

Module 4: Connecting to Additional Resources



In this module, you will learn:

- To connect to databases.
- To connect to files.
- To connect to JMS queues.
- To connect to SaaS applications.
- To discover and install connectors not bundled with Anypoint Studio.
- About developing your own custom connectors with Anypoint Connector DevKit.

Walkthrough 4-1: Connect to a database (MySQL)

In this walkthrough, you will connect to a MySQL database that contains information about American flights. You will:

- Create a new flow to receive requests at <http://localhost:8081/american>.
- Add a Database connector endpoint that connects to a MySQL database.
- Write a query to return all flights from the database for the static destination SFO.
- Modify the query to use a dynamic destination from a query parameter.
- Use the Object to String transformer to return the results as a string.

Note: You will get a destination dynamically from a form in a later module.

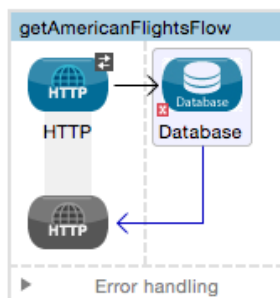


Create a new flow with an HTTP Listener connector endpoint

1. Return to `apessentials.xml`.
2. Create a new flow above the Delta flow and name it `getAmericanFlightsFlow`.
3. In the HTTP Properties view, set the connector configuration to the existing `HTTP_Listener_Configuration`.
4. Set the path to `/american` and the allowed methods to GET.

Add and configure a Database connector endpoint

5. Add a Database connector to the process section of the flow.



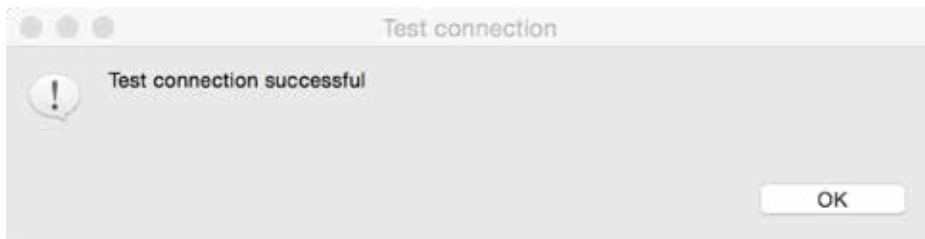
6. In the Database Properties view, change the display name to American DB Request.
7. Click the Add button next to connector configuration.
8. In the Choose Global Type dialog box, select Connector Configuration > MySQL Configuration and click OK.
9. In the Global Element Properties dialog box, set the name to Training_MySQL_Configuration.
10. Set the server, port, user, password, and database values to the values listed in the course snippets.txt file for the MySQL database.
11. Click the Add File button next to the MySQL driver required dependency.
12. Browse to the mysql-connector-java-{version}-bin.jar file located in the MUEssentials3.7_studentFiles_{date}/jars folder and click Open.

The screenshot shows the 'Global Element Properties' dialog box for a 'MySQL Configuration' element. The 'General' tab is selected, showing the following fields and options:

- Name:** Training_MySQL_Configuration
- Database configuration parameters:**
 - Host:** mudb.mulesoft-training.com
 - Port:** 3306
 - User:** mule
 - Password:** (masked with dots) ☐ Show password
 - Database:** training
- Configure via spring-bean:** ☐
DataSource Reference: (empty field with dropdown and icons)
- Database URL:** ☐
URL: (empty field)
- ☒ **Enable DataSense**
- Required dependencies:**
 - ☒ **MySQL Driver** (src/main/resources/mysql-connector-java-5.0.8-bin.jar)

At the bottom of the dialog are buttons for '?', 'Test Connection...', 'Cancel', and 'OK'.

- Back in the Global Element Properties dialog box, click the Test Connection button; you should get a successful test dialog box.



Note: If your database connectivity test fails, make sure you are not behind a firewall restricting access to port 3306. If you cannot access port 3306, use the setup instructions to install and set up VirtualBox to use the MuleSoft Training server image.

- Click OK to close the dialog box.
- Click OK to close the Global Element Properties dialog box.

Write a query to return all flights

- In the Properties view for the database endpoint, select the Select operation.
- In the parameterized query text box, write a query to select all records from the flights database.

```
SELECT *  
FROM flights
```

Test the application

- Save the file and run the application.
- Make a request to <http://localhost:8081/american>; you should get some garbled plain text displayed or contained in a download file, which is the tool's best representation of an ArrayList.

A screenshot of a file explorer window. The 'File Path' is set to '~/Downloads/american'. The file 'american' is selected. The content of the file is displayed as garbled text, including fragments like 'sr java.util.LinkedList)SJJ`à" xpw sr\$org.mule.util.CaseInsensitiveHashMap', 'Boeing 787t code2t 0001t toAirportt LAXt takeOffDatesr', 'java.sql.Date `Fh?5f6 xr java.util.DatehjÅ KYt xpw K Z`xt fromAirportt MUAt price', 'Boeing 747q~ t 0123q~t CLEq~', 'sq~wK fxq~ t MUAq~ sq~ ,q~ t American Airlinesq~ t 7q~ t rreexsq~w?@ q~ t', 'Boeing 777q~ t 0192q~t LAXq~', 'sq~wK Z`xq~ t MUAq~ sq~ ,q~ t American Airlinesq~ t noneq~ t rreexsq~w?@ q~ t', 'Airbus 800q~ t 100q~t KULq~', 'sq~wK Z`xq~ t MUAq~ sq~ Ûq~ tMas Airlinesq~ t 20q~ t masxsq~w?@ q~ t', 'Boeing 737q~ t 1000q~t CLEq~', and 'sq~wK Z`xq~ t MUAq~ sq~ xq~ t American Airlinesq~ t 5q~ t rreexsq~w?@ q~ t'.

Debug the application

20. Add a Logger component after the Database connector endpoint and add a breakpoint to it.
21. Save the file, debug the application, and make a request to <http://localhost:8081/american>.
22. In the Mule Debugger view, drill-down and look at the data contained in the payload.
23. Find the name of the field for the flight destination.

The screenshot shows the Mule Debugger interface with the 'Logger' tab selected. The console displays the following data:

Name	Value	Type
Message		org.mule.DefaultMuleMessage
Message Processor	Logger	org.mule.api.processor.LoggerMessageProcess
payload	size = 12	java.util.LinkedList
0	{planeType=Boeing 787, code2=00...	org.mule.util.CaseInsensitiveHashMap
0	planeType=Boeing 787	org.apache.commons.collections.map.Abstrac.
1	code2=0001	org.apache.commons.collections.map.Abstrac.
2	toAirport=LAX	org.apache.commons.collections.map.Abstrac.
3	takeOffDate=2015-01-20	org.apache.commons.collections.map.Abstrac.
4	fromAirport=MUA	org.apache.commons.collections.map.Abstrac.
5	price=541	org.apache.commons.collections.map.Abstrac.
6	airlineName=American Airlines	org.apache.commons.collections.map.Abstrac.
7	seatsAvailable=none	org.apache.commons.collections.map.Abstrac.
8	code1=rree	org.apache.commons.collections.map.Abstrac.
1	{planeType=Boeing 747, code2=01...	org.mule.util.CaseInsensitiveHashMap
10	{planeType=Boeing 777, code2=45...	org.mule.util.CaseInsensitiveHashMap
11	{planeType=Boeing 737, code2=45...	org.mule.util.CaseInsensitiveHashMap

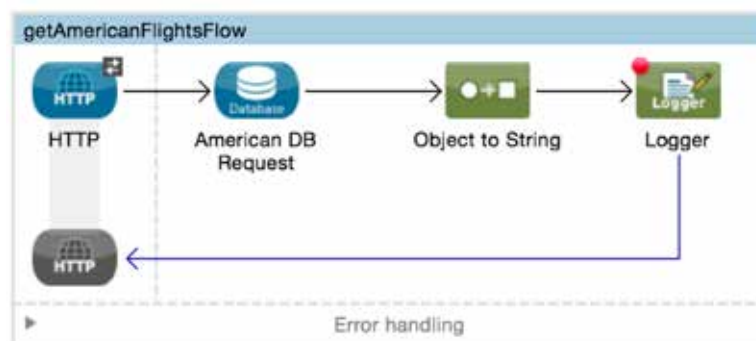
Below the table, the full payload is shown as a JSON object:

```
{planeType=Boeing 787, code2=0001, toAirport=LAX, takeOffDate=2015-01-20, fromAirport=MUA, price=541, airlineName=American Airlines, seatsAvailable=none, code1=rree}
```

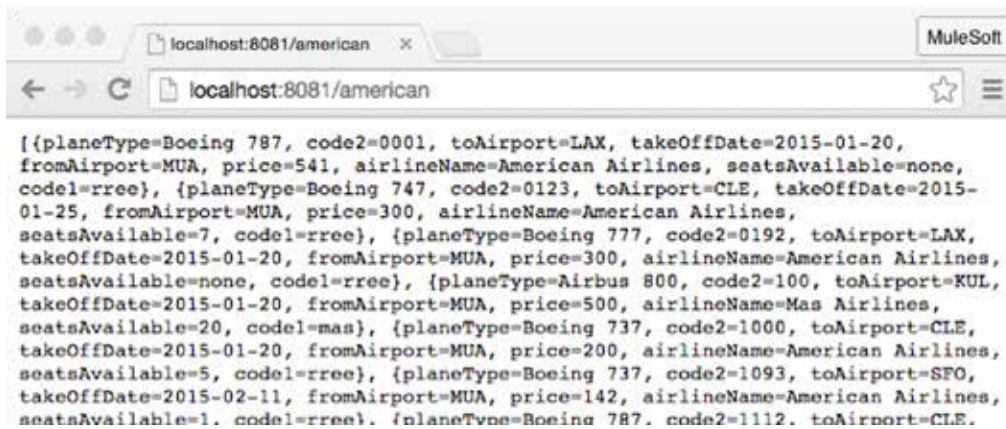
24. Stop the Mule Debugger.

Display the return data

25. Set the Logger message to #[payload].
26. Add an Object to String transformer before the Logger component.

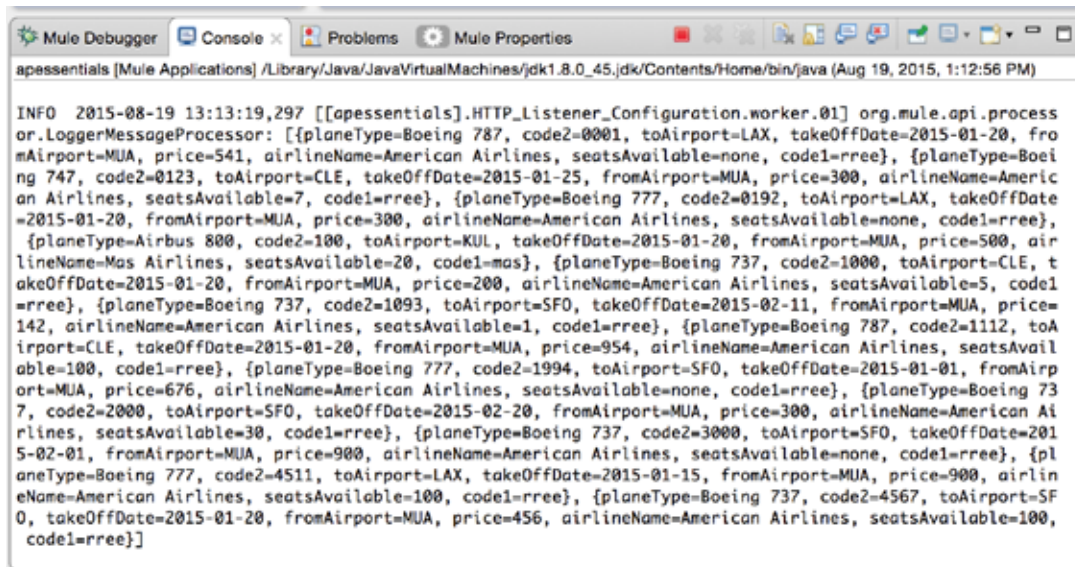


27. Save and run the application.
28. Make another request to <http://localhost:8081/american>; you should see the query results displayed.



```
[{"planeType": "Boeing 787", "code2": "0001", "toAirport": "LAX", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 541, "airlineName": "American Airlines", "seatsAvailable": "none", "code1": "rree"}, {"planeType": "Boeing 747", "code2": "0123", "toAirport": "CLE", "takeOffDate": "2015-01-25", "fromAirport": "MUA", "price": 300, "airlineName": "American Airlines", "seatsAvailable": 7, "code1": "rree"}, {"planeType": "Boeing 777", "code2": "0192", "toAirport": "LAX", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 300, "airlineName": "American Airlines", "seatsAvailable": "none", "code1": "rree"}, {"planeType": "Airbus 800", "code2": "100", "toAirport": "KUL", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 500, "airlineName": "Mas Airlines", "seatsAvailable": 20, "code1": "mas"}, {"planeType": "Boeing 737", "code2": "1000", "toAirport": "CLE", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 200, "airlineName": "American Airlines", "seatsAvailable": 5, "code1": "rree"}, {"planeType": "Boeing 737", "code2": "1093", "toAirport": "SFO", "takeOffDate": "2015-02-11", "fromAirport": "MUA", "price": 142, "airlineName": "American Airlines", "seatsAvailable": 1, "code1": "rree"}, {"planeType": "Boeing 787", "code2": "1112", "toAirport": "CLE", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 954, "airlineName": "American Airlines", "seatsAvailable": 100, "code1": "rree"}, {"planeType": "Boeing 777", "code2": "1994", "toAirport": "SFO", "takeOffDate": "2015-01-01", "fromAirport": "MUA", "price": 676, "airlineName": "American Airlines", "seatsAvailable": "none", "code1": "rree"}, {"planeType": "Boeing 737", "code2": "2000", "toAirport": "SFO", "takeOffDate": "2015-02-20", "fromAirport": "MUA", "price": 300, "airlineName": "American Airlines", "seatsAvailable": 30, "code1": "rree"}, {"planeType": "Boeing 737", "code2": "3000", "toAirport": "SFO", "takeOffDate": "2015-02-01", "fromAirport": "MUA", "price": 900, "airlineName": "American Airlines", "seatsAvailable": "none", "code1": "rree"}, {"planeType": "Boeing 777", "code2": "4511", "toAirport": "LAX", "takeOffDate": "2015-01-15", "fromAirport": "MUA", "price": 900, "airlineName": "American Airlines", "seatsAvailable": 100, "code1": "rree"}, {"planeType": "Boeing 737", "code2": "4567", "toAirport": "SFO", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 456, "airlineName": "American Airlines", "seatsAvailable": 100, "code1": "rree"}]
```

29. Look at the console; you should also see the query results there.



```
INFO 2015-08-19 13:13:19,297 [[apessentials].HTTP_Listener_Configuration.worker.01] org.mule.api.processor.LoggerMessageProcessor: [{"planeType": "Boeing 787", "code2": "0001", "toAirport": "LAX", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 541, "airlineName": "American Airlines", "seatsAvailable": "none", "code1": "rree"}, {"planeType": "Boeing 747", "code2": "0123", "toAirport": "CLE", "takeOffDate": "2015-01-25", "fromAirport": "MUA", "price": 300, "airlineName": "American Airlines", "seatsAvailable": 7, "code1": "rree"}, {"planeType": "Boeing 777", "code2": "0192", "toAirport": "LAX", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 300, "airlineName": "American Airlines", "seatsAvailable": "none", "code1": "rree"}, {"planeType": "Airbus 800", "code2": "100", "toAirport": "KUL", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 500, "airlineName": "Mas Airlines", "seatsAvailable": 20, "code1": "mas"}, {"planeType": "Boeing 737", "code2": "1000", "toAirport": "CLE", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 200, "airlineName": "American Airlines", "seatsAvailable": 5, "code1": "rree"}, {"planeType": "Boeing 737", "code2": "1093", "toAirport": "SFO", "takeOffDate": "2015-02-11", "fromAirport": "MUA", "price": 142, "airlineName": "American Airlines", "seatsAvailable": 1, "code1": "rree"}, {"planeType": "Boeing 787", "code2": "1112", "toAirport": "CLE", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 954, "airlineName": "American Airlines", "seatsAvailable": 100, "code1": "rree"}, {"planeType": "Boeing 777", "code2": "1994", "toAirport": "SFO", "takeOffDate": "2015-01-01", "fromAirport": "MUA", "price": 676, "airlineName": "American Airlines", "seatsAvailable": "none", "code1": "rree"}, {"planeType": "Boeing 737", "code2": "2000", "toAirport": "SFO", "takeOffDate": "2015-02-20", "fromAirport": "MUA", "price": 300, "airlineName": "American Airlines", "seatsAvailable": 30, "code1": "rree"}, {"planeType": "Boeing 737", "code2": "3000", "toAirport": "SFO", "takeOffDate": "2015-02-01", "fromAirport": "MUA", "price": 900, "airlineName": "American Airlines", "seatsAvailable": "none", "code1": "rree"}, {"planeType": "Boeing 777", "code2": "4511", "toAirport": "LAX", "takeOffDate": "2015-01-15", "fromAirport": "MUA", "price": 900, "airlineName": "American Airlines", "seatsAvailable": 100, "code1": "rree"}, {"planeType": "Boeing 737", "code2": "4567", "toAirport": "SFO", "takeOffDate": "2015-01-20", "fromAirport": "MUA", "price": 456, "airlineName": "American Airlines", "seatsAvailable": 100, "code1": "rree"}]
```

Note: To set the width of the console, select Anypoint Studio > Preferences > Run/Debug > Console and select Fixed width console and set a maximum character width.

Set a flow variable to hold the value of a query parameter

30. Select the Set Destination transformer in the getUnitedFlightsFlow and from the main menu, select Edit > Copy (or press Cmd+C/Ctrl+C).
31. Click in the process section of the getAmericanFlightsFlow and from the main menu, select Edit > Paste (or press Cmd+V/Ctrl+V); you should see a copy of the transformer added to the flow.
32. Move the transformer before the American DB Request endpoint.

33. In the Variable Properties view, change the display name to Set Destination and review the expression.

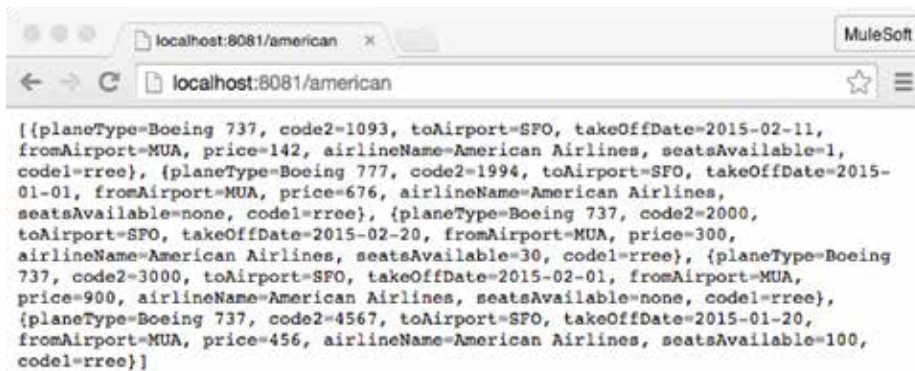


Modify the query to use a dynamic destination

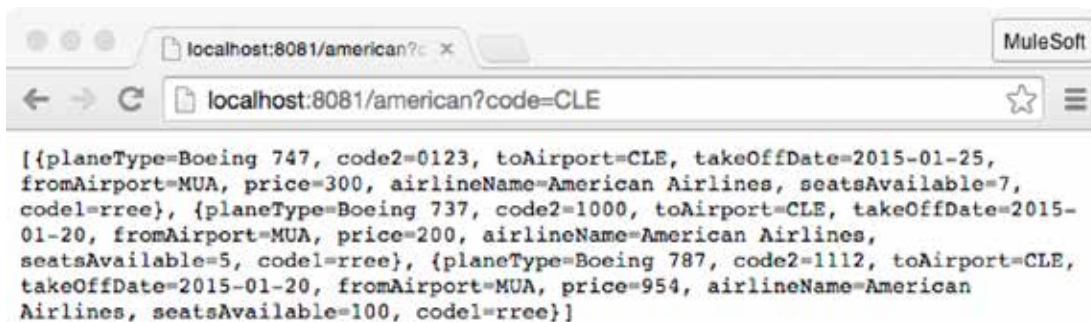
34. In the Properties view for the Database connector endpoint, modify the query to use the destination flow variable.

```
SELECT *  
FROM flights  
WHERE toAirport = #[flowVars.destination]
```

35. Save to redeploy the application and make another request to <http://localhost:8081/american>; you should now only see flights to SFO.



36. Make another request to <http://localhost:8081/american?code=CLE>; you should now see flights to CLE.

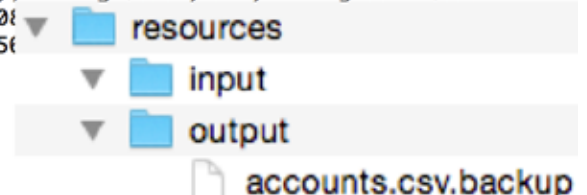


Walkthrough 4-2: Connect to a file (CSV)

In this walkthrough, you will load data from a local CSV file. You will:

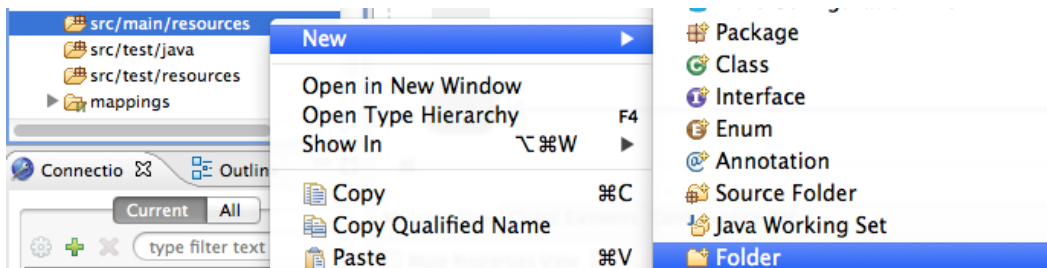
- Add and configure a File connector endpoint to watch an input directory, read the contents of any added files, and then rename the files and move them to an output folder.
- Add a CSV file to the input directory and watch it renamed and moved.
- Restrict the type of file read.
- Use the File to String transformer to return the results as a string.

```
INFO 2015-08-19 13:44:16,688 [[apessentials].getCSVAccountsFlow.stage1.02] org.mule.api.processor.Logger
MessageProcessor: Billing Street,Billing City,Billing Country,Billing State,Name,BillingPostalCode
111 Boulevard Hausmann,Paris,France,,Dog Park Industries,75008
400 South St,San Francisco,USA,CA,Iguana Park Industries,91156
777 North St,San Francisco,USA,CA,Cat Park Industries,91156
```

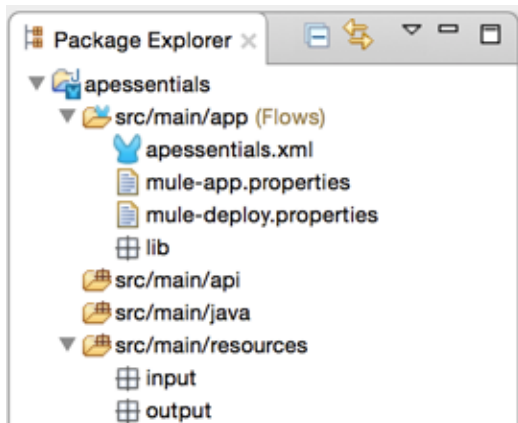


Create new directories

1. Return to apessentials.xml.
2. Right-click the src/main/resources folder in the Package Explorer and select New > Folder.

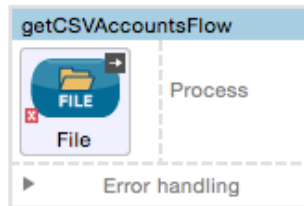


3. Set the folder name to input and click Finish.
4. Create a second folder called output.

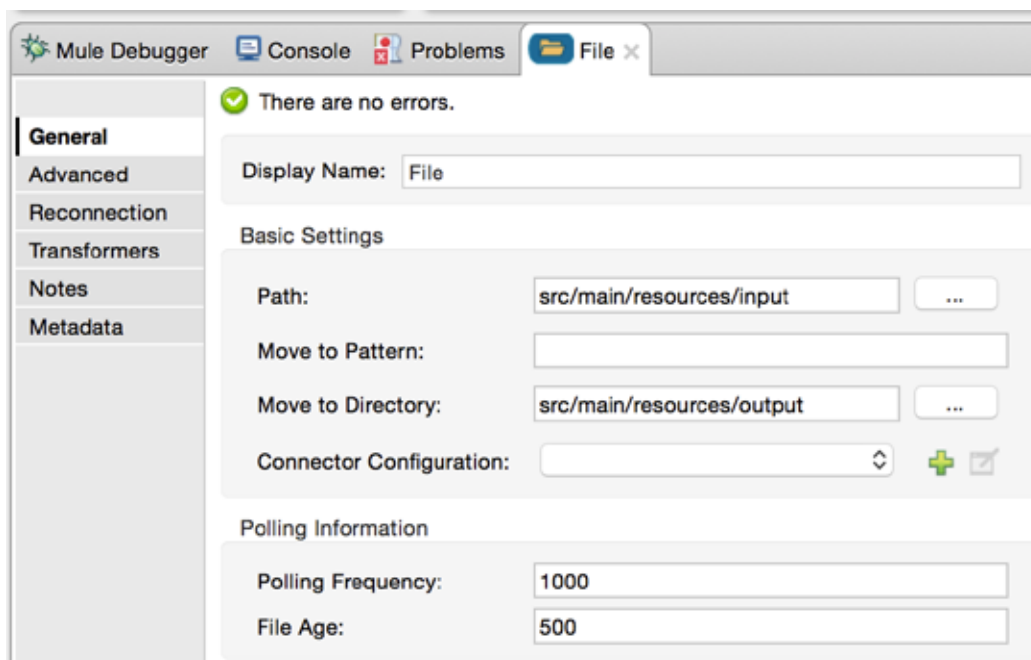


Add and configure a File connector endpoint

5. Drag a File connector from the palette and drop it in the canvas to create a new flow.
6. Rename the flow to getCSVAccountsFlow.



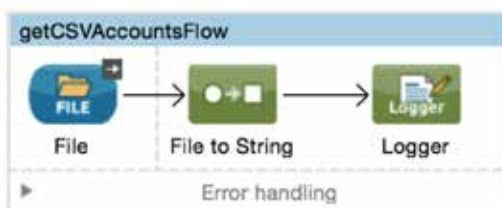
7. In the File Properties view, set the path to src/main/resources/input.
8. Set the move to directory to src/main/resources/output.



9. Look at the default polling information; the endpoint checks for incoming messages every 1000 milliseconds and sets a minimum of 500 milliseconds a file must wait before it is processed.

Display the file contents

10. Add a File to String transformer to the process section of the flow.
11. Add a Logger after the transformer.



12. In the Logger Properties view, set the message to display the payload.

Run the application

13. Save the file to redeploy the application.

14. Leave the application running.

Add a CSV file to the input directory

15. Right-click src/main/resources in the Package Explorer and select Show In > System Explorer.

16. In another window, navigate to the student files folder for the course:

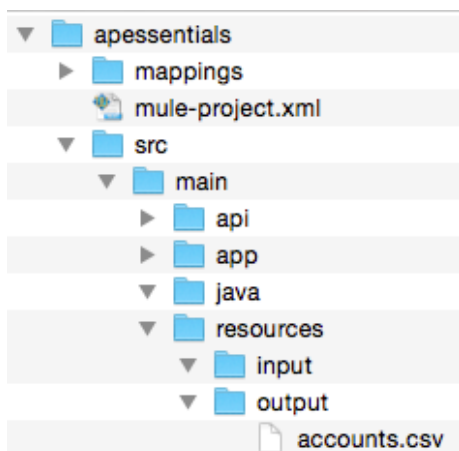
MUEssentials3.7_studentFiles_{date}.

17. Locate the resources/accounts.csv file.

18. Make a copy of this file and put it in the src/main/resources folder of the apessentials project.

Test the application

19. Drag the CSV into the input folder; you should see it almost immediately moved to the output folder.



20. Drag it back into the input folder; it will be read again and then moved to the output folder.

21. Leave this folder open.

Debug the application

22. Return to apessentials.xml.

23. Add a breakpoint to the Logger.

24. Debug the application.

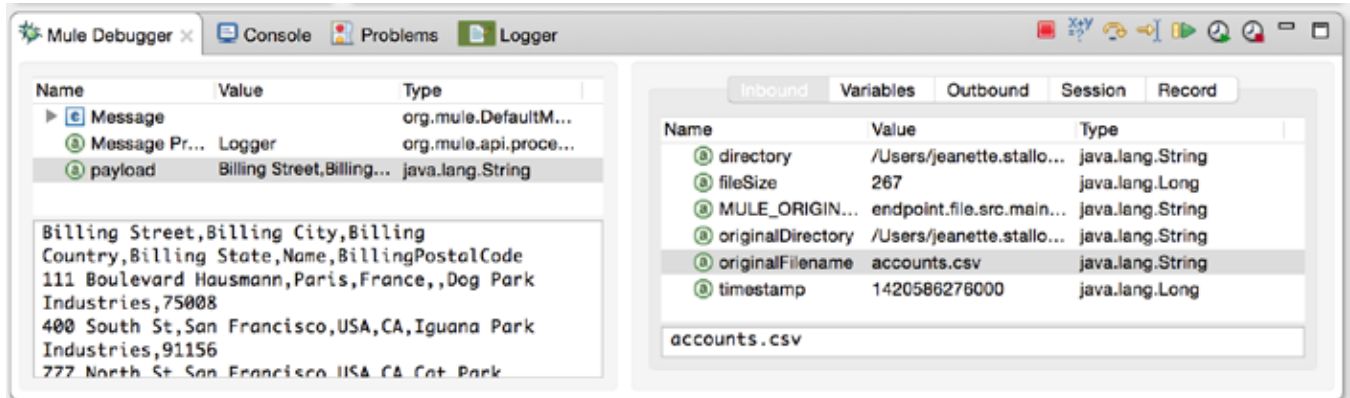
25. Move the accounts.csv file from the output folder back to the input folder.

Note: You can do this in your operating system window or in Anypoint Studio. If you use Anypoint Studio, you will need to right-click on the project and select Refresh to see the file.

26. Wait until code execution stops at the Logger.

27. Drill-down and look at the payload.

28. Look at the inbound message properties and locate the original filename.



29. Click the Resume button in the Mule Debugger view.

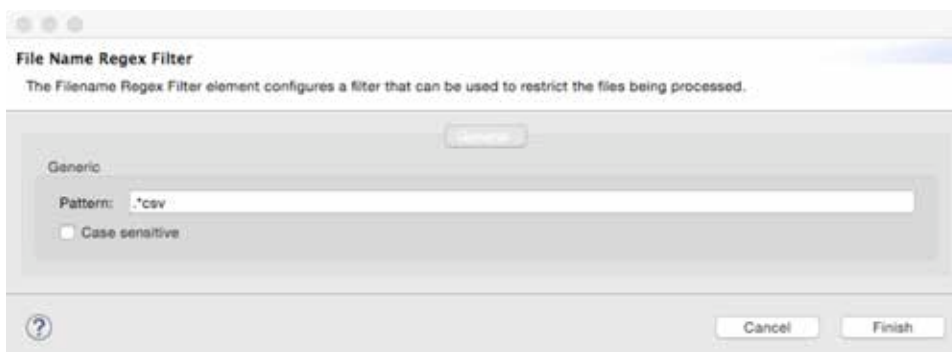
30. Stop the debugger.

Restrict the type of file read

31. Open the Properties view for the File endpoint.

32. Click the Add button next to file name regex filter.

33. In the File Name Regex Filter dialog box, set the pattern to *.csv and uncheck case sensitive.

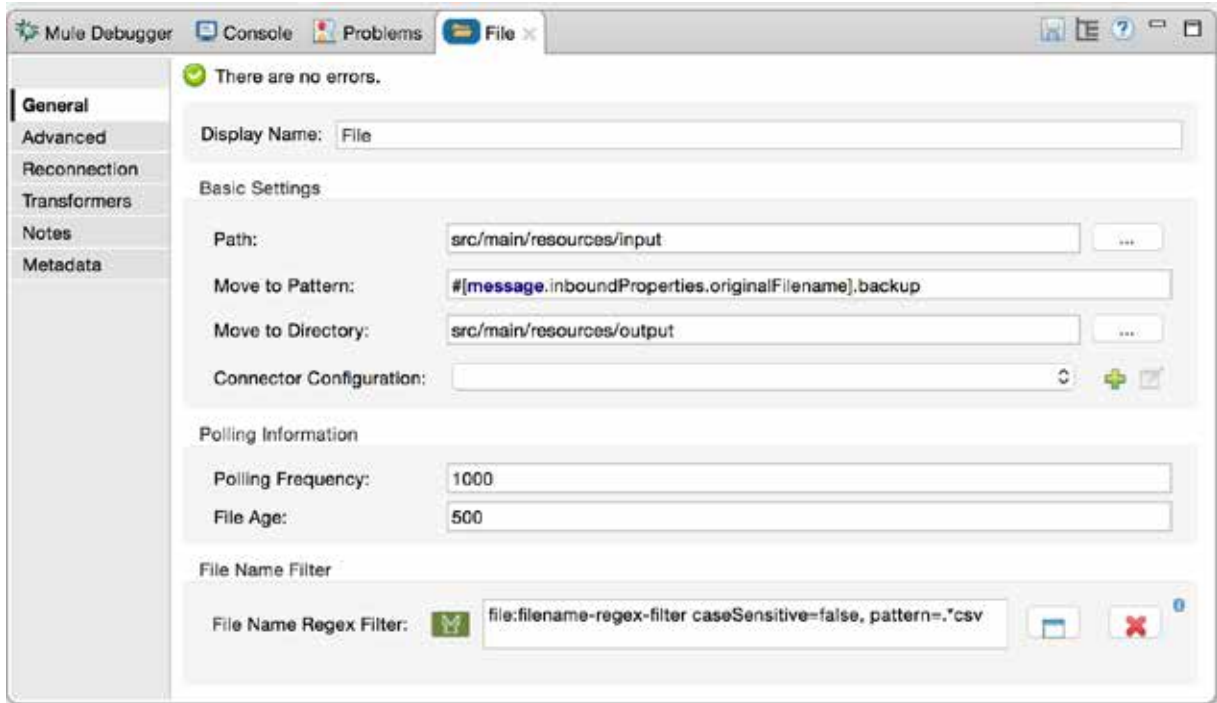


34. Click Finish.

Rename the file

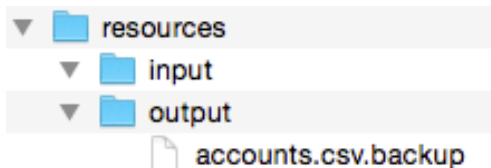
35. Set the move to pattern to append .backup to the original filename.

```
#[message.inboundProperties.originalFilename].backup
```



Test the application

36. Save the file and run the application.
37. Move the accounts.csv file from the output folder back to the input folder; you should see it appear in the output folder with its new name.



38. Look at the console; you should see the contents of the file displayed.

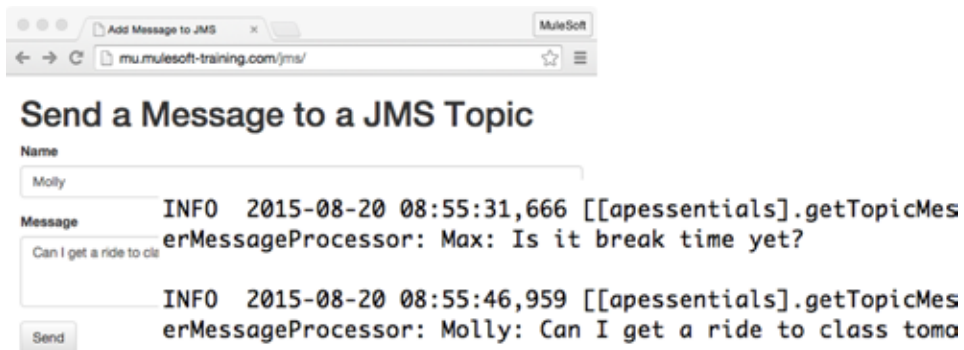
```
INFO 2015-08-19 13:44:16,688 [[apessentials].getCSVAccountsFlow.stage1.02] org.mule.api.processor.Logger
MessageProcessor: Billing Street,Billing City,Billing Country,Billing State,Name,BillingPostalCode
111 Boulevard Hausmann,Paris,France,,Dog Park Industries,75008
400 South St,San Francisco,USA,CA,Iguana Park Industries,91156
777 North St,San Francisco,USA,CA,Cat Park Industries,91156
```

39. Move the accounts.csv.backup file from the output folder back to the input folder; it should not be processed and moved to the output folder.

Walkthrough 4-3: Connect to a JMS queue (ActiveMQ)

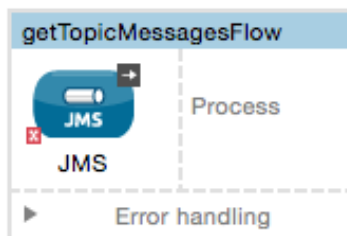
In this walkthrough, you will read and write messages from a JMS topic. You will:

- Create a flow accessible at <http://localhost:8081/jms>.
- Add and configure an ActiveMQ connector.
- Use a JMS endpoint to retrieve messages from a JMS topic.
- Add messages to the topic using a web form.
- Use a JMS endpoint to send messages to a JMS topic.



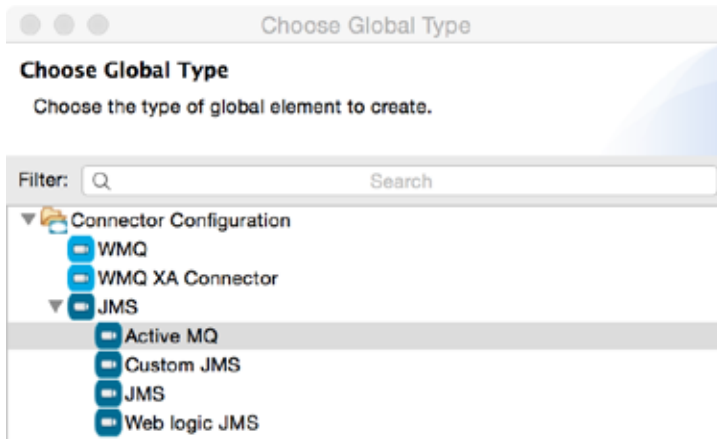
Create a JMS inbound-endpoint

1. Return to apessentials.xml.
2. Drag out a JMS connector and drop it at the bottom of the canvas to create a new flow.
3. Give the flow a new name of getTopicMessagesFlow.



4. In the JMS Properties view, select topic and set it to apessentials.
5. Click the Add button next to connector configuration.

6. In the Choose Global Type dialog box, select Connector Configuration > JMS > Active MQ and click OK.



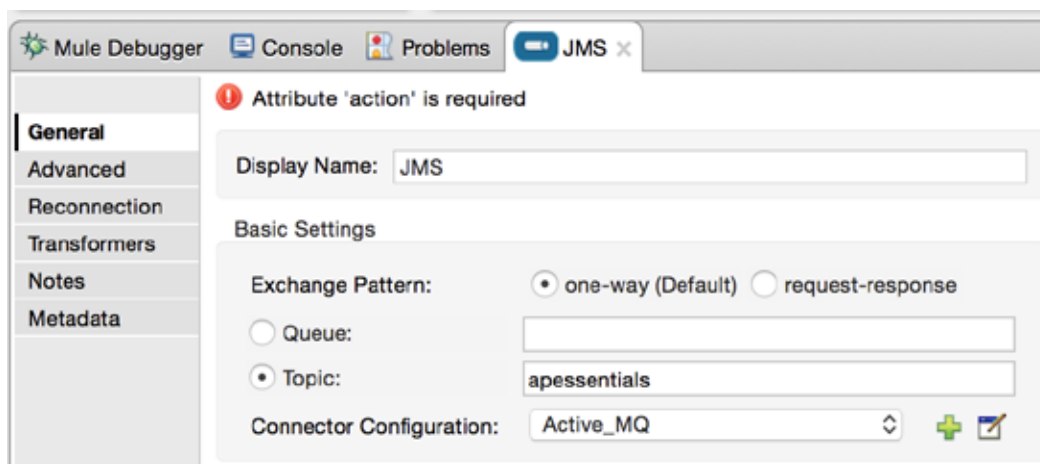
7. In the Global Element Properties dialog box, change the broker URL to the JMS ActiveMQ Broker URL listed in the course snippets.txt file.
8. Set the Specification to v1.1.



9. Click the Advanced tab and take a look at the various settings.
10. Click OK.
11. Click the Apply Changes button in the Properties view.

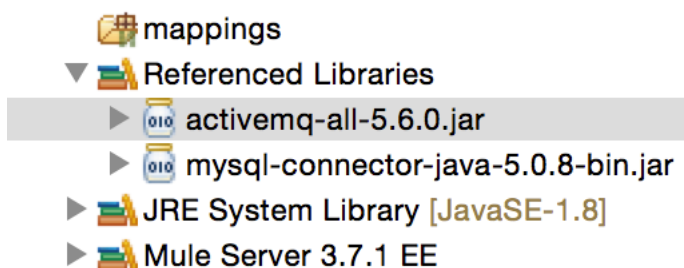
12. If you see an Attribute 'action' is required warning, ignore it.

Note: This is a bug in the Anypoint Studio 5.2.1.



Add the ActiveMQ library

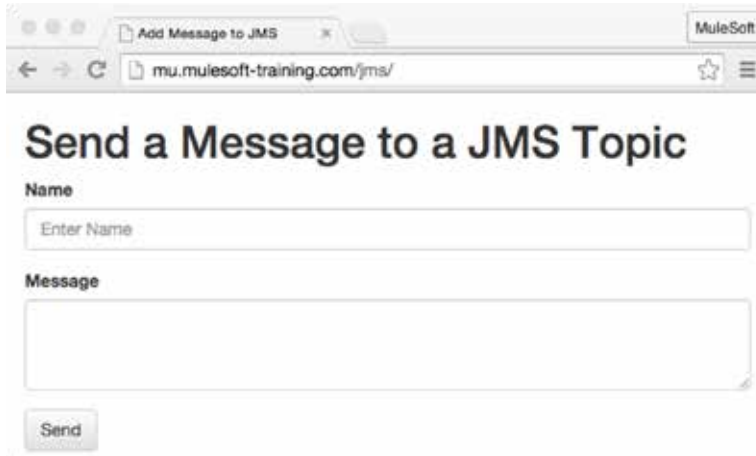
13. In the Package Explorer, right-click apessentials and select New > Folder.
14. In the New Folder dialog box, set the folder name to lib and click Finish.
15. Locate activemq-all-{version}.jar file located in the MUEssentials3.7_studentFiles_{date}/jars folder.
16. Copy and paste or drag the JAR file into your lib folder.
17. Right-click the JAR file in the Package Explorer and select Build Path > Add to Build Path; you should now see it under Referenced Libraries.



Note: Adding activemq-all.jar can create conflicts with other dependencies in projects, so it is recommended that you only add only the jar files you need in relation to what you need ActiveMQ for. For more details, see the documentation at <http://www.mulesoft.org/documentation/display/current/ActiveMQ+Integration>.

Test the application and receive messages

18. Add a Logger to the process section of the flow and set its message to #[payload].
19. Save the file and run the application.
20. Make a request to the JMS form URL listed in the course snippets file.
21. In the form, enter your name and a message and click Send.



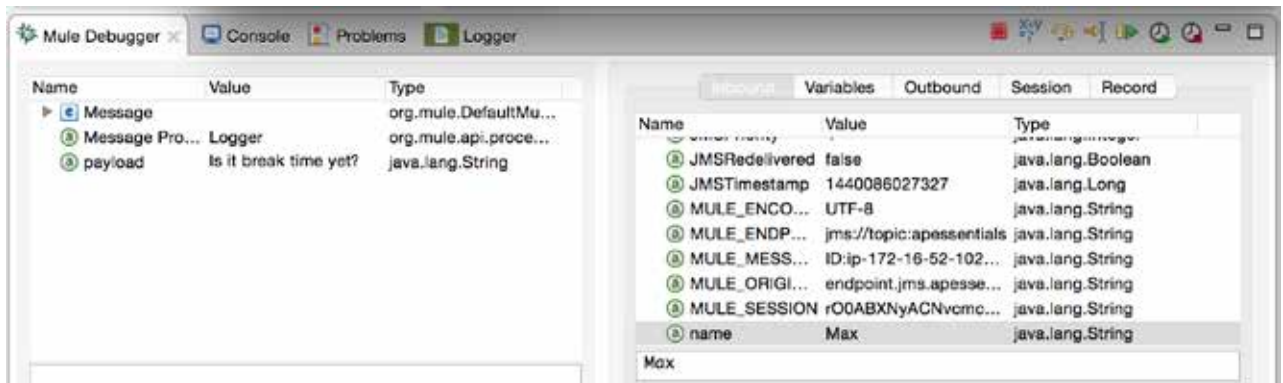
22. Return to the console in Anypoint Studio; you should see your message along with those from your classmates – but you don't see the names of the people who sent the messages.

```
INFO 2015-08-20 08:52:41,053 [[apessentials].getTopicMessagesFlow.stage1.02] org.mule.api.processor.LoggerMessageProcessor: Is it break time yet?
```

```
INFO 2015-08-20 08:52:57,613 [[apessentials].getTopicMessagesFlow.stage1.02] org.mule.api.processor.LoggerMessageProcessor: Can I get a ride to class tomorrow?
```

Debug the application

23. Add a breakpoint to the Logger.
24. Debug the application.
25. Make another request to the JMS form URL and submit another message.
26. In the Mule Debugger view, look at the payload and inbound message properties.



Name	Value	Type
JMSRedelivered	false	java.lang.Boolean
JMSTimestamp	1440086027327	java.lang.Long
MULE_ENCO...	UTF-8	java.lang.String
MULE_ENDP...	jms://topic:apessentials	java.lang.String
MULE_MESS...	ID:ip-172-16-52-102...	java.lang.String
MULE_ORIGI...	endpoint:jms.apesse...	java.lang.String
MULE_SESSION	r00ABXNyACNvcmc...	java.lang.String
name	Max	java.lang.String

27. Stop the Mule Debugger.

Display names with the messages

28. In the Logger Properties view, change the message to use an expression to display the name and message.

```
#[message.inboundProperties.name + ": " + payload]
```

Test the application

29. Save the file and run the application.

30. Return to the message form and submit a new message.

31. Look at the console; you should now see names along with messages.

```
INFO 2015-08-20 08:55:31,666 [[apessentials].getTopicMessagesFlow.stage1.02] org.mule.api.processor.LoggerMessageProcessor: Max: Is it break time yet?  
  
INFO 2015-08-20 08:55:46,959 [[apessentials].getTopicMessagesFlow.stage1.02] org.mule.api.processor.LoggerMessageProcessor: Molly: Can I get a ride to class tomorrow?
```

Create a JMS outbound-endpoint

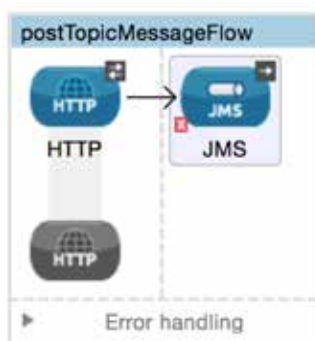
32. Drag out another HTTP connector and drop it in the canvas to create a new flow.

33. Give the flow a new name of postTopicMessageFlow.

34. In the HTTP Properties view, set the connector configuration to the existing HTTP_Listener_Configuration.

35. Set the path to /jms and the allowed methods to GET.

36. Drag out another JMS connector and drop it into the process section of the flow.



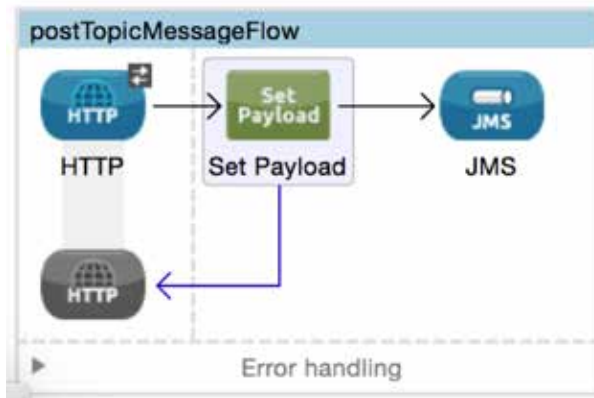
37. In the JMS Properties view, select topic and set it to apessentials.

38. Set the connector configuration to the existing Active_MQ.

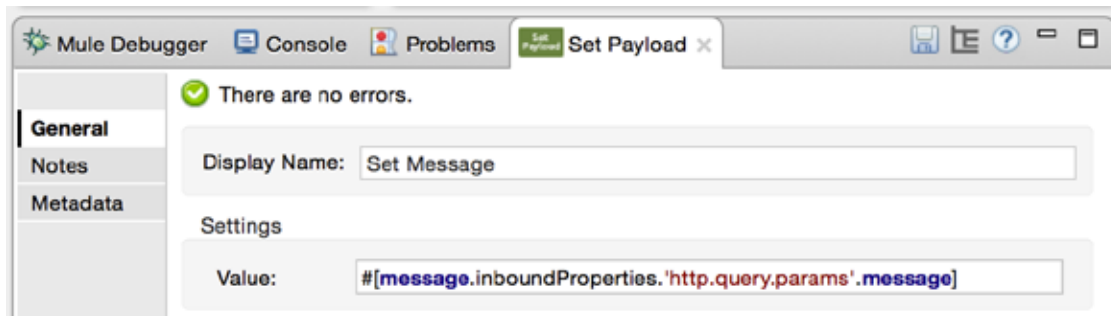
39. If you see an Attribute 'action' is required warning, ignore it.

Set a message

40. Add a Set Payload transformer between the HTTP and JMS connector endpoints.



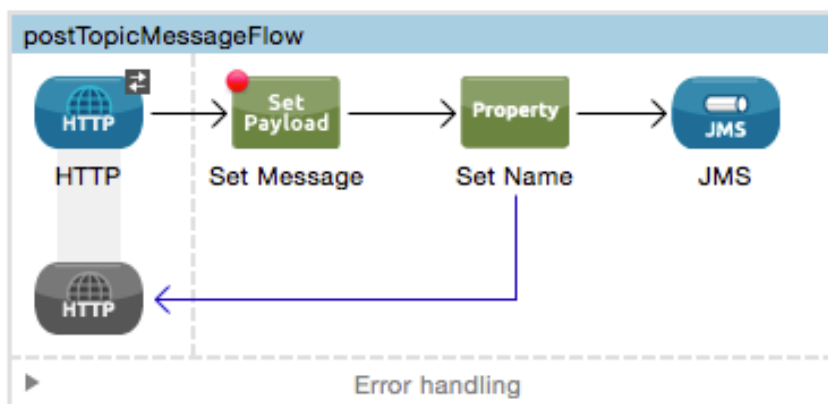
41. In the Set Payload Properties view, change the display name to Set Message and set the value to a message query parameter.



42. Add a breakpoint to the Set Payload transformer.

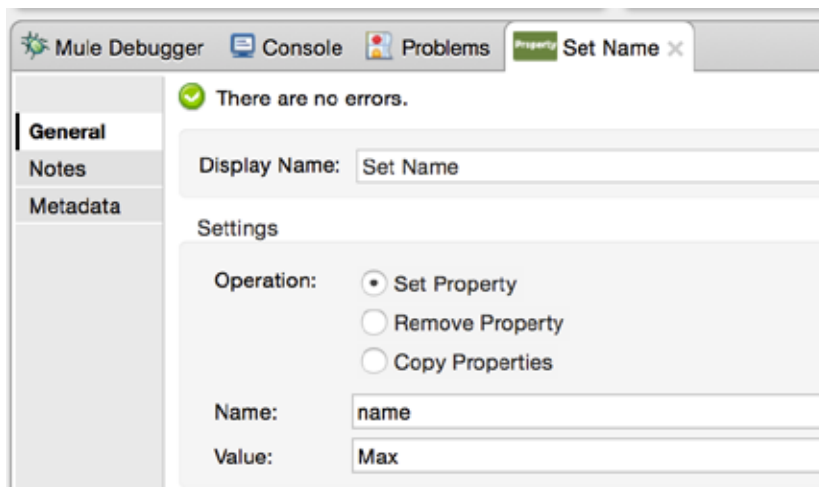
43. Add a Property transformer after the Set Payload transformer.

44. In the Properties view, change the display name to Set Name.



45. Select Set Property and set the name to name and the value to your name.

Note: You can set this to a query parameter instead if you prefer.



Test the application and post messages

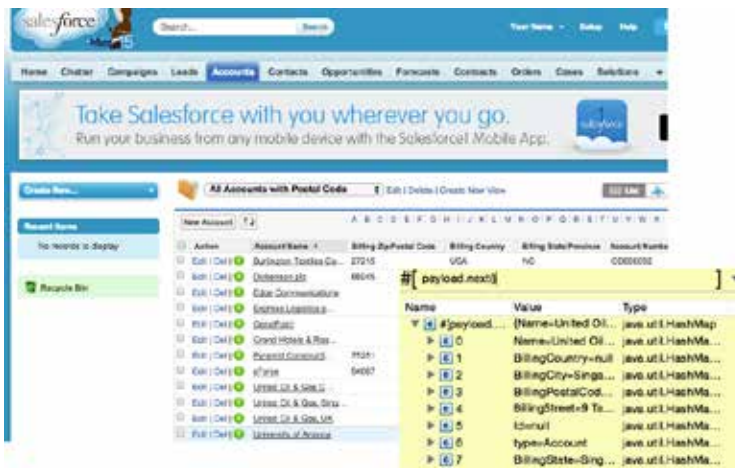
46. Save the file to redeploy the application and make a request to <http://localhost:8081/jms?message=Hello>.
47. Look at the console; you should see your name and message displayed – along with those of your classmates.

```
INFO 2015-08-20 09:01:28,411 [[apessentials].getTopicMessagesFlow.stage1.02] org.mule.api.processor.LoggerMessageProcessor: Max: Hello
```

Walkthrough 4-4: Connect to a SaaS application (Salesforce)

In this walkthrough, you will retrieve account records from Salesforce. You will:

- Browse Salesforce data on <http://salesforce.com>.
- Create a flow accessible at <http://localhost:8081/sfdc>.
- Add a Salesforce connector endpoint to retrieve accounts for a specific postal code.
- Use the Query Builder to write a query.
- Use the Object to JSON transformer to return the results as a string.



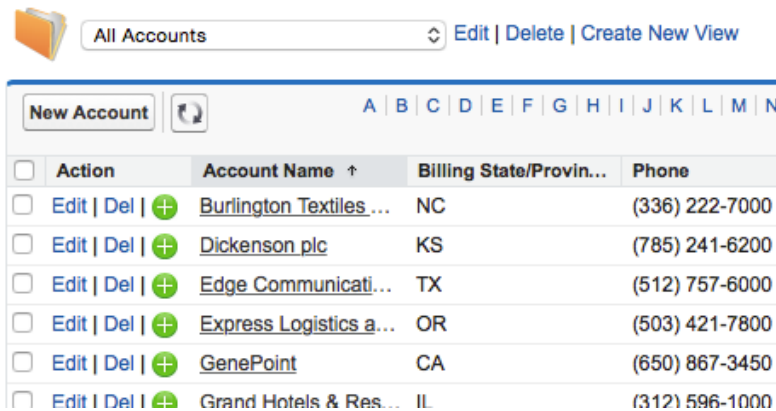
Note: To complete this walkthrough, you need a Salesforce Developer account. Instructions for creating a Salesforce developer account and getting an API access token are included in the Setup instructions at the beginning of this student manual.

Look at existing Salesforce account data

1. In a browser, go to <http://login.salesforce.com/> and log in with your Salesforce Developer account.
2. Click the Accounts link in the main menu bar.
3. In the view drop-down, select All Accounts and click the Go button.

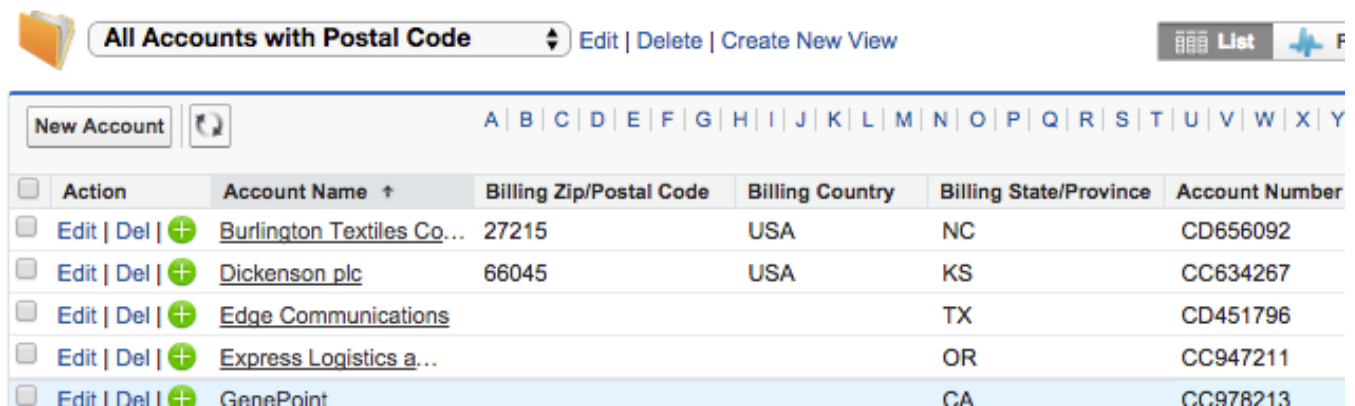


- Look at the existing account data; a Salesforce Developer account is populated with some sample data.



Action	Account Name	Billing State/Province	Phone
Edit Del +	Burlington Textiles ...	NC	(336) 222-7000
Edit Del +	Dickenson plc	KS	(785) 241-6200
Edit Del +	Edge Communicati...	TX	(512) 757-6000
Edit Del +	Express Logistics a...	OR	(503) 421-7800
Edit Del +	GenePoint	CA	(650) 867-3450
Edit Del +	Grand Hotels & Res...	IL	(312) 596-1000

- Notice that countries and postal codes are not displayed by default.
- Click the Create New View link next to the drop-down displaying All Accounts.
- Set the view name to All Accounts with Postal Code.
- Locate the Select Fields to Display section.
- Select Billing Zip/Postal Code as the available field and click the Add button.
- Add the Billing Country and Account Number fields.
- Use the Up and Down buttons to order the fields as you prefer.
- Click the Save button; you should now see all the accounts with postal codes and countries.



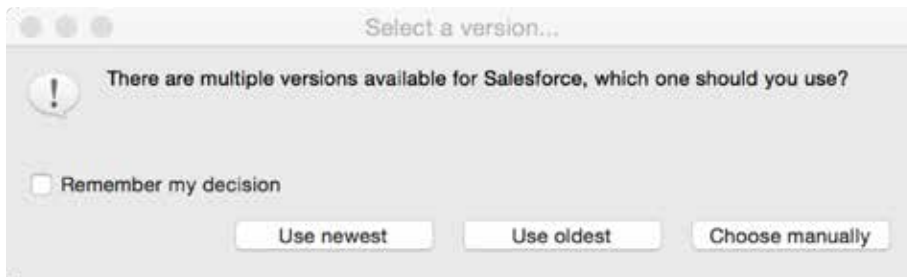
Action	Account Name	Billing Zip/Postal Code	Billing Country	Billing State/Province	Account Number
Edit Del +	Burlington Textiles Co...	27215	USA	NC	CD656092
Edit Del +	Dickenson plc	66045	USA	KS	CC634267
Edit Del +	Edge Communications			TX	CD451796
Edit Del +	Express Logistics a...			OR	CC947211
Edit Del +	GenePoint			CA	CC978213

Create a new project and flow

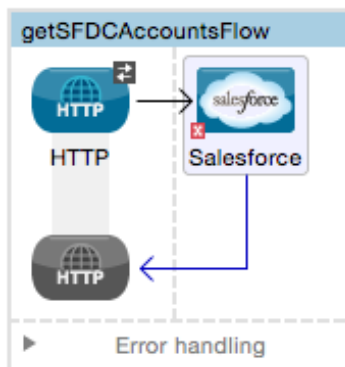
- Return to apessentials.xml in Anypoint Studio.
- Drag out an HTTP connector and drop it above the getCSVAccountsFlow to create a new flow.
- Give the flow a new name of getSFDCAccountsFlow.
- In the HTTP Properties view, set the connector configuration to the existing HTTP_Listener_Configuration.
- Set the path to /sfdc and the allowed methods to GET.

Add a Salesforce connector endpoint

18. Drag out a Salesforce connector and drop it in the process section of the flow.
19. In the Select a version dialog box, click Use newest.

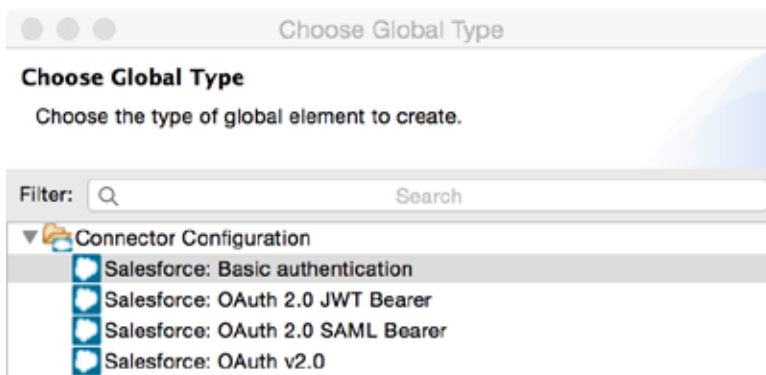


20. Examine the flow.



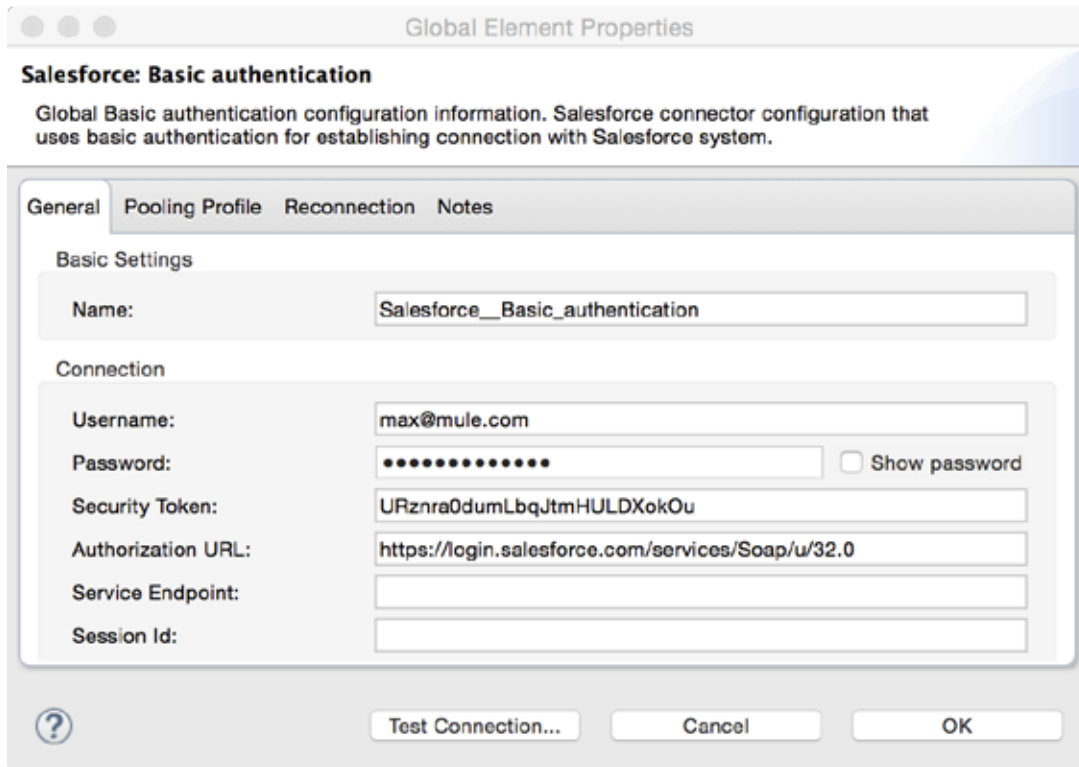
Configure the Salesforce connector

21. In the Salesforce Properties view, click the Add button next to Connector Configuration.
22. In the Choose Global Type dialog box, select Connector Configuration > Salesforce: Basic authentication and click OK.



23. In the Global Element Properties dialog box, enter your email username, password, and security token.

Note: Instructions for creating a Salesforce Developer account and getting a security token are included in the Setup instructions at the beginning of this student manual.



The screenshot shows the 'Global Element Properties' dialog box with the 'Salesforce: Basic authentication' tab selected. The 'General' sub-tab is active, displaying 'Basic Settings' and 'Connection' sections. The 'Name' field is set to 'Salesforce__Basic_authentication'. The 'Connection' section includes fields for 'Username' (max@mule.com), 'Password' (masked with dots), 'Security Token' (URznra0dumLbqJtmHULDxOkOu), 'Authorization URL' (https://login.salesforce.com/services/Soap/u/32.0), 'Service Endpoint', and 'Session Id'. A 'Show password' checkbox is next to the password field. At the bottom, there are buttons for 'Test Connection...', 'Cancel', and 'OK'.

24. Click the Test Connection button; you will get a Test Connection dialog box letting you know if the connection succeeds or fails.



25. Click OK to close the Test connection dialog box.
26. If your test connection failed, fix it; do not proceed until it is working.

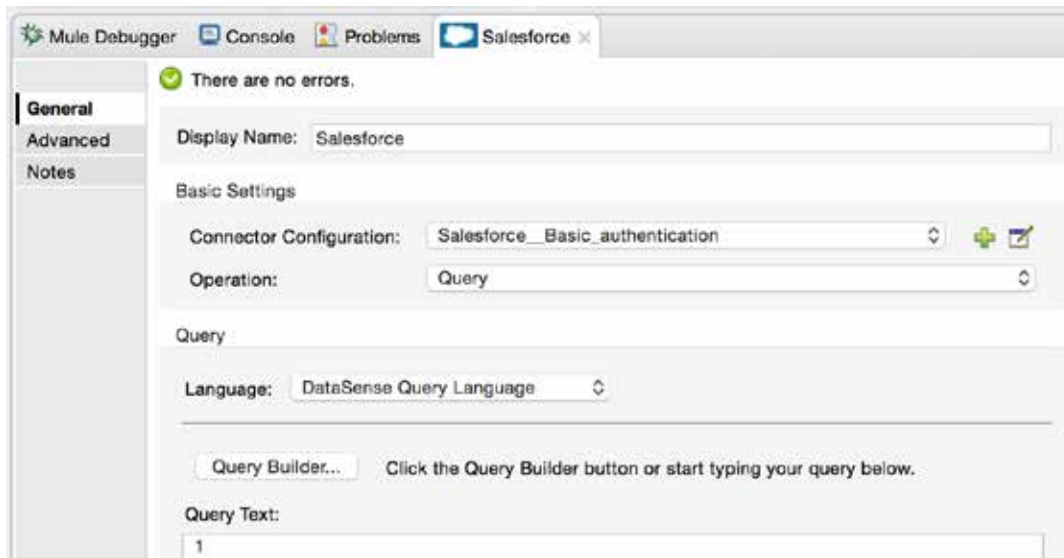
Note: If it failed, check to see if you have any extra whitespace in your entries.

27. Click OK to close the Global Elements Properties dialog box.

Write the query

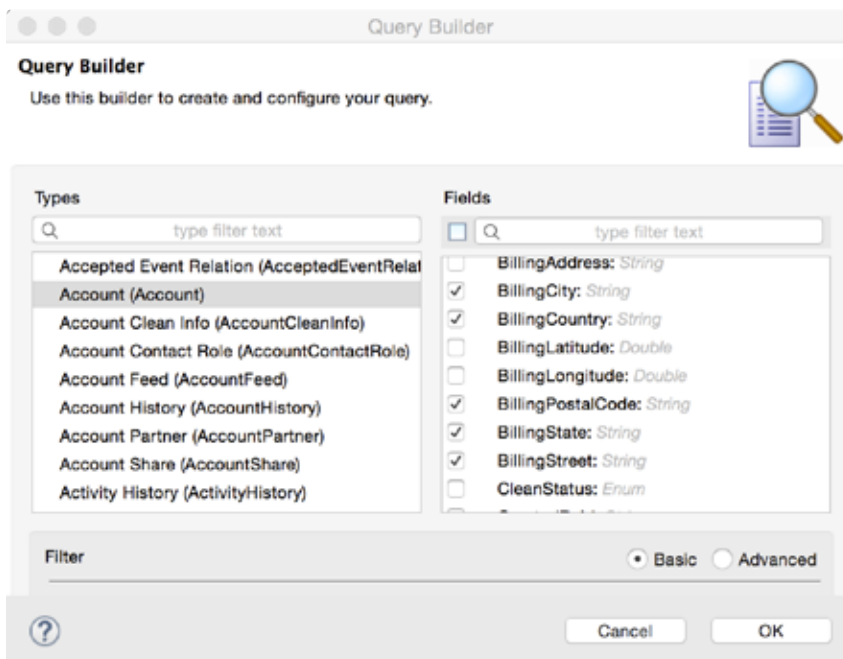
28. In the Salesforce Properties view, select an operation of Query.

29. Click the Query Builder button.



30. In the Query Builder dialog box, select a type of Account (Account).

31. Select fields BillingCity, BillingCountry, BillingPostalCode, BillingState, BillingStreet, and Name.



32. Click OK.

33. Examine the generated query.

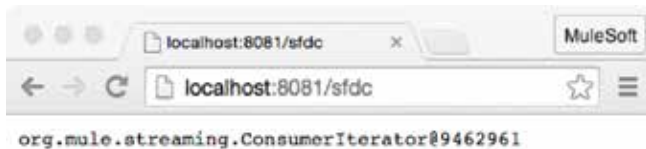


Test the application

34. Save the file to redeploy the application.

Note: If you get a SAXParseException, stop the Mule runtime and run the application again. If you still get an exception, close and reopen the project. If that does not work, restart Anypoint Studio.

35. Make a request to <http://localhost:8081/sfdc>; you should see the type of Java object created.

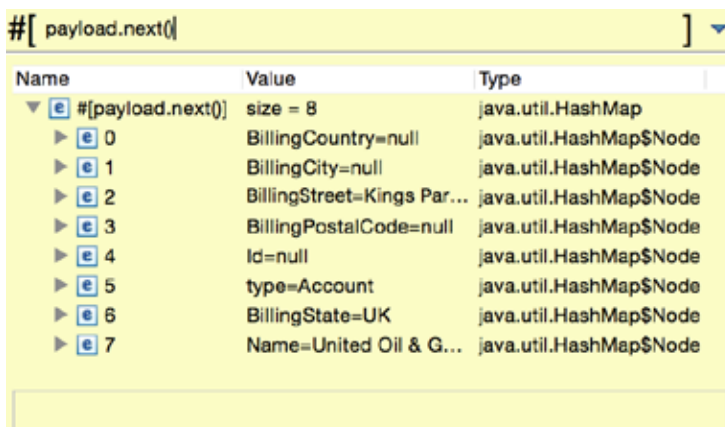


Debug the application

36. Add a Logger component after the Salesforce endpoint.
37. Set the Logger message to `#[payload]`.
38. Add a breakpoint to the Logger.
39. Save the file and debug the application.
40. Make another request to <http://localhost:8081/sfdc>.
41. Step to the Logger and drill-down into the payload variable.
42. Click the Evaluate Mule Expression button in the Mule Debugger view.



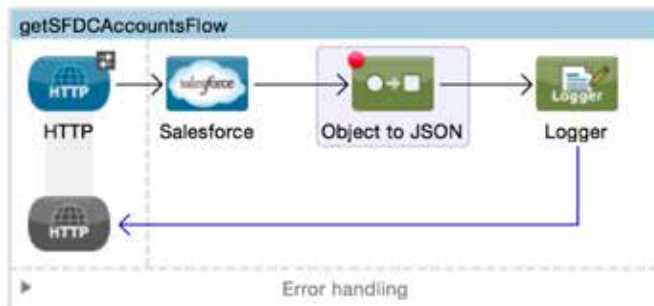
43. Enter the following expression and press the Enter key.
`#[payload.next()]`
44. Expand the results; you should see the account data for the first account.



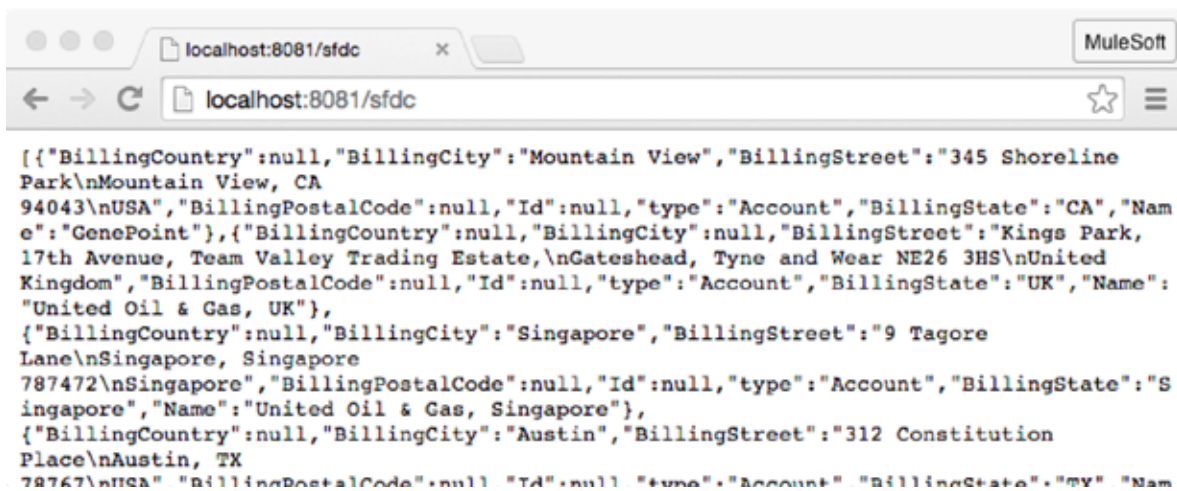
45. Click anywhere outside the expression window to close it.
46. Stop the Mule Debugger.

Display the return data

47. Add an Object to JSON transformer before the Logger component.



48. Save the file and run the application.
49. Make another request to <http://localhost:8081/sfdc>; you should see the Salesforce accounts displayed.



Modify the query

50. In the Salesforce Properties view, click the Query Builder button.
51. In the Query Builder dialog box, click the Add Filter button.

52. Create a filter for BillingPostalCode equal to one of the postal codes (like 27215) in your Salesforce account data.

Query Builder

Use this builder to create and configure your query.

Types

type filter text

- Account (Account)
- Account Clean Info (AccountCleanInfo)
- Account Contact Role (AccountContactRole)
- Account Feed (AccountFeed)
- Account History (AccountHistory)
- Account Partner (AccountPartner)
- Account Share (AccountShare)
- Activity History (ActivityHistory)
- Additional Directory Number (AdditionalNumber)

Fields

type filter text

- ☐ BillingAddress: String
- ☒ BillingCity: String
- ☒ BillingCountry: String
- ☐ BillingLatitude: Double
- ☐ BillingLongitude: Double
- ☒ BillingPostalCode: String
- ☒ BillingState: String
- ☒ BillingStreet: String
- ☐ CleanStatus: Enum

Filter

Basic Advanced

BillingPostalCode = 27215

Add Filter

Cancel OK

53. Click OK.
54. Examine the generated query.

Query Text:

```
1 SELECT BillingCity,BillingCountry,BillingPostalCode,BillingState,BillingStreet,Name
2 FROM Account
3 WHERE BillingPostalCode = '27215'
```

Test the application

55. Save the file to redeploy the application.
56. Make another request to <http://localhost:8081/sfdc>; you should only the accounts with the specified postal code displayed.

localhost:8081/sfdc

localhost:8081/sfdc

```
[{"BillingCountry":"USA","BillingCity":"Burlington","BillingStreet":"525 S. Lexington Ave","BillingPostalCode":"27215","Id":null,"type":"Account","BillingState":"NC","Name":"Burlington Textiles Corp of America"}]
```

Walkthrough 4-5: Find and install not-in-the-box connectors

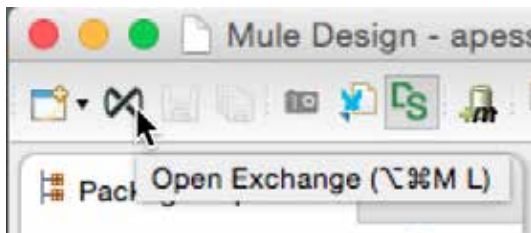
In this walkthrough, you will learn how to add a new connector to Anypoint Studio. You will:

- Browse the Anypoint Exchange.
- Add a new connector to Anypoint Studio.



Browse the Anypoint Exchange from Anypoint Studio

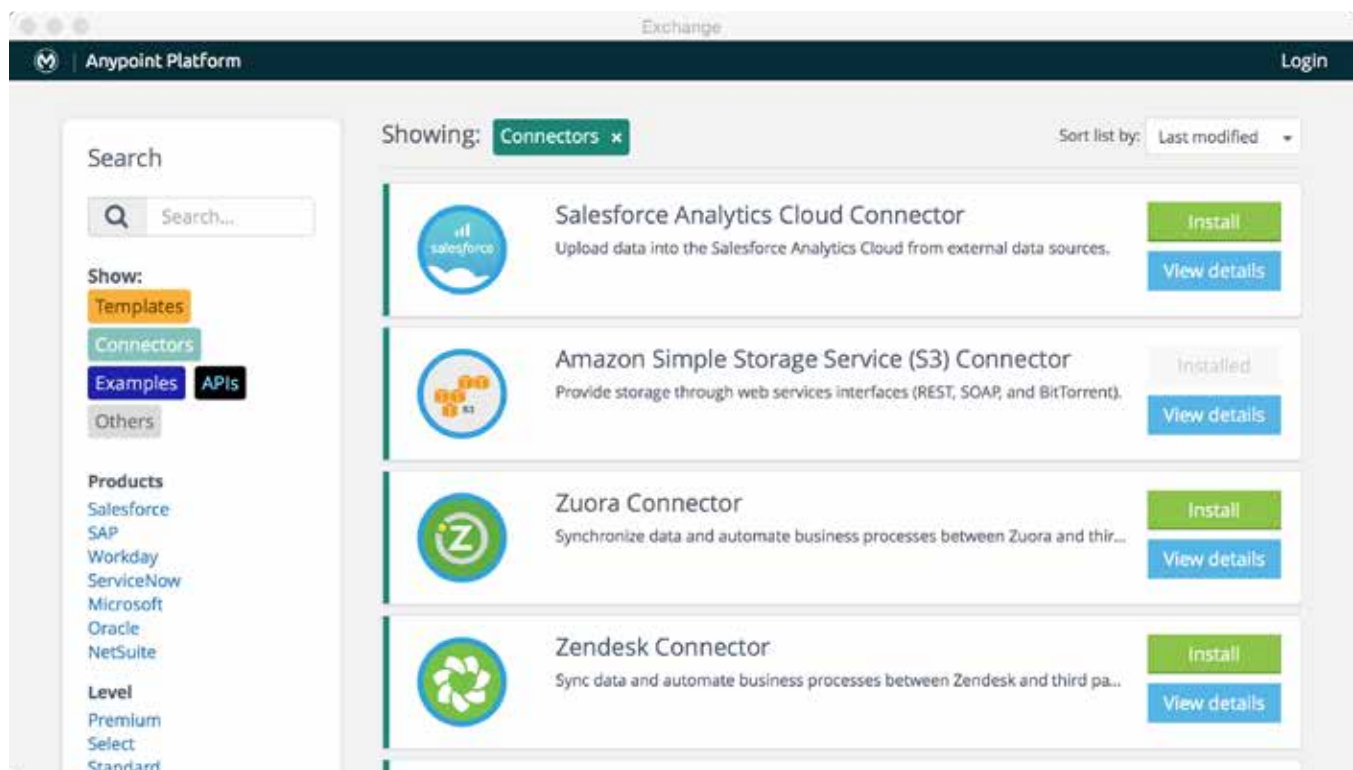
1. In Anypoint Studio, click the Open Exchange button; the Anypoint Exchange should open in a new window.



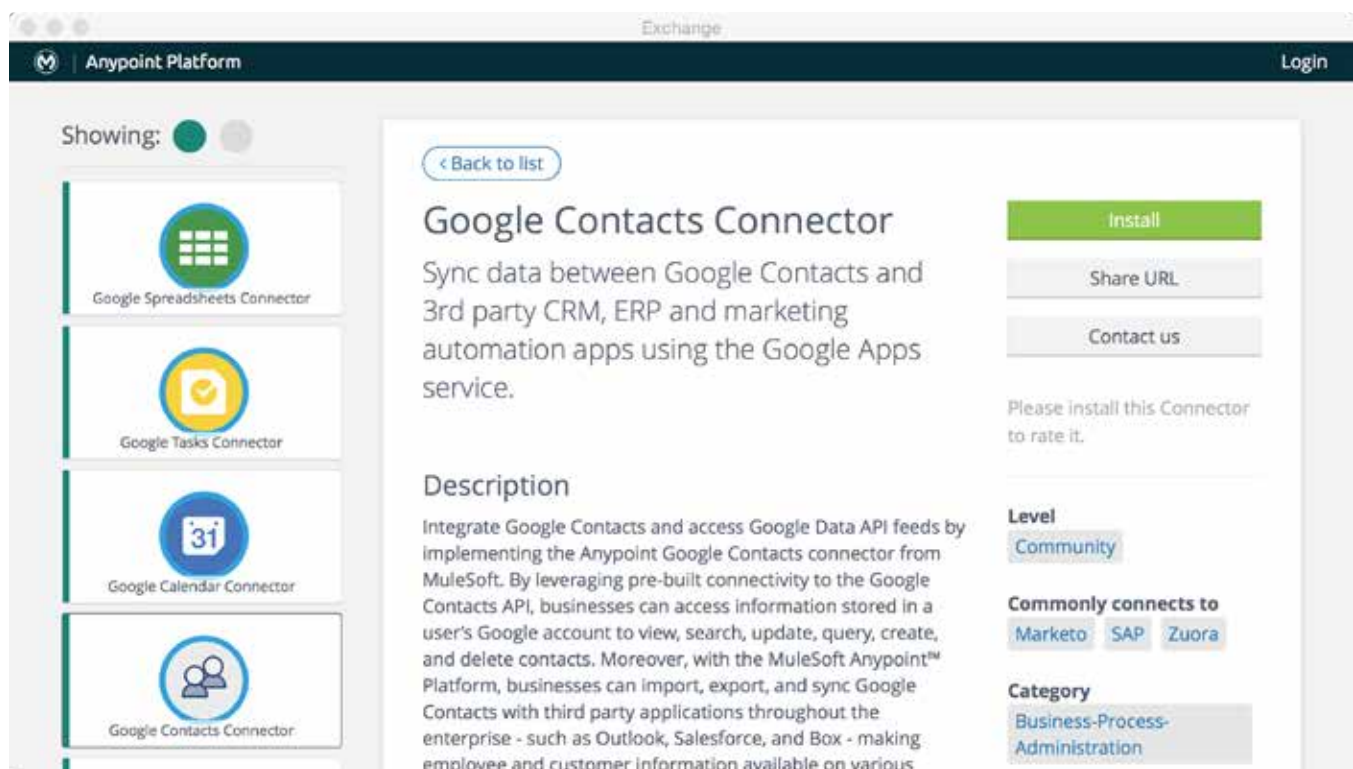
2. Click the Connectors button.



3. Browse the connectors, exploring the different levels and categories.

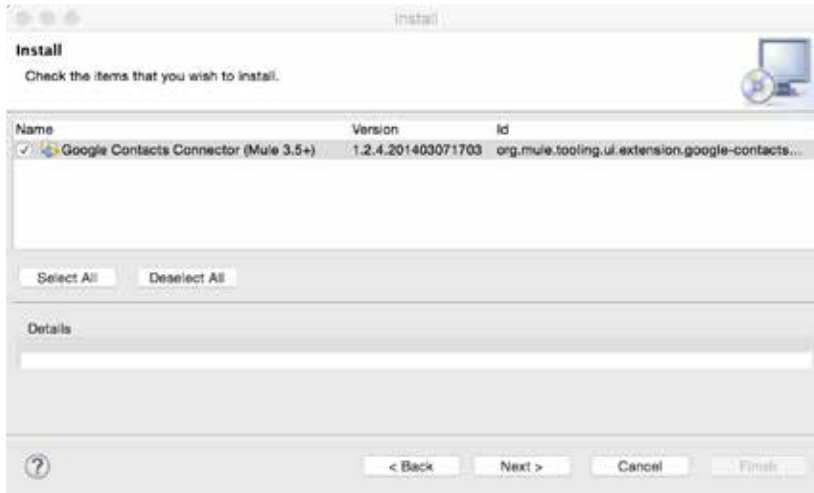


4. Locate the Google Contacts connector (or any other connector you are interested in).
5. Click the View details button and browse the connector's documentation.



Install the connector

6. In the Anypoint Exchange, click the connector's Install button.
7. If you get a Terms of Service dialog box, enter your personal information and click Install.
8. Wait until an Install dialog box appears in Anypoint Studio.



Note: If you do not see the connector listed, resize the Install dialog box.

Note: You can also install connector's directly from Anypoint Studio. From the main menu bar, select Help > Install New Software. In the Install dialog box, click the Work with drop-down button and select Anypoint Connectors Update Site. Drill-down into Community and select the Google Contacts Connector (or some other connector).

9. Click Next.
10. On the Install Details page, click Next.
11. On the Review Licenses page, select the I accept the terms radio button.
12. Click Finish; the software will be installed and then you will get a message to restart Anypoint Studio.
13. In the Software Updates dialog box, click Yes to restart Anypoint Studio.
14. After Anypoint Studio restarts, locate the new connector in the Connectors section of the palette.

