Lab 2 -Function Outbound Endpoint

Ver. 1.0.0

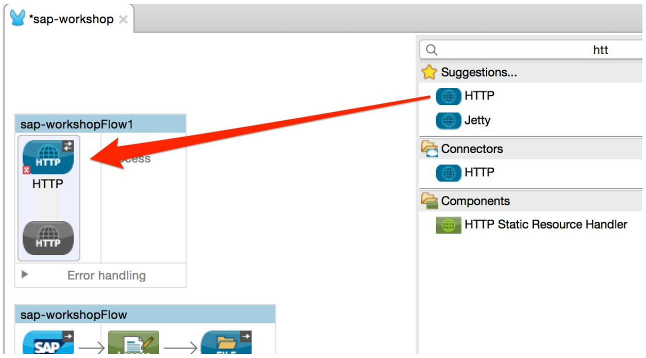
Overview

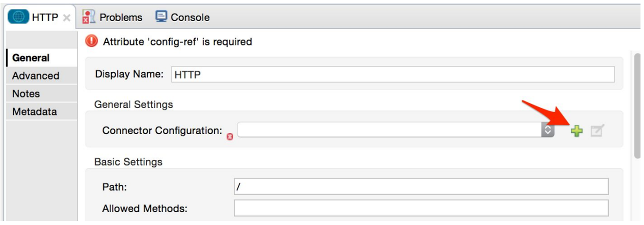
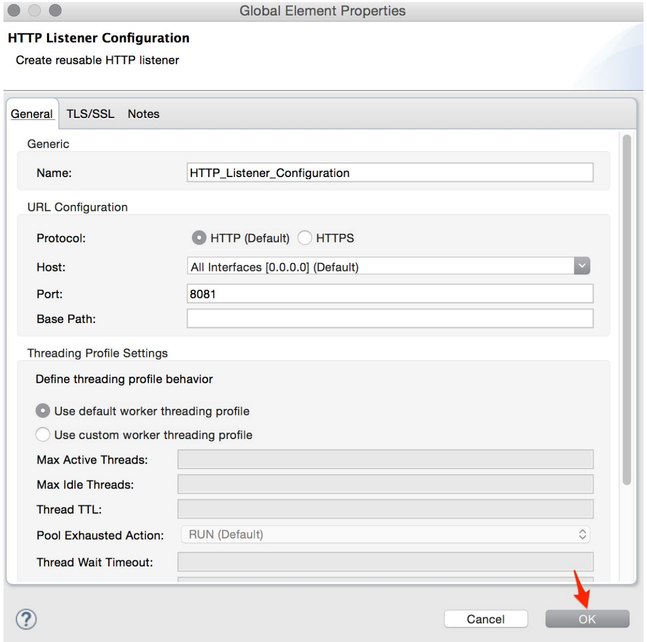
This next lab will focus on using the SAP Connector as a Function Outbound Endpoint. The flow that you create will call a BAPI on your SAP server to retrieve a list of customers. Using the previous Mule project from Lab 1, we'll extend and add a new flow to the project.

Reminder: This lab requires the use of your own SAP instance. If you don't have one, you can leverage a hosted SAP service. For the purposes of this lab, we utilized Sandbox SAP.

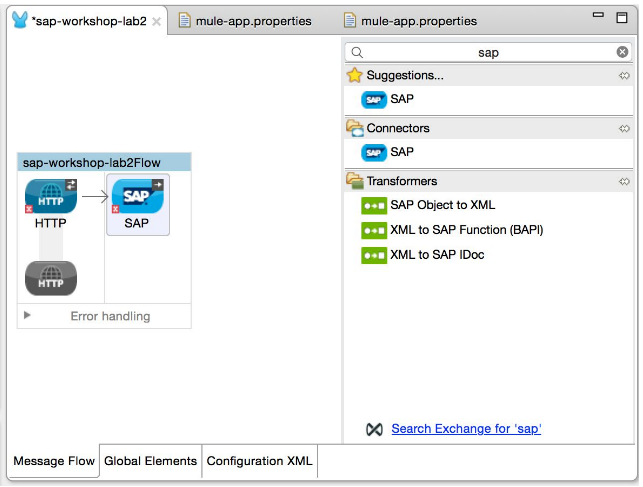
Steps

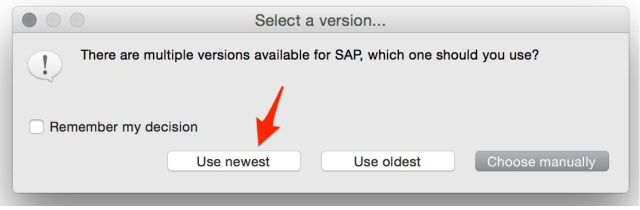
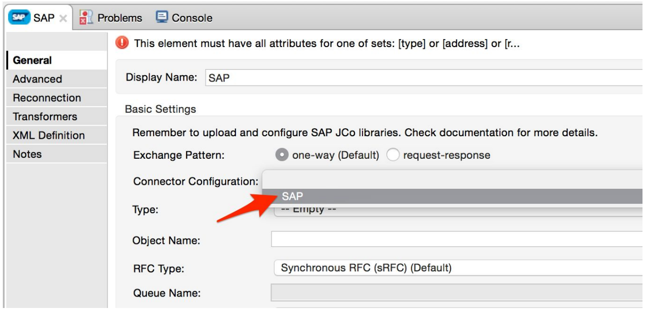
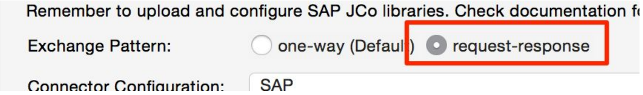
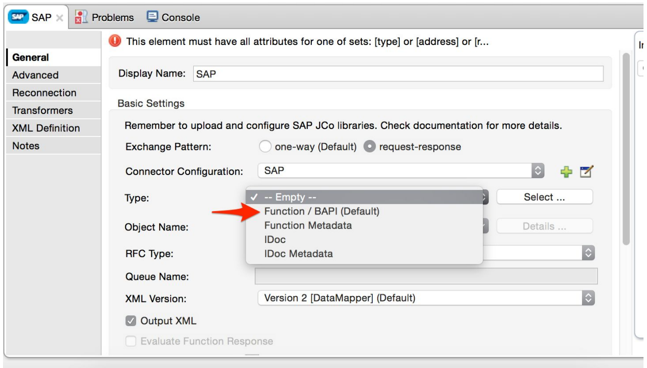
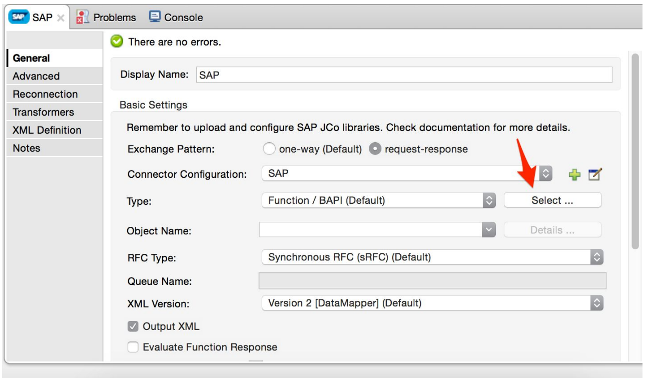
1. Setup HTTP Connector

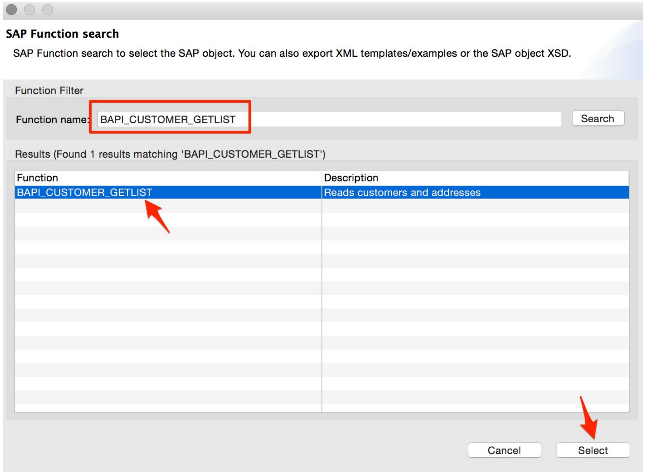
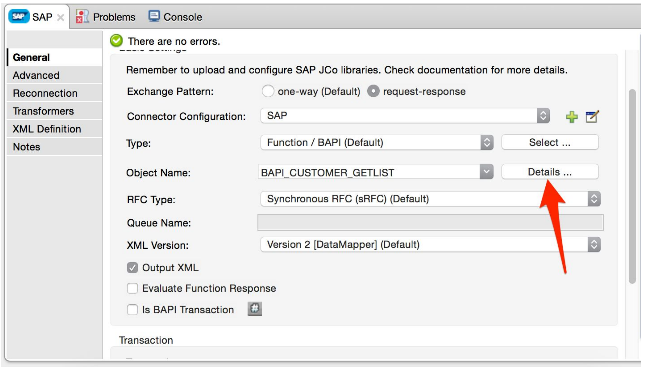
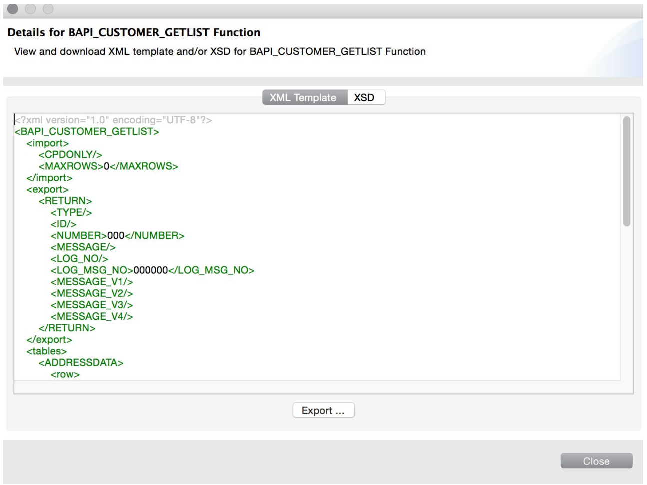
1.1	In the toolkit, search for the HTTP Connector by typing in 'http' in the Search bar. Drag and drop the **HTTP Connector** onto the canvas.	 *sap-workshop htt Suggestions... HTTP Jetty Connectors HTTP Components HTTP Static Resource Handler sap-workshopFlow1 HTTP HTTP Error handling sap-workshopFlow

1.2	In the **Mule Properties** tab for the **HTTP Connector**, click on the **'+'** sign.	 HTTP ✕ Problems Console Attribute 'config-ref' is required General Advanced Notes Metadata Display Name: HTTP General Settings Connector Configuration: Basic Settings Path: / Allowed Methods:
1.3	Accept the default values and click **OK**. When you run the application later, you will use the following URL: http://localhost:8081	 Global Element Properties **HTTP Listener Configuration** Create reusable HTTP listener General TLS/SSL Notes Generic Name: HTTP_Listener_Configuration URL Configuration Protocol: ● HTTP (Default) ○ HTTPS Host: All Interfaces [0.0.0.0] (Default) Port: 8081 Base Path: Threading Profile Settings Define threading profile behavior ● Use default worker threading profile ○ Use custom worker threading profile Max Active Threads: Max Idle Threads: Thread TTL: Pool Exhausted Action: RUN (Default) Thread Wait Timeout: Cancel OK

2. Configure SAP Connector

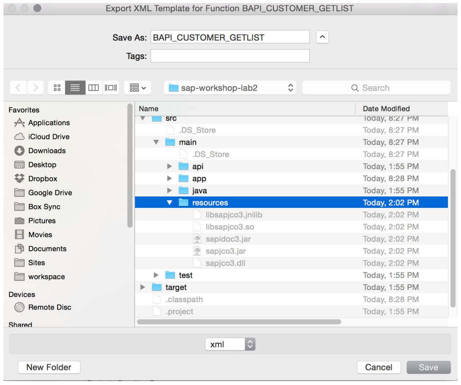
2.1	Next, let's add the SAP Connector. In the toolkit, search for the SAP Connector by typing in 'SAP' in the Search bar. Drag and drop the **SAP Connector** onto the canvas. This will set the **SAP Connector** as an outbound endpoint.	 *sap-workshop-lab2 ✕ mule-app.properties mule-app.properties sap Suggestions... SAP Connectors SAP Transformers SAP Object to XML XML to SAP Function (BAPI) XML to SAP IDoc Search Exchange for 'sap' sap-workshop-lab2Flow HTTP SAP Error handling Message Flow Global Elements Configuration XML

2.2	If the **Select a version...** window pops up, select **Use newest**	 Select a version... There are multiple versions available for SAP, which one should you use? □ Remember my decision Use newest · Use oldest · Choose manually
2.3	In the **Connector Configuration** field, select the configuration that we created in Lab 1. It should be listed as **SAP**	 SAP · Problems · Console General · Advanced · Reconnection · Transformers · XML Definition · Notes This element must have all attributes for one of sets: [type] or [address] or [r... Display Name: SAP Basic Settings Remember to upload and configure SAP JCo libraries. Check documentation for more details. Exchange Pattern: ● one-way (Default) ○ request-response Connector Configuration: SAP Type: -- Empty -- Object Name: RFC Type: Synchronous RFC (sRFC) (Default) Queue Name:
2.4	Next, select **request-response** for the **Exchange Pattern** field.	 Remember to upload and configure SAP JCo libraries. Check documentation f Exchange Pattern: ○ one-way (Default) ● request-response Connector Configuration: SAP
2.5	For the **Type** field, select **Function / BAPI (Default)**	 SAP · Problems · Console General · Advanced · Reconnection · Transformers · XML Definition · Notes This element must have all attributes for one of sets: [type] or [address] or [r... Display Name: SAP Basic Settings Remember to upload and configure SAP JCo libraries. Check documentation for more details. Exchange Pattern: ○ one-way (Default) ● request-response Connector Configuration: SAP Type: ✓ -- Empty -- / Function / BAPI (Default) / Function Metadata / IDoc / IDoc Metadata · Select ... Object Name: · Details ... RFC Type: Queue Name: XML Version: Version 2 [DataMapper] (Default) ☑ Output XML □ Evaluate Function Response
2.6	On the same line, next to the **Type** field, click on the **Select** button to search for the SAP object to be used in the transaction.	 SAP · Problems · Console General · Advanced · Reconnection · Transformers · XML Definition · Notes There are no errors. Display Name: SAP Basic Settings Remember to upload and configure SAP JCo libraries. Check documentation for more details. Exchange Pattern: ○ one-way (Default) ● request-response Connector Configuration: SAP Type: Function / BAPI (Default) · Select ... Object Name: · Details ... RFC Type: Synchronous RFC (sRFC) (Default) Queue Name: XML Version: Version 2 [DataMapper] (Default) ☑ Output XML □ Evaluate Function Response

2.7	Enter the following name in the **Function name** field: **BAPI_CUSTOMER_GETLIST** and click on **Search.** Select the result that is returned and click on **Select**	 **SAP Function search** SAP Function search to select the SAP object. You can also export XML templates/examples or the SAP object XSD. Function Filter Function name: BAPI_CUSTOMER_GETLIST [Search] Results (Found 1 results matching 'BAPI_CUSTOMER_GETLIST') Function: BAPI_CUSTOMER_GETLIST — Description: Reads customers and addresses [Cancel] [Select]
2.8	The **SAP Connector** provides a way to retrieve the XML and XSD for the SAP Object. Click on the **Details** button next to the **Object Name** field.	 There are no errors. Remember to upload and configure SAP JCo libraries. Check documentation for more details. Exchange Pattern: ○ one-way (Default) ● request-response Connector Configuration: SAP Type: Function / BAPI (Default) [Select ...] Object Name: BAPI_CUSTOMER_GETLIST [Details ...] RFC Type: Synchronous RFC (sRFC) (Default) Queue Name: XML Version: Version 2 [DataMapper] (Default) ☑ Output XML ☐ Evaluate Function Response ☐ Is BAPI Transaction Transaction
2.9	In the **Details for BAPI_CUSTOMER_GETLIST Function** window, you can view and download the XML template and the XSD for the **BAPI_CUSTOMER_GETLIST** function.	 **Details for BAPI_CUSTOMER_GETLIST Function** View and download XML template and/or XSD for BAPI_CUSTOMER_GETLIST Function [XML Template] [XSD] [Export ...] [Close]

```
<?xml version="1.0" encoding="UTF-8"?>
<BAPI_CUSTOMER_GETLIST>
  <import>
    <CPDONLY/>
    <MAXROWS>0</MAXROWS>
  </import>
  <export>
    <RETURN>
      <TYPE/>
      <ID/>
      <NUMBER>000</NUMBER>
      <MESSAGE/>
      <LOG_NO/>
      <LOG_MSG_NO>000000</LOG_MSG_NO>
      <MESSAGE_V1/>
      <MESSAGE_V2/>
      <MESSAGE_V3/>
      <MESSAGE_V4/>
    </RETURN>
  </export>
  <tables>
    <ADDRESSDATA>
      <row>
```

2.10 Click on **Export** and save the XML Template to the **src/main/resources** folder.



Export XML Template for Function BAPI_CUSTOMER_GETLIST

Save As: BAPI_CUSTOMER_GETLIST

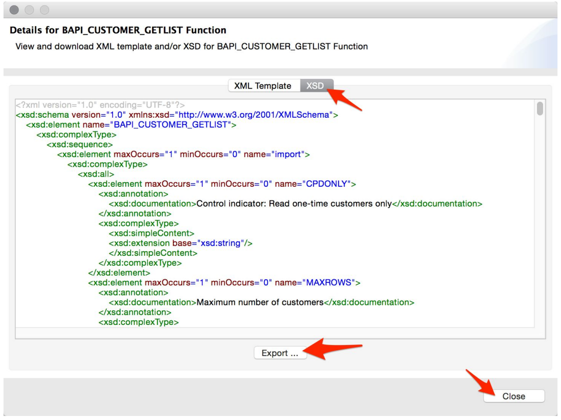
Tags:

sap-workshop-lab2

Name	Date Modified
src	Today, 8:27 PM
.DS_Store	Today, 8:27 PM
main	Today, 8:27 PM
.DS_Store	Today, 8:27 PM
api	Today, 1:55 PM
app	Today, 8:28 PM
java	Today, 1:55 PM
resources	Today, 2:02 PM
libsapjco3.jnilib	Today, 2:02 PM
libsapjco3.so	Today, 2:02 PM
sapidoc3.jar	Today, 2:02 PM
sapjco3.jar	Today, 2:02 PM
sapjco3.dll	Today, 2:02 PM
test	Today, 1:55 PM
target	Today, 1:55 PM
.classpath	Today, 8:28 PM
.project	Today, 1:55 PM

xml

2.11 Repeat the process for the **XSD** file and store it in the same location.

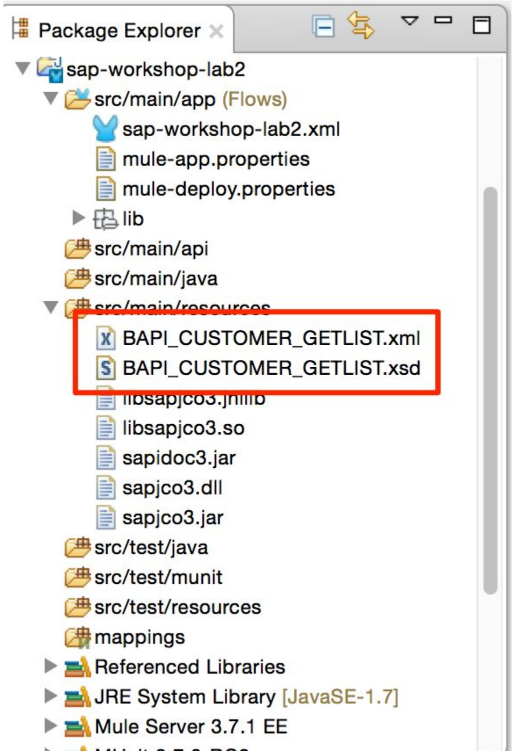


Details for BAPI_CUSTOMER_GETLIST Function

View and download XML template and/or XSD for BAPI_CUSTOMER_GETLIST Function

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="BAPI_CUSTOMER_GETLIST">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element maxOccurs="1" minOccurs="0" name="import">
          <xsd:complexType>
            <xsd:all>
              <xsd:element maxOccurs="1" minOccurs="0" name="CPDONLY">
                <xsd:annotation>
                  <xsd:documentation>Control indicator: Read one-time customers only</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                  <xsd:simpleContent>
                  <xsd:extension base="xsd:string"/>
                  </xsd:simpleContent>
                </xsd:complexType>
              </xsd:element>
              <xsd:element maxOccurs="1" minOccurs="0" name="MAXROWS">
                <xsd:annotation>
                  <xsd:documentation>Maximum number of customers</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
```

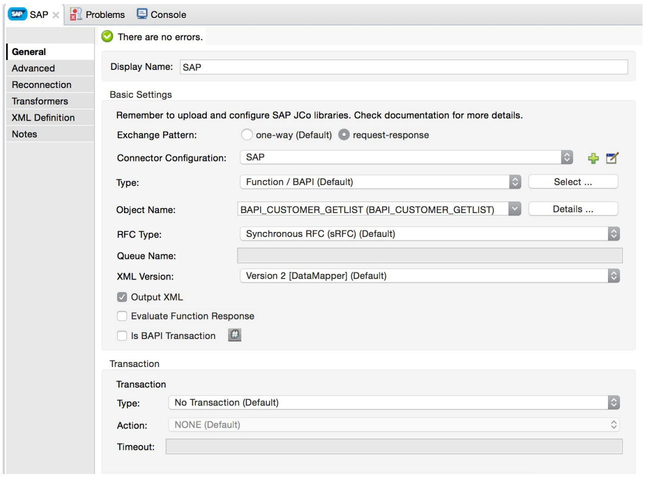
2.12 The **Package Explorer** for the project should contain the following files.



```
sap-workshop-lab2
  src/main/app (Flows)
    sap-workshop-lab2.xml
    mule-app.properties
    mule-deploy.properties
    lib
  src/main/api
  src/main/java
  src/main/resources
    BAPI_CUSTOMER_GETLIST.xml
    BAPI_CUSTOMER_GETLIST.xsd
    libsapjco3.jnilib
    libsapjco3.so
    sapidoc3.jar
    sapjco3.dll
    sapjco3.jar
  src/test/java
  src/test/munit
  src/test/resources
  mappings
  Referenced Libraries
  JRE System Library [JavaSE-1.7]
  Mule Server 3.7.1 EE
```

2.13

Keep the default values for the remaining fields in the SAP Configuration Properties window.



There are no errors.

Display Name: SAP

Basic Settings

Remember to upload and configure SAP JCo libraries. Check documentation for more details.

Exchange Pattern: ○ one-way (Default) ● request-response

Connector Configuration: SAP

Type: Function / BAPI (Default)

Object Name: BAPI_CUSTOMER_GETLIST (BAPI_CUSTOMER_GETLIST)

RFC Type: Synchronous RFC (sRFC) (Default)

Queue Name:

XML Version: Version 2 [DataMapper] (Default)

☑ Output XML

□ Evaluate Function Response

□ Is BAPI Transaction

Transaction

Transaction

Type: No Transaction (Default)

Action: NONE (Default)

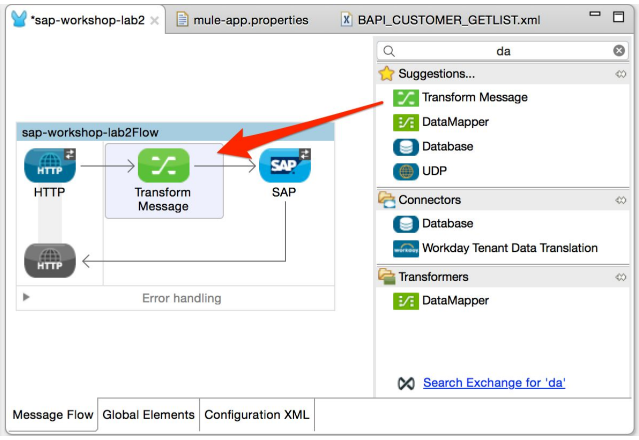
Timeout:

3. Use DataWeave to Create a BAPI Function

3.1

Now that the SAP Connector is configured with the correct credentials and SAP Object we intend to use, we need to setup the message to pass in.

In the toolkit, search for the DataWeave component. Type in ‘data’ in the search and then drag and drop the **Transform Message** component into the canvas. Drop it between the **HTTP Connector** and the **SAP Connector**



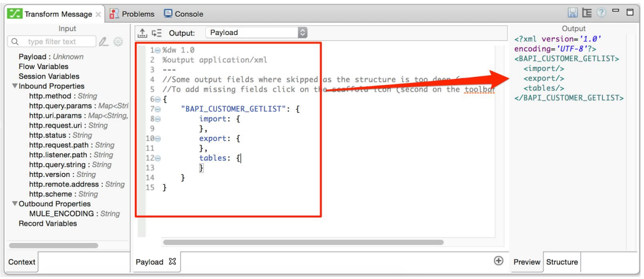
3.2

In the **Transform Message Configuration Properties** window, you can see how the component scaffolds out the expected message to be passed to the **SAP Connector** using **DataWeave**

The **Output** on the right shows a preview of the payload message as you modify the message using DataWeave

More information about the DataWeave language can be found here:

http://mulesoft.github.io/data-weave



```
%dw 1.0
%output application/xml
---
//Some output fields where skipped as the structure is too deep (
//To add missing fields click on the scaffold icon (second on the toolbar
{
    "BAPI_CUSTOMER_GETLIST": {
        import: {
        },
        export: {
        },
        tables: {
        }
    }
}
```

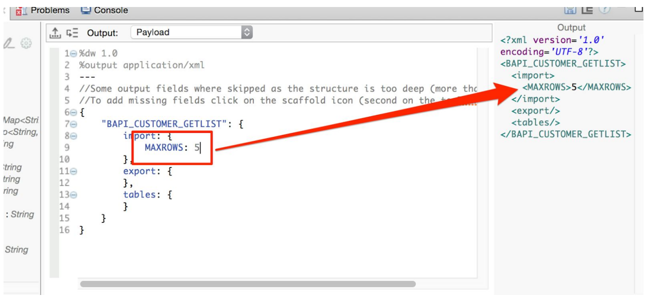
```
<?xml version='1.0'
encoding='UTF-8'?>
<BAPI_CUSTOMER_GETLIST>
  <import/>
  <export/>
  <tables/>
</BAPI_CUSTOMER_GETLIST>
```

3.3 Let’s make some modifications to the message. First, we’ll set the maximum number of rows to return from SAP.

Under the **import** node, add the following line:

MAXROWS: 5

Notice how the preview on the right under **Output** updates the XML message that will be passed to SAP.



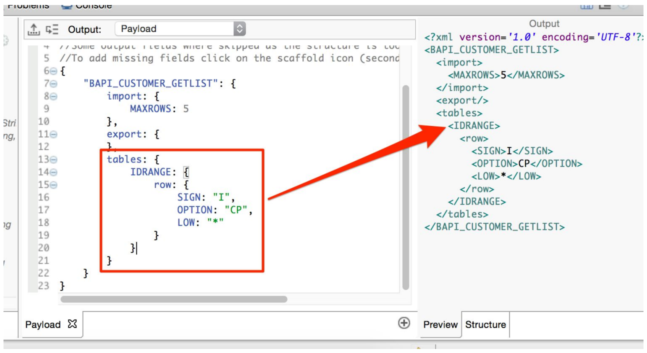
```
%dw 1.0
%output application/xml
---
//Some output fields where skipped as the structure is too deep (more tha
//To add missing fields click on the scaffold icon (second on the to
{
    "BAPI_CUSTOMER_GETLIST": {
        import: {
            MAXROWS: 5
        },
        export: {
        },
        tables: {
        }
    }
}
```

```
<?xml version='1.0'
encoding='UTF-8'?>
<BAPI_CUSTOMER_GETLIST>
  <import>
    <MAXROWS>5</MAXROWS>
  </import>
  <export/>
  <tables/>
</BAPI_CUSTOMER_GETLIST>
```

3.4 Next we need the ID range of customer records we want returned:

Under the tables node, add the following lines:

```
IDRANGE: {
        row: {
                SIGN: "I",
                OPTION: "CP",
                LOW: "*"
        }
}
```



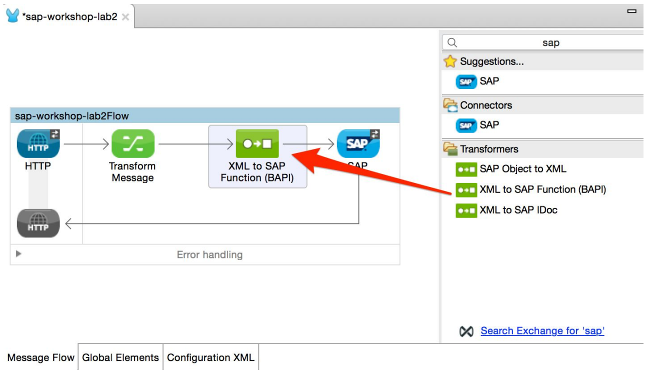
```
//Some output fields where skipped as the structure is too
//To add missing fields click on the scaffold icon (second
{
    "BAPI_CUSTOMER_GETLIST": {
        import: {
            MAXROWS: 5
        },
        export: {
        },
        tables: {
            IDRANGE: {
                row: {
                    SIGN: "I",
                    OPTION: "CP",
                    LOW: "*"
                }
            }
        }
    }
}
```

```
<?xml version='1.0' encoding='UTF-8'?>
<BAPI_CUSTOMER_GETLIST>
  <import>
    <MAXROWS>5</MAXROWS>
  </import>
  <export/>
  <tables>
    <IDRANGE>
      <row>
        <SIGN>I</SIGN>
        <OPTION>CP</OPTION>
        <LOW>*</LOW>
      </row>
    </IDRANGE>
  </tables>
</BAPI_CUSTOMER_GETLIST>
```

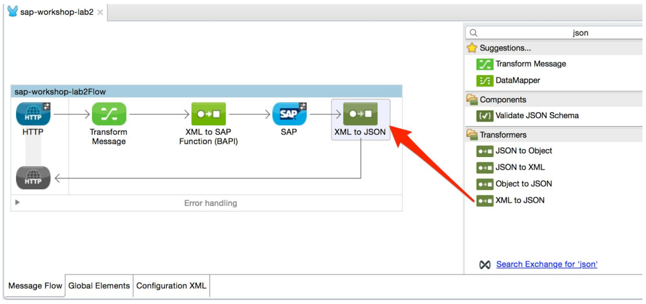
4. Transform XML to SAP Function

4.1 Now that the XML has the correct parameters, we need to transform it from XML to an SAP Function.

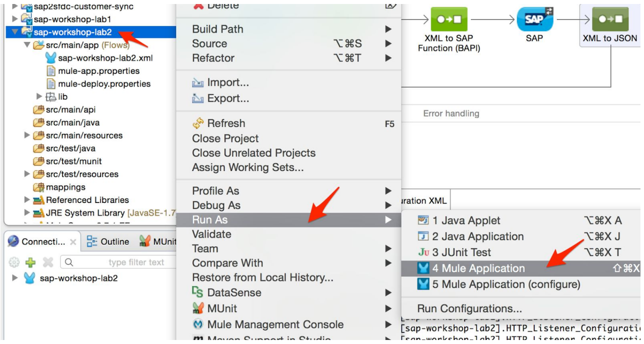
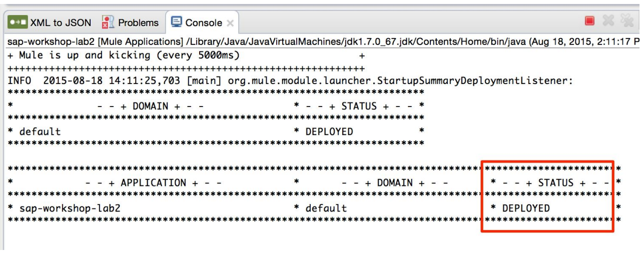
From the toolkit, search for **SAP** and drag and drop the **XML to SAP Function (BAPI)** transformer and place it between **Transform Message** and the **SAP Connector**



<table>
<tr><td>4.2</td><td>Once the SAP Connector processes the function the data will be returned as XML. Let’s go ahead and transform that and output the customer list as JSON data.

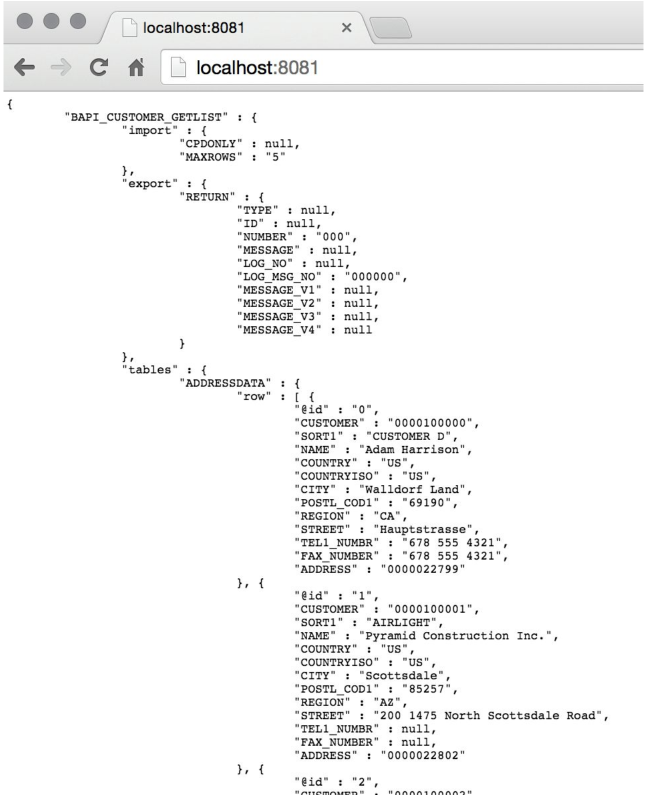
Search for <b>json</b> and then drag and drop the <b>XML to JSON</b> transformer and place it after the <b>SAP Connector</b>.</td><td>
</td></tr>
</table>

5. Test Project

<table>
<tr><td>5.1</td><td>Our next step is to test the flow we’ve built. In the <b>Package Explorer</b>, right click on the project folder, select <b>Run As</b>, and the click on <b>Mule Application</b></td><td>
</td></tr>
<tr><td>5.2</td><td>The <b>Console</b> tab should pop-up now. Wait for the status to show <b>DEPLOYED</b> before moving onto the next step.</td><td>

<pre>
sap-workshop-lab2 [Mule Applications] /Library/Java/JavaVirtualMachines/jdk1.7.0_67.jdk/Contents/Home/bin/java (Aug 18, 2015, 2:11:17 P
+ Mule is up and kicking (every 5000ms)   +
+++
INFO 2015-08-18 14:11:25,703 [main] org.mule.module.launcher.StartupSummaryDeploymentListener:
****
*   - - + DOMAIN + - -   * - - + STATUS + - - *
****
* default   * DEPLOYED   *
****

****
*   - - + APPLICATION + - -   *   - - + DOMAIN + - -   * - - + STATUS + - - *
****
* sap-workshop-lab2   * default   * DEPLOYED   *
****
</pre></td></tr>
</table>

5.3	Let’s test out our flow now. Switch to your browser and enter the following URL: http://localhost:8081 If everything was configured correctly, you should see the following screen on the right.	

```
{
        "BAPI_CUSTOMER_GETLIST" : {
                "import" : {
                        "CPDONLY" : null,
                        "MAXROWS" : "5"
                },
                "export" : {
                        "RETURN" : {
                                "TYPE" : null,
                                "ID" : null,
                                "NUMBER" : "000",
                                "MESSAGE" : null,
                                "LOG_NO" : null,
                                "LOG_MSG_NO" : "000000",
                                "MESSAGE_V1" : null,
                                "MESSAGE_V2" : null,
                                "MESSAGE_V3" : null,
                                "MESSAGE_V4" : null
                        }
                },
                "tables" : {
                        "ADDRESSDATA" : {
                                "row" : [ {
                                        "@id" : "0",
                                        "CUSTOMER" : "0000100000",
                                        "SORT1" : "CUSTOMER D",
                                        "NAME" : "Adam Harrison",
                                        "COUNTRY" : "US",
                                        "COUNTRYISO" : "US",
                                        "CITY" : "Walldorf Land",
                                        "POSTL_COD1" : "69190",
                                        "REGION" : "CA",
                                        "STREET" : "Hauptstrasse",
                                        "TEL1_NUMBR" : "678 555 4321",
                                        "FAX_NUMBER" : "678 555 4321",
                                        "ADDRESS" : "0000022799"
                                }, {
                                        "@id" : "1",
                                        "CUSTOMER" : "0000100001",
                                        "SORT1" : "AIRLIGHT",
                                        "NAME" : "Pyramid Construction Inc.",
                                        "COUNTRY" : "US",
                                        "COUNTRYISO" : "US",
                                        "CITY" : "Scottsdale",
                                        "POSTL_COD1" : "85257",
                                        "REGION" : "AZ",
                                        "STREET" : "200 1475 North Scottsdale Road",
                                        "TEL1_NUMBR" : null,
                                        "FAX_NUMBER" : null,
                                        "ADDRESS" : "0000022802"
                                }, {
                                        "@id" : "2",
                                        "CUSTOMER" : "0000100002"
```

Summary

This lab demonstrated how simple it is to retrieve a list of customers using the SAP Connector.

In the next lab, you will learn how to use the SAP Connector as an IDoc Outbound Endpoint to insert a customer into SAP

Lab 3 - IDoc Outbound Endpoint