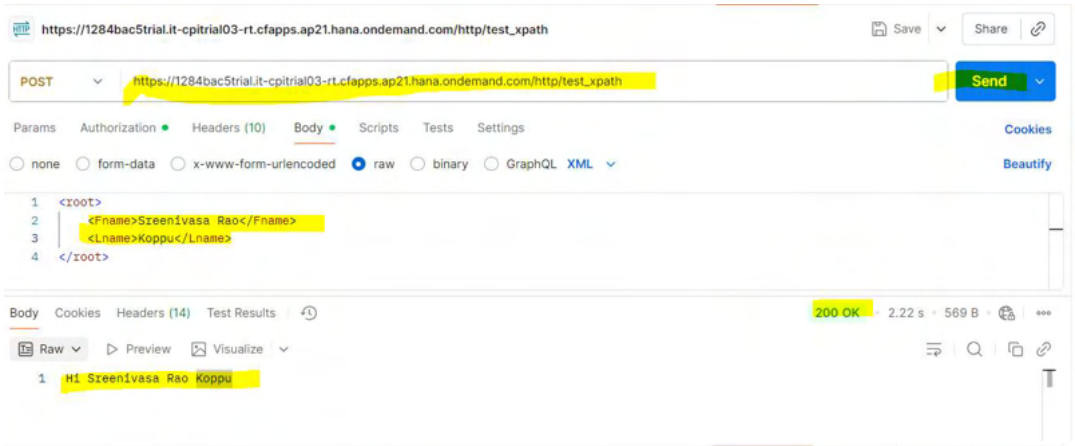
Note: “Source Value” (//Fname) to be given exact HTML tag in XML file. Or path can be given line “/root/Fname”.

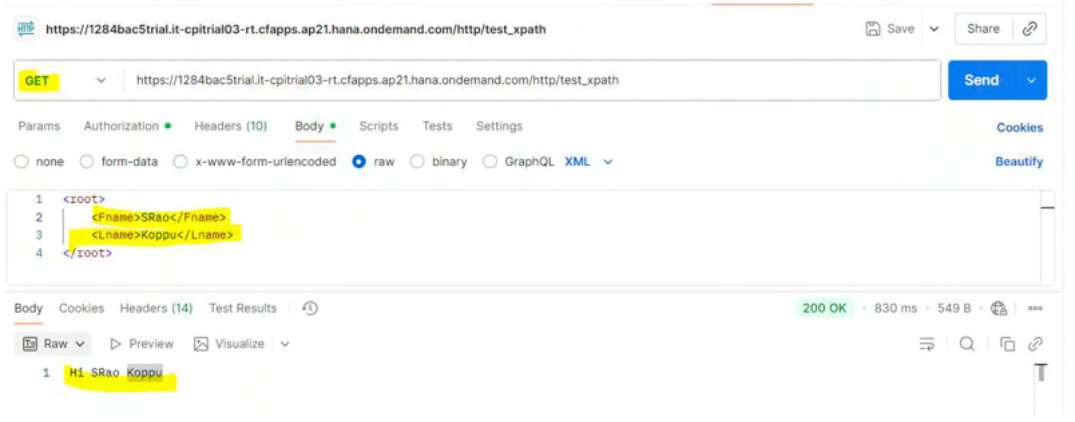




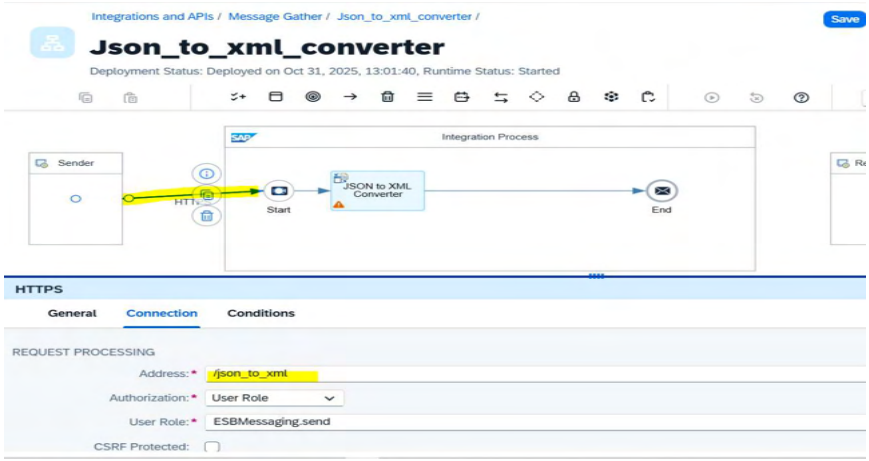
- Call the URL from POSTMAN to see the output.

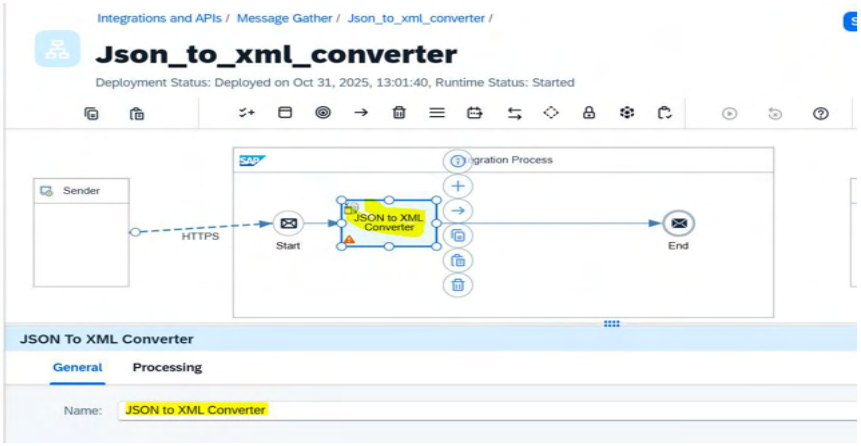


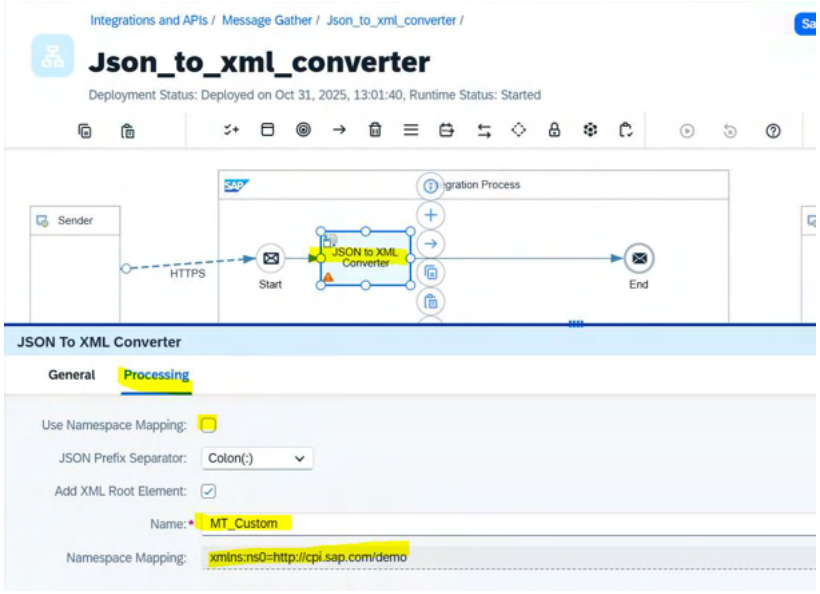




Scenario 10: Converter (JSON to XML)





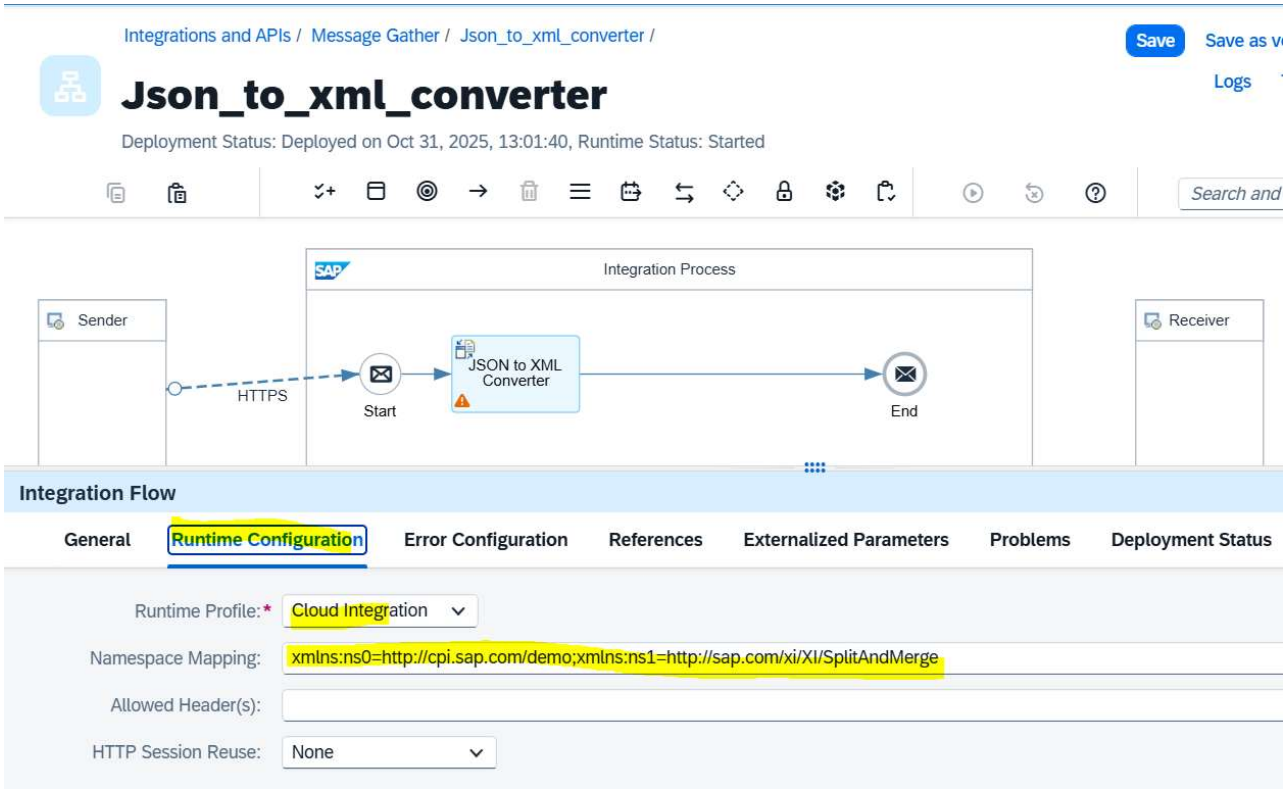


Note: Here “MT_Custom” is the name space in the XML file (i.,e default is “root”)
Namespace Mapping: copy / configure the Namespace path:

Namespace configuration:

- Click on outside iFlow

- Select “Runtime Configuraiton”
- Add the namespace Mapping (the below snamp shot namespace taken from the tutorial where it is working) → xmlns:ns0=http://cpi.sap.com/demo;xmlns:ns1=http://sap.com/xi/XI/SplitAndMerge



- Save and deploy and execute from POSTMAN