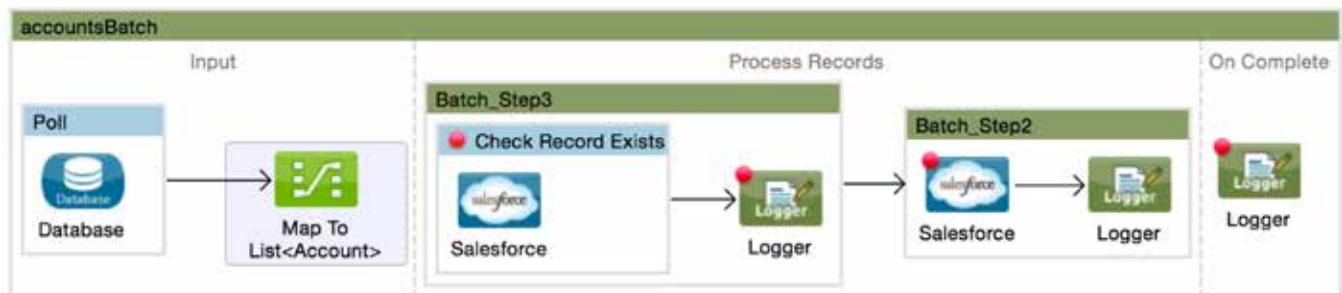


Module 9: Processing Records



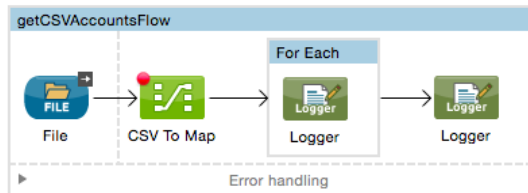
In this module, you will learn:

- To process items in a collection individually.
- To use the Batch Job element (EE).
 - To process individual records.
 - To store information at the record level.
 - To report results of a batch job.
 - To skip records.
 - To use batch jobs to process items in a CSV file.
 - To use batch jobs to process records from a database using a poll.

Walkthrough 9-1: Process items in a collection individually

In this walkthrough, you will split a collection and process each item in it individually. You will:

- Read a CSV file and use the DataMapper to convert it to a collection of objects.
- Use the DataMapper to create mappings using sample input data.
- Use the For Each scope element to process each item in a collection individually.



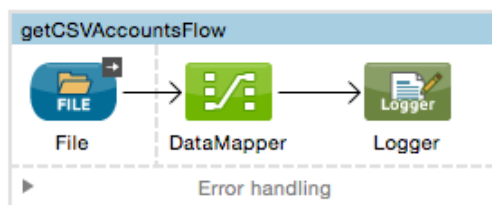
Modify the File endpoint to not rename the files

1. Open accounts.xml.
2. In the Package Explorer, expand the src/main/resources/input folder.
3. Right-click accounts.csv.backup and select Refactor > Rename.
4. In the Rename Resource dialog box, set the new name to accounts.csv and click OK.
5. In getCSVAccountsFlow, delete the File-to-String transformer.
6. In the File properties view, delete the move to pattern.

Note: This will make it easier to test the application because you won't have to keep renaming the file as you move it back to the input directory.

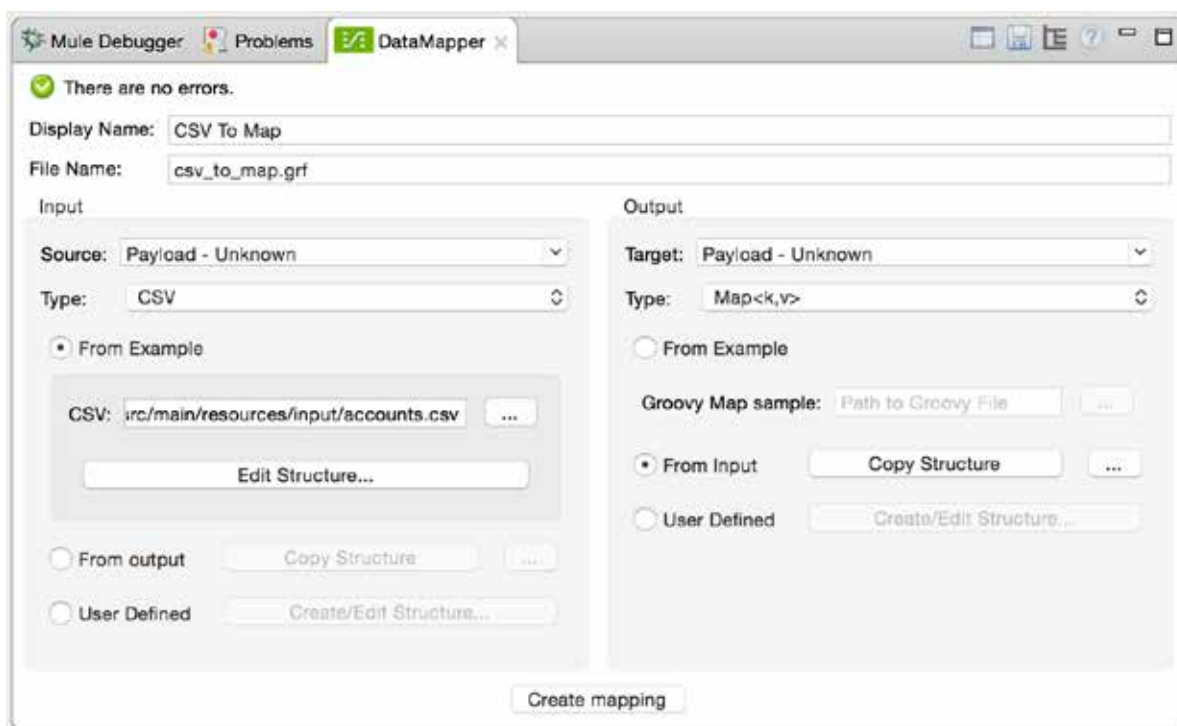
Convert a CSV file to a collection of objects

7. Add a DataMapper transformer between the File and the Logger.



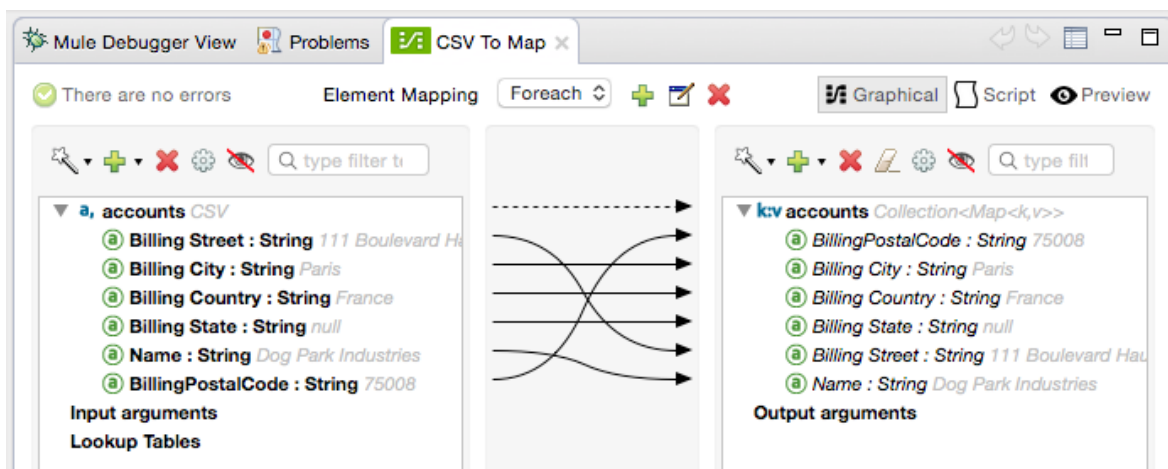
8. In the Input section of the DataMapper Properties view, set the type to CSV.
9. In the options that appear beneath it, make sure From Example is selected.
10. Click the Browse button next to CSV.
11. In the Open dialog box, browse to the src/main/resources/input/accounts.csv file and click Open.
12. In the Output section of the DataMapper Properties view, set the type to Map<k,v>.

13. Select From Input and click Copy Structure.



14. Click Create mapping.

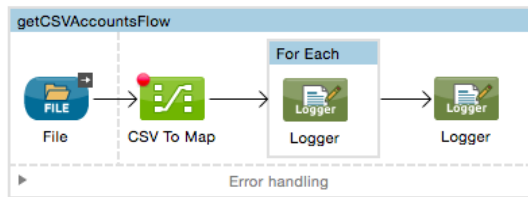
15. Look at the generated mapping.



Add a For Each scope element

16. Drag a For Each scope element form the palette and drop it after the DataMapper.

17. Add a Logger to the For Each scope.



Process each element

18. In the Logger Properties view, set the message to #[payload].

Debug the application

19. Add a breakpoint to the DataMapper.

20. Save the file and debug the application; application execution should stop at the DataMapper.

Note: There is a bug with the File connector in the Anypoint Studio 2015 release. If the file is not being read, try changing the path and move to directory values to absolute paths. If that does not work, try adding any transformer after the scope. If that does not work, try creating a connector configuration for the File endpoint.

21. In the Mule Debugger view, watch the payload value.

22. Step to the For Each scope; the payload should be an ArrayList.

23. Step through the For Each; the payload should be a LinkedHashMap.

The screenshot shows the Mule Studio interface. The top part displays the message flow diagram for 'getCSVAccountsFlow', which includes a File connector, a CSV To Map transformer, a For Each scope with a Logger, and another Logger connector. Below the diagram are tabs for 'Message Flow', 'Global Elements', and 'Configuration XML'. The bottom part of the screenshot shows the 'Mule Debugger View' with a table of message details.

Name	Value	Type
▶ [e] Message		org.mule.DefaultMuleMessage
▶ [a] Message Processor	Logger	org.mule.api.processor.Logg...
▶ [e] payload	{Billing Street=111 Boulevard...	java.util.LinkedHashMap
▶ [e] 0	Billing Street=111 Boulevard...	java.util.LinkedHashMap\$Entry
▶ [e] 1	Billing City=Paris	java.util.LinkedHashMap\$Entry
▶ [e] 2	Billing Country=France	java.util.LinkedHashMap\$Entry
▶ [e] 3	Name=Dog Park Industries	java.util.LinkedHashMap\$Entry
▶ [e] 4	BillingPostalCode=75008	java.util.LinkedHashMap\$Entry

24. Watch the console as you step through; you should see each record displayed.

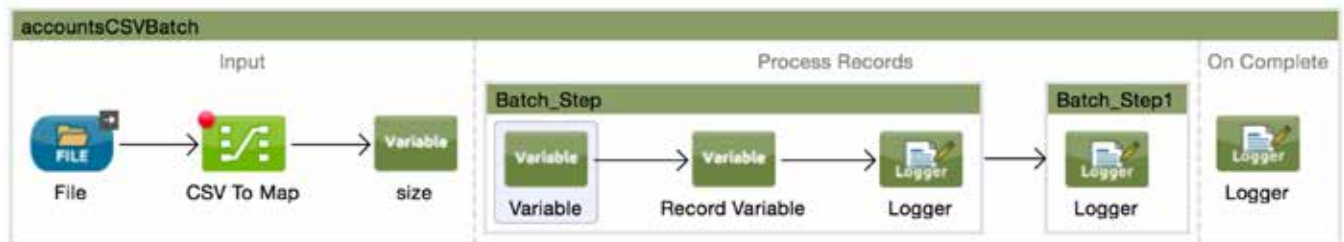
```
INFO 2015-08-31 11:06:34,737 [[apessentials].getCSVAccountsFlow.stage1.02] org.mule.api.processor.Logger
MessageProcessor: {Billing Street=111 Boulevard Hausmann, Billing City=Paris, Billing Country=France, Name=Dog Park Industries, BillingPostalCode=75008}
INFO 2015-08-31 11:06:35,317 [[apessentials].getCSVAccountsFlow.stage1.02] org.mule.api.processor.Logger
MessageProcessor: {Billing Street=400 South St, Billing City=San Francisco, Billing Country=USA, Billing State=CA, Name=Iguana Park Industries, BillingPostalCode=91156}
INFO 2015-08-31 11:06:35,899 [[apessentials].getCSVAccountsFlow.stage1.02] org.mule.api.processor.Logger
MessageProcessor: {Billing Street=777 North St, Billing City=San Francisco, Billing Country=USA, Billing State=CA, Name=Cat Park Industries, BillingPostalCode=91156}
INFO 2015-08-31 11:06:36,651 [[apessentials].getCSVAccountsFlow.stage1.02] org.mule.api.processor.Logger
```

25. Step through to the Logger after the For Each; the payload should be an ArrayList again.

Walkthrough 9-2: Create a batch job for records in a file

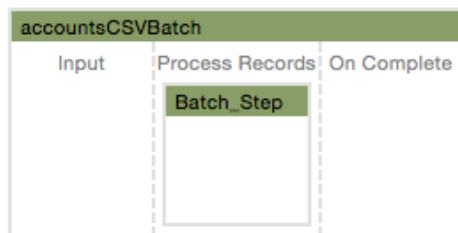
In this walkthrough, you will create a batch job to process records in a CSV file. You will:

- Create a new flow containing a batch job.
- Explore flow and record variable persistence across batch steps and phases.
- In the input phase, check for CSV files every second and convert them to a collection of objects.
- In the process records phase, create two batch steps for setting and tracking variables.
- In the on complete phase, display the number of records processed and failed.



Create a batch job

1. Return to accounts.xml.
2. Drag a Batch scope element from the palette and drop it above the getCSVAccountsFlow.
3. Change the batch job name to accountsCSVBatch.

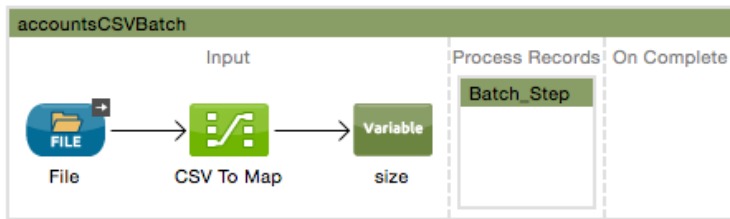


Note: Remember, you can click an item in the canvas once and then click it again to edit it in the canvas.

Add elements to the input phase

4. Select the File and DataMapper elements in getCSVAccountsFlow and select Edit > Copy or press Cmd+C/Ctrl+C.
5. Click in the accountsCSVBatch input phase and select Edit > Paste or press Cmd+V/Ctrl+V.

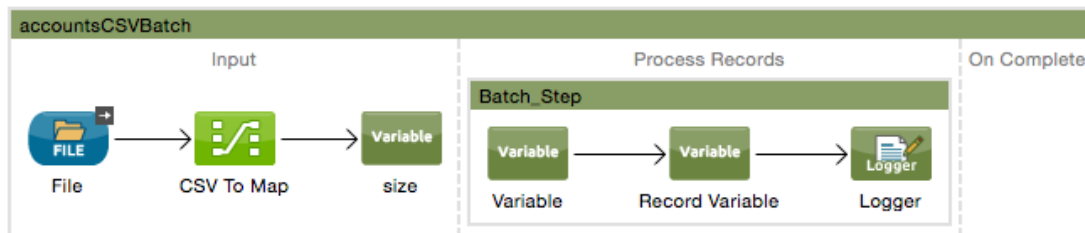
6. Rename the elements to remove Copy_of_ from their names.
7. Add a Variable transformer after the DataMapper in the input phase.
8. In the Variable Properties view, change the display name to size and select Set Variable.



9. Set the name to size and the value to `#[payload.size()]`.

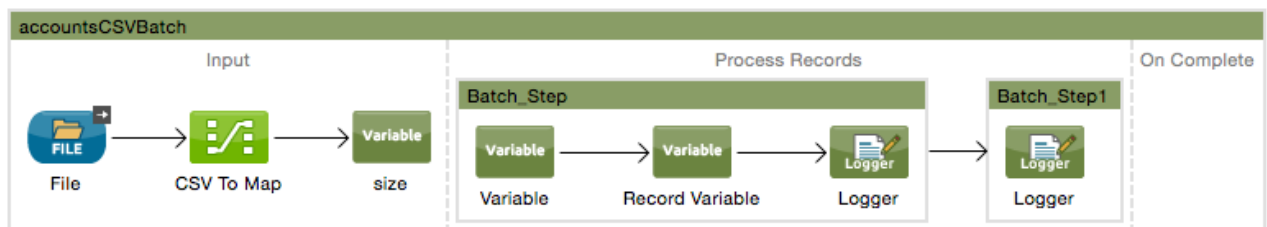
Add processors to a batch step in the process records phase

10. Add a Variable transformer to the batch step in the process records phase.
11. In the Properties view, select Set Variable and set the name to frame and the value to `#[payload.Name]`.
12. Add a Record Variable transformer to the batch step.
13. In the Properties view, select Set Record Variable and set the name to rname and the value to `#[payload.Name]`.
14. Add a Logger component to the batch step.



Create a second batch step

15. Drag a Batch Step scope element from the palette and drop it in the process records phase after the first batch step.
16. Add a Logger to the second batch step.



Add processors to the on complete phase

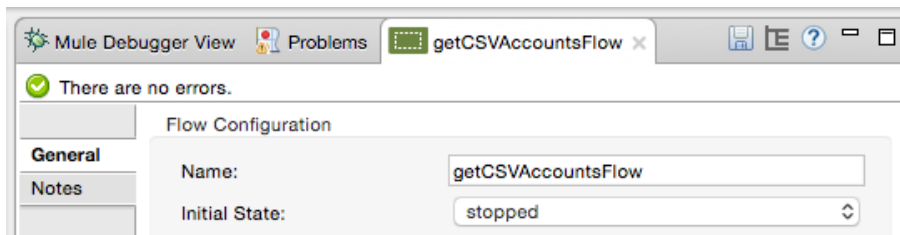
17. Add a Logger component to the on complete phase.
18. In the Logger Properties view, set the message to display the number of processed records and failed records.

```
#['\n\nProcessed: ' + payload.processedRecords + ' Failed: ' +  
payload.failedRecords]
```

Note: If you want to add line breaks, add one or more '\n' to your MEL expression – meaning it must be inside the #[].

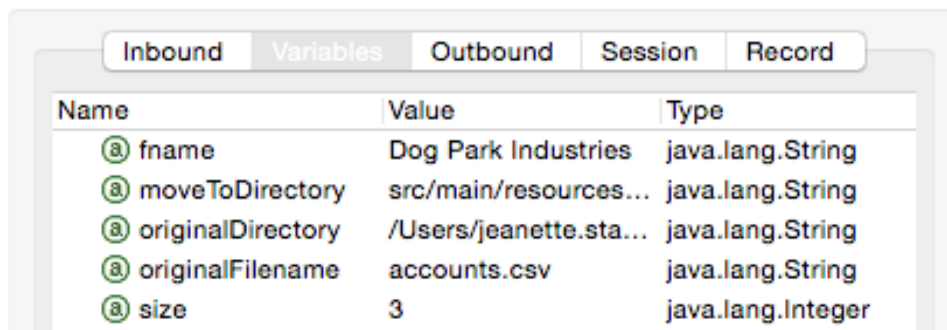
Stop getCSVAccountsFlow from running

19. Double-click getCSVAccountsFlow.
20. In the Properties view, set the initial state to stopped.

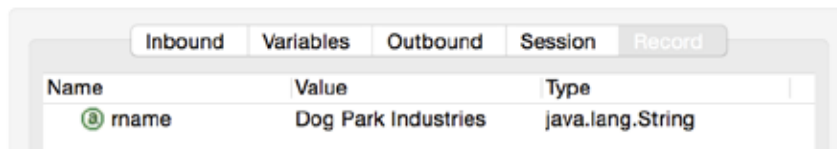


Debug the application

21. Add a breakpoint to the DataMapper in accountCSVBatch.
22. Save the file to redeploy the application in debug mode.
23. After the application starts, drag the accounts.csv file from the output folder to the input folder.
24. In the Mule Debugger view, watch the payload value.
25. Step into the process records phase.
26. Click the Variables tab; you should see the size flow variable.
27. Step to the Logger in the first batch step; you should see the fname flow variable.



28. Click the Record tab; you should see the rname flow variable.



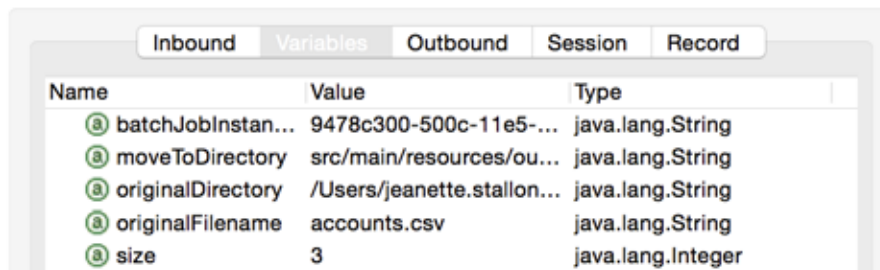
Name	Value	Type
③ rname	Dog Park Industries	java.lang.String

Note: If you do not see the record variable, update Anypoint Studio. The January 2015 release of Anypoint Studio does not display record variables in the Mule Debugger.

29. Click the Variables tab.

30. Step through the rest of the records in the first batch step and watch the payload, the frame flow variable, and the rname record variable.

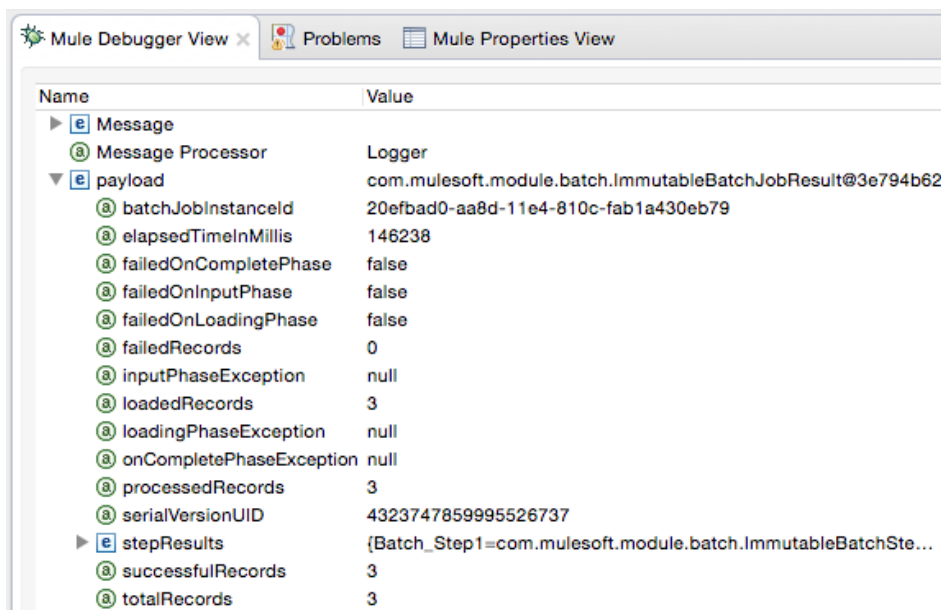
31. Step into the second batch step and look at the payload, the flow variables, and the record variables; you should see the rname record variable and size flow variable but not the fname flow variable.



Name	Value	Type
③ batchJobInstan...	9478c300-500c-11e5-...	java.lang.String
③ moveToDirectory	src/main/resources/ou...	java.lang.String
③ originalDirectory	/Users/jeanette.stallon...	java.lang.String
③ originalFilename	accounts.csv	java.lang.String
③ size	3	java.lang.Integer

32. Step through the rest of the records in the second batch step.

33. Step into the on complete phase; you should see the payload is an ImmutableBatchJobResult.



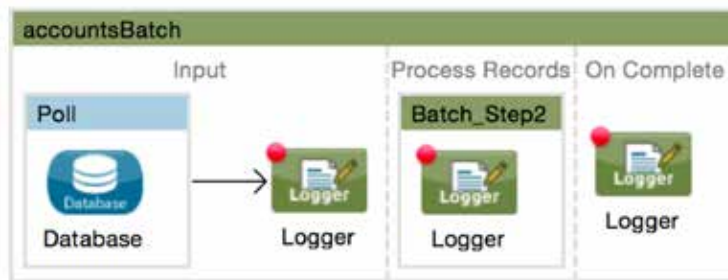
Name	Value
▶ [e] Message	
③ Message Processor	Logger
▼ [e] payload	com.mulesoft.module.batch.ImmutableBatchJobResult@3e794b62
③ batchJobInstanceId	20efbad0-aa8d-11e4-810c-fab1a430eb79
③ elapsedTimeInMillis	146238
③ failedOnCompletePhase	false
③ failedOnInputPhase	false
③ failedOnLoadingPhase	false
③ failedRecords	0
③ inputPhaseException	null
③ loadedRecords	3
③ loadingPhaseException	null
③ onCompletePhaseException	null
③ processedRecords	3
③ serialVersionUID	4323747859995526737
▶ [e] stepResults	{Batch_Step1=com.mulesoft.module.batch.ImmutableBatchSte...
③ successfulRecords	3
③ totalRecords	3

34. Step to the end of the application; you should see the number of process records phase and failed records in the console.

Walkthrough 9-3: Create a batch job for records in a database

In this walkthrough, you will create a batch job that retrieves records from a legacy database. You will:

- Go to a form and add multiple accounts for a specific postal code to the database.
- Create a new flow containing a batch job that polls a MySQL database every 45 seconds for records with a specific postal code.
- Use the Poll scope.



Get familiar with the data in the database

1. In a browser, go to the account list URL for the MySQL database listed in the course snippets.txt file.
2. Look at the existing data and the name of the columns, which match the names of the database fields.

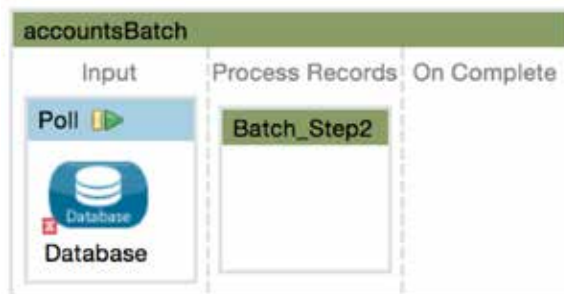
The screenshot shows a web browser window with the address bar displaying 'mu.mulesoft-training.com/essentials/accounts/show'. The page title is 'MUA Accounts'. There is a green button labeled 'Create More Accounts' in the top right corner. Below the title is a table with the following columns: accountID, name, street, city, state, postal, and country. The table contains several rows of data, with account IDs ranging from 1104 down to 1098. The data is partially obscured by horizontal lines, suggesting a large dataset.

accountID	name	street	city	state	postal	country
1104						
1103						
1102						
1101						
1100						
1099						
1098						

3. Click the Create More Accounts button.
4. Add one or more new records with a specific postal code; you will retrieve these records in this walkthrough and insert them into your Salesforce accounts in the next walkthrough.

Create a batch job that polls a database

5. Return to accounts.xml.
6. Drag a Batch scope element onto the canvas from the palette to create a new batch job.
7. Add a Poll scope element to the Input phase.
8. In the Properties view, select the fixed frequency scheduler.
9. Set the frequency to 30 and the time unit to seconds.
10. Add a Database connector to the poll.



11. Set the connector configuration to the existing Training_MySQL_Configuration global element.
12. Set the operation to Select.
13. Add a query to select the data for your postal code.

```
SELECT *  
FROM accounts  
WHERE postal = '94108'
```

Log each record to the console in the process records phase

14. Add a Logger component to the Process Records phase.
15. Display the record – the message payload – and other text or information you wish.

```
#['\n\nRECORD: ' + payload]
```

Log processed records in on complete phase

16. Add a Logger component to the on complete phase.
17. Set the Logger message to display the number of processed records and failed records.

```
#['\n\nProcessed: ' + payload.processedRecords + ' Failed: ' +  
payload.failedRecords]
```

Test the application

18. Run the application and watch the console; you should see records displayed every 30 seconds.

```
Console x
<terminated> apessentials [Mule Applications] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Aug 31, 2015, 1:23:51,484) [batch-job-accountsBatch-work-manager.02] org.mule.api.processor.LoggerMessageProcessor:

Processed: 2 Failed: 0
INFO 2015-08-31 13:23:51,484 [batch-job-accountsBatch-work-manager.02] com.mulesoft.module.batch.engine.DefaultBatchEngine: Finished execution of onComplete phase for instance 31815a70-501e-11e5-96b9-fe4c2d1fdc03 of job accountsBatch
INFO 2015-08-31 13:23:51,484 [batch-job-accountsBatch-work-manager.02] com.mulesoft.module.batch.engine.DefaultBatchEngine: Finished execution for instance '31815a70-501e-11e5-96b9-fe4c2d1fdc03' of job 'accountsBatch'. Total Records processed: 2. Successful records: 2. Failed Records: 0
INFO 2015-08-31 13:23:51,484 [batch-job-accountsBatch-work-manager.02] com.mulesoft.module.batch.engine.DefaultBatchEngine:

INFO 2015-08-31 13:23:51,489 [batch-job-accountsBatch-work-manager.02] com.mulesoft.module.batch.DefaultBatchStep: Step Batch_Step2 finished processing all records for instance 31815a70-501e-11e5-96b9-fe4c2d1fdc03 of job accountsBatch
INFO 2015-08-31 13:24:21,441 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Created instance 4361ec02-501e-11e5-96b9-fe4c2d1fdc03 for batch job accountsBatch
INFO 2015-08-31 13:24:21,442 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Starting input phase
INFO 2015-08-31 13:24:21,442 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Input phase completed
INFO 2015-08-31 13:24:21,451 [pool-27-thread-1] com.mulesoft.module.batch.engine.queue.BatchQueueLoader: Starting loading phase for instance '4361ec02-501e-11e5-96b9-fe4c2d1fdc03' of job 'accountsBatch'
INFO 2015-08-31 13:24:21,455 [pool-27-thread-1] com.mulesoft.module.batch.engine.queue.BatchQueueLoader: Finished loading phase for instance 4361ec02-501e-11e5-96b9-fe4c2d1fdc03 of job accountsBatch. 2 records were loaded
INFO 2015-08-31 13:24:21,458 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Started execution of instance '4361ec02-501e-11e5-96b9-fe4c2d1fdc03' of job 'accountsBatch'
INFO 2015-08-31 13:24:21,460 [batch-job-accountsBatch-work-manager.02] org.mule.api.processor.LoggerMessageProcessor:

RECORD: {accountID=1105, country=United States, street=77 Geary Street, state=CA, name=Max Mule, city=San Francisco, postal=94118}
INFO 2015-08-31 13:24:21,461 [batch-job-accountsBatch-work-manager.02] org.mule.api.processor.LoggerMessageProcessor:

RECORD: {accountID=1106, country=United States, street=77 Geary Street, state=CA, name=Molly Mule, city=San Francisco, postal=94118}
INFO 2015-08-31 13:24:21,473 [batch-job-accountsBatch-work-manager.02] com.mulesoft.module.batch.engine.DefaultBatchEngine: Starting execution of onComplete phase for instance 4361ec02-501e-11e5-96b9-fe4c2d1fdc03 of job accountsBatch
INFO 2015-08-31 13:24:21,473 [batch-job-accountsBatch-work-manager.02] org.mule.api.processor.LoggerMessageProcessor:

Processed: 2 Failed: 0
INFO 2015-08-31 13:24:21,474 [batch-job-accountsBatch-work-manager.02] com.mulesoft.module.batch.engine.DefaultBatchEngine: Finished execution of onComplete phase for instance 4361ec02-501e-11e5-96b9-fe4c2d1fdc03 of job accountsBatch
```

Note: Right now, all records with matching postal code are retrieved – over and over again. In the next walkthrough, you will modify this so only new records with the matching postal code are retrieved.

Walkthrough 9-4: Restrict processing using a poll watermark

In this walkthrough, you will modify the poll so it only retrieves *new* database records with a specific postal code. You will:

- Modify the Poll to use a watermark to keep track of the last record returned from the database.
- Modify the database query to use the watermark.
- Clear application data.

Parameterized query:

```
SELECT *  
FROM accounts  
WHERE postal = '94108' AND accountID > #[flowVars.lastAccountID]
```

Use a watermark to keep track of the last record

1. Return to accounts.xml.
2. In the Poll Properties view, select Enable watermark.
3. Set the watermark to store the max accountID returned by setting the following values.
 - Variable name: lastAccountID
 - Default expression: 0
 - Selector: MAX
 - Selector Expression: #[payload.accountID]

The screenshot shows the Mule Debugger interface with the 'Poll' configuration window open. The 'General' tab is selected, and the 'Watermark' section is expanded. The 'Enable watermark' radio button is selected. The 'Variable Name' is set to 'lastAccountID', the 'Default Expression' is '0', the 'Update Expression' is empty, the 'Selector' is set to 'MAX', the 'Selector Expression' is '#[payload.accountID]', and the 'Object Store' is empty. A message at the top says 'There are no errors.'

Debug and examine the watermark value

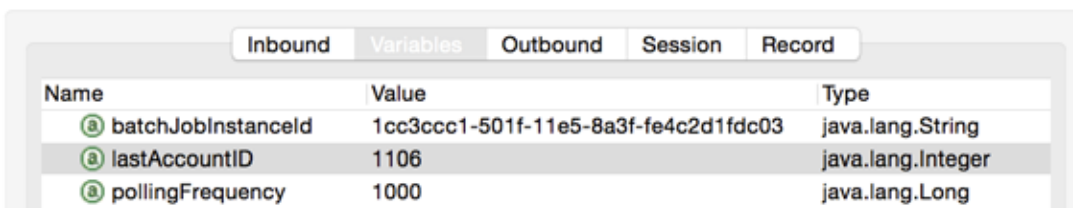
4. Place a breakpoint on the Logger in the On Complete phase.
5. Debug the application.
6. Find your watermark variable in the Variables section of the Mule Debugger view; initially, you should see a default value of zero.



The screenshot shows the 'Variables' tab in the Mule Debugger. It contains a table with three variables: batchJobInstanceId, lastAccountID, and pollingFrequency. lastAccountID has a value of 0.

Name	Value	Type
batchJobInstanceId	0b0ce1b0-501f-11e5-8a3f-fe4c2d1fdc03	java.lang.String
lastAccountID	0	java.lang.String
pollingFrequency	1000	java.lang.Long

7. Run the application though until the next polling event – the next time it retrieves records from the accounts table; the watermark variable should now be equal to the max accountID for training accounts records with the country you are using.



The screenshot shows the 'Variables' tab in the Mule Debugger. The lastAccountID variable now has a value of 1106, indicating it has been updated to the maximum account ID from the previous batch.

Name	Value	Type
batchJobInstanceId	1cc3ccc1-501f-11e5-8a3f-fe4c2d1fdc03	java.lang.String
lastAccountID	1106	java.lang.Integer
pollingFrequency	1000	java.lang.Long

8. Resume the application multiple times; the same records should still be selected over and over again.

Modify the database query to use the watermark

9. In the Database Properties view, modify the query so it only returns records for your postal code and with accountID values greater than the watermark lastAccountID value.

Parameterized query:

```
SELECT *  
FROM accounts  
WHERE postal = '94108' AND accountID > #[flowVars.lastAccountID]
```

Test the application

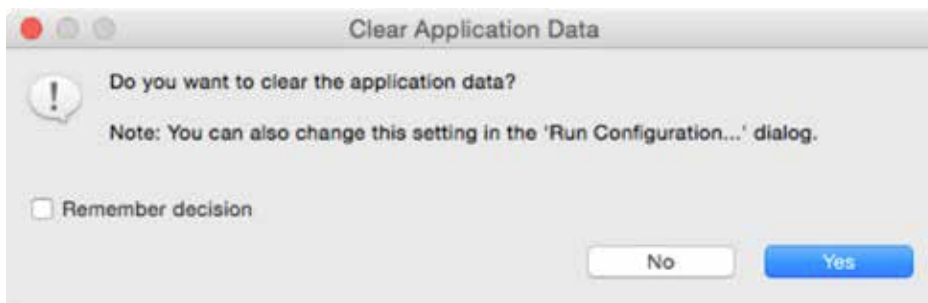
10. Run the application and look at the console; you should see that no records are retrieved at all this time.

```
Processed: 0 Failed: 0
INFO 2015-08-31 13:33:19,804 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Finished execution of onComplete phase for instance 843f092c-501f-11e5-802f-fe4c2d1fdc03 of job accountsBatc
```

Note: By default, the watermark is stored in a persistent object store so its value is retained between different executions of the application.

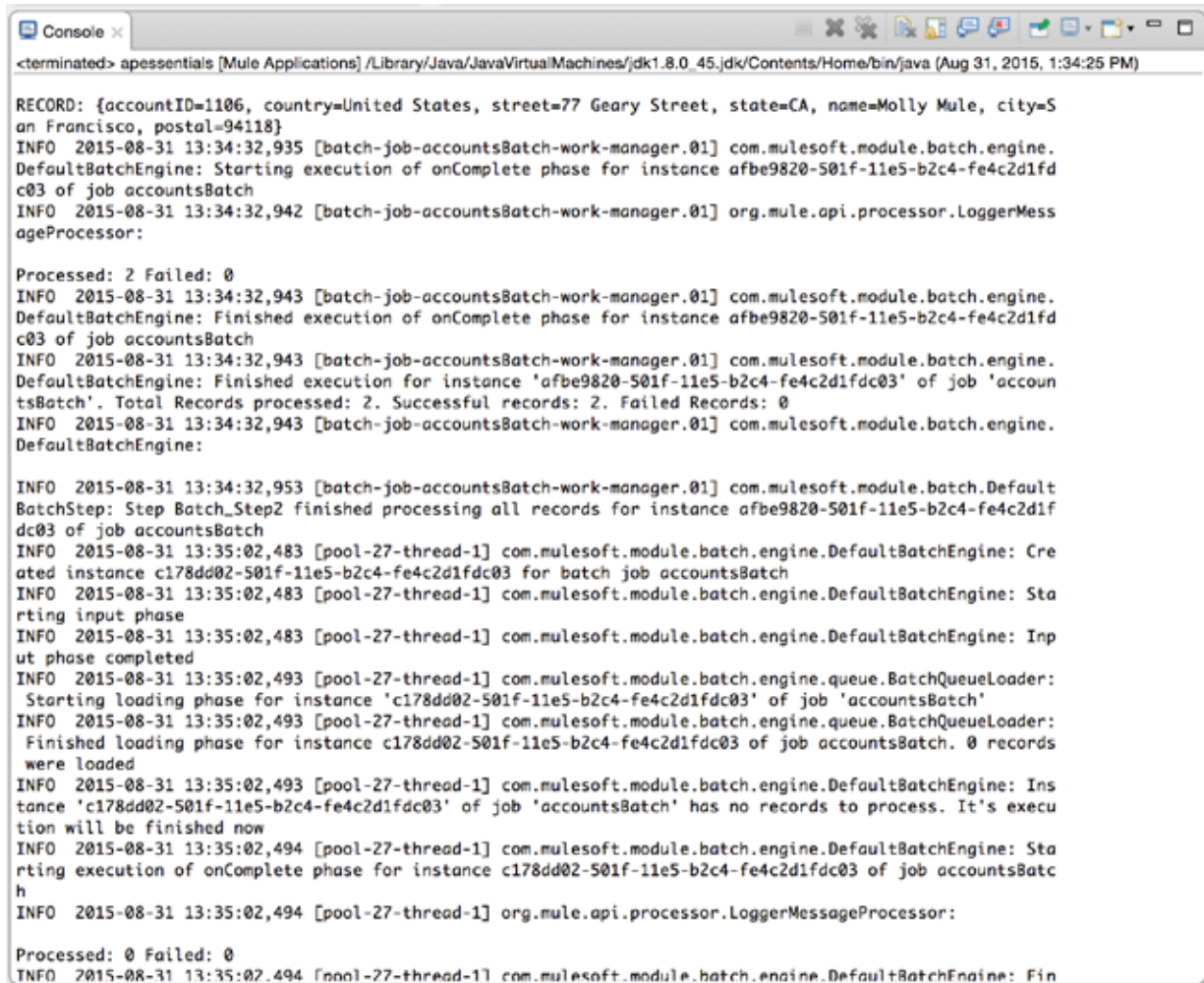
Clear application data

11. Select Run > Run Configurations.
12. Make sure your apessentials project is selected and then on the General tab, change Clear Application Data from Never to Prompt.
13. Run or debug your application again; you should be prompted to clear the application data.
14. Click Yes.



15. Look at the console; you should see the latest matching records retrieved from the legacy database again – but this time, only once.

16. Watch the console and see that all subsequent polling events retrieve no records.



```
<terminated> apessentials [Mule Applications] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Aug 31, 2015, 1:34:25 PM)

RECORD: {accountID=1106, country=United States, street=77 Geary Street, state=CA, name=Molly Mule, city=San Francisco, postal=94118}
INFO 2015-08-31 13:34:32,935 [batch-job-accountsBatch-work-manager.01] com.mulesoft.module.batch.engine.DefaultBatchEngine: Starting execution of onComplete phase for instance afbe9820-501f-11e5-b2c4-fe4c2d1fdc03 of job accountsBatch
INFO 2015-08-31 13:34:32,942 [batch-job-accountsBatch-work-manager.01] org.mule.api.processor.LoggerMessageProcessor:

Processed: 2 Failed: 0
INFO 2015-08-31 13:34:32,943 [batch-job-accountsBatch-work-manager.01] com.mulesoft.module.batch.engine.DefaultBatchEngine: Finished execution of onComplete phase for instance afbe9820-501f-11e5-b2c4-fe4c2d1fdc03 of job accountsBatch
INFO 2015-08-31 13:34:32,943 [batch-job-accountsBatch-work-manager.01] com.mulesoft.module.batch.engine.DefaultBatchEngine: Finished execution for instance 'afbe9820-501f-11e5-b2c4-fe4c2d1fdc03' of job 'accountsBatch'. Total Records processed: 2. Successful records: 2. Failed Records: 0
INFO 2015-08-31 13:34:32,943 [batch-job-accountsBatch-work-manager.01] com.mulesoft.module.batch.engine.DefaultBatchEngine:

INFO 2015-08-31 13:34:32,953 [batch-job-accountsBatch-work-manager.01] com.mulesoft.module.batch.DefaultBatchStep: Step Batch_Step2 finished processing all records for instance afbe9820-501f-11e5-b2c4-fe4c2d1fdc03 of job accountsBatch
INFO 2015-08-31 13:35:02,483 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Created instance c178dd02-501f-11e5-b2c4-fe4c2d1fdc03 for batch job accountsBatch
INFO 2015-08-31 13:35:02,483 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Starting input phase
INFO 2015-08-31 13:35:02,483 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Input phase completed
INFO 2015-08-31 13:35:02,493 [pool-27-thread-1] com.mulesoft.module.batch.engine.queue.BatchQueueLoader: Starting loading phase for instance 'c178dd02-501f-11e5-b2c4-fe4c2d1fdc03' of job 'accountsBatch'
INFO 2015-08-31 13:35:02,493 [pool-27-thread-1] com.mulesoft.module.batch.engine.queue.BatchQueueLoader: Finished loading phase for instance c178dd02-501f-11e5-b2c4-fe4c2d1fdc03 of job accountsBatch. 0 records were loaded
INFO 2015-08-31 13:35:02,493 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Instance 'c178dd02-501f-11e5-b2c4-fe4c2d1fdc03' of job 'accountsBatch' has no records to process. Its execution will be finished now
INFO 2015-08-31 13:35:02,494 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Starting execution of onComplete phase for instance c178dd02-501f-11e5-b2c4-fe4c2d1fdc03 of job accountsBatch
INFO 2015-08-31 13:35:02,494 [pool-27-thread-1] org.mule.api.processor.LoggerMessageProcessor:

Processed: 0 Failed: 0
INFO 2015-08-31 13:35:02,494 [pool-27-thread-1] com.mulesoft.module.batch.engine.DefaultBatchEngine: Fin
```


Walkthrough 9-5: Restrict processing using a message enricher and a batch step filter

In this walkthrough, you will add logic to check and see if an account already exists in Salesforce before adding it. You will:

- Add a first batch step with a Message Enricher scope element that checks if a record already exists in Salesforce (an account with the same Name) and stores the result in a record variable and retains the original payload.
- Modify the second batch step to use a filter that only allows new records (records that don't already exist) to be processed.
- (Optional) Add the record(s) to Salesforce.



Add a Salesforce account manually

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 with Postal Code and click the Go button.
4. Click the New Account button.
5. Enter an account name (one that matches one of the accounts you added with your postal code to the MySQL database) and click Save.

The screenshot shows the 'Account Edit' form in Salesforce, titled 'New Account'. It includes a header with 'Account Edit' and buttons for 'Save', 'Save & New', and 'Cancel'. The form is divided into sections for 'Account Information' and 'Web'.

Account Information:

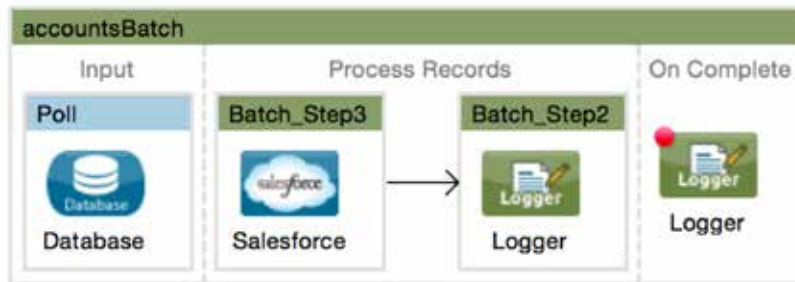
- Account Owner: Max Mule
- Account Name: Molly Mule
- Parent Account: (empty field)
- Account Number: (empty field)
- Account Site: (empty field)
- Type: --None--
- Industry: --None--
- Annual Revenue: (empty field)

Web:

- RA
- PH
- Web
- Ticker Syn
- Owner
- Employ
- SIC C

Check to see if a record already exists in Salesforce

6. Return to Anypoint Studio.
7. Return to accountsBatch in accounts.xml.
8. Drag out a new Batch Step scope element and drop it at the beginning of the process records phase.
9. Add a Salesforce connector to the new batch step.

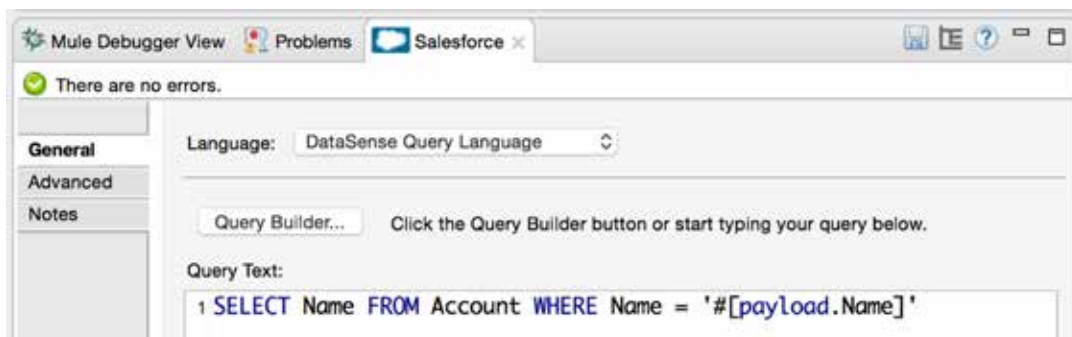


10. Configure the Salesforce connector endpoint to use the existing Salesforce connector configuration.
11. Set the operation to Query.
12. Click the Query Builder button.
13. Set the type to Account (Account).
14. Select a field of Name; it does not matter what you select, you just want to see if any records are returned.
15. Click the Add Filter button.
16. Create a filter to check if the Name field in the record being processed already exists in the database.

Name = #[payload.Name]

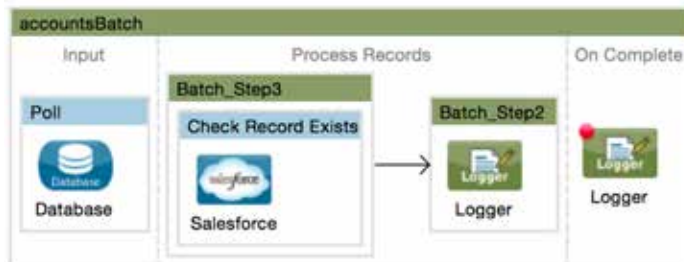
Note: Right now, you can use payload.name or payload.Name because the payload is a caseSensitiveHashMap. Later this walkthrough, however, you will transform the Map to an Account object with a Name property and will have to refer to this as payload.Name.

17. Click OK.



Add a Message Enricher

18. Add a Message Enricher scope element to the first batch step.
19. Move the Salesforce connector endpoint so it is inside the first batch step.
20. In the Message Enricher Properties view, set the display name to Check Record Exists.

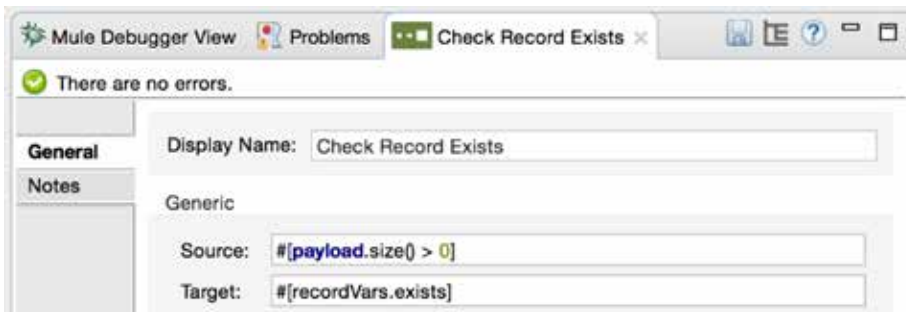


21. Set the source to an expression that checks to see if any records were returned, i.e. if there is a payload.

```
#[payload.size() > 0]
```

22. Set the target to assign the result of the source expression to a record variable called exists.

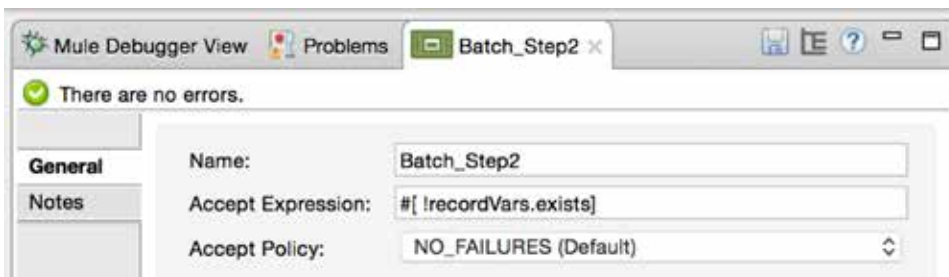
```
#[recordVars.exists]
```



Set a filter for the insertion step

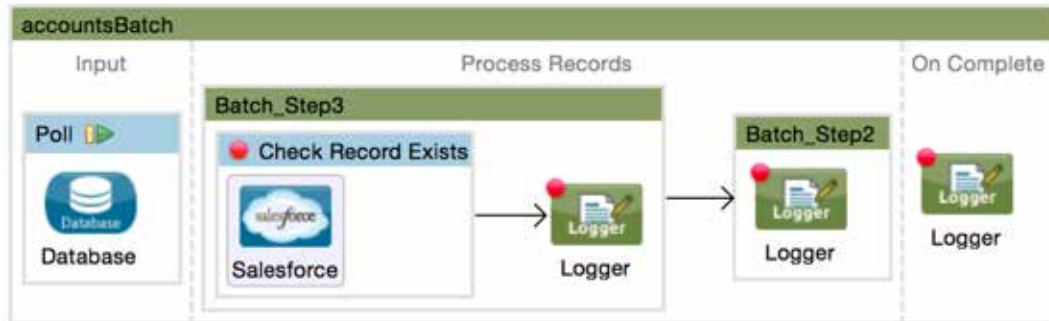
23. Double-click the second batch step that will/would insert the record into Salesforce.
24. In the Batch Step Properties view, set the Accept Expression so that only records that have a record variable exists set to false are processed.

```
#[!recordVars.exists]
```



Test the application

25. Add a breakpoint to the Message Enricher.
26. Add a Logger after the Message Enricher in the first batch step and add a breakpoint to it.



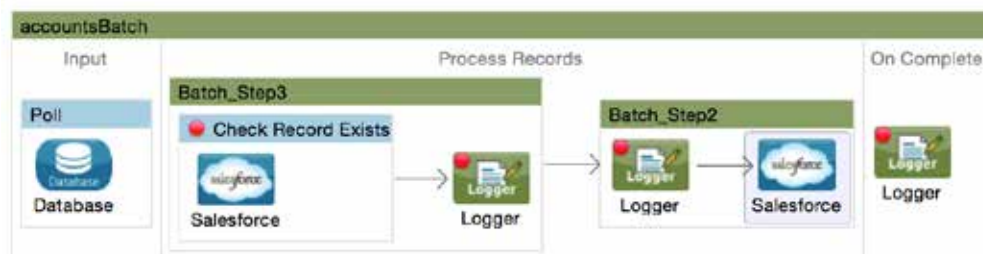
27. Add a breakpoint to the Logger in the second batch step.
28. Debug the application and clear the application data.
29. Step through the application and watch the record variables; you should see exists set to false for records with names that don't exist in Salesforce and true for those that do.

Inbound			Variables			Outbound			Session			Record		
Name			Value			Type								
exists			false			java.lang.Boolean								

Inbound			Variables			Outbound			Session			Record		
Name			Value			Type								
exists			true			java.lang.Boolean								

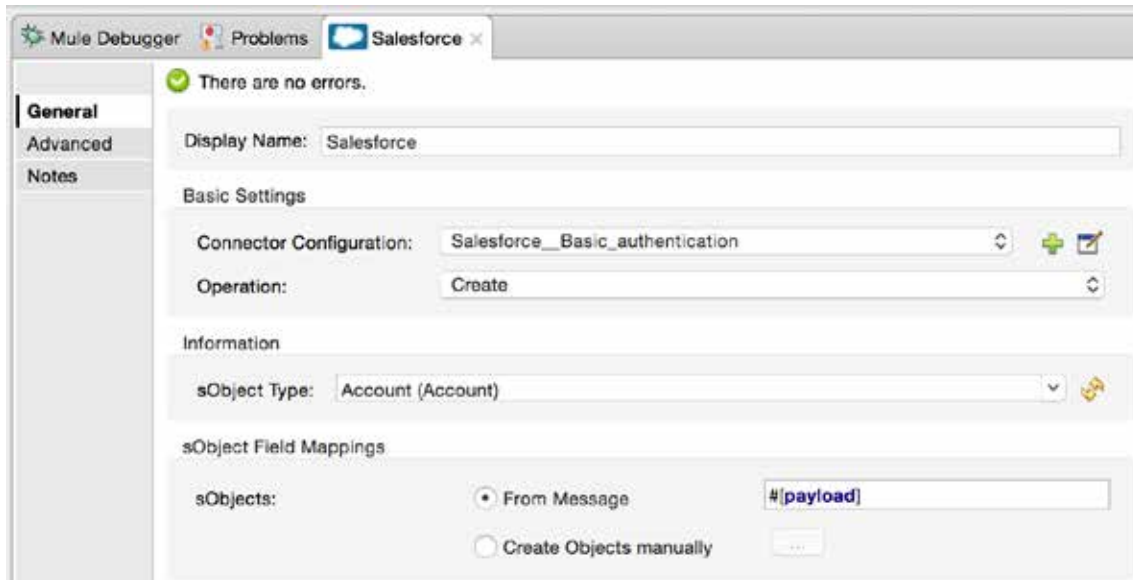
Commit new account records to Salesforce (optional)

30. Add a Salesforce connector to the second batch step in the processing phase.

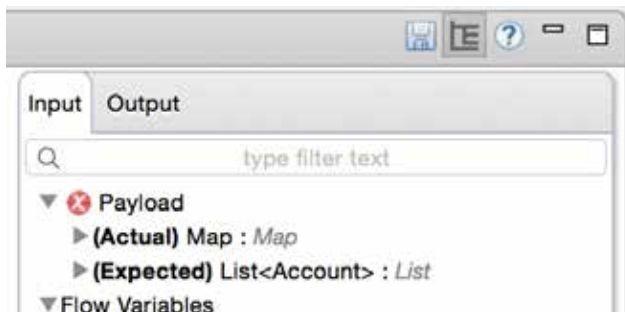


31. Configure the Salesforce connector endpoint to use the existing Salesforce connector configuration.
32. Set the operation to Create.

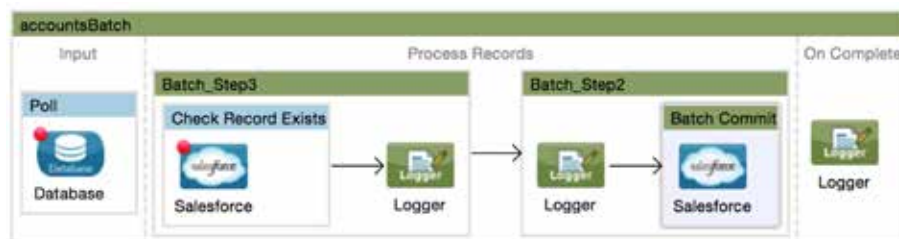
33. Set the sObject Type to Account (Account).
34. Leave the sObject Field Mappings to From Message #[payload].



35. Look at the DataSense Explorer and see a problem indicated for the payload; it is a Map but the outbound connector is expecting a List of Account objects.



36. Drag out a Batch Commit scope and drop it in the second batch step.
37. Add the Salesforce endpoint to it.



38. In the Batch Commit Properties view, set the commit size to 100.

Transform the record data

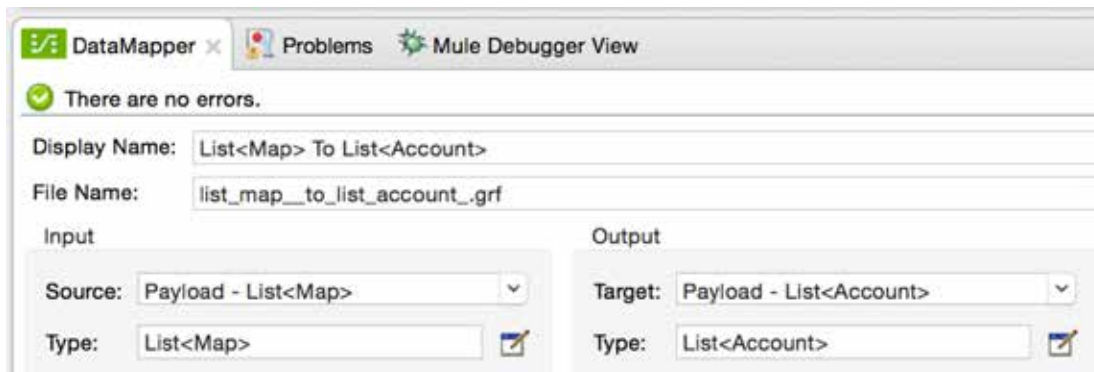
39. Copy the Salesforce endpoint in the second batch step and paste it after the poll in the input phase.

Note: You are temporarily adding this Salesforce endpoint after the Database endpoint so DataSense will work to create the mappings. You could also add the DataMapper to the process records phase, but this would add additional overhead.

40. Add a DataMapper between the Poll scope and the Salesforce endpoint.



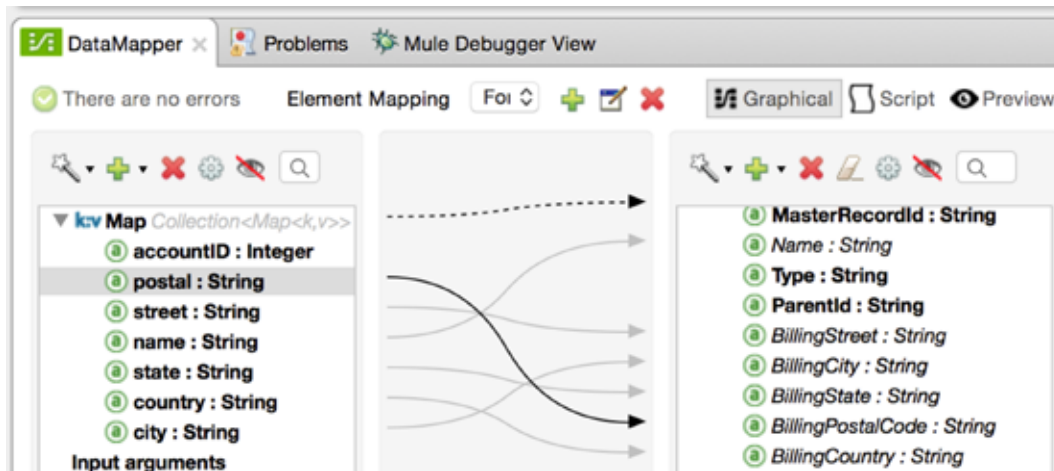
41. Look at the DataMapper Properties view; DataSense should work to automatically map List of Map objects to a List of Account objects.



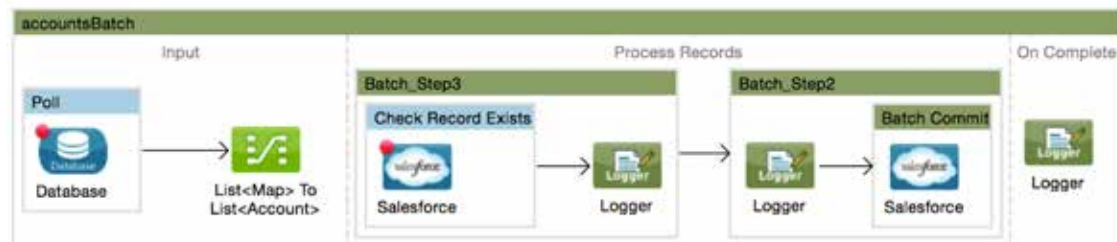
Note: If DataSense does not pick up the metadata for either the input or the output, go to the Properties view for that endpoint and then click in the white space of the canvas for the Input phase. This should trigger recaching of the metadata.

42. Click Create mapping.

43. Map all the fields except the accountID.



44. Delete the Salesforce endpoint in the input phase.



45. Save the file.

Test the application

46. Locate the Salesforce endpoint in getSFDCAccountsFlow.

47. Modify the query so it selects accounts with the postal code you have been using this module.

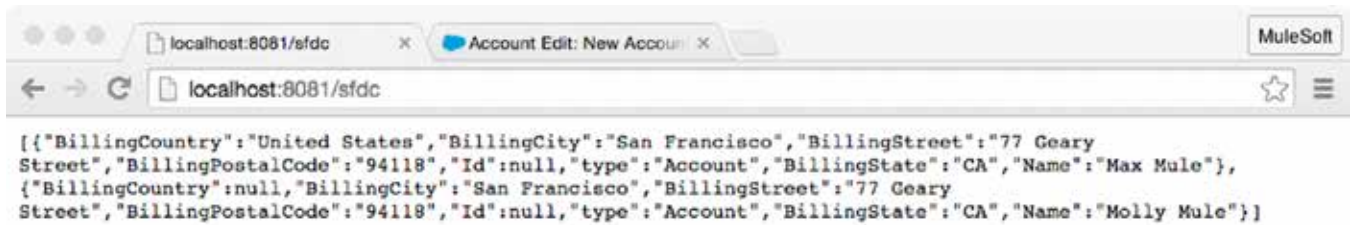
Query Text:

```
1 SELECT BillingCity,BillingCountry,BillingPostalCode,BillingState,BillingStreet,Name
2 FROM Account
3 WHERE BillingPostalCode = '94108'
```

48. Run the application and clear the application data.

49. Check the console and make sure you see your new records processed.

50. Make a request to <http://localhost:8081/sfdc>; you should see your new records from the legacy MySQL database now in the Salesforce database.



```
[{"BillingCountry": "United States", "BillingCity": "San Francisco", "BillingStreet": "77 Geary Street", "BillingPostalCode": "94118", "Id": null, "type": "Account", "BillingState": "CA", "Name": "Max Mule"}, {"BillingCountry": null, "BillingCity": "San Francisco", "BillingStreet": "77 Geary Street", "BillingPostalCode": "94118", "Id": null, "type": "Account", "BillingState": "CA", "Name": "Molly Mule"}]
```

51. Run the application again; no records should be processed.
52. Return to salesforce.com and locate your new record(s); they should have been inserted only once.

<input type="checkbox"/>	Action	Account Name ↑	Billing Country
<input type="checkbox"/>	Edit Del +	Burlington Textile...	USA
<input type="checkbox"/>	Edit Del +	Dickenson plc	USA
<input type="checkbox"/>	Edit Del +	Edge Communicati...	
<input type="checkbox"/>	Edit Del +	Express Logistics...	
<input type="checkbox"/>	Edit Del +	GenePoint	
<input type="checkbox"/>	Edit Del +	Grand Hotels & ...	
<input type="checkbox"/>	Edit Del +	Max Mule	United States
<input type="checkbox"/>	Edit Del +	Molly Mule	
<input type="checkbox"/>	Edit Del +	Pyramid Constru...	France
<input type="checkbox"/>	Edit Del +	sForce	US
<input type="checkbox"/>	Edit Del +	United Oil & Gas ...	
<input type="checkbox"/>	Edit Del +	United Oil & Gas, Si...	
<input type="checkbox"/>	Edit Del +	United Oil & Gas, UK	
<input type="checkbox"/>	Edit Del +	University of Arizona	