

SAP BTP Integration Suite

Contents

About:.....	2
Scenario 1: Encoder or Decoder	2
Scenario 2: “Combine” and “Enrich” options by using Odata Protocol (iFlow) (.....	4
Scenario 3: CSV to XML conversion	11
Scenario 4: Filtering	13
Scenario 5: Filter and display the output for records that contain the “contact” field or a specific tag/field in the input data.....	16
Scenario 6: How to Capture the Source and Target logs by using Groovy Script.....	18
Scenario 7: Data Store and Fetch	23
Scenerio 8: Creating 2 iflows and Call 1 Iflow from anotehr IFlow	28
Scenario 9: XPath in Cloud Integration	32
Scenario 10: Converter (JSON to XML)	34

About:

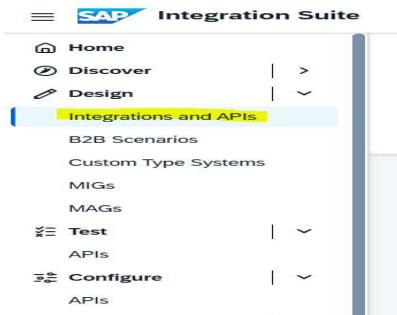
This document provides a step-by-step guide on creating iFlows using the SAP BTP Integration Suite. Due to confidentiality and security restrictions, details from the live project cannot be shared. Therefore, this Proof of Concept (POC) document demonstrates similar scenarios and examples for reference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

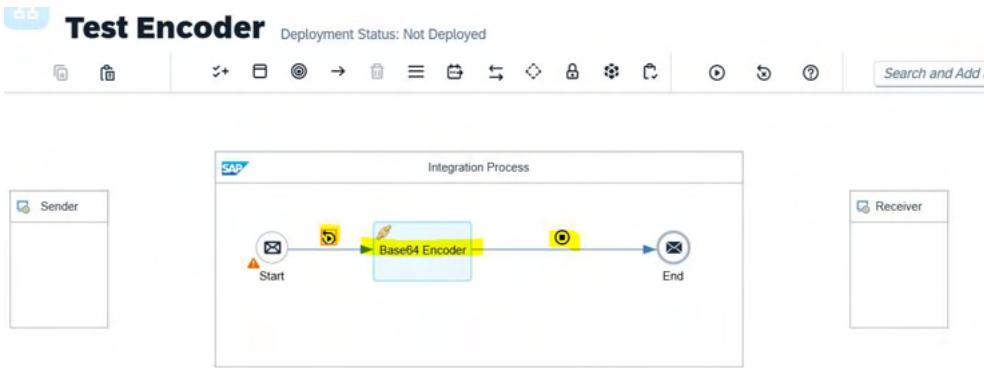
A screenshot of the SAP Integration Suite interface showing a package named 'Encoder and Decoder'. The package ID is 'EncoderandDecoder'. It shows details such as Vendor: Editable, Version: 1.0.0, and various creation and modification dates and emails. The 'Overview' tab is selected.

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

A screenshot of the SAP Integration Suite interface showing the 'Artifacts' tab for the 'Encoder and Decoder' package. A context menu is open over the 'Add' button, with 'Integration Flow' highlighted. The list of artifacts includes Value Mapping, Imported Archives, API, Message Mapping, Function Libraries, Integration Flow, Integration Adapter, Script Collection, Service Interface, Data Type, and Message Type.

The screenshot shows the 'Encoder and Decoder' interface in SAP Integration Studio. A modal window titled 'Add Integration Flow' is open, prompting for the creation of a new flow. The flow is named 'Test Encoder' with ID 'Test_Encoder'. The runtime profile is set to 'Cloud Integration'. The 'Description' field contains 'Test Encoder'. The 'Sender' is listed as 'Sender' and the 'Receiver' as 'Receiver'. At the bottom of the modal are buttons for 'Ans' (Cancel), 'Add and Open in Editor' (highlighted in yellow), and 'Save'.

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

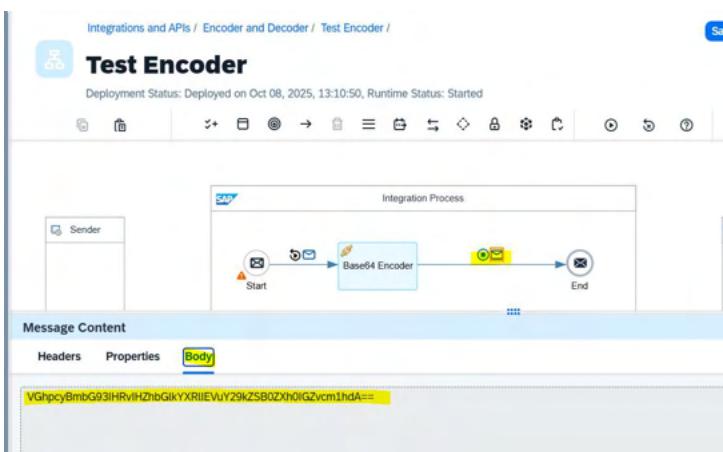


Add the information / input in “Simulator start body”

The screenshot shows the 'Add Simulation Input' dialog. It has sections for 'Header', 'Properties', and 'Body'. In the 'Body' section, there is a 'File:' input field and a 'Content:' text area containing the note: 'This row to validate Encode text format.' Buttons at the bottom include 'Upload from File System', 'OK', and 'Cancel'.

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/>)

Decode from Base64 format
Simply enter your data then push the decode button.

VGhpcyBmbG93IHRvIHZhbgIKYXRlEVuY29kZSB0ZXh0IGZvcmlhdA==

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

Source character set: UTF-8

Decode each line separately (useful when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE Decodes your data into the area below.

This flow to validate Encode text format

Scenario 2: “Combine” and “Enrich” options by using Odata Protocol (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/>

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<service xmlns="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom" xml:base="https://services.odata.org/V2/OData/Odata.svc/">
  <@workspace>
    <atom:title>Default</atom:title>
    <@collection href="Products">
      <atom:title>Products</atom:title>
    </collection>
    <@collection href="Categories">
      <atom:title>Categories</atom:title>
    </collection>
    <@collection href="Suppliers">
      <atom:title>Suppliers</atom:title>
    </collection>
  </@workspace>
</service>
```

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

Integrations and APIs / Encoder and Decoder /

Encoder and Decoder

Add Integration Flow

Create Upload

Name: * Odata Enricher

ID: * Odata_Enricher

Runtime Profile: Cloud Integration

Description: Odata Enriched

Sender: Sender

Receiver: Receiver

Add Add and Open in Editor Cancel

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

Integrations and APIs / Encoder and Decoder / Odata Enricher

Odata Enricher Deployment Status: Not Deployed

Integration Process

```

graph LR
    StartTimer1((Start Timer 1)) --> ReqReply[Reqe[ue] Reply]
    ReqReply --> ContentEnricher1[Content Enricher 1]
    ContentEnricher1 --> End((End))
    
```

OData

General Connection Processing

CONNECTION DETAILS

Address: <https://services.odata.org/V2/OData/OData.svc/>

Proxy Type: Internet

Authentication: None

Click on “Processing” → “select”

Integrations and APIs / Encoder and Decoder / Odata Enricher

Odata Enricher Deployment Status: Deployed on Oct 11, 2025, 17:23:39, Runtime Status: Started

Search and Add a Step

General Connection Processing

PROCESSING DETAILS

Operation Details: Query (GET)

Resource Path: Products

Query Options: \$select=ID,Name,Description,ReleaseDate,DiscontinuedDate,Rating,Price

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

Model Operation

1. Connect to System

2. Select Entity & Define Operation

1. Connect to System

Connection Source: Local EDMX File

EDMX File: services_odata_org_V2_OData_OData.svc.edmx

2. Select Entity & Define Operation

Operation: Query (GET)

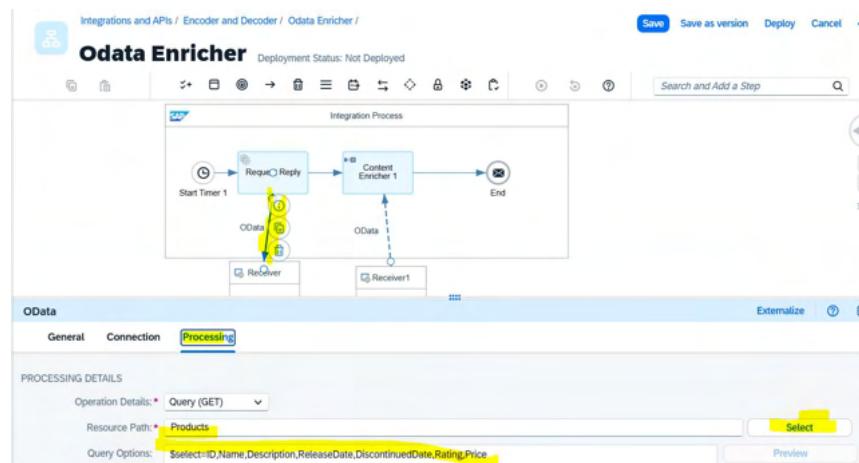
Select Entity: Products

Generate XML Schema Definition

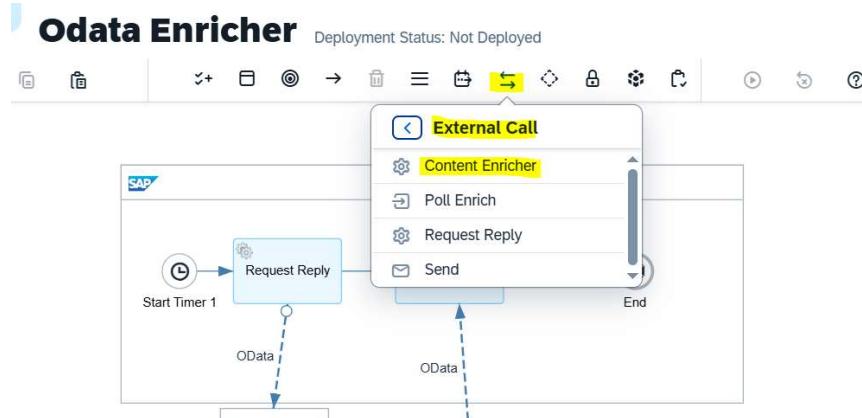
Fields Filter: Select All Fields

Click finish

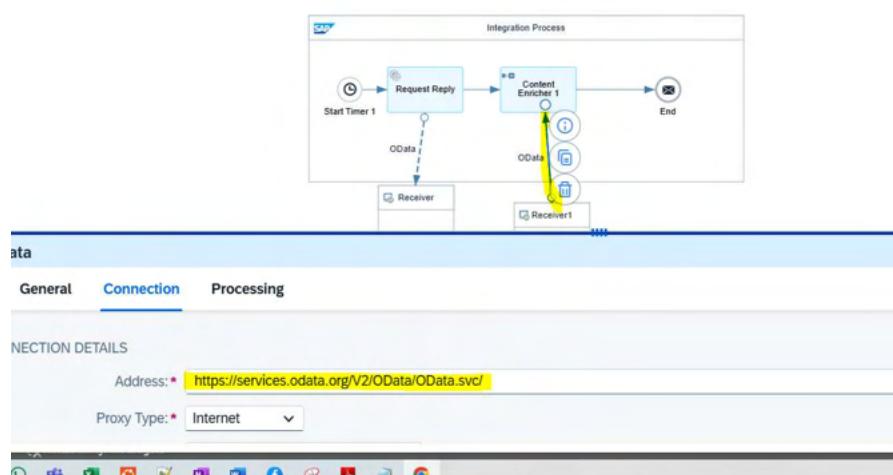
After “finish” iFlow as below:



Add the “External call → select “Content Enricher”



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



Select the “Supplier” option from Processing

Model Operation

1 Connect to System 2 Select Entity & Define Operation

2. Select Entity & Define Operation

Operation :* Query (GET) Sub Levels : 0

Select Entity :* Suppliers

Generate XML Schema Definition

Fields Filter

Fields

- ID
- Name
- Concurrency
- Address

Select “Content Enricher 1” and configure as below:

Integrations and APIs / Encoder and Decoder / Odata Enricher /

Odata Enricher

Deployment Status: Deployed on Oct 11, 2025, 17:09:25, Runtime Status: Started

Content Enricher

General Processing

Aggregation Algorithm: Enrich

ORIGINAL MESSAGE

Path to Node: */Products/Product
Key Element: ID

LOOKUP MESSAGE

Path to Node: */Suppliers/Supplier
Key Element: ID

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

Integrations and APIs / Encoder and Decoder / Odata Enricher /

Odata Enricher Deployment Status: Not Deployed

OData

General Connection Processing

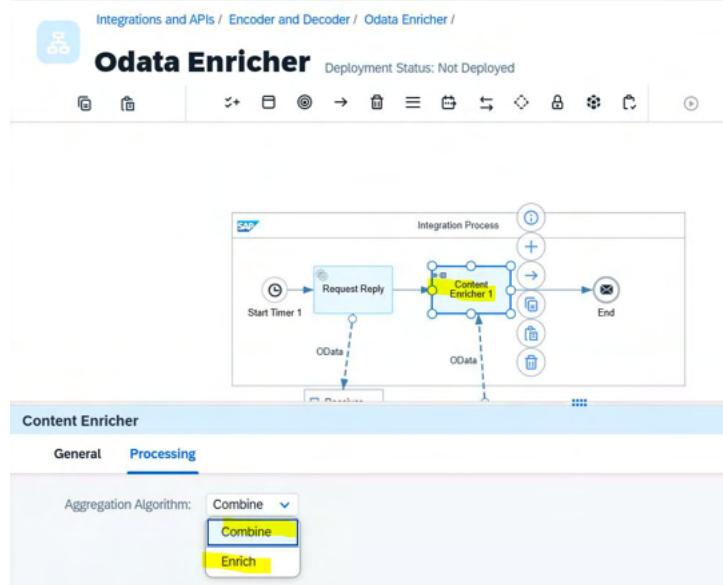
PROCESSING DETAILS

Operation Details: Query (GET)
Resource Path: Suppliers
Query Options: \$select=ID,Name,Concurrency,Address

Note: Content Enrich having 2 options

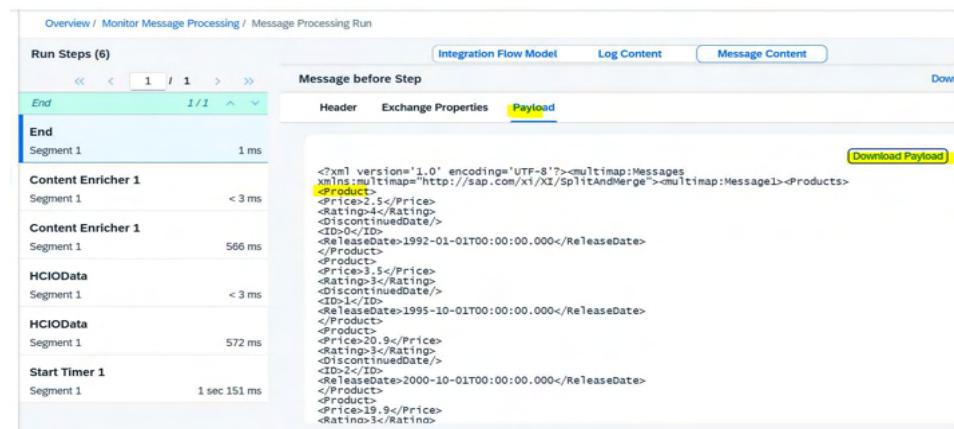
1. Combined (it will combine “product” and “Supplier” data in the end/output)
 2. Enrich (it will combine matched data of “product” and “Supplier data” based on the common filed i.e., “ID” will nest within XML and non-matched data at the end of output)

- **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>
</Suppliers>

```

- Output from “Enriched”:

Changes to be made below:

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

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

Integrations and APIs / Encoder and Decoder / Odata Enricher /

Odata Enricher

Deployment Status: Deployed on Oct 11, 2025, 17:09:25, Runtime Status: Started

Save

Content Enricher

General Processing

Aggregation Algorithm: Enrich

ORIGINAL MESSAGE

Path to Node: */Products/Product

Key Element: ID

LOOKUP MESSAGE

Path to Node: */Suppliers/Supplier

Key Element: ID

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

Overview / Monitor Message Processing / Message Processing Run

Run Steps (6) Integration Flow Model Log Content **Message Content**

Message before Step Down

Header Exchange Properties **Payload**

[Download Payload](#)

End Segment 1 2 ms

Content Enricher 1 Segment 1 < 3 ms

Content Enricher 1 Segment 1 538 ms

HCIOData Segment 1 < 3 ms

HCIOData Segment 1 566 ms

Start Timer 1 Segment 1 1 sec 119 ms

```

<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>
    <DiscontinuedDate/>
  </Product>

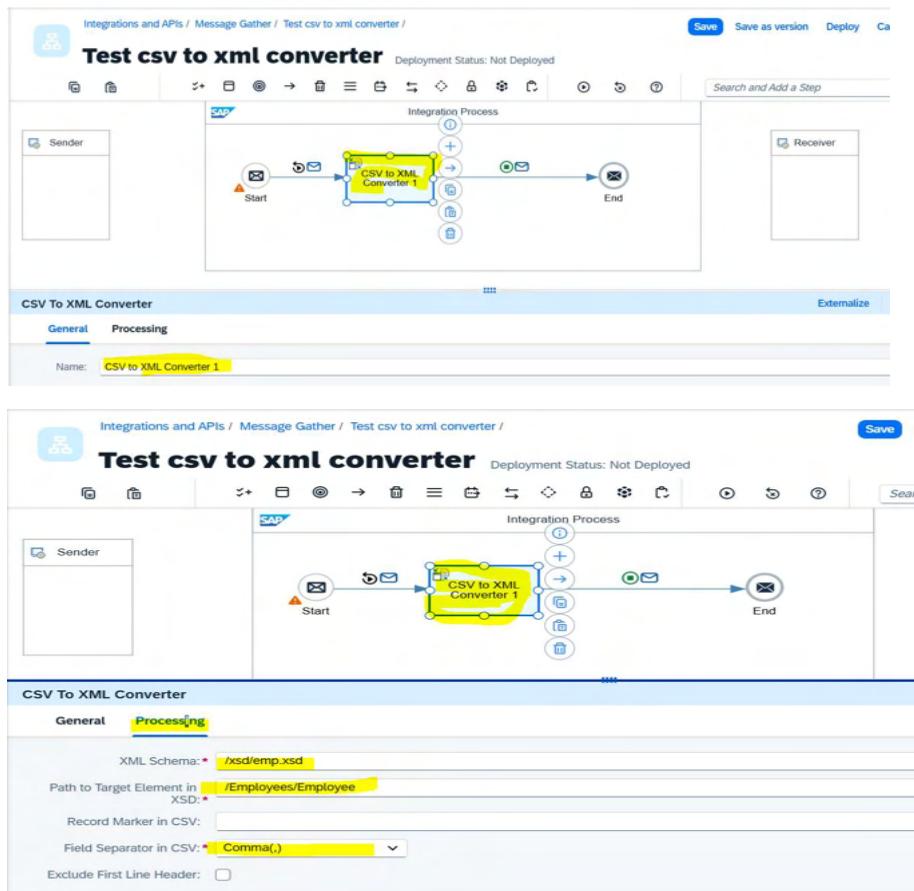
```

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.

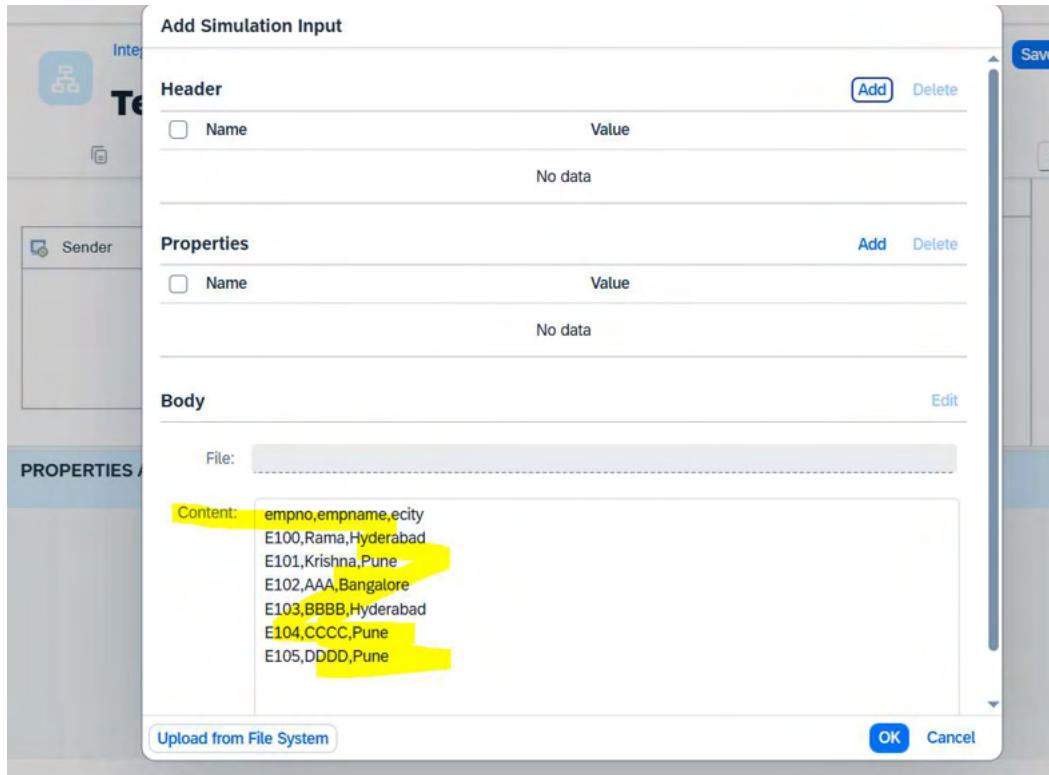


emp.xsd

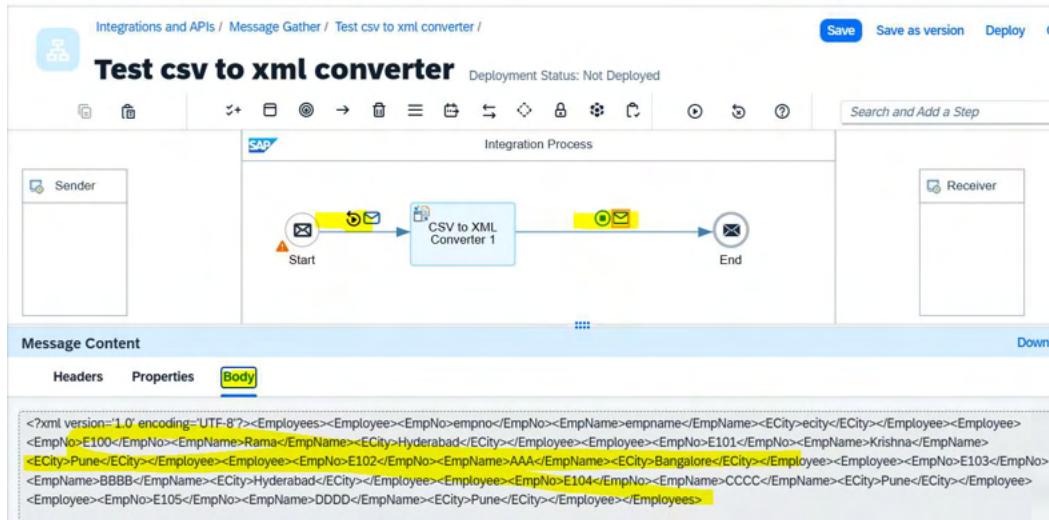
Sample data in CSV file:

```
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

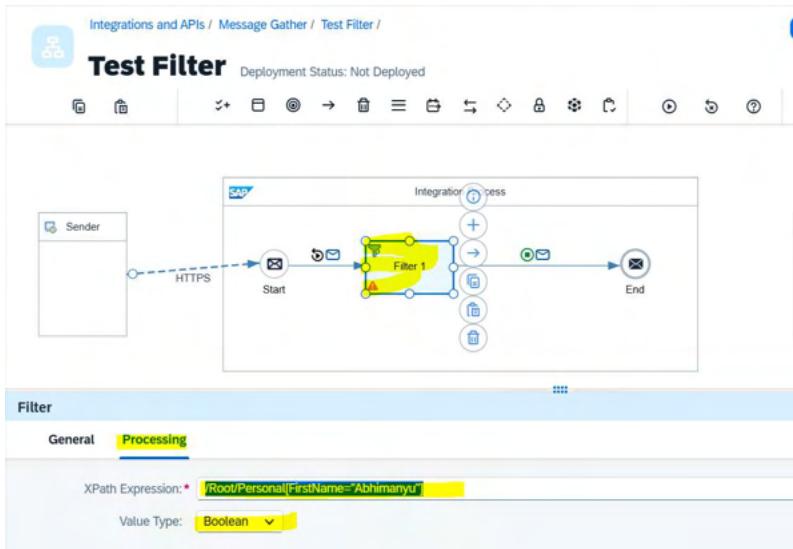
The below sample data used for this POC:



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

In the example we configured 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).



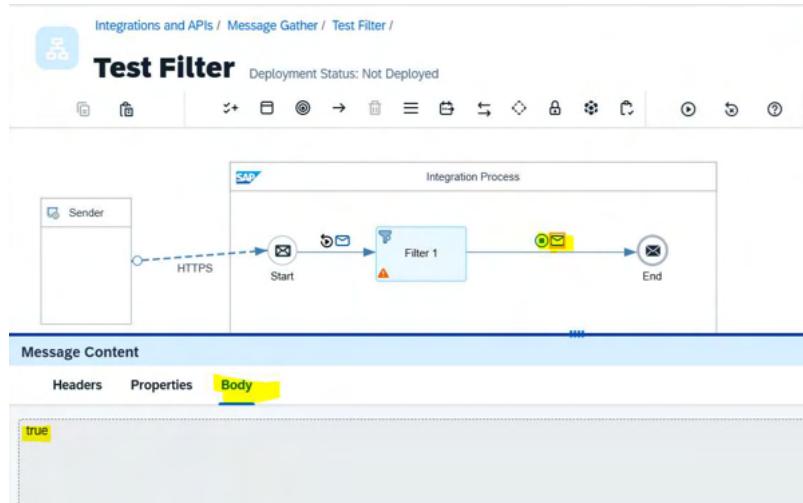
Input provided in the “simulation start body”

A screenshot of the SAP Integration Tester interface showing the "Add Simulation Input" dialog. It has three sections: "Properties" (empty), "Body" (with a file content editor showing XML code), and "OK" and "Cancel" buttons at the bottom. The "Body" section contains the following XML content:

```
<Root>
<Personal>
<FirstName>Abhimanyu</FirstName>
<LastName>Yadav</LastName>
</Personal>
<Personal>
<FirstName>Abhishek</FirstName>
<LastName>Yadav</LastName>
</Personal>
<Personal>
```

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**.

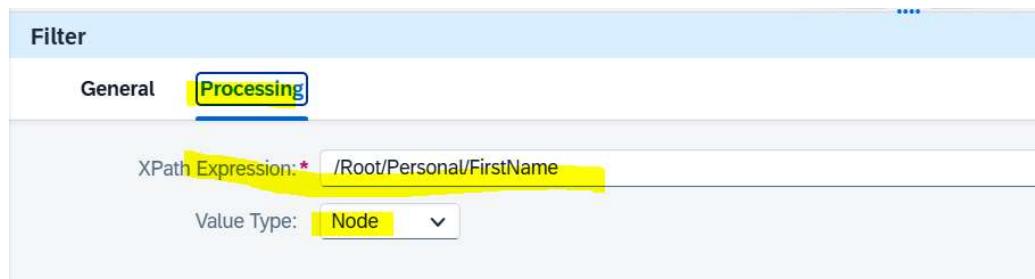
Seen the below output:



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 “Firname” values

Input in the simulation body:

The screenshot shows the "Add Simulation Input" dialog. It has sections for "Header", "Properties", and "Body". In the "Header" section, there is a table with one row and two columns: "Name" and "Value", both currently empty. In the "Properties" section, there is a similar table with one row and two columns: "Name" and "Value", both currently empty. In the "Body" section, there is a "File:" input field and a "Content:" text area. The "Content:" area contains the following XML code:

```

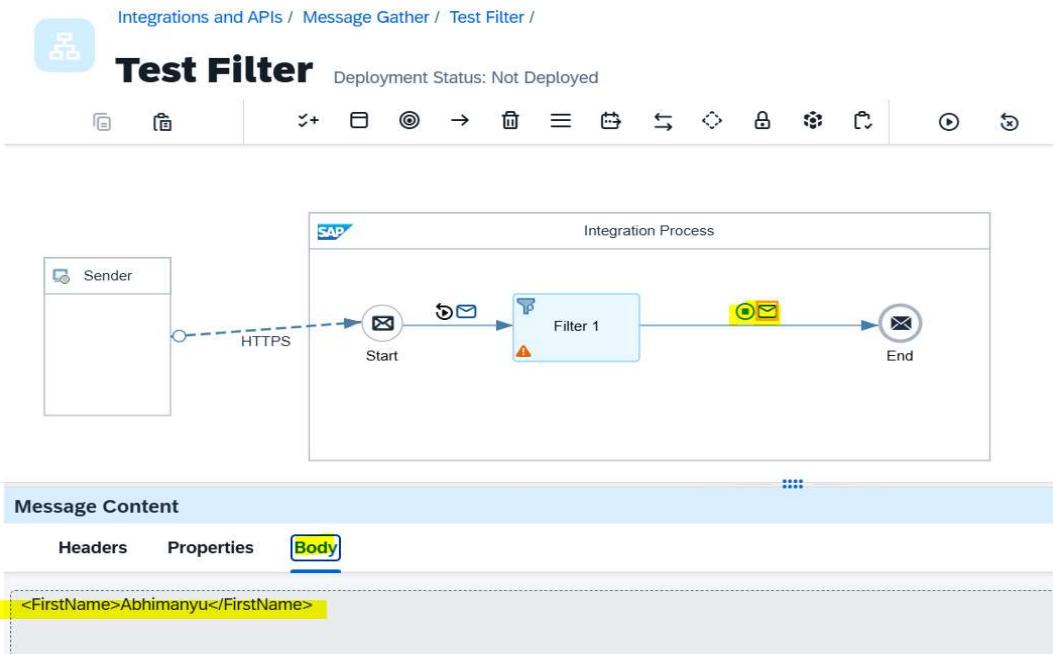
<Root>
  <Personal>
    <FirstName>Abhimanyu</FirstName>
    <LastName>Yadav</LastName>
  </Personal>
  <Personal>
    <FirstName>Abhishek</FirstName>
    <LastName>Yadav</LastName>
  </Personal>

```

At the bottom of the dialog are "OK" and "Cancel" buttons.

Ouput:

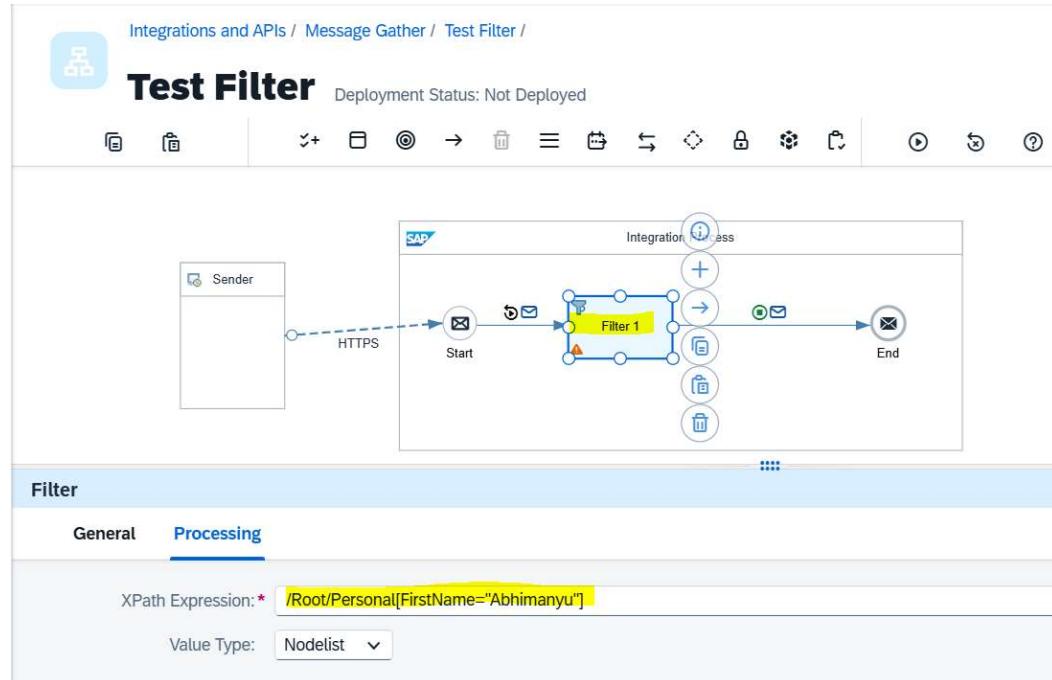
Eventhough "Firstanme" having mulitple times in the input (approx.. 3 times) output will **consider First Occurance** and display that value:



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

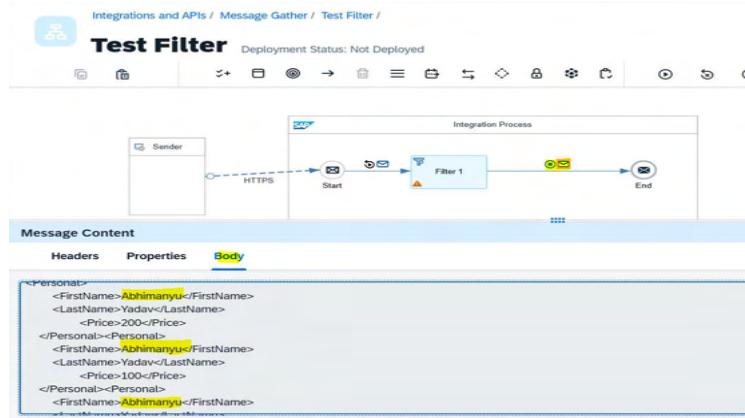
Outwill display all matched list / details (Single/multiple output)

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.

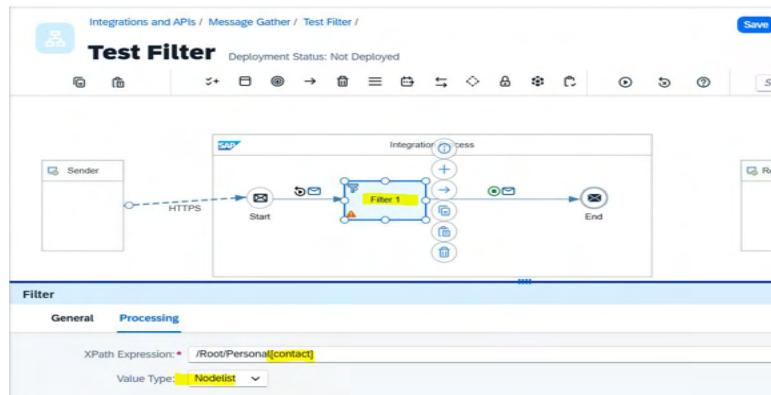


Scenario 5: Filter and display the output for records that contain the “contact” field or a specific tag/field in the input data.

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:

The screenshot shows the SAP Integration Tester interface for a 'Test Filter' deployment. The 'Integration Process' diagram at the top illustrates a flow from a 'Sender' node to a 'Start' event, followed by a 'Filter 1' node, and finally an 'End' event. The 'Message Content' section displays XML data under the 'Body' tab. The XML includes fields for FirstName, LastName, and phone numbers, with one entry for 'Yadav' highlighted.

```
<FirstName>Abhishek</FirstName>
<LastName>Yadav</LastName>
<<contact>
<phone>2244555 </phone>
</contact>
</Personal><Personal>
<FirstName>Shekhar</FirstName>
<LastName>Yadav</LastName>
<<contact>
<phone>99933388</phone>
```

Double filter: Scenario → Filter condition is Contact should be there along with Last Name as "Yadav"

This screenshot shows a more complex filter configuration. The 'Filter' section has the 'Processing' tab selected. The 'XPath Expression' field contains the condition '/Root/Personal[contact and LastName="Yadav"]'. The 'Value Type' dropdown is set to 'Nodelist'.

XPath Expression: /Root/Personal[contact and LastName="Yadav"]

Value Type: Nodelist

Output: Output is only Last name is "Yadav" along with that, that person having "contact".

The final screenshot shows the 'Message Content' after applying the double filter. Only the entry for 'Yadav' is present in the XML, as it is the only one that matches both the 'LastName' and the presence of a 'contact' node.

```
<FirstName>Abhishek</FirstName>
<LastName>Yadav</LastName>
<<contact>
<phone>2244555 </phone>
</contact>
</Personal><Personal>
<FirstName>Shekhar</FirstName>
<LastName>Yadav</LastName>
<<contact>
<phone>99933388</phone>
```

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

Integrations and APIs / Message Gather / Common testing /

Save Save as version

Common testing

Deployment Status: Deployed on Oct 17, 2025, 16:50:35, Runtime Status: Started

Search and Add a Step

Integration Process

Sender → HTTPS → Start → Groovy Script 1 → Base64 Encoder → Base64 Decoder → Groovy Script 2 → End

Groovy Script

General Processing

Script File: `/script/script1.groovy`

Select

Script Function:

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:

Integrations and APIs / Message Gather / Common testing / script1.groovy / [Apply](#)

script1.groovy

```
1 import com.sap.gateway.ip.core.customer.util.Message;
2
3 - def Message processMessage(Message message) {
4     def body = message.getBody(java.lang.String) as String;
5     def messageLog = messageLogFactory.getMessageLog(message);
6
7 -     if (messageLog != null) {
8         messageLog.setStringProperty("Logging#1", "Printing Payload As Attachment");
9         messageLog.addAttachmentAsString("Source Payload", body, "text/plain");
10    }
11
12    return message;
13 }
14
```

Problems

[Upgrade Readiness Checks](#) [General Script Checks](#)

Type	Description	Actions
------	-------------	---------

Groovy Script at Target:

Integrations and APIs / Message Gather / Common testing /

Save Save as...

Common testing

Deployment Status: Deployed on Oct 17, 2025, 16:50:35, Runtime Status: Started

Search and...

Integration Process

Sender → HTTPS → Start → Groovy Script 1 → Base64 Encoder 1 → Base64 Decoder 1 → Groovy Script 2 (Parallel Paths) → Groovy Script 2 → End

Groovy Script

General Processing

Script File: `/script/script2.groovy`

Select

Script Function:

Groovy script at Target:

```

Integrations and APIs / Message Gather / Common testing / script2.groovy /
script2.groovy

1 import com.sap.gateway.ip.core.customdev.util.Message;
2
3 def processData(Message message) {
4     def body = message.getBody(java.lang.String) as String;
5     def messageLog = messageLogFactory.getMessageLog(message);
6
7     if (messageLog != null) {
8         messageLog.setStringProperty("Logging#1", "Printing Payload As Attachment");
9         messageLog.addAttachmentAsString("Target Payload:", body, "text/plain");
10    }
11
12    return message;
13 }
14

```

Problems

Upgrade Readiness Checks General Script Checks

Type	Description
No data	

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

Overview / Monitor Message Processing Message Status Overview

Time: Past Hour Status: All Type: All Package: All Artifacts: Common test ID: Message, Correlation or Ap...
Oct 17, 2025, 16:09:38 - Oct 17, 2025, 17:09:38

Common testing
Last Updated at: Oct 17, 2025, 16:53:32

Status	Properties	Logs	Attachments	Artifact Details
Status				
✓ Message processing completed successfully. Processing Time: 105 ms				
Properties				
Message ID: AGlyJzS1gfAGIOjUyM6cJeFdUVd Correlation ID: AGlyJzRxDmOrTVLYxp8VfpGSJij				
Retention Periods				

Overview / Monitor Message Processing Message Status Overview Hide Filter Bar

Time: Past Hour Status: All Type: All Package: All Artifacts: Common test ID: Message, Correlation or Ap...
Oct 17, 2025, 16:09:38 - Oct 17, 2025, 17:09:38

Common testing
Last Updated at: Oct 17, 2025, 16:53:32

Status	Properties	Logs	Attachments	Artifact Details															
Attachments																			
Entries (2) <table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> <th>Modified At</th> <th>Size</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>Source Payload</td> <td>text/plain</td> <td>Oct 17, 2025, ...</td> <td>1 KB</td> <td></td> </tr> <tr> <td>Target Payload</td> <td>text/plain</td> <td>Oct 17, 2025, ...</td> <td>1 KB</td> <td></td> </tr> </tbody> </table>					Name	Type	Modified At	Size	Action	Source Payload	text/plain	Oct 17, 2025, ...	1 KB		Target Payload	text/plain	Oct 17, 2025, ...	1 KB	
Name	Type	Modified At	Size	Action															
Source Payload	text/plain	Oct 17, 2025, ...	1 KB																
Target Payload	text/plain	Oct 17, 2025, ...	1 KB																

Source Payload / Log:

Overview / Monitor Message Processing / Message Processing Log Attachments

Common testing

Last Updated at: Oct 17, 2025, 16:53:32 Log Level: Info
Status: Completed Processing Time: 105 ms

Log **Attachments**

Source Payload **Target Payload:**

This is my first HTTPS iflow of Protocol, testing with Groovy script for login !!!

Overview / Monitor Message Processing / Message Processing Log Attachments

Common testing

Last Updated at: Oct 17, 2025, 16:53:32 Log Level: Info
Status: Completed Processing Time: 105 ms

Log **Attachments**

Source Payload **Target Payload:**

This is my first HTTPS iflow of Protocol, testing with Groovy script for login !!!

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

Integration Process

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 /

Common testing Deployment Status: Deployed on Oct 17, 2025, 16:50:35, Runtime Status: Started

Edit Configure Search Step

Integration Process

```

graph LR
    Sender[Sender] -- "HTTPS" --> Start((Start))
    Start --> GS1[Groovy Script 1]
    GS1 --> B64E[Base64 Encoder]
    B64E --> B64D[Base64 Decoder]
    B64D --> GS2[Groovy Script 2]
    GS2 --> End((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 /

Common testing Deployment Status: Deployed on Oct 17, 2025, 16:50:35, Runtime Status: Started

Edit Configure Search Step

Integration Process

```

graph LR
    Sender[Sender] -- "HTTPS" --> Start((Start))
    Start --> GS1[Groovy Script 1]
    GS1 --> B64E[Base64 Encoder]
    B64E --> B64D[Base64 Decoder]
    B64D --> GS2[Groovy Script 2]
    GS2 --> End((End))

```

Message Content

Headers Properties **Body**

VGhpcyBpcyBmb3IgR3Jvb3Z5IHnjcmIwdCB0ZXN0aW5nIHrvdGNhcHR1cmUgdGhlGkvZyBmb3Ig
Ym90aCZb3VvY2UgY5klHrcmdldCaoRW5jcnIwdGluZyBhbmOgRGVjcnIwdGluZyk=

Integrations and APIs / Message Gather / Common testing /

Common testing Deployment Status: Deployed on Oct 17, 2025, 16:50:35, Runtime Status: Started

Edit Configure Deploy Search Step

Integration Process

```

graph LR
    Sender[Sender] -- "HTTPS" --> Start((Start))
    Start --> GS1[Groovy Script 1]
    GS1 --> B64E[Base64 Encoder]
    B64E --> B64D[Base64 Decoder]
    B64D --> GS2[Groovy Script 2]
    GS2 --> End((End))

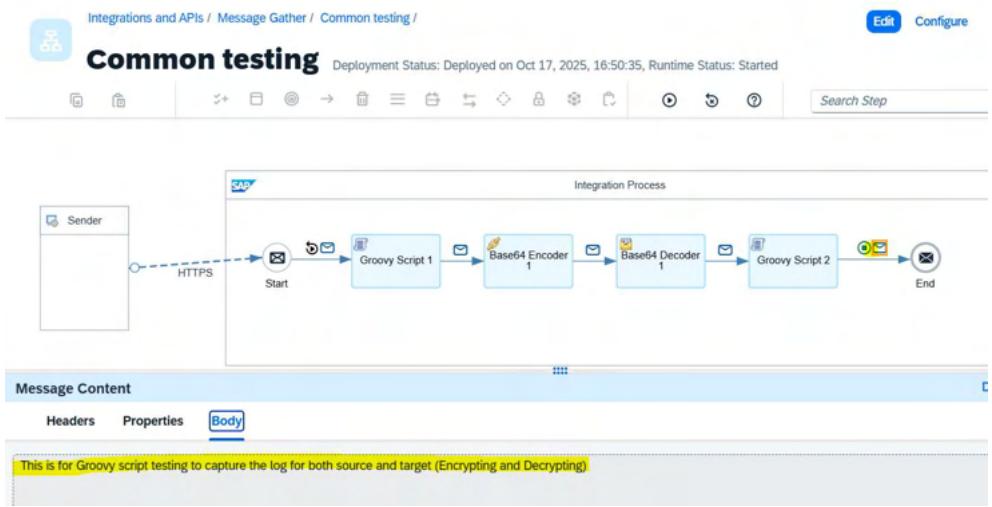
```

Message Content

Headers Properties **Body**

This is for Groovy script testing to capture the log for both source and target (Encrypting and Decrypting)

Download



Logs from the iflow:

Overview / Monitor Message Processing

Time: Past Hour Status: Co... Type: All Package: All Artifacts: All ID: Message, Correlation or Applic... Nov 07, 2025, 13:06:39 - Nov 07, 2025, 14:06:39

Messages (1)	
Artifact Name	Status
Common testing	Completed
Nov 07, 2025, 14:04:17	352 ms

Common testing

Last Updated at: Nov 07, 2025, 14:04:17

Status Properties Logs **Attachments** Artifact Details

Attachments

Entries (2)				
Name	Type	Modified At	Size	Action
Source Payload	text/plain	Nov 07, 2025, ...	1 KB	Download
Target Payload:	text/plain	Nov 07, 2025, ...	1 KB	Download

Artifact Details

Source payload log:

SAP Integration Suite

Home Discover Design Test Configure Monitor Analyze Engage Inspect Monetize Settings

Overview / Monitor Message Processing / Message Processing Log Attachments

Common testing

Last Updated at: Nov 07, 2025, 14:04:17 Log Level: Trace
Status: Completed Processing Time: 352 ms

Log **Attachments**

Source Payload Target Payload

This is for Groovy script testing to capture the log for both source and target (Encrypting and Decrypting)

Target payload log:

SAP Integration Suite

Home > Overview / Monitor Message Processing / Message Processing Log Attachments

Common testing

Last Updated at: Nov 07, 2025, 14:04:17 Log Level: Trace
Status: Completed Processing Time: 352 ms

Log **Attachments**

Source Payload **Target Payload**

This is for Groovy script testing to capture the log for both source and target (Encrypting and Decrypting)

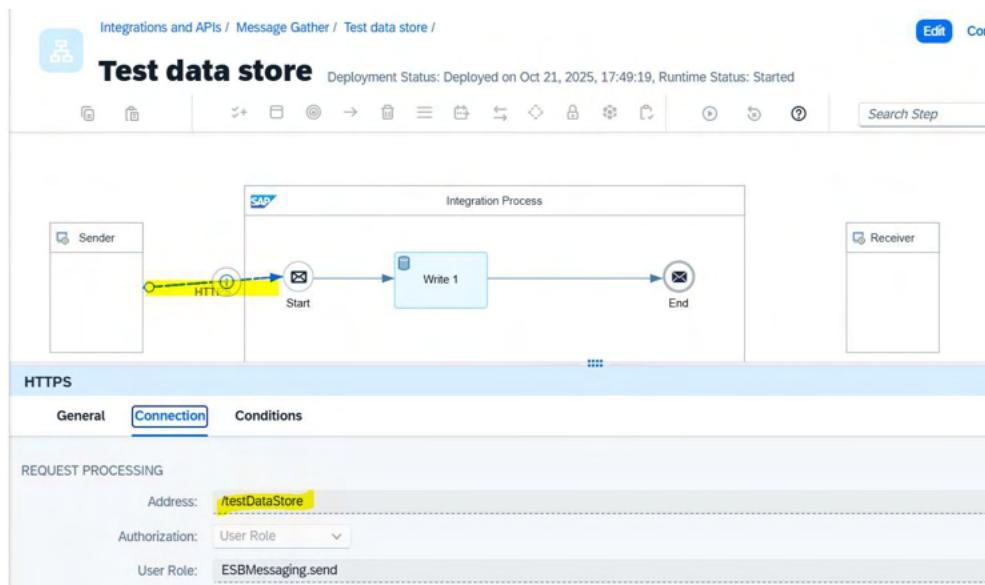
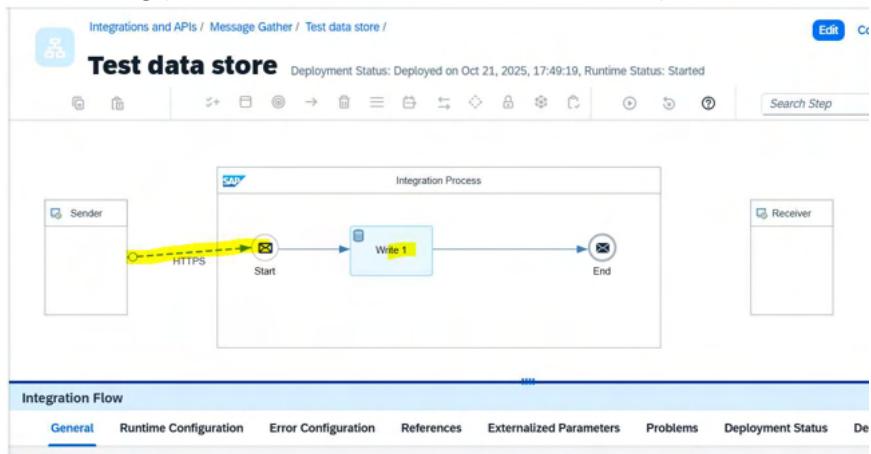
=====

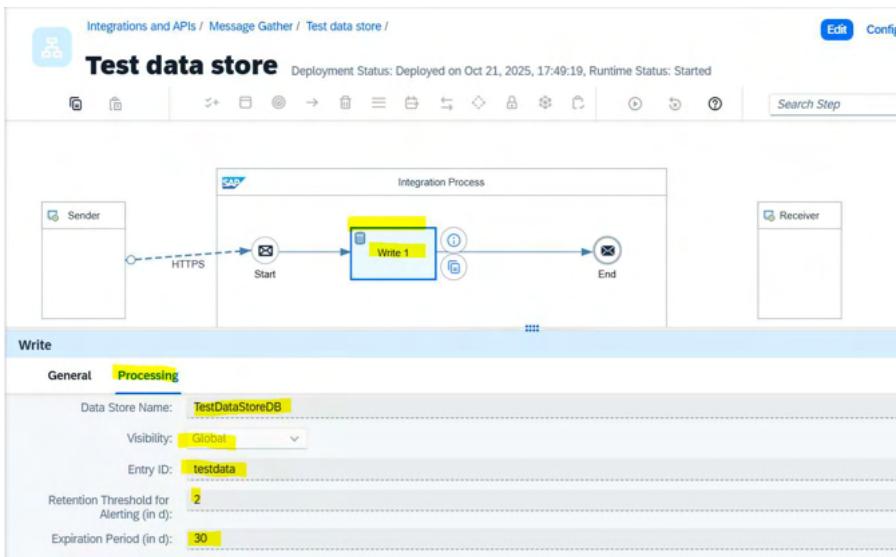
Scenario 7: Data Store and Fetch

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 POST method option to store in the DB and table)

https://1284bac5trial.it-cptrial03-rt.cfapps.ap21.hana.ondemand.com/http/testDataStore

POST https://1284bac5trial.it-cptrial03-rt.cfapps.ap21.hana.ondemand.com/http/testDataStore

Body (raw) XML

```

<root>
  <Personal>
    <FirstName>Abhimanyu</FirstName>
    <LastName>Yadav</LastName>
    <Price>200</Price>
  </Personal>
  <Personal>
    <FirstName>Abhishek</FirstName>
    <LastName>Yadav</LastName>
    <contact>
      <phone>2244855</phone>
    </contact>
  </Personal>
  <Personal>
    <FirstName>Shekhar</FirstName>
    <LastName>Yadav</LastName>
    <Price>300</Price>
    <contact>
      <phone>99933388</phone>
    </contact>
  </Personal>

```

200 OK · 526 ms · 1.84 KB

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

Integration Suite

Home

Discover

Design

Integrations and APIs

B2B Scenarios

Custom Type Systems

MIGs

MAGs

Test

Configure

Monitor

Integrations and APIs

B2B Scenarios

Analyze

Engage

Inspect

Monetize

Settings

Overview

Connectivity Tests

Manage Stores

Data Stores

Variables

Access Logs

System Log Files

1

Overview / Manage Data Stores

Data Stores (1) Filter by Name

TestDataStoreDB	1
Global	

TestDataStoreDB

Global

Entries (1) Filter by ID Delete Download

ID	Status	Due At	Created At
testdata	Waiting	Oct 23, 2025, 17:49:28	Oct 21, 2025, 17:49:28

Message ID: AGj3eIBMlaOyaa2wPN62Vcc7XNK9

Home Discover Design Integrations and APIs B2B Scenarios Custom Type Systems MIGs MAGs Test Configure Monitor Integrations and APIs B2B Scenarios Analyze Engage Inspect Monetize Settings

Overview / Manage Integration Content

Integration Content (8) Filter by Name or ID

Name	Status
Test data fetch	Started
Integration Flow	Started
Test data store	Started
Integration Flow	Started
Test Aggregator	Started
Integration Flow	Started
Custom testing	Started
Integration Flow	Started

Test data store

Restart Undeploy

Deployed On: Oct 21, 2025, 17:49:19 ID: Test_data_store Package: Message Gather Deployed By: koppuraso@gmail.com Version: 1.0.0

Endpoints Status Details Artifact Details Log Configuration

Endpoints

<https://1284baac3trial1.vt-cfapps.ap21.hana.ondemand.com/http/testDataStore>

Status Details

The Integration Flow is deployed successfully.

Artifact Details

Monitor Message Processing View deployed Artifact

Overview / Monitor Message Processing

Time: Status: Type: Package: Artifacts: or

Past Hour All All All Test data ...

Oct 21, 2025, 17:12:27 - Oct 21, 2025, 18:12:27

Messages (4) 1 / 1

Artifact Name	Status
Test data store	Completed
Test data store	Failed
Test data store	Failed
Test data store	Completed

Test data store

Last Updated at: Oct 21, 2025, 17:49:28

Status Properties Logs Artifact Details

Logs

Trace data is removed after the configured retention time

Log Level: Trace Instance ID: 0

Artifact Details

Manage Integration Content View deployed Artifact Navigate to Artifact Editor

Name: Test data store

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

Overview / Monitor Message Processing / Message Processing Run

Run Steps (4) 1 / 1

Step	Segment 1	Time
End	Segment 1	3 ms
Write 1	Segment 1	17 ms
HTTPS	Segment 1	3 ms
HTTPS	Segment 1	< 3 ms

Integration Flow Model Log Content Message Content

Message before Step

Header Exchange Properties Payload

Dov

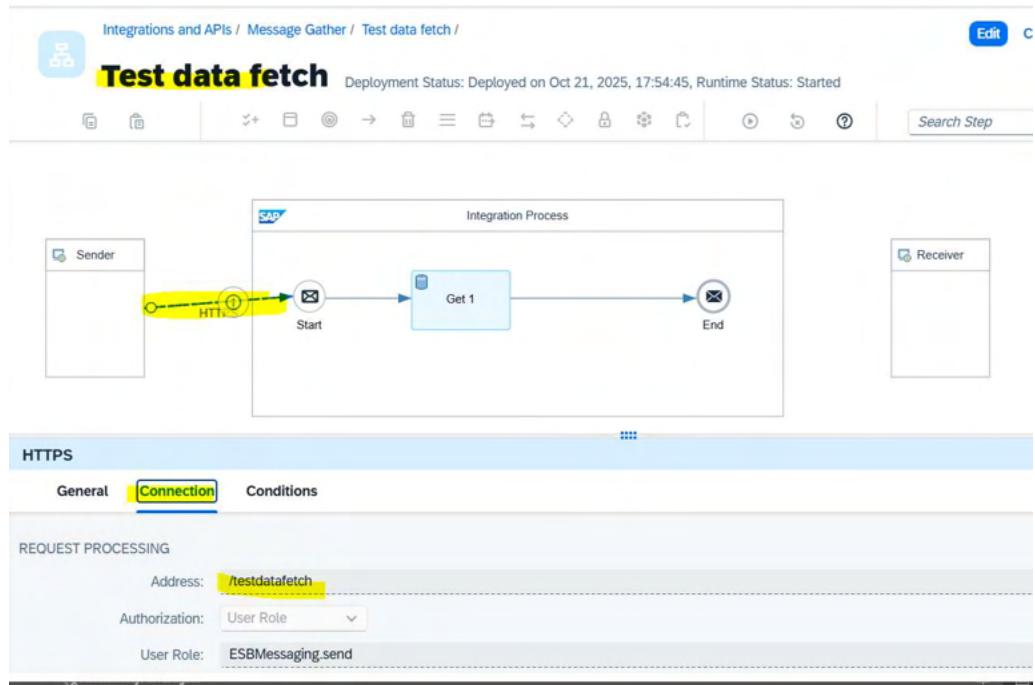
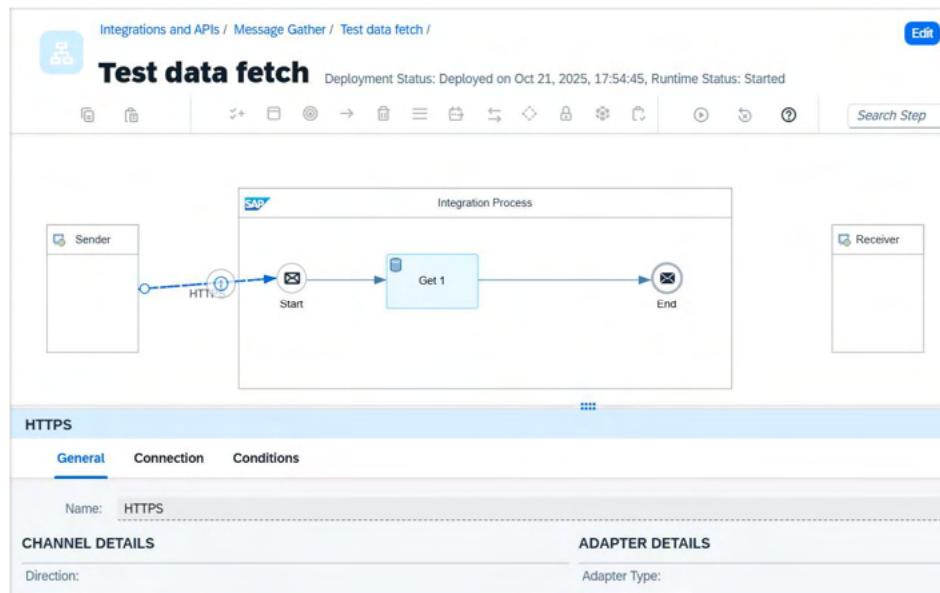
```

<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>

```

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

Note: This DB is configured as “Global”



Integrations and APIs / Message Gather / Test data fetch /

Test data fetch Deployment Status: Deployed on Oct 21, 2025, 17:54:45, Runtime Status: Started

Search Step

Get

General Processing

Data Store Name: **TestDataStoreDB**

Visibility: Global

Entry ID: **testdata**

Delete On Completion:

Throw Exception on Missing

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

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

https://1284bac5trial.it-cptrial03-rt.cfapps.ap21.hana.ondemand.com/http/testdatafetch

GET https://1284bac5trial.it-cptrial03-rt.cfapps.ap21.hana.ondemand.com/http/testdatafetch Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL XML

1 Ctrl+Alt+P to Ask AI

Body Cookies Headers (18) Test Results 200 OK 696 ms · 1.91 KB · ***

Raw Preview Visualize

```

<Root>
  <Personal>
    <FirstName>Abhimanyu</FirstName>
    <LastName>Yadav</LastName>
    <Price>200</Price>
  </Personal>
  <Personal>
    <FirstName>Anilshuk</FirstName>
    <LastName>Yadav</LastName>
    <contact>
      <phone>2244555 </phone>
    </contact>
  </Personal>
  <Personal>
    <FirstName>Shekhar</FirstName>
    <LastName>Yadav</LastName>
    <Price>300</Price>
  </Personal>

```

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

Integration Suite

Overview / Monitor Message Processing / Message Processing Run

Run Steps (4)

Integration Flow Model Log Content Message Content

Message before Step Header Exchange Properties Payload

End Segment 1 2 ms

Get 1 Segment 1 12 ms

HTTPS Segment 1 4 ms

HTTPS Segment 1 < 3 ms

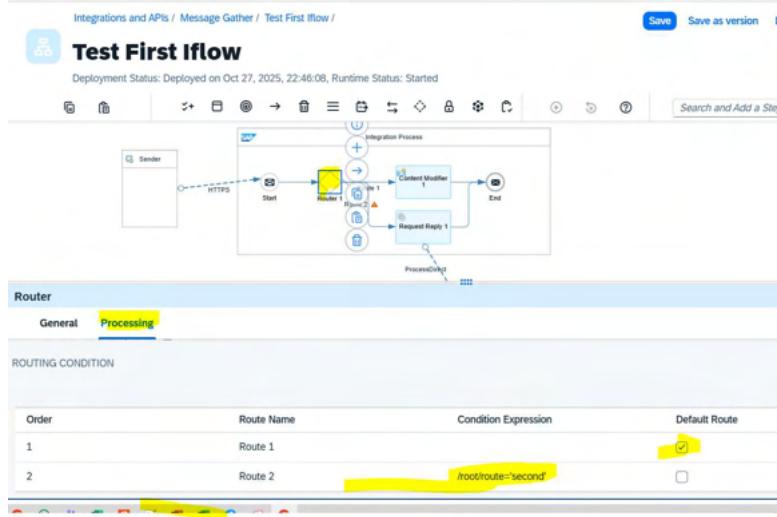
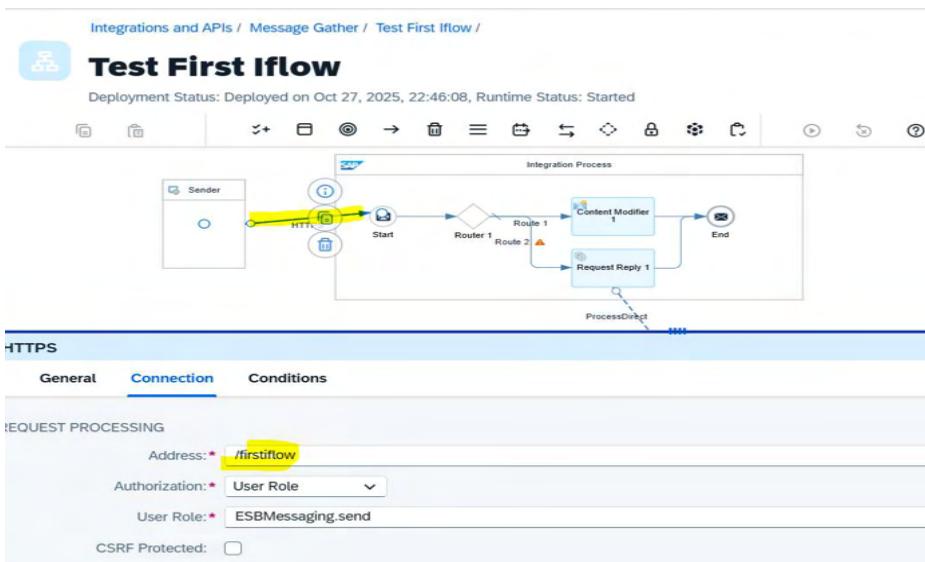
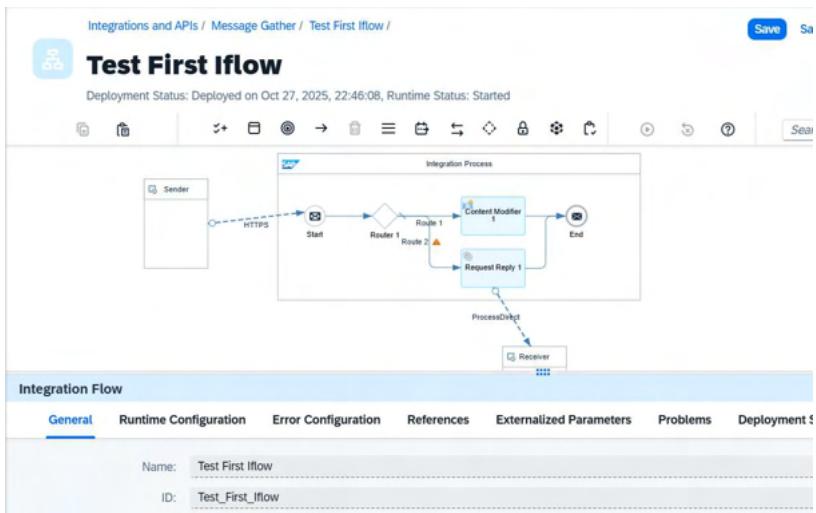
```

<Root>
  <Personal>
    <FirstName>Abhimanyu</FirstName>
    <LastName>Yadav</LastName>
    <Price>200</Price>
  </Personal>
  <Personal>
    <FirstName>Anilshuk</FirstName>
    <LastName>Yadav</LastName>
    <contact>
      <phone>2244555 </phone>
    </contact>
  </Personal>
  <Personal>
    <FirstName>Shekhar</FirstName>
    <LastName>Yadav</LastName>
    <Price>300</Price>
  </Personal>

```

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

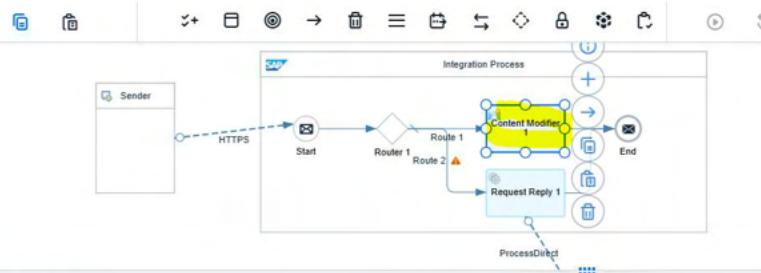
- Create first IFlow:





Test First Iflow

Deployment Status: Deployed on Oct 27, 2025, 22:46:08, Runtime Status: Started



Content Modifier

General Message Header Exchange Property **Message Body**

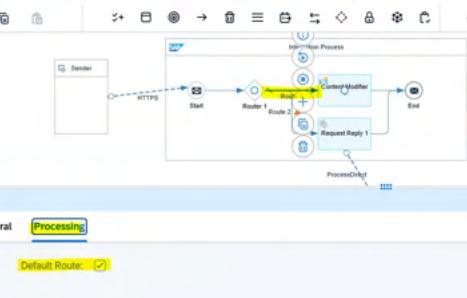
Type: Constant

Body: Data from First Iflow



Test First Iflow

Deployment Status: Deployed on Oct 27, 2025, 22:46:08, Runtime Status: Started



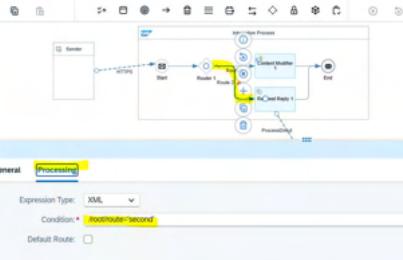
Route

General **Processing**

Default Route:

Test First Iflow

Deployment Status: Deployed on Oct 27, 2025, 22:46:08, Runtime Status: Started



Route

General **Processing**

Expression Type: XML

Condition: /root/route::second

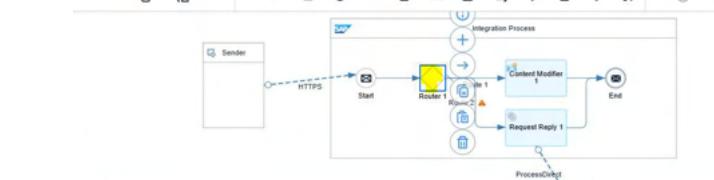
Default Route:

Save Save as versi

Test First Iflow

Deployment Status: Deployed on Oct 27, 2025, 22:46:08, Runtime Status: Started

Search and Ad



Router

General **Processing**

ROUTING CONDITION

Order	Route Name	Condition Expression	Default Route
1	Route 1		<input checked="" type="checkbox"/>
2	Route 2	/root/route::second	<input type="checkbox"/>

Test First Iflow

Deployment Status: Deployed on Oct 27, 2025, 22:46:08, Runtime Status: Started

Integration Process

Sender → Start → Router → Content Modifier → End

ProcessDirect

General Connection

CONNECTION DETAILS

Address: * /secondiflow

- Create 2nd Iflow:

second iflow

Deployment Status: Deployed on Oct 27, 2025, 22:46:40, Runtime Status: Started

Integration Process

Sender → Start → Content Modifier 1 → End

ProcessDirect

General Connection

CONNECTION DETAILS

Address: /secondiflow

second iflow

Deployment Status: Deployed on Oct 27, 2025, 22:46:40, Runtime Status: Started

Content Modifier

General Message Header Exchange Property Message Body

Type: Constant

Body: Message/Data from second 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.

The image contains two side-by-side screenshots of the POSTMAN application interface, demonstrating API requests to two different iFlows.

Screenshot 1: Request to http://firstiflow

- URL:** https://1284bac5trial.it-cptrial03-rt.cfapps.ap21.hana.ondemand.com/http/firstiflow
- Method:** POST
- Body Content:**

```
1 <root>
2   <route>first</route>
3   <!--
4   <route>first</route>
5   <route>second</route>
6   -->
7 </root>
```
- Response:** 200 OK | 601 ms | 544 B | Data from First Iflow

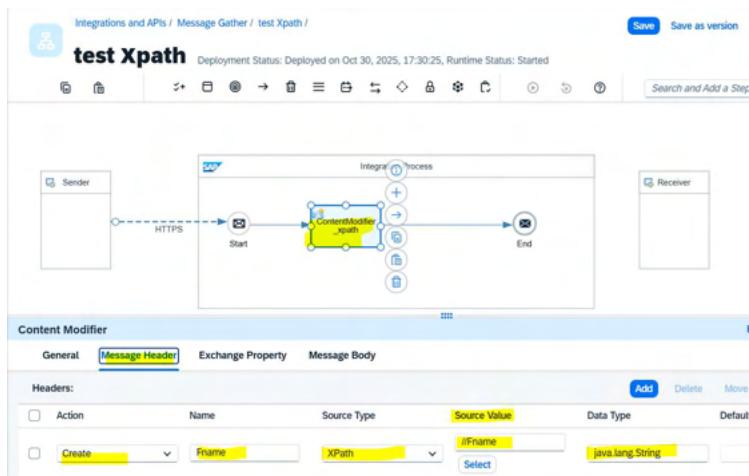
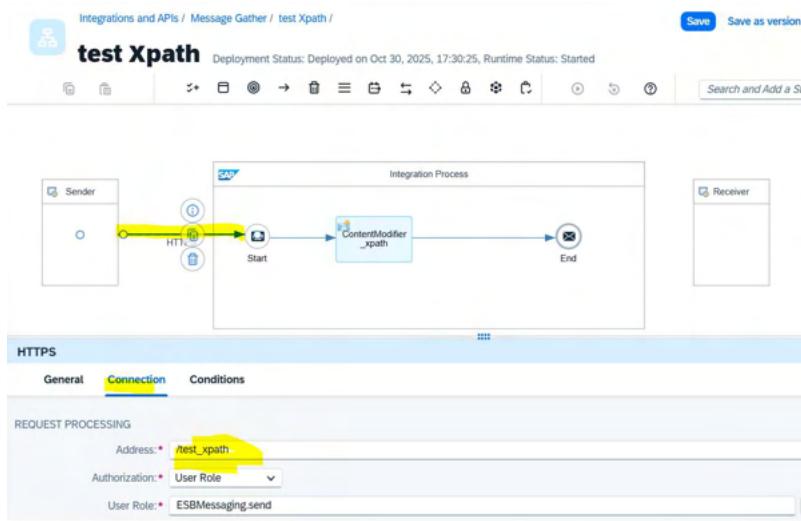
Screenshot 2: Request to http://secondiflow

- URL:** https://1284bac5trial.it-cptrial03-rt.cfapps.ap21.hana.ondemand.com/http/secondiflow
- Method:** POST
- Body Content:**

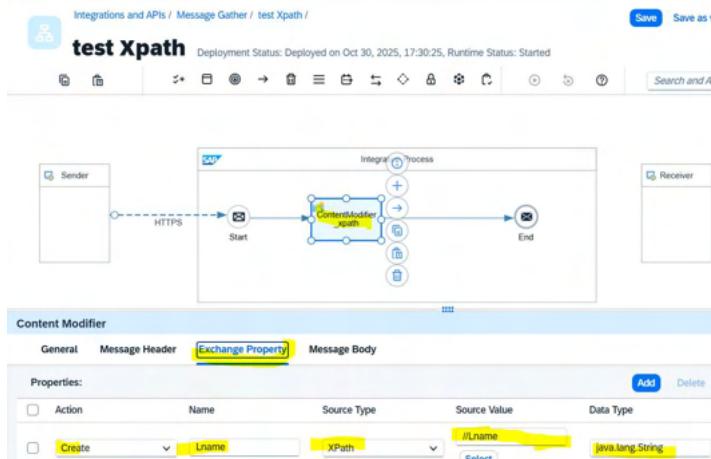
```
1 <root>
2   <route>second</route>
3   <!--
4   <route>first</route>
5   <route>second</route>
6   -->
7 </root>
```
- Response:** 200 OK | 74 ms | 659 B | Message /Data from second iflow

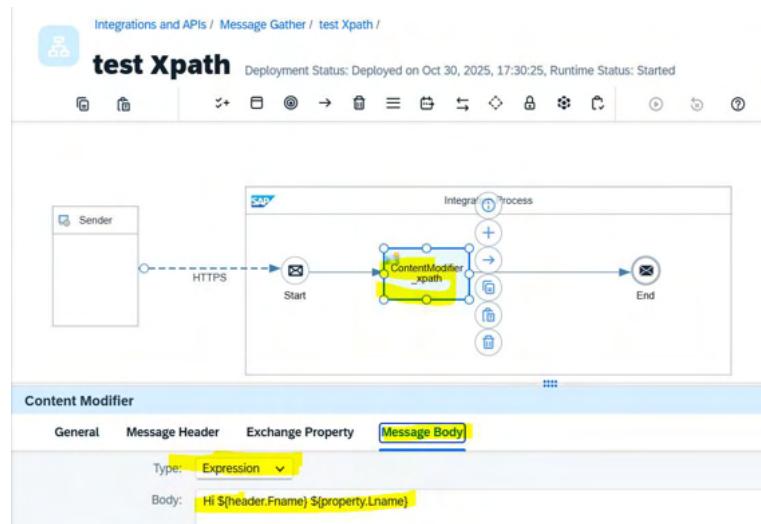
Scenario 9: XPath in Cloud Integration

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 like "/root/Fname".





- Call the URL from POSTMAN to see the output.

HTTP https://1284bac5trial.it-cpitrial03-rt.cfapps.ap21.hana.ondemand.com/http/test_xpath

POST https://1284bac5trial.it-cpitrial03-rt.cfapps.ap21.hana.ondemand.com/http/test_xpath

Params Authorization Headers (10) **Body** Scripts Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL XML

```
1 <root>
2   <Fname>Sreenivasa Rao</Fname>
3   <Lname>K</Lname>
4 </root>
```

Body Cookies Headers (14) Test Results 41 200 OK 180 ms 565 B

Raw Preview Visualize 1 Hi Sreenivasa Rao

HTTP https://1284bac5trial.it-cpitrial03-rt.cfapps.ap21.hana.ondemand.com/http/test_xpath

POST https://1284bac5trial.it-cpitrial03-rt.cfapps.ap21.hana.ondemand.com/http/test_xpath

Params Authorization Headers (10) **Body** Scripts Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL XML

```
1 <root>
2   <Fname>Sreenivasa Rao</Fname>
3   <Lname>Koppu</Lname>
4 </root>
```

Body Cookies Headers (14) Test Results 41 200 OK 2.22 s 569 B

Raw Preview Visualize 1 Hi Sreenivasa Rao Koppu

HTTP https://1284bac5trial.it-cpitrial03-rt.cfapps.ap21.hana.ondemand.com/http/test_xpath

GET https://1284bac5trial.it-cpitrial03-rt.cfapps.ap21.hana.ondemand.com/http/test_xpath

Params Authorization Headers (10) **Body** Scripts Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL XML

```
1 <root>
2   <Fname>SRao</Fname>
3   <Lname>Koppu</Lname>
4 </root>
```

Body Cookies Headers (14) Test Results 41 200 OK 830 ms 549 B

Raw Preview Visualize 1 Hi SRao Koppu

Scenario 10: Converter (JSON to XML)

Integration Process Diagram:

```

graph LR
    Sender[Sender] -- "HTTPS" --> Start((Start))
    Start --> JSONToXML[JSON to XML Converter]
    JSONToXML --> End((End))
    
```

HTTPS Request Processing Configuration:

- Address: `/json_to_xml`
- Authorization: `User Role`
- User Role: `ESBMessaging.send`
- CSRF Protected:

JSON To XML Converter Configuration (General Tab):

- Name: `JSON to XML Converter`

JSON To XML Converter Configuration (Processing Tab):

- Use Namespace Mapping:
- JSON Prefix Separator: `Colon(:)`
- Add XML Root Element:
- Name: `MT_Custom`
- Namespace Mapping: `xmns:ns0=http://cpi.sap.com/demo`

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 Configuration"
- 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

Integrations and APIs / Message Gather / Json_to_xml_converter /

Json_to_xml_converter

Deployment Status: Deployed on Oct 31, 2025, 13:01:40, Runtime Status: Started

Save Save as v Logs

Integration Flow

General Runtime Configuration Error Configuration References Externalized Parameters Problems Deployment Status

Runtime Profile: * Cloud Integration

Namespace Mapping: xmlns:ns0=http://cpi.sap.com/demo;xmlns:ns1=http://sap.com/xi/XI/SplitAndMerge

Allowed Header(s):

HTTP Session Reuse: None

```

graph LR
    Start((Start)) --> JSON[JSON to XML Converter]
    JSON --> End((End))
    subgraph Sender [Sender]
        Start
    end
    subgraph Receiver [Receiver]
        End
    end
    style JSON fill:#e0f2f1,stroke:#336699,color:#336699
    style Start fill:#d9e1f2,stroke:#336699,color:#336699
    style End fill:#d9e1f2,stroke:#336699,color:#336699
    
```

- Save and deploy and execute from POSTMAN

https://1284bac5trial.it-cptrial03-rt.cfapps.ap21hana.ondemand.com/http/test_xpath

GET https://1284bac5trial.it-cptrial03-rt.cfapps.ap21hana.ondemand.com/http/json_to_xml

Params Authorization Headers (10) Body Scripts Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2     "row": [
3         {
4             "CustomerID": "NEWCID01",
5             "CompanyName": "NEWCOMPANY1",
6             "Country": "Finance"
7         },
8         {
9             "CustomerID": "NEWCID02",
10            "CompanyName": "NEWCOMPANY2",
11            "Country": "Italy"
12        }
13    ]
}
  
```

200 OK 791 ms 944 B

Body Cookies Headers (15) Test Results

XML Preview Visualize

```

1 <?xml version='1.0' encoding='UTF-8'?>
2 <ns0:MT_Custom xmlns:ns0="http://cpi.sap.com/demo">
3     <row>
4         <CustomerID>NEWCID01</CustomerID>
5         <CompanyName>NEWCOMPANY1</CompanyName>
6         <Country>Finance</Country>
7     </row>
8     <row>
9         <CustomerID>NEWCID02</CustomerID>
10        <CompanyName>NEWCOMPANY2</CompanyName>
11        <Country>Italy</Country>
12    </row>
13 </ns0:MT_Custom>
  
```