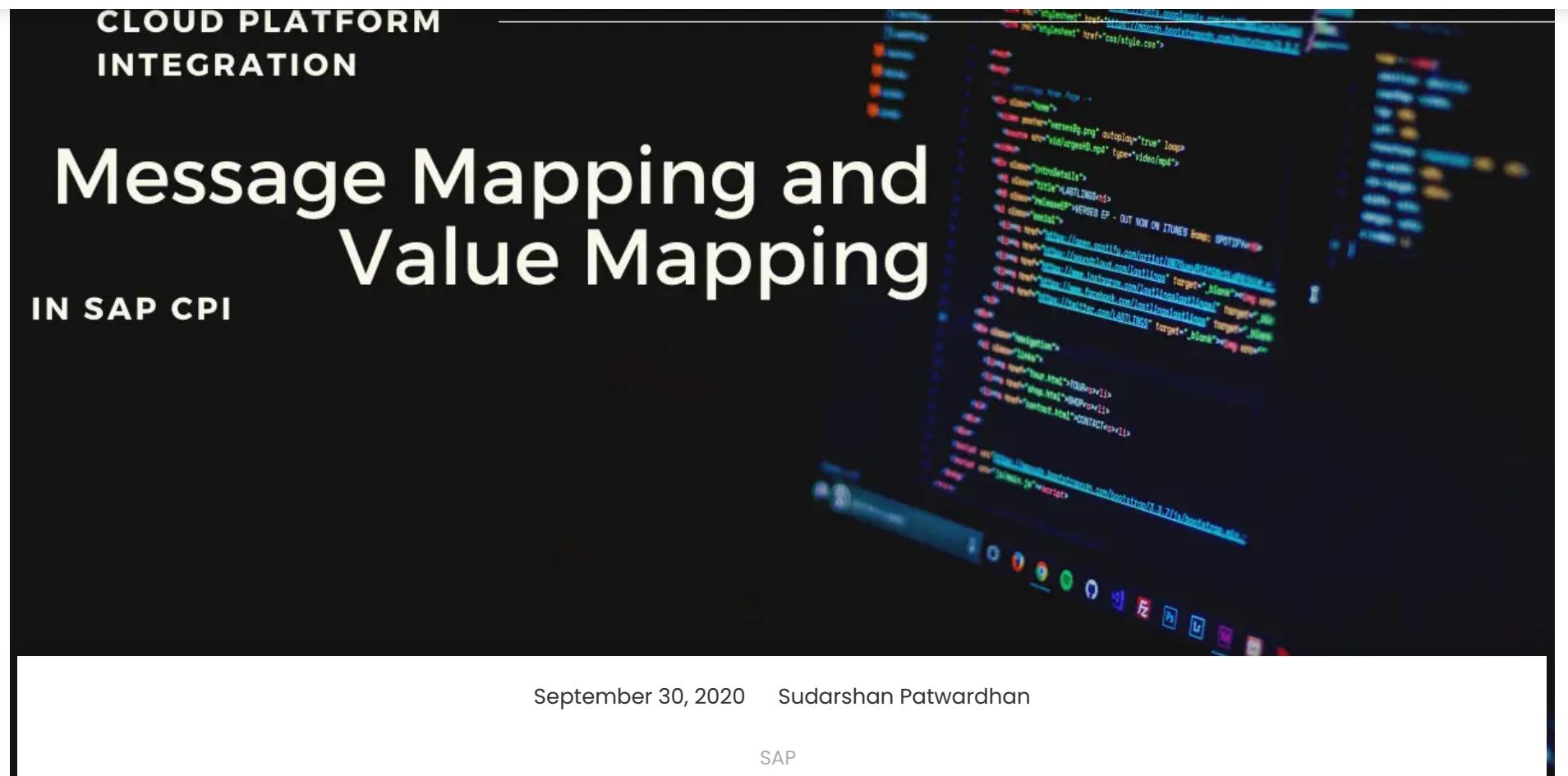


 Menu

 Menu

**CLOUD PLATFORM
INTEGRATION**

Message Mapping and Value Mapping

IN SAP CPI

September 30, 2020 Sudarshan Patwardhan

SAP

Message Mapping and Value Mapping in SAP CPI

Let's have a look at what is Message mapping in SAP CPI, In CPI message mapping is required the type of message is different on your sender and receiver systems. In SAP PO message mapping we call this has Graphical mapping or Java Mapping according to the way we do this. This article will help you understand the Implementation of Message Mapping and Value Mapping in SAP CPI.

Back To Top ↑

[Table of Contents](#)

☰ Menu

- 4. SELF- TRY :
- 5. SAP CPI Message Mapping Tutorial
- 6. FINAL VERDICT :

CREATING SAP CPI VALUE MAPPING

Step 1: Go to **Artifacts** tab of your Package. **Add → Value Mapping**, This will add sap cpi value mapping.

The screenshot shows the SAP CPI interface with the 'Artifacts' tab selected. A dropdown menu is open at the top right, showing options: 'Add' (with a dropdown arrow), 'Delete', and 'Actions'. Below this, a list of artifact types is shown: 'Integration Flow', 'Value Mapping' (which is highlighted with a yellow box), 'OData Service', and 'Integration Adapter'. The 'Value Mapping' option is the one intended for selection.

Step 2: Select **Create** Radio Button and Give a **valid Name**. (This step is sap cpi namespace)

 Menu

Create Upload

Name: *

ID: *

Description:

Description:

Target:

Step 3 : Click on the new Value Mapping Created.

Header Overview Artifacts (1) Documents Comments Tags

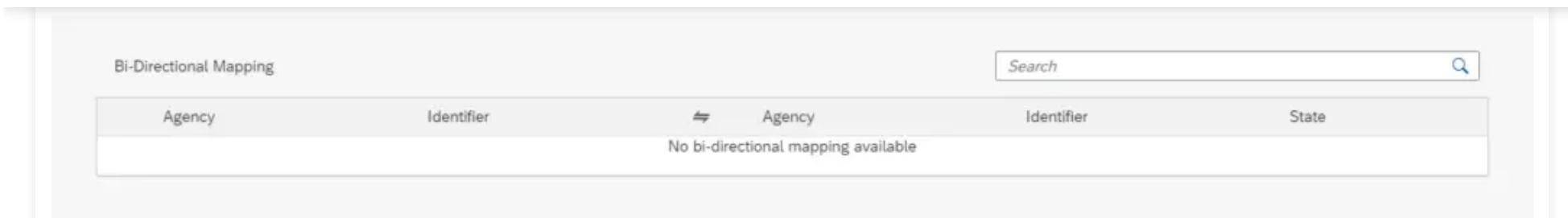
Add Delete Actions Filter Artifacts

<input type="checkbox"/> Name	Type	Version	Actions
<input type="checkbox"/> ID_Conversion	Value Mapping	1.0.0	
<input type="checkbox"/> Created			

Step 4 : Click on edit



Back To Top

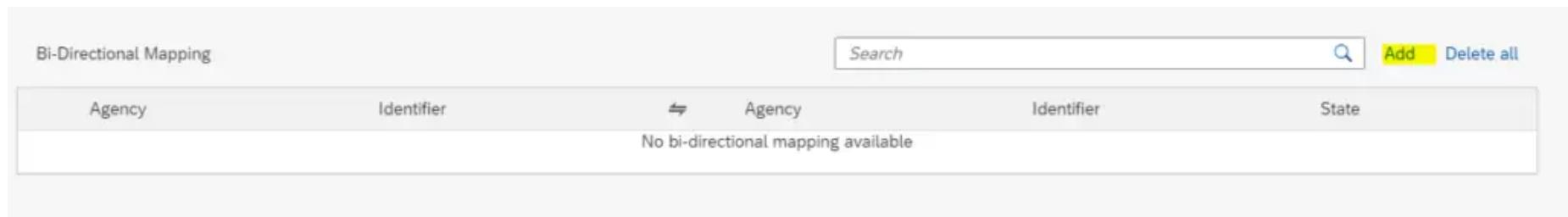
 Menu

Bi-Directional Mapping

Search 

Agency	Identifier	Agency	Identifier	State
				No bi-directional mapping available

Step 5 : Click on **Add**.



Bi-Directional Mapping

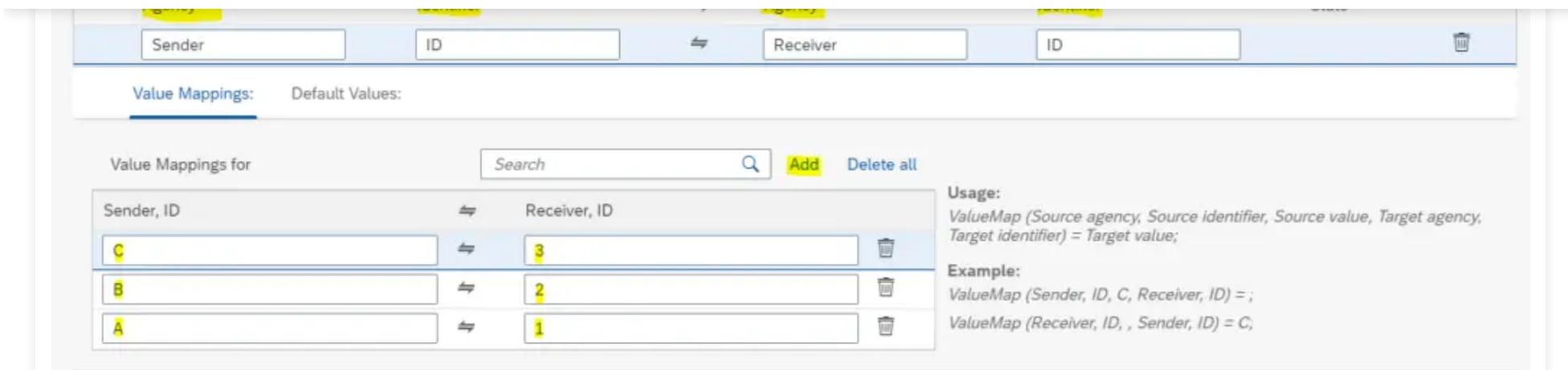
Search  **Add**  Delete all

Agency	Identifier	Agency	Identifier	State
				No bi-directional mapping available

Step 6 : Fill the required details and Conversion Values.

Source	Target
A	1
B	2
C	3

 Menu

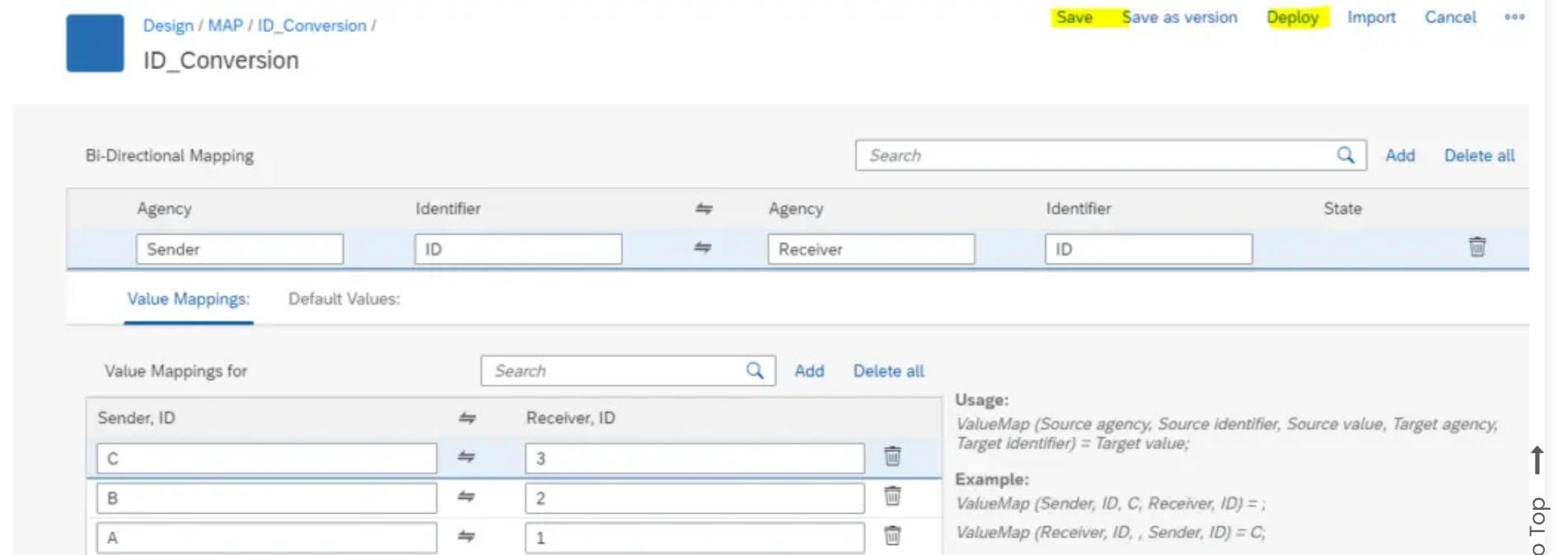


The screenshot shows the SAP CPI Value Mapping interface. At the top, there are four input fields: 'Sender' (containing 'C'), 'ID' (containing '3'), 'Receiver' (containing 'A'), and another 'ID' field (containing '1'). Below these are tabs for 'Value Mappings' (selected) and 'Default Values'. A search bar and 'Add' and 'Delete all' buttons are also present. The main area displays 'Value Mappings for' with three entries:

Sender, ID	Receiver, ID
C	3
B	2
A	1

Usage:
 $\text{ValueMap}(\text{Source agency}, \text{Source identifier}, \text{Source value}, \text{Target agency}, \text{Target identifier}) = \text{Target value};$

Example:
 $\text{ValueMap}(\text{Sender, ID, C, Receiver, ID}) = ;$
 $\text{ValueMap}(\text{Receiver, ID, , Sender, ID}) = C,$

Step 7: Save and Deploy.


The screenshot shows the SAP CPI Bi-Directional Mapping interface. At the top, there are two input fields: 'Sender' (containing 'C') and 'ID' (containing '3'). Below these are tabs for 'Value Mappings' (selected) and 'Default Values'. A search bar and 'Add' and 'Delete all' buttons are also present. The main area displays 'Value Mappings for' with three entries:

Agency	Identifier	Agency	Identifier	State
Sender	ID	Receiver	ID	

Usage:
 $\text{ValueMap}(\text{Source agency}, \text{Source identifier}, \text{Source value}, \text{Target agency}, \text{Target identifier}) = \text{Target value};$

Example:
 $\text{ValueMap}(\text{Sender, ID, C, Receiver, ID}) = ;$
 $\text{ValueMap}(\text{Receiver, ID, , Sender, ID}) = C,$

Step 8 : In Overview Window, Make sure that the Value Mapping is started.

Menu

The screenshot shows the SAP CPI Integration Content interface. At the top left, it says "Integration Content (5)". There is a search bar labeled "Filter by Name or ID" and some filter icons. On the right, there is a large title "ID_Conversion" and deployment information: "Deployed On: Sep 22, 2020, 16:23:57" and "Deployed By: sudpat123123@gmail.com". Below the main title, there is a table with columns "Name" and "Status". One row is highlighted in blue, showing "ID_Conversion" with a status of "Started". Another row below it is "Value Mapping".

CREATING INTEGRATION FLOW

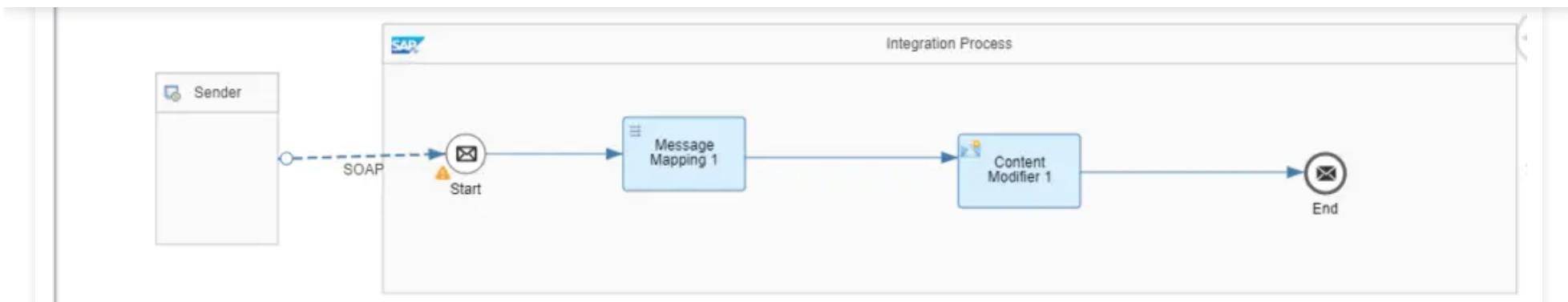
Step 9 : Go to **Design** Window and **Create an Integration Flow** under Artifacts.

The screenshot shows the SAP CPI Artifacts list. At the top, there are tabs: Header, Overview, **Artifacts (2)**, Documents, Comments, and Tags. Below the tabs is a header row with columns: "Name", "Type", "Version", and "Actions". There are two entries:

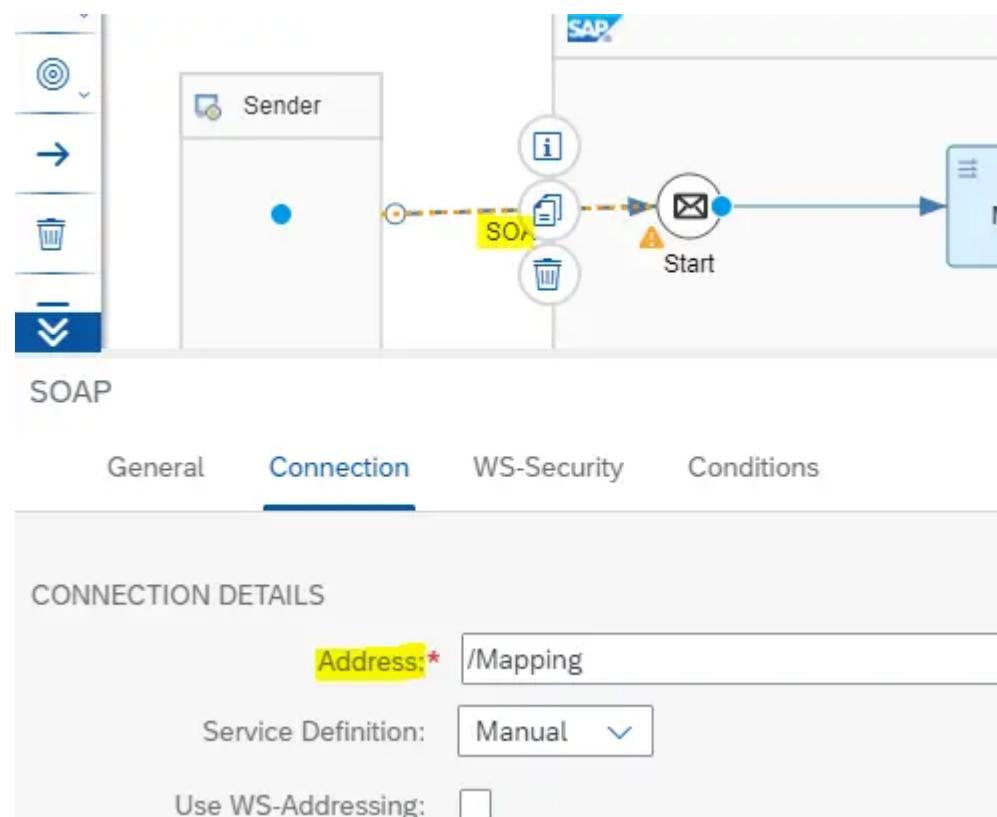
- ID_Conversion**: Type "Value Mapping", Version "Draft", Actions button (edit icon).
- MappingLogic**: Type "Integration Flow", Version "1.0.0", Actions button (edit icon).

Step 10 : Open the Integration Flow in **edit** Mode. Add a **Message Mapping** and a **Content Modifier**.

Choose **Sender Adapter** as **SOAP**.

 Menu


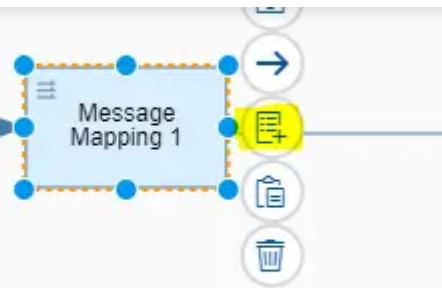
Step 11 : Select **SOAP Adapter**. Under **Connection** tab, Give a relative **address** to define the Endpoint. (/Mapping)



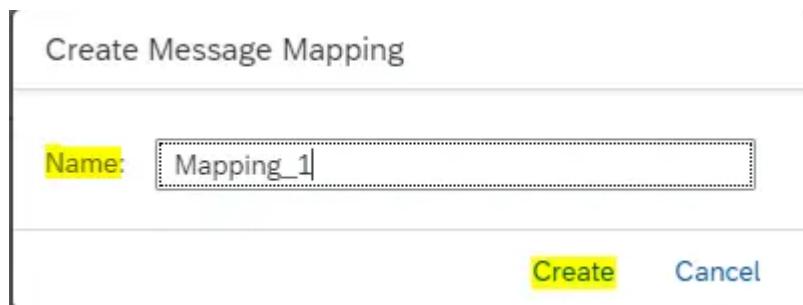
The screenshot shows the SAP CPI interface with the 'SOAP' adapter selected. The 'Connection' tab is active, displaying the following details:

CONNECTION DETAILS	
Address:	<input type="text" value="/Mapping"/>
Service Definition:	<input type="button" value="Manual"/>
Use WS-Addressing:	<input type="checkbox"/>

Step 12 : Select **Message Mapping** and Click on **Create** button highlighted below.

 Menu

Step 13 : Give any Valid **name** for your Mapping. Click on **Create**.

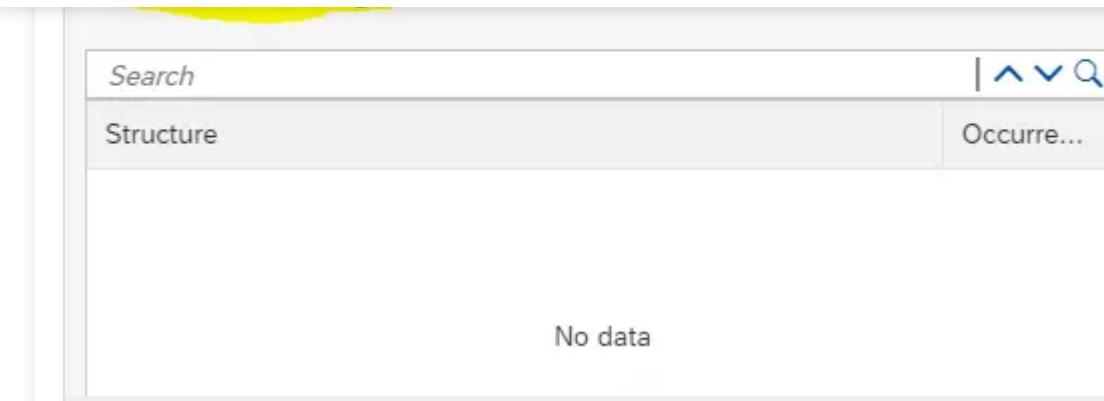


Create Message Mapping

Name:

Create **Cancel**

Step 14 : Click on **Add Source Message**.

 Menu

Step 15 : Click on ***Upload from File System***. Browse and select the **Source WSDL**.

Copy the below sample WSDL for Source message and save to upload.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="SendOrder_Async"
    targetNamespace="http://cpi.sap.com/demo" xmlns:p1="http://cpi.sap.com/demo"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:documentation />
    <wsp:UsingPolicy wsdl:required="true" />
    <wsp:Policy wsu:id="OP_SendOrder_Async" />
    <wsdl:types>
        <xsd:schema targetNamespace="http://cpi.sap.com/demo"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://cpi.sap.com/demo">
            <xsd:element name="Order_MT" type="Order_DT" />
            <xsd:complexType name="Order_DT">
                <xsd:sequence>
```

 Menu

```
        </xsu:sequence>
    </xsd:complexType>
</xsd:schema>
</wsdl:types>
<wsdl:message name="Order_MT">
    <wsdl:documentation />
    <wsdl:part name="Order_MT" element="p1:Order_MT" />
</wsdl:message>
<wsdl:portType name="SendOrder_Async">
    <wsdl:documentation />
    <wsdl:operation name="SendOrder_Async">
        <wsdl:documentation />
        <wsp:Policy>
            <wsp:PolicyReference URI="#OP_SendOrder_Async" />
        </wsp:Policy>
        <wsdl:input message="p1:Order_MT" />
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="SendOrder_AsyncBinding" type="p1:SendOrder_Async">
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" />
    <wsdl:operation name="SendOrder_Async">
        <soap:operation soapAction="http://sap.com/xi/WebService/soap1.1"
                      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" />
        <wsdl:input>
            <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" />
        </wsdl:input>
    </wsdl:operation>
```

 Menu

Click on Upload from the file system.

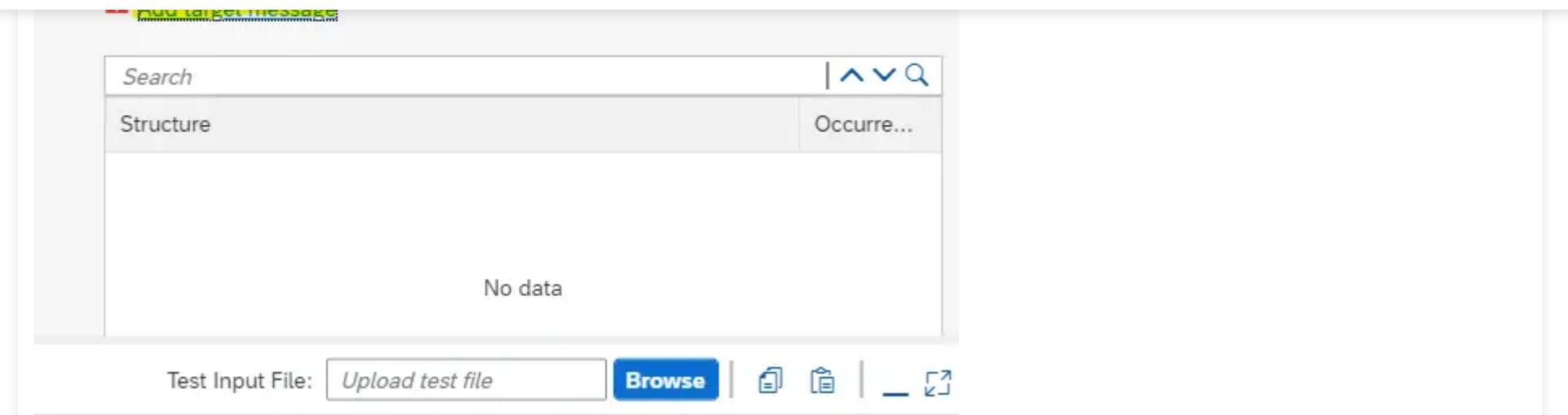
Step 16 : Repeat the Step 14 and Step 15 for Target message and choose target WSDL.

Copy the below sample WSDL for Target message and save to upload.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="SendOrder_Async"
    targetNamespace="http://cpi.sap.com/demo" xmlns:p2="http://cpi.sap.com/demo"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:documentation />
    <wsp:UsingPolicy wsdl:required="true" />
    <wsp:Policy wsu:id="OP_SendOrder_Async" />
    <wsdl:types>
        <xsd:schema targetNamespace="http://cpi.sap.com/demo"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://cpi.sap.com/demo">
            <xsd:element name="Order_MT" type="Order_DT" />
            <xsd:complexType name="Order_DT">
                <xsd:sequence>
                    <xsd:element name="orderNumber" type="xsd:string" />
                    <xsd:element name="Environment" type="xsd:string" />
                </xsd:sequence>
            </xsd:complexType>
        </xsd:schema>
    </wsdl:types>
```

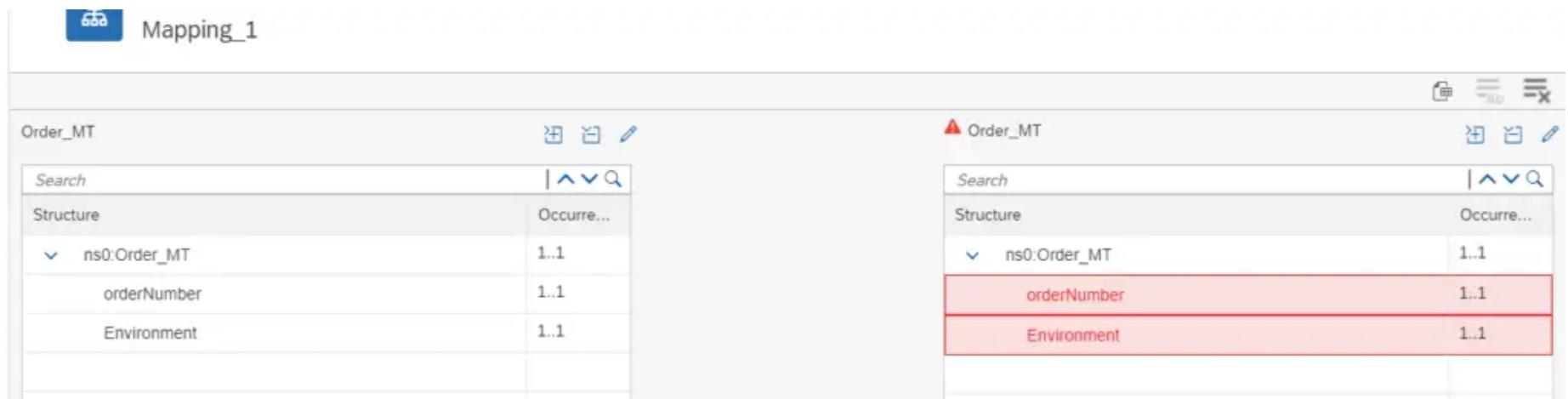
 Menu

```
<wsdl:part name="Order_MT" element="p2:Order_MT" />
</wsdl:message>
<wsdl:portType name="SendOrder_Async">
    <wsdl:documentation />
    <wsdl:operation name="SendOrder_Async">
        <wsdl:documentation />
        <wsp:Policy>
            <wsp:PolicyReference URI="#OP_SendOrder_Async" />
        </wsp:Policy>
        <wsdl:input message="p2:Order_MT" />
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="SendOrder_AsyncBinding" type="p2:SendOrder_Async">
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" />
    <wsdl:operation name="SendOrder_Async">
        <soap:operation soapAction="http://sap.com/xi/WebService/soap1.1"
                      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" />
        <wsdl:input>
            <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" />
        </wsdl:input>
    </wsdl:operation>
</wsdl:binding>
</wsdl:definitions>
```

 Menu

The screenshot shows the SAP CPI Recode Hive interface. At the top, there is a search bar with a magnifying glass icon. Below it, a table titled 'Structure' has a single row with the text 'Occurre...'. In the center of the screen, it says 'No data'. At the bottom, there is a 'Test Input File:' section with a 'Upload test file' button, a 'Browse' button, and other file-related icons.

Step 17 : It will look as below.



The screenshot shows two side-by-side tables for the 'Order_MT' message. The left table is labeled 'Order_MT' and contains the following data:

Structure	Occurrence
ns0:Order_MT	1..1
orderNumber	1..1
Environment	1..1

The right table is also labeled 'Order_MT' and contains the same data:

Structure	Occurrence
ns0:Order_MT	1..1
orderNumber	1..1
Environment	1..1

In both tables, the first three rows (ns0:Order_MT, orderNumber, and Environment) are highlighted in red, indicating they are source elements.

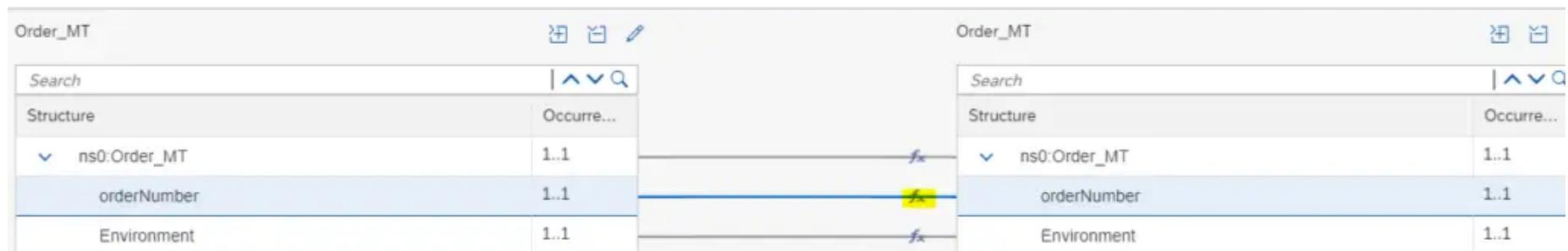
Step 18 : Click on Source Element and drag to the target element to map respective fields. Now it will look as below.



Back To Top

☰ Menu


Step 19 : Click on **fx** Symbol to edit the mapping for that element.

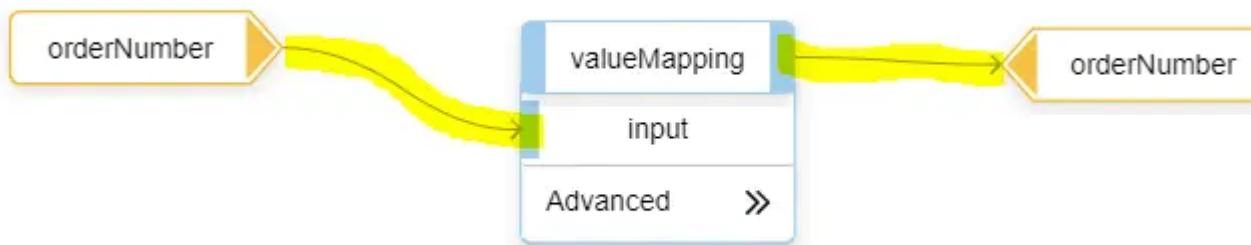


Step 20 : From the left window, choose **Conversions** → **Value Mapping**



 Menu

Step 21 : Place the Value Mapping block in center and **Connect Source to input , valueMapping to Target** as shown in the below picture.



Step 22 : Click on **Advanced** and fill the required details same as the **deployed Value Mapping** in [Step 7](#).

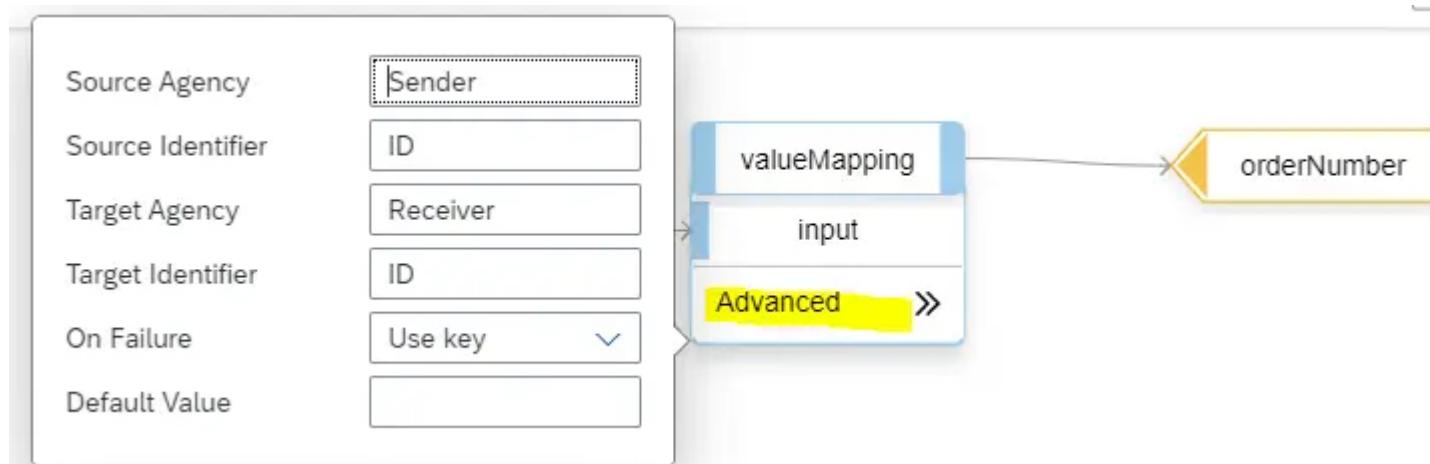
On Failure :

On Failure	Expected Outcomes when the incoming Message has any other value apart from the once maintained in the Value Mapping Used (in our case any value other than A, B , C)
-------------------	--

☰ Menu

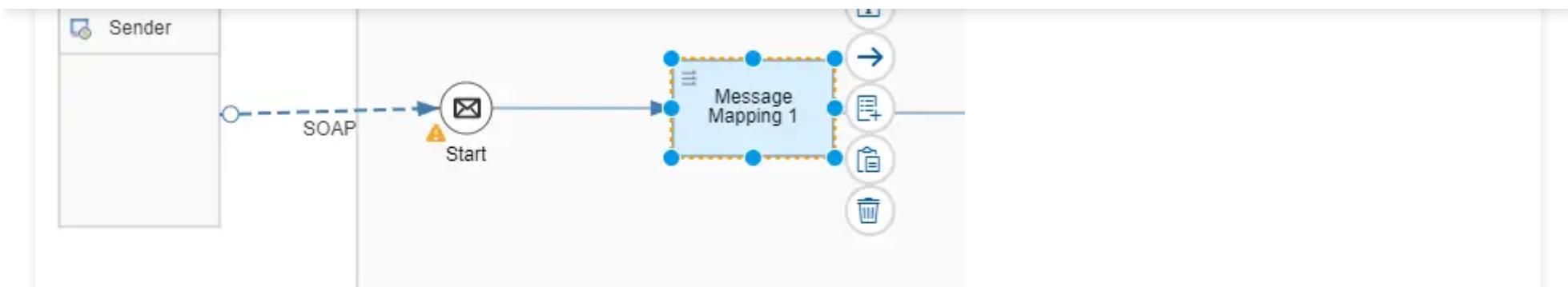
Use Default Value	It will pass the default value that is specified in the Default value field.
Throw Exception	This will throw an exception.

I have used Use Key.



Step 23 : Click on **OK** at top right corner. It will redirect you to the Main Integration Flow.

Step 24 : You can **edit** your mapping any time from **Processing** tab of Message Mapping.

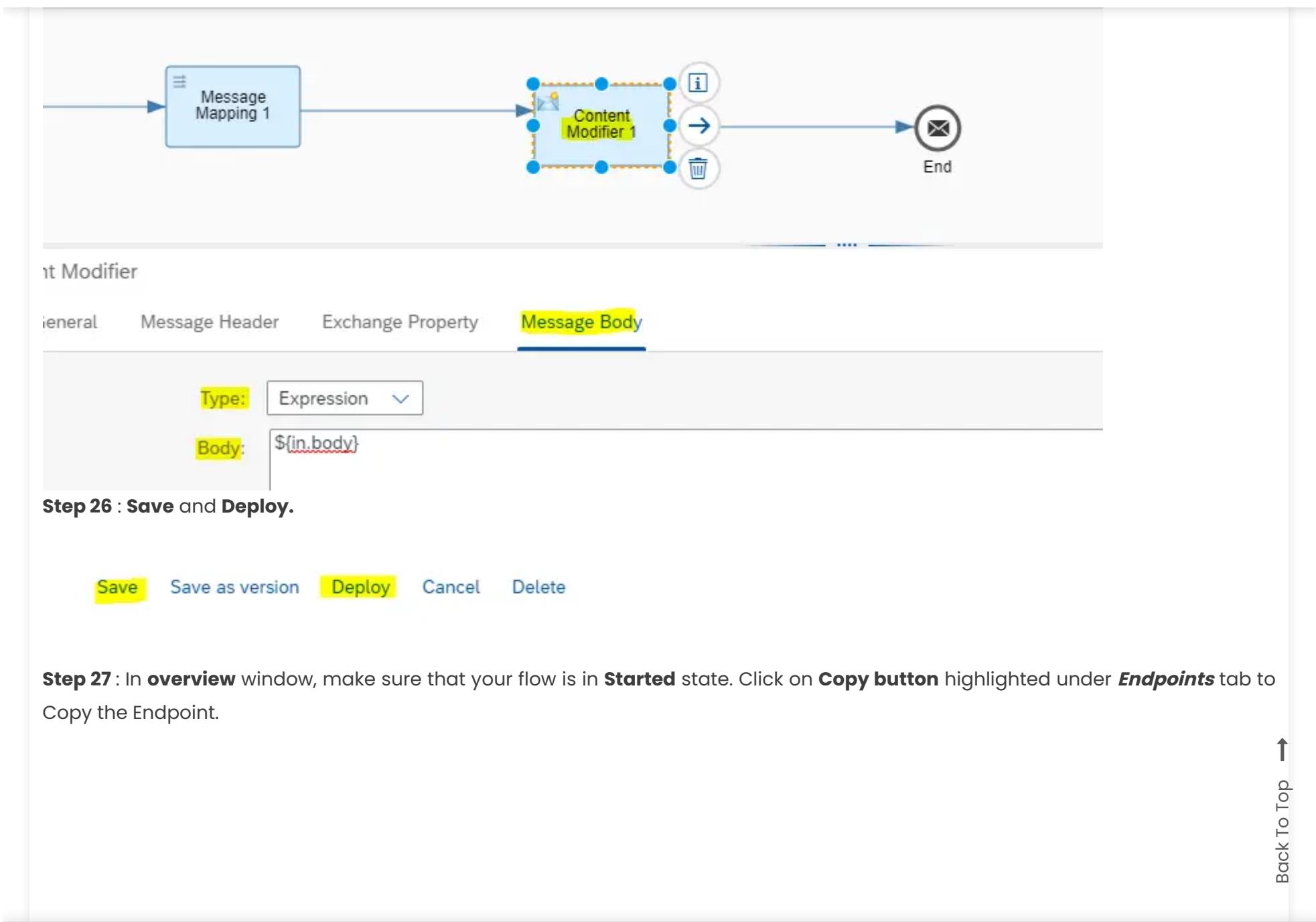
 Menu

Mapping

General Processing

Resource: * /Mapping_1.mmap

Step 25 : Select the **Content Modifier**. Under **Message Body** tab, choose **Type=Expression** and **Body : \${in.body}** to capture the incoming payload.

 Menu

The screenshot shows a message flow diagram and its configuration details:

Message Flow Diagram:

```
graph LR; Start(( )) --> MM1[Message Mapping 1]; MM1 --> CM1[Content Modifier 1]; CM1 --> End((End))
```

Content Modifier Configuration:

General Message Header Exchange Property **Message Body**

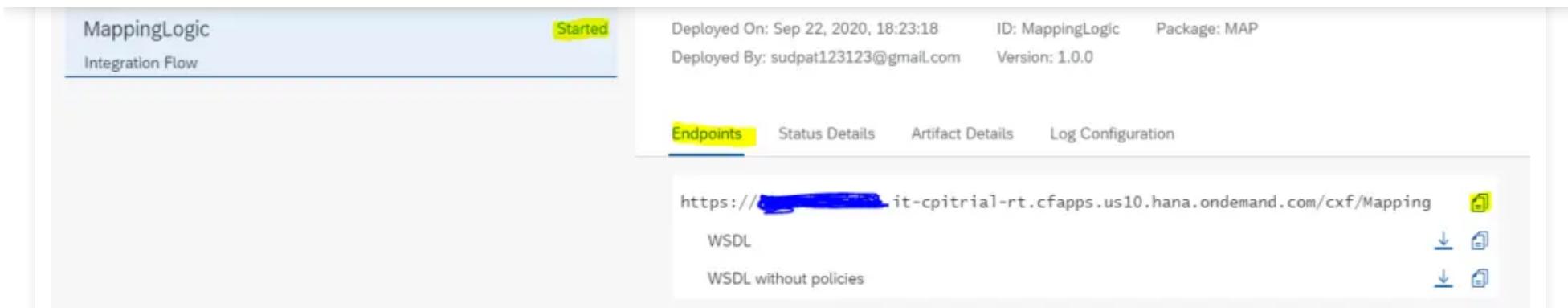
Type: Expression
Body: \${in.body}

Step 26 : Save and Deploy.

Save **Save as version** Deploy Cancel Delete

Step 27 : In **overview** window, make sure that your flow is in **Started** state. Click on **Copy button** highlighted under **Endpoints** tab to Copy the Endpoint.

↑
Back To Top

 Menu

MappingLogic
Integration Flow

Started

Deployed On: Sep 22, 2020, 18:23:18 ID: MappingLogic Package: MAP
Deployed By: sudpat123123@gmail.com Version: 1.0.0

Endpoints Status Details Artifact Details Log Configuration

https://[REDACTED].it-cpitrial-rt.cfapps.us10.hana.ondemand.com/cxf/Mapping

WSDL 

WSDL without policies 

TESTING THE ARTIFACTS

Step 28 : Open **POSTMAN**. Create a new request with type **POST**.

Paste the **Endpoint** copied in previous step in **URL** tab.

Type : Basic Auth

Username : Client ID

Password : Client Secret

Client Id and Client Secret you will get when you create Process Integration Runtime Instance during tenant setup. (Refer **Step 34** of [this post](#))

 Menu

The screenshot shows the Postman interface with the 'Authorization' tab selected. Under 'TYPE', 'Basic Auth' is chosen. A note says: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.' Below are fields for 'Username' and 'Password', both of which are redacted.

Step 29 : Under **Body** tab, Choose **Raw** radio button and **XML** message Type. **Type** the below input message. Click on **Send**.

```
<soapenv:Envelope
  xmlns:demo="http://cpi.sap.com/demo">
  <soapenv:Header/>
  <soapenv:Body>
    <demo:Order_MT>
      <orderNumber>A</orderNumber>
      <Environment>CS</Environment>
    </demo:Order_MT>
  </soapenv:Body>
</soapenv:Envelope>
```

xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"

 Menu



The screenshot shows the Postman interface with the 'Body' tab selected. The 'XML' option is chosen from the dropdown menu. The XML code in the body is:

```

1 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:demo="http://cpi.sap.com/demo">
2   <soapenv:Header/>
3   <soapenv:Body>
4     <demo:Order_MT>
5       <orderNumber>A</orderNumber>
6       <Environment>CS</Environment>
7     </demo:Order_MT>
8   </soapenv:Body>
9 </soapenv:Envelope>

```

Step 30 : You will get back the result as below. Note that the ***OrderNumber is changed from A to 1 in output.*** This is due to the use of Value Mapping.



The screenshot shows the Postman interface with the 'Body' tab selected. The 'XML' option is chosen from the dropdown menu. The XML code in the body is identical to the one above, but the 'orderNumber' element is highlighted with a yellow box.

```

1 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:demo="http://cpi.sap.com/demo">
2   <soapenv:Header/>
3   <soapenv:Body>
4     <demo:Order_MT>
5       <orderNumber>A</orderNumber>
6       <Environment>CS</Environment>
7     </demo:Order_MT>

```

Body Cookies Headers (16) Test Results



The screenshot shows the Postman interface with the 'Test Results' tab selected. The 'XML' option is chosen from the dropdown menu. The XML code in the response body is:

```

1 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
2   <soap:Header/>
3   <soap:Body>
4     <ns0:Order_MT xmlns:ns0="http://cpi.sap.com/demo">
5       <orderNumber>1</orderNumber>
6       <Environment>CS</Environment>
7     </ns0:Order_MT>
8   </soap:Body>

```

Back To Top ↑

Step 31 : Change **orderNumber** to **B** and **C** and click on **Send**. You should get the response as **2** and **3** respectively.

 Menu

SAP Training Class

Join Us Now

Awarded No.1 SAP Education Partner. 15+ Years of Experience. Apply Now.

sapeducation.atos.net

OPEN

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

```
1 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:demo
2   <soapenv:Header/>
3   <soapenv:Body>
4     <demo:Order_MT>
5       <orderNumber>45</orderNumber>
6       <Environment>CS</Environment>
7     </demo:Order_MT>
```

Body Cookies Headers (16) Test Results

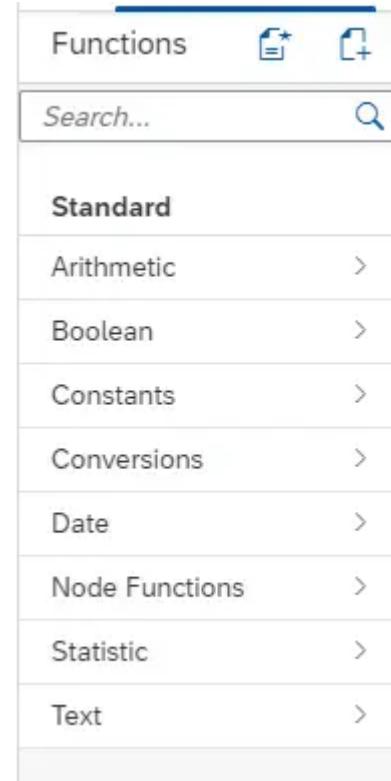
Pretty Raw Preview Visualize XML ▾

```
2 <soap:Envelope>
3   <soap:Header/>
4   <soap:Body>
5     <ns0:Order_MT xmlns:ns0="http://cpi.sap.com/demo">
6       <orderNumber>45</orderNumber>
7       <Environment>CS</Environment>
8     </ns0:Order_MT>
9   </soap:Body>
10 </soap:Envelope>
```

SELF- TRY:

☰ Menu

2. Try adding different functions available in Mapping window to transform source to target and test the same.



SAP CPI Message Mapping Tutorial

The below is a detailed example with SAP CPI **message mapping using if conditions**, and how to do namespace mapping. If you are good to go with Message mapping in CPI let's look into a bit advanced version which is **importing value mapping to SAP CPI**.



Back To Top

 Menu

FINAL VERDICT :

In conclusion, I hope you enjoyed reading this article on "**Message Mapping and Value Mapping in SAP CPI**", If yes, then don't forget to spread the word about it. Do send the feedback and to know more about it. Also, have a sneak peek at our [SAP CPI Tutorials](#). Signing off Sudarshan@recodehive.com

[Adaptation](#)[Growth Hacking](#)[Developer Bundle](#)[Hope and Focus](#)[My Day at Google Campus](#)[Time Machine](#)[Manifestation of life after death](#)

What is Multi Mapping in SAP CPI?

It's a way of using multiple message in sender or receiver system for your message mapping.

How Message Mapping work?

It uses Internal Queues, basically these are XML structure of the source message and nodes of these function is categorised into context. This can be used to trace the programs etc.



Back To Top

 Menu

0 comments Sort by Newest



Add a comment...

Facebook Comments Plugin

326641.5K70 Shares

Tagged [CPI](#), [Learning](#), [Mapping](#), [SAP](#), [SAP CPI](#)

RELATED POSTS

 Menu[!\[\]\(c7774dea93eb10ead3ed0542c77a8534_img.jpg\) Header based Routing & Local Integration Process](#)[!\[\]\(f15da8627380db409bac161a6cb03047_img.jpg\) Importing Value Mapping in SAP CPI !\[\]\(2b67363fd725a7774f62e5fccbbf7d79_img.jpg\)](#)

Back To Top

☰ Menu

 Menu

 Menu

☰ Menu

Back To Top ↑

 Menu

 Menu

☰ Menu

 Menu

 Menu

 Menu

 Menu

 Menu

 Menu

☰ Menu

Back To Top ↑

☰ Menu

 Menu

☰ Menu

Back To Top ↑

 Menu

☰ Menu

Back To Top ↑

 Menu

[Terms and conditions](#) | [Privacy Policy](#) | [Career](#) | [Contact us](#) | [About us](#)

Recode Hive © 2022 | All right reserved

Back To Top ↑