Developing Mule Applications for Performance

Student Manual

May 29, 2017

Table of Contents

Introduction

Use case

In these exercises, you will use a single use case, the ACME Bank.

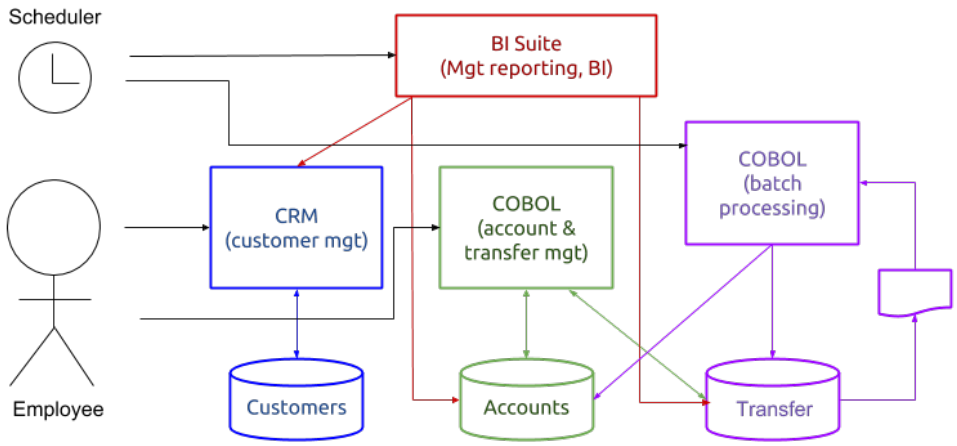
ACME Bank's heterogeneous IT landscape consists of a variety of systems, application, and databases. The systems lack proper integration, though there is some point-to-point integration for specific purposes. It recently became clear that the current landscape does not scale and does not provide enough flexibility and capacity to expand. It is also increasingly difficult to maintain some old legacy systems, which are no longer capable of handling the load caused by the natural growth of ACME Bank.

The newly appointed CTO's main responsibility is to lead ACME Bank into the digital era by providing new and innovative applications and functionalities. The focus will be on quickly enabling web based and mobile applications, which are yet to be developed. Also, in due time, several legacy systems are to be replaced by custom Java applications. To that end, a new digital platform will be created.

These systems are currently part of the IT landscape:

- A COBOL mainframe application that contains the main business logic for account management and transfer management. This application is difficult to use in an integrated environment as it does not provide proper interfacing methods. Bank employees use this application directly using a terminal. ACME wishes to have better interfacing capabilities. In due time, the entire application is due to be phased and replaced with a new (and yet to be developed) Java based application.
- An IBM DB/2 database that contains money transfers data, but also accounts data and is being used by the COBOL application for customer and accounts management.
- A CRM system from a commercial vendor with customizations for customer management. Bank employees use the web based UI on top of this Java based CRM system for accessing and managing customer data. The underlying database is an Oracle database.
- A commercial BI suite for providing management reporting, insight in transfers and customer data. The BI suite connects directly to the DB/2 databases and CRM system.

Partial, high level overview of ACME's applications and systems:



The platform will allow development of new services and the integration of old and new resources (systems, applications, databases, etc).

Module 1: Measuring Performance

Objectives:

- Understand performance.
- Understand performance requirements.
- Learn how to define performance.
- Learn how to measure and monitor performance.

Prerequisites

- Installation of Anypoint Studio (free download).
- Installation of Apache JMeter (free download).
- Installation of a stand-alone Mule EE runtime (30-day trial version can be downloaded free of charge).

  Please consult the setup document for more details.

Walkthrough 1-1: Gather performance metrics

Objectives

In this walkthrough, you will:

- Use a basic method of gathering performance metrics and information, using built-in application logging.

Background Story

One of ACME's in-house developers has built a prototype application that mimics the business logic of the old application, but it does not perform and scale very well and does not seem to meet the non-functional requirements. Further investigation is needed.

Import a Mule application in Anypoint Studio

1. Open Anypoint Studio.
2. Select File > Import.
3. In the dialog box, select Anypoint Studio > Anypoint Studio generated Deployable Archive (.zip).
4. Select Next.
5. Select file AcmeBankingServices.zip from the student files location, folder Exercises/Module01.
6. Select Finish.

Familiarize yourself with the imported application

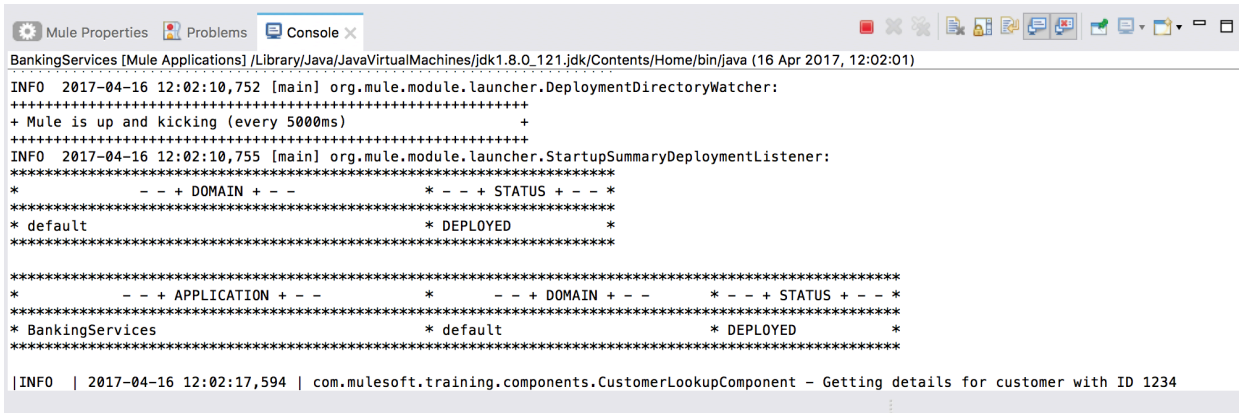
7. Open flow configuration file src/main/app/transferservice.xml.
8. Inspect the message processors of flow transferServiceFlow.
9. Open flow configuration files customerservice.xml.
10. Inspect the message processors of flow customerServiceFlow.
11. Open and inspect the Java classes in src/main/java, package com.mulesoft.training.components.
12. Open and inspect the java class CustomerLookupComponent.java in src/main/java, package com.mulesoft.training.components.

Run the application

13. From the main menu, select Run > Run As > Mule Application.
14. Test the application by opening a browser and entering this address location:

```
http://localhost:8081/customer?custid=1234
```

15. Verify the results in JSON format in the browser.
16. Check the console in Anypoint Studio for log messages.



17. Stop the application.

Add logging to the application

18. Open the customerservice.xml flow configuration file.
19. From the palette, drag a Variable transformer to flow customerServiceFlow; be sure to place it before the existing Choice router with label “CustID?”.
20. In the Properties view, set the name of the variable to “*STARTTIME*”.
21. Set the Display name to “*Set STARTTIME*”.

22. Add the following MEL expression as a value:

```
#[server.nanoTime()]
```

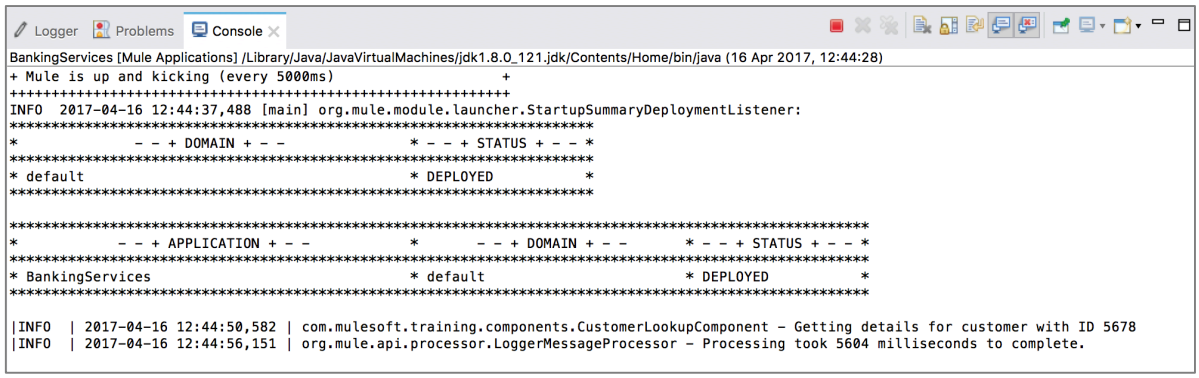
23. Save your changes.
24. At the end of the customerServiceFlow flow, add a Logger component.
25. Set this value as message:

```
#['Processing took ' +(server.nanoTime() - flowVars.STARTTIME)/1000000 +'
milliseconds to complete. ']
```

26. Start the application and send another request to the Customer service flow using a browser:

```
http://localhost:8081/customer?custid=5678
```

27. Check the new log message in the Console.



28. Stop the application.

Question 1: What is the processing time of a CustomerService request?

Answer: _____

Note: there will be more questions. Please open file questions.txt from the StudentFiles location in a text editor and add your answers to that file.

Add more performance logging

29. Open flow configuration file transferservice.xml.
30. From the palette, drag a Variable transformer to flow TransferServiceFlow; make sure to place the transformer at the beginning of the flow, directly after the message source (HTTP inbound endpoint).
31. In the Properties view, set the name of the variable to “STARTTIME”.
32. Set the Display name to “Set STARTTIME”.

33. Add the following MEL expression as a value:

```
#[server.nanoTime()]
```

34. Locate the Logger component at the end of the flow.
35. Set the message value to:

```
#['Processing and auditing took ' +(server.nanoTime() - flowVars.STARTTIME)
/1000000 +' milliseconds to complete'].
```

36. Save your changes.
37. Do not start the application at this time.

Walkthrough 1-2: Gather detailed performance metrics

Objectives

In this walkthrough, you will:

- Use a more advanced method of gathering performance metrics and information, using Apache JMeter.

Prerequisites

- Completion of walkthrough 1-1.
- If you did not complete Walkthrough 1-1, import the solution for this walkthrough from the studentFiles, folder 02-Solution/Module01, file AcmeBankingServices-WT1-1.zip.

Background story

ACME Bank expects to grow steadily in the next two years in terms of new customers, but also with regard to services, transfers, and new products.

The newly appointed CTO defined to following non-functional requirements:
NF001: The transfer processing service must be able to process 2000 transfers within 5 seconds.
NF002: The customer service must be able to handle 200 concurrent requests.

Your task is to create performance tests to verify that the newly developed Mule application meets these requirements.

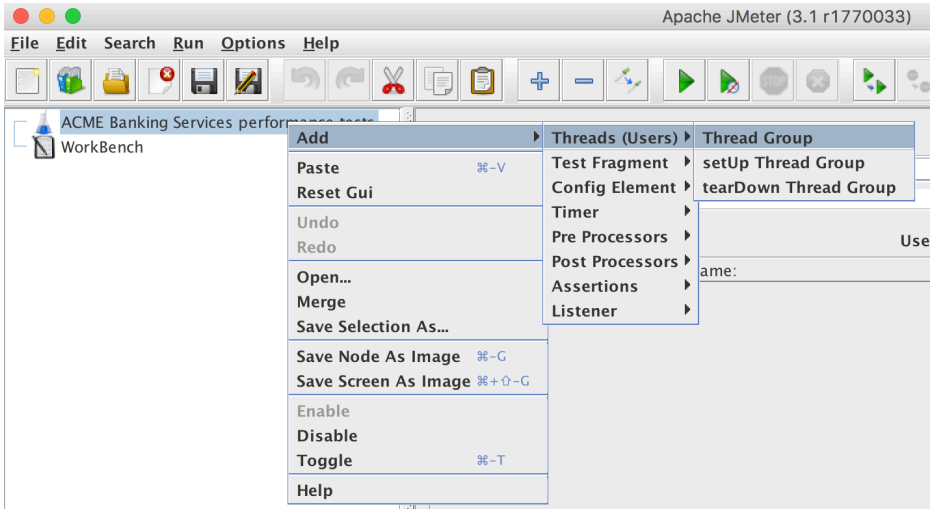
Run the BankingServices application in Anypoint Studio

1. Start Anypoint Studio.
2. Open and select the AcmeBankingServices project.
3. From the main menu, select Run > Run As > Mule Application.

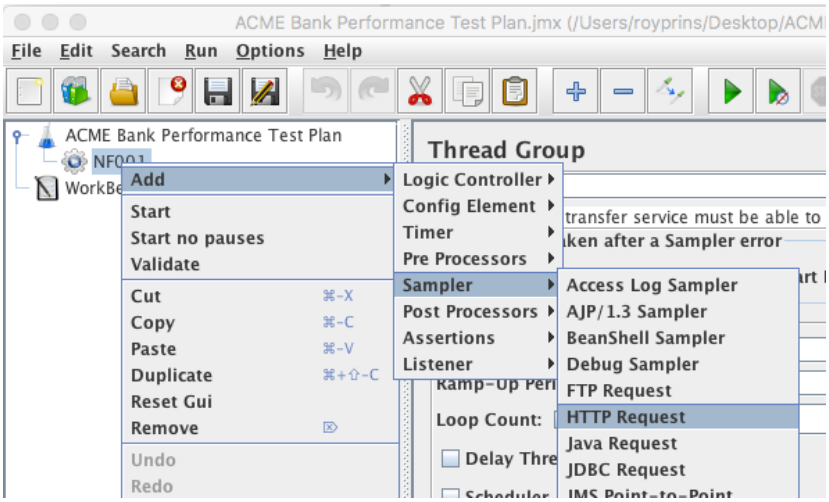
Create a performance test in Apache JMeter

4. Start Apache JMeter.
    - On Window, use the script JMETER_HOME\bin/jmeter.bat
    - On Linux/UNIX/MacOS, use the script JMETER_HOME/bin/jmeter.sh
5. In the main UI, select the Test Plan element in the left-hand panel.
6. Set the name of the test plan to “ACME Bank Performance Test Plan”.
7. In the main menu, select File > Save Test Plan As.

8. Save the test plan as “ACME Bank Performance Test Plan.jmx” to a location of choice, for example your desktop folder.
9. Right-click the test plan and select Add > Threads (Users) > Thread Group.



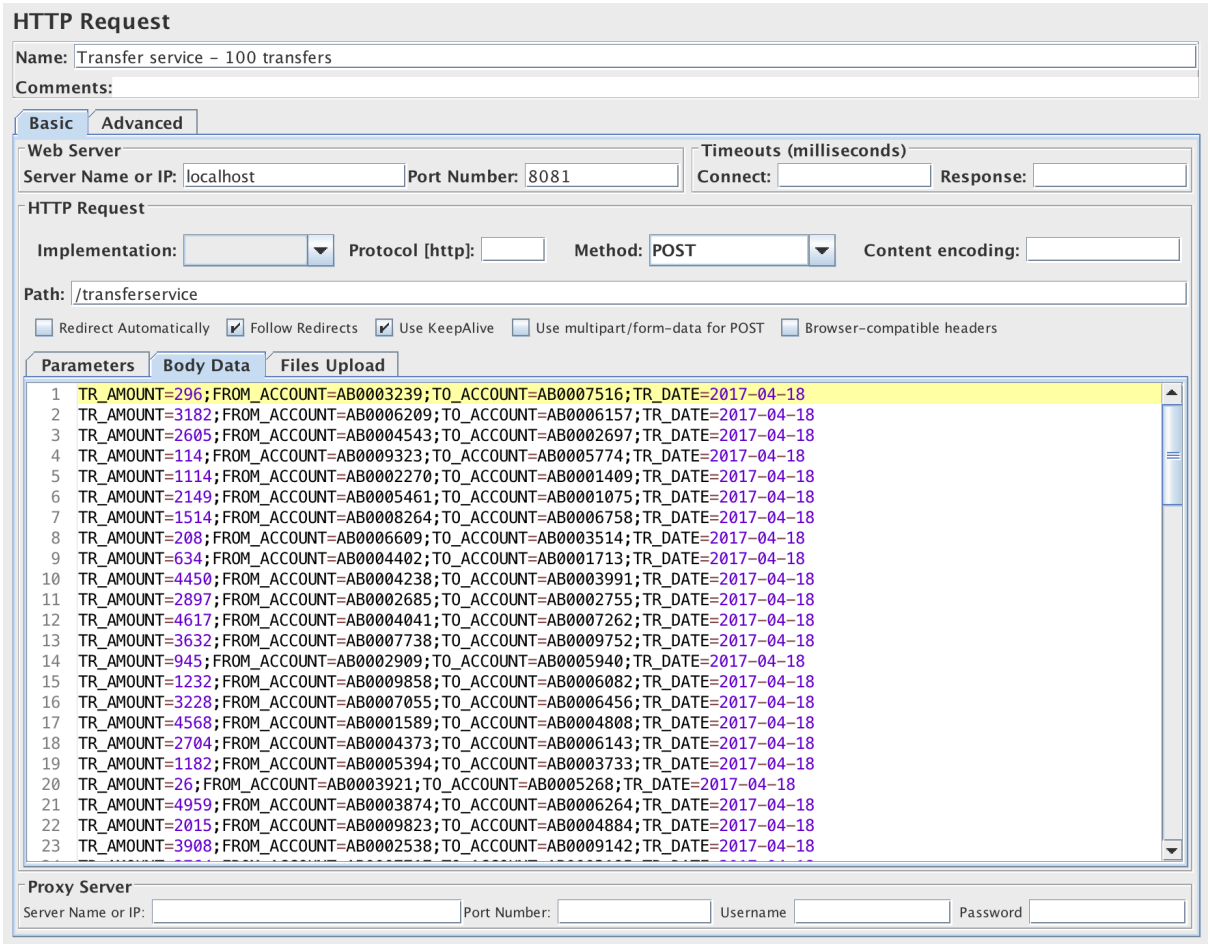
10. Set the name of the new thread group to “NF001”.
11. Set the Comments to “The transfer service must be able to process 2000 transfers within 5 seconds”.
12. Do not configure any other settings.
13. Save your changes by clicking File > Save from the main menu.
14. Right-click thread group NF001 and select Add > Sampler > HTTP Request.



15. Set the name of the HTTP request to “Transfer service – 100 transfers”.
16. In the tab labeled “Basic”, set Server Name to “localhost”.
17. Set Port Number to “8081”.
18. Set Method to “POST”.
19. Set Path to “/transferservice”.
20. Save your changes (CTRL+S on Windows or CMD-S on macOS).

Add sample data to your test

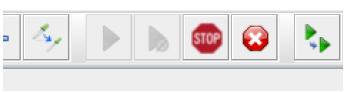
21. From the student files, open file “transfers_100.txt” in folder Exercises/Module01; use a text editor of choice.
22. Return to JMeter.
23. In the tab labeled “Body Data”, paste the contents of file transfers_100.txt”.



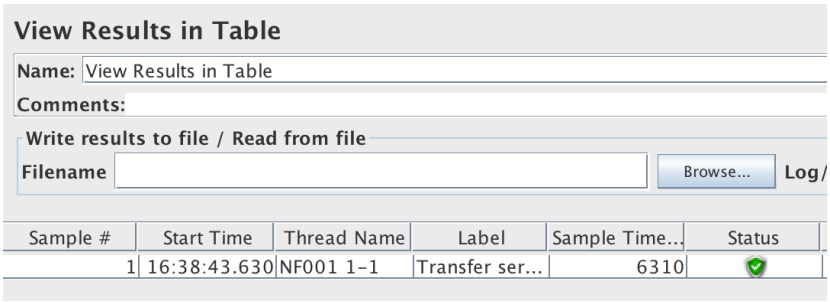
24. Right-click the HTTP request (Transfer service – 100 request). Select Add > Listener > View Results in Table.
25. Save your changes (CTRL+S on Windows, CMD+S on macOS).

Run the performance test

26. Right-click thread group NF001 and select Start.
27. Note how Anypoint Studio’s console will display log messages.
28. Note how JMeter’s toolbar will change. A red “Stop” icon will be displayed while the test runs.



29. After completing the test, the stop icon will be disabled.
30. Click “View Results in Table”.



31. The details panel will display the test results.
32. Try to locate a value labeled “Sample Time”.

Question 2: How long does it take to process 100 transfers? Answer: _____

Create a second performance test with validation

33. Right-click thread group NF001 and select Add > Sampler > HTTP Request.
34. Set the name of the HTTP request to “Transfer service – 2000 transfers”.
35. In the tab labeled “Basic”, set Server Name to “localhost”.
36. Set Port Number to “8081”.
37. Set Method to “POST”.
38. Set Path to “/transferservice”.
39. Save your changes (CTRL+S / CMD+S).
40. From the student files, open file “transfers_2000.txt” in folder Exercises/Module01; use a text editor of choice.
41. Return to JMeter.
42. In the tab labeled “Body Data”, paste the contents of file transfers_2000.txt.
43. Right-click “HTTP request (Transfer service – 2000 request")” and select Add > Assertions > Duration Assertion.
44. Set the value for Duration in milliseconds to “5000”.
45. Right-click “HTTP request (Transfer service – 2000 request)” and select Add > Listener > View Results in Table.
46. Right-click the HTTP request (Transfer service – 100 request) and select Disable.
47. Save your changes (CTRL+S / CMD+S).

Run the performance test

48. Right-click thread group NF001 and select Start.

    Note: This will take considerably longer than the previous test using only 100 transfers.

Check the results

49. After completing the test, click “View Results in Table”.
50. Try to locate the value for “Sample Time”.

Label	Sample Tim...	Status
Transfer service - 2000 transfers	106531	

51. Switch to Anypoint Studio and check the results in the Console.

```
|INFO  | 2017-04-16 17:08:21,504 | com.mulesoft.training.components.TransferProcessingComponent - Processing 2000 transfers
|INFO  | 2017-04-16 17:10:06,988 | com.mulesoft.training.components.TransferAuditingComponent - Selecting transfer with highest amount for auditing
|INFO  | 2017-04-16 17:10:06,989 | com.mulesoft.training.components.TransferAuditingComponent - Auditing transfer with ID 25508470-2302-11e7-8505-4c32759d5595
|INFO  | 2017-04-16 17:10:08,005 | org.mule.api.processor.LoggerMessageProcessor - Processing took 106516 milliseconds to complete.
```

Interpret the results

52. Return to JMeter.
53. Click “View Results in Table”.
54. Check the value for “Status” and note how it has a red symbol.
55. Compare the processing time in the Mule logs to the Sample Time value.

Question 3: How long does it take to process 2000 transfers?

Answer: _____

Question 4: Does the transfer service meet its non-functional requirements? How do you know?

Answer: _____

Deploy the banking services app to a stand-alone Mule runtime

56. Return to Anypoint Studio.
57. Stop the running Mule application.
58. Right-click the project in the Package Explorer and select Export.
59. Select Mule > Anypoint Studio Project to Mule Deployable Archive.

60. Click Next.
61. Enter a file name for the deployable archive, such as BankingServices.zip.
62. Click Finish.
63. Copy the generated deployable archive (.zip format) to $MULE_HOME/apps (where $MULE_HOME is the directory where you installed the Mule runtime).
64. Start the Mule runtime from command line. Use %MULE_HOME\bin\mule.bat (Windows) or %MULE_HOME/bin/mule (macOS/Linux/UNIX).
65. Return to JMeter.
66. Right-click thread group NF001 and select Start; the test will now run on the Mule runtime instead of from within Anypoint Studio (using the embedded runtime).
67. After completing, run the test at least one more time.
68. When done, click “View Results in Table”.
69. Compare the values of Sample Time of the various test runs.
70. Stop the Mule runtime.

Sample #	Start Time	Th...	Label	Sample Tim...	Status	Bytes	Sent Bytes	Latency	Connect Ti...
1	18:08:22.599	N...	Transfer service - 2000 transfers	106904		227	157765	106904	2
2	18:18:59.311	N...	Transfer service - 2000 transfers	107138		227	157765	107138	1
3	18:22:40.532	N...	Transfer service - 2000 transfers	107173		227	157765	107173	2
4	18:25:12.931	N...	Transfer service - 2000 transfers	103990		227	157765	103990	2
5	18:28:23.181	N...	Transfer service - 2000 transfers	107907		227	157765	107907	4

Red = results from running in Anypoint Studio

Blue = results from running in stand-alone runtime

Module 2: Analyzing Performance

Objectives:

- Learn how to analyze performance and performance measurements.
- Learn how to find bottlenecks.

Prerequisites

- Completion of previous walkthroughs.
    - If you did not complete Module 1, import the solutions for the walkthrough from the student files folder 02-Solution/Module01:
        - Import file AcmeBankingServices-WT1-1.zip into Anypoint Studio.
        - Open file “ACME Bank Performance Test Plan.jmx” in Apache JMeter.
- Installation of a stand-alone 3.8.x Mule runtime.
- Deployment of AcmeBankingServices app to Mule runtime. See WT 1-2 for details.
    - If you did not complete Module 1, copy file AcmeBankingServices-WT1-1.zip to $MULE_HOME/apps.

Walkthrough 2-1: Inspect Mule applications

Objectives

In this walkthrough, you will:

- learn how to inspect a running Mule application.
- Deploy to a stand-alone Mule runtime.
- Observe behavior.

Background story

ACME Bank expects to grow steadily in the next two years in terms of new customers, but also with regard to services, transfers and new products.

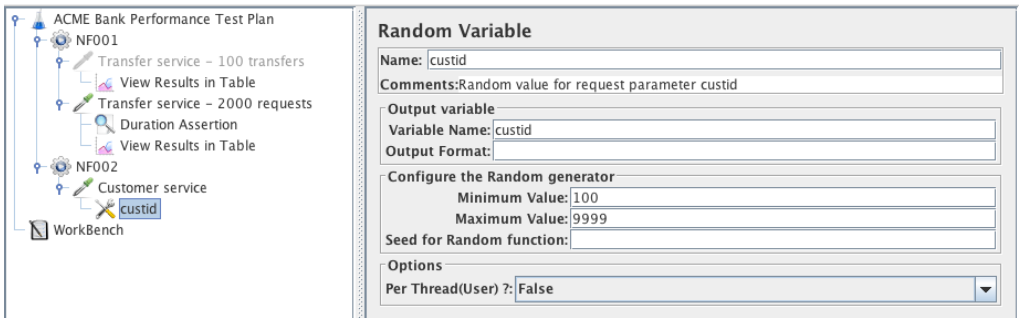
The newly appointed CTO defined to following non-functional requirements:

- NF001: The transfer processing service must be able to process 2000 transfers within 5 seconds.
- NF002: The customer service must be able to handle 200 concurrent requests.

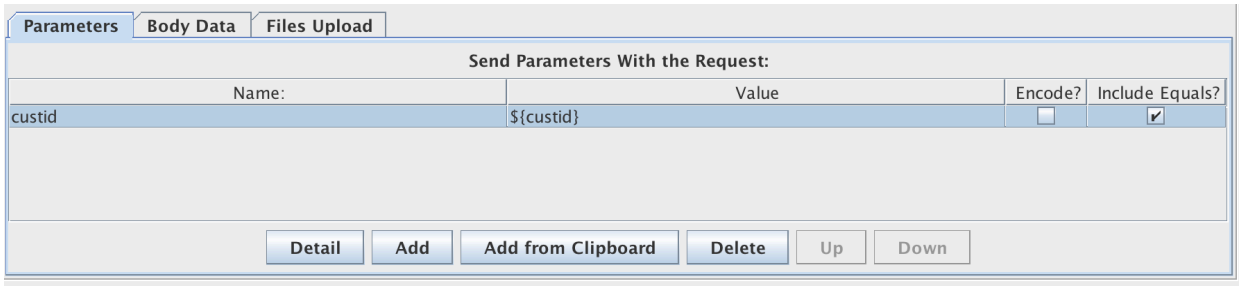
Generate a new load test

1. Return to JMeter.
2. Right-click the test plan and select Add > Threads (Users) > Thread Group.
3. Set the name of the new thread group to “NF002”.
4. Set the Comments to “The customer service must be able to handle 200 concurrent requests”.
5. Set the value for Number of Threads (users) to 200.
6. Save your changes (CTRL+S / CMD+S).
7. Right-click thread group NF002 and select Add > Sampler > HTTP Request.
8. Set the name of the HTTP request to “Customer service”.
9. In the tab labeled “Basic”, set Server Name to “localhost”.
10. Set Port Number to “8081”.
11. Set Method to “GET”.
12. Set Path to “/customer”.
13. Right-click Customer service and select Add > Config Element > Random Variable.
14. Set the name to “custid”.
15. Set the value for Comments to “Random value for request parameter custid”.
16. Set the value for Variable Name (under Output Variable) to “custid”.
17. Set the value for Minimum Value to “1000”.

18. Set the value for Maximum Value to “9999”.



19. Click Customer service.
20. In the tab Parameters, click “Add”.
21. Set Name to “custid”.
22. Set Value to “${custid}”.



23. Right-click Customer service and select Add > Listener > View Results Tree.
24. Save your changes.

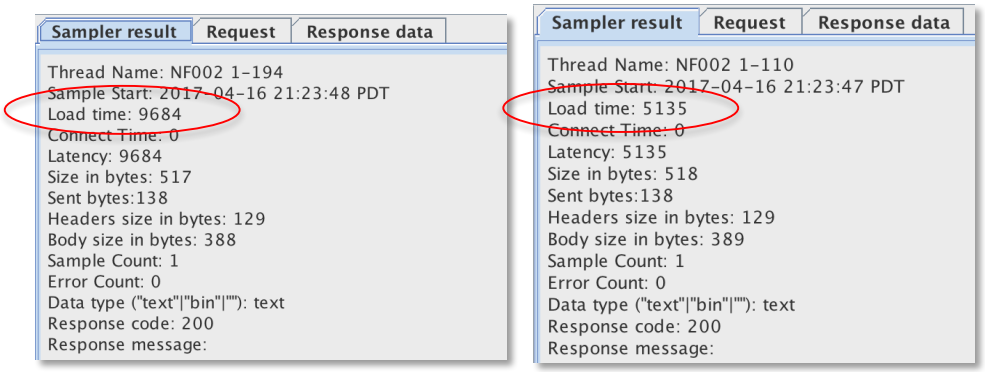
Run the load test

25. Make sure the Mule runtime is started and the banking services application has been deployed.
26. Right-click thread group NF001 and select “Disable” (Alternative: CTRL+T/CMD+T).
27. Save your changes.
28. Right-click thread group NF002 and select “Start”.
29. Observe the output of the Mule runtime in the terminal.

```
INFO  | 2017-04-16 21:19:01,912 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 9000
INFO  | 2017-04-16 21:19:01,912 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 1917
INFO  | 2017-04-16 21:19:01,912 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 8987
INFO  | 2017-04-16 21:19:01,912 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 3159
INFO  | 2017-04-16 21:19:01,913 | org.mule.api.processor.LoggerMessageProcessor - Processing took 5001 milliseconds to complete.
INFO  | 2017-04-16 21:19:01,915 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 8205
INFO  | 2017-04-16 21:19:01,915 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 6184
INFO  | 2017-04-16 21:19:01,916 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 9671
INFO  | 2017-04-16 21:19:01,916 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 7769
INFO  | 2017-04-16 21:19:01,916 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 9770
INFO  | 2017-04-16 21:19:01,916 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 7836
INFO  | 2017-04-16 21:19:01,916 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 3559
INFO  | 2017-04-16 21:19:01,916 | com.mulesoft.training.components.CustomerLookupComponent - Getting details for customer with ID 3189
INFO  | 2017-04-16 21:19:06,882 | org.mule.api.processor.LoggerMessageProcessor - Processing took 5003 milliseconds to complete.
INFO  | 2017-04-16 21:19:06,882 | org.mule.api.processor.LoggerMessageProcessor - Processing took 5003 milliseconds to complete.
INFO  | 2017-04-16 21:19:06,882 | org.mule.api.processor.LoggerMessageProcessor - Processing took 5004 milliseconds to complete.
INFO  | 2017-04-16 21:19:06,882 | org.mule.api.processor.LoggerMessageProcessor - Processing took 5003 milliseconds to complete.
```

30. Return to JMeter.

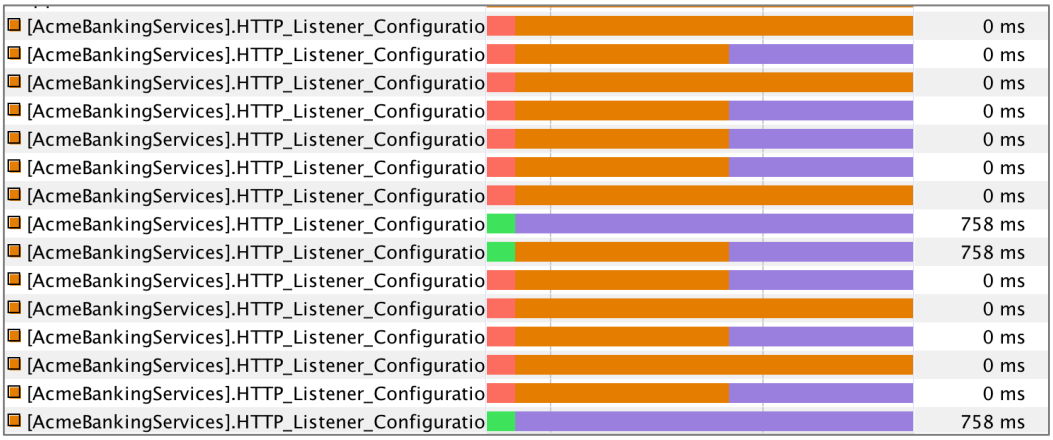
31. Select thread group NF002 > Customer service request > View Results Tree.
32. Inspect a few results from the top.
33. Inspect a few results from the bottom of the list; pay attention to the Load time value in the tab "Sampler result".



34. Inspect Response data for a few results.

Inspect a running Mule application with VisualVM

35. Open a new terminal/command window.
36. Start Java VisualVM by entering the command "jvisualvm".
37. Connect to process "org.mule.module.reboot.MuleContainerBootstrap" by double-clicking it.
38. Select tab "Threads".
39. Observe the various threads.
40. Return to JMeter.
41. OPTIONAL: Try to arrange the windows of JMeter and VisualVM side by side.
42. Right-click thread group NF002 and select "Start".
43. Return to VisualVM and observe the results of running the JMeter load test.



44. Stop VisualVM.

Stop the running Mule app

45. Stop the Mule runtime.

Walkthrough 2-2: Monitor Mule applications

Objectives

In this walkthrough, you will:

- Learn how to monitor a running Mule application using Java Mission Control, which can provide detailed information on live applications.

Prerequisites

- Having completed walkthrough 2-1.
- If you have not completed it successfully:
    - Import the solution (AcmeBankingServices-WT1-1.zip) from the student files, folder Solutions/Module01/ in Anypoint Studio.
    - Open the solution (ACME Bank Performance Test Plan.jmx) from the student files, folder Solutions/Module02/ in Apache JMeter.

Before you start

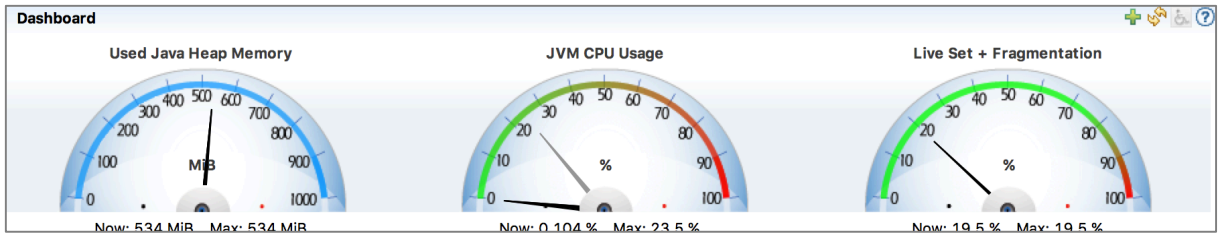
1. Make sure the Mule runtime is started and the banking services application has been deployed.

Run Java Mission Control

2. Open a new terminal/command window.
3. Start Java Mission Control by entering the command “jmc”.
4. OPTIONAL: Try to arrange the windows of JMeter and VisualVM side by side.
5. Connect to process “org.mule.module.reboot.MuleContainerBootstrap” by double-clicking it.
6. Familiarize yourself with the UI.

Run tests using JMeter

7. Return to JMeter.
8. OPTIONAL: Try to arrange the windows of JMeter and VisualVM side by side.
9. Right-click thread group NF002. Select “Start”.
10. Observe the dials of Java Mission Control.



Add custom dials for monitoring Mule applications

11. Click the green “plus” icon in the tab Dashboard.
12. Select Mule.AcmeBankingServices > Flow > customerServiceFlow > AverageProcessingTime. Click “Next”.
13. Select value “Duration” for Content Type.
14. Select units of 1 ms (millisecond).
15. Click “Finish”.
16. Do the same for flow transferServiceFlow.
17. Delete all other dials by right-clicking them and selecting “Delete”.
18. Change the display names of the remaining dials to “AverageProcessingTime - Customer service” and “AverageProcessingTime - Transfer service”.

19. From the main menu, select File > Save.
20. Explore Java Mission Control by adding several other dials; feel free to select one or more metrics of choice.

Stop the running Mule app

21. Save your changes.
22. Exit Java Mission Control.
23. Stop the Mule runtime.

Module 3: Developing for Performance

Objectives:

- Understand performance design considerations and tradeoffs.
- Learn how to build applications for performance.
- Learn how to build applications for scalability and reliability.

Prerequisites

- Having completed the previous walkthroughs.
- If you have not completed it successfully:
    - Import the solution (AcmeBankingServices-WT1-1.zip) from the student files, folder Solutions/Module01/ in Anypoint Studio.
    - Open the solution (ACME Bank Performance Test Plan.jmx) from the student files, folder Solutions/Module02/ in Apache JMeter.

Walkthrough 3-1: Refactoring for performance

Objectives

In this walkthrough, you will:

- Refactor a Mule application to improve its performance.
- Learn how to build scalable applications.
- Leverage the VM transport for auto-load balancing.

Create a copy of the transferservice.xml configuration file

1. Return to Anypoint Studio.
2. Open project AcmeBankingServices.
3. Copy flow configuration file transferservice.xml using CTRL+C/CTRL+V or CMD+C/CMD+V on respectively Windows or macOS.
4. Rename to copied file to transferservice-refactored.xml.
5. Open transferservice-refactored.xml.
6. Rename the flow to “transferServiceAquisitionFlow”.

Refactor properties to avoid conflicts

7. Select the HTTP inbound endpoint.
8. Change the Path to “/transferservice2”.
9. Save your changes; any error messages should disappear.

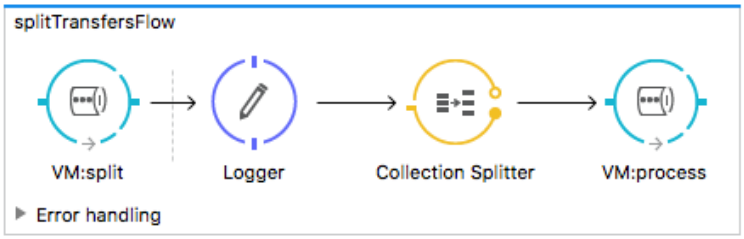
Create new flows

10. Create a new flow named “splitTransfersFlow” by dragging a Flow scope element from the Palette to the canvas.
11. Create another flow named “transferServiceProcessingFlow”.
12. Move components “Process transfer” and “Audit transfer” from flow transferServiceAquisitionFlow to flow transferServiceProcessingFlow by dragging them.

Add asynchronous behavior to transferService

13. Add a Request-Reply scope to transferServiceAquisitionFlow, directly after StringToTransferListTransformer, but before the set Payload transformer.
14. Add a VM transport element in both request and reply sections.

15. Select the VM transport in the Request section and set the Queue Path to “split”.
16. Select the VM transport in the Reply section and set the Queue Path to “return”.
17. Make sure the Exchange Pattern for both VM transports it set to “one-way (Default)”.
18. Add a Collection Splitter element to the process section of splitTransfersFlow.
19. At the beginning and end of both splitTransfersFlow and transferServiceProcessingFlow, add VM transport elements.
20. Set the Queue Path of the inbound VM endpoint in splitTransfersFlow to “split”.
21. Set the Queue Path of the outbound VM endpoint in splitTransfersFlow to “process”.
22. Set the Queue Path of the inbound VM endpoint in transferServiceProcessingFlow to “process”.
23. Set the Queue Path of the outbound VM endpoint in transferServiceProcessingFlow to “return”.

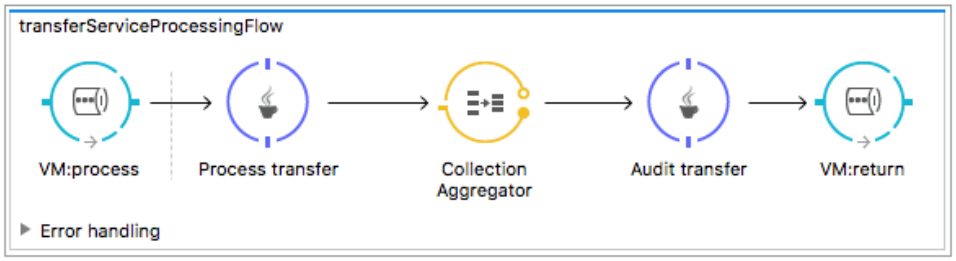


Finishing up

24. Add a Logger component before the Collection Splitter in splitTransfersFlow.
25. Set the log message to

```
#['Processing ' +message.payload.size() +'  transfers']
```

26. Add a Collection Aggregator between the two Java component in transferServiceProcessingFlow.



27. Run the application.

Testing the application

28. Start JMeter.
29. Open the project your created in Walkthrough 2-1.
30. Right-click thread group NF002. Select “Disable”.

31. Right-click thread group NF001. Select “Enable”.
32. Expand thread group NF001.
33. Make sure HTTP Request “Transfer service – 100 transfers” is disabled.
34. Make sure HTTP Request “Transfer service – 2000 transfers” is enabled.
35. Right-click thread group NF001. Select “Start”.
36. Check the results in “View Results in Table”.

Testing the refactored flow

37. Select “Transfer service – 2000 transfers”.
38. Change the Path to “/transferservice2”.
39. Save your changes (CTRL+S / CMD+S).
40. Right-click thread group NF001. Select “Start”.
41. Check the results in “View Results in Table” and compare to the previous test run.

Question 5: How long does it take to process 2000 transfers after refactoring?

Answer: _____

Walkthrough 3-2: Refactoring for reliability

Objectives

In this walkthrough, you will:

- Refactor a Mule application to improve its reliability.
- Use a persisted messaging transport (using ActiveMQ) to reliably exchange message between flows.
- Learn how to develop for zero-message-loss scenarios.

Prerequisites

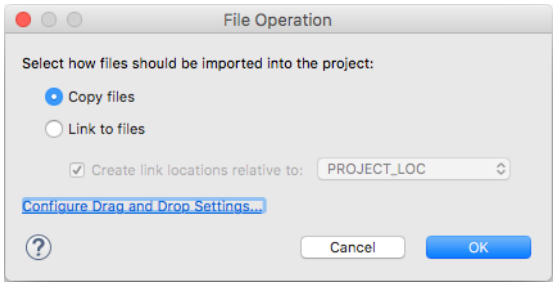
- Having completed Walkthrough 3-1.
- If you have not completed it successfully:
    - Import the solution (AcmeBankingServices-WT3-1.zip) from the student files, folder Solutions/Module03/ in Anypoint Studio.
    - Open the solution (ACME Bank Performance Test Plan.jmx) from the student files, folder Solutions/Module03/ in Apache JMeter.

Create a copy of the BankingServices application

1. Open Anypoint Studio.
2. Make sure the AcmeBankingServices application is not running.
3. Select the project AcmeBankingServices in the Package Explorer.
4. Copy the entire project.
5. Rename to “AcmeBankingServices-reliable”.

Add a driver for ActiveMQ

6. Right-click the project (AcmeBankingServices-reliable), select New > Folder.
7. Name the new folder “lib” and click Finish.
8. From the student files, folder Exercises/Module03, copy file activemq-all-5.13.2.jar to the lib folder. Make sure to select “Copy files” in the dialogue that will be displayed.



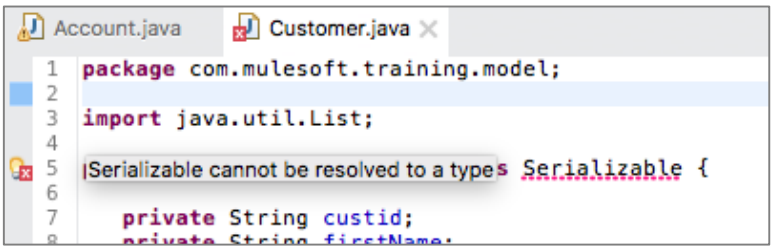
9. Right-click the jar file in the lib folder, select Build Path > Add to Build Path.
10. Expand the Referenced Libraries folder to verify that the jar files was added to the build path.

Refactor Java classes for serialization

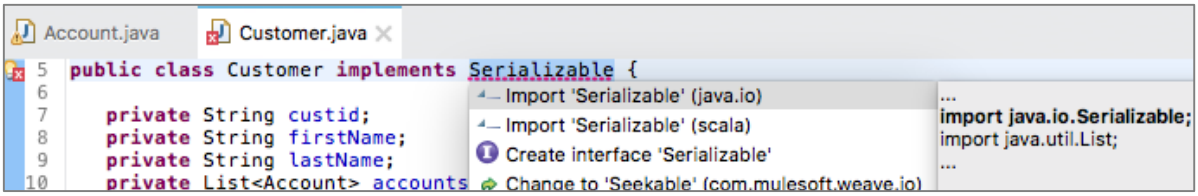
11. Navigate to folder src/main/java and open the package com.mulesoft.training.model.
12. Open ALL Java files inside the model package.
13. Modify the lines, starting with "public class Xxxx" by adding " `implements Serializable`" before the curly bracket.

```
1  package com.mulesoft.training.model;
2
3  import java.io.Serializable;
4  import java.util.List;
5
6  public class Customer implements Serializable {
7
8     private String custid;
9     private String firstName;
10    private String lastName;
11    private List<Account> accounts;
```

14. Look and see if you get the error message "Serializable cannot be resolved to a type".
15. To resolve this error, click the red icon in the left gutter and select "Import 'Serializable' (java.io)".



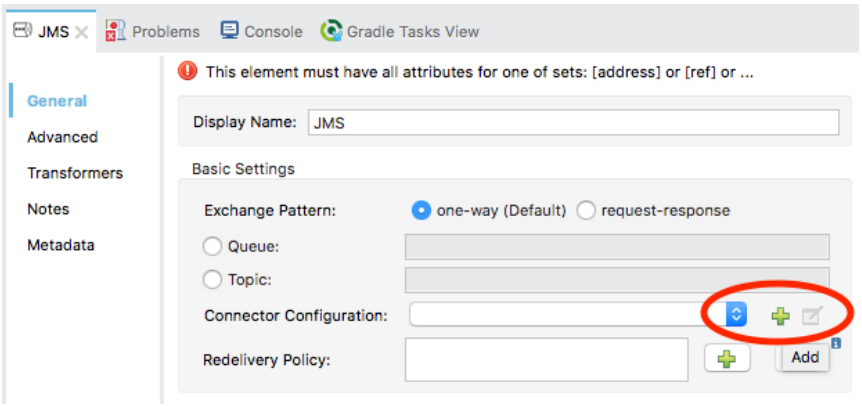
16. Save the Java class file; the error message should disappear.



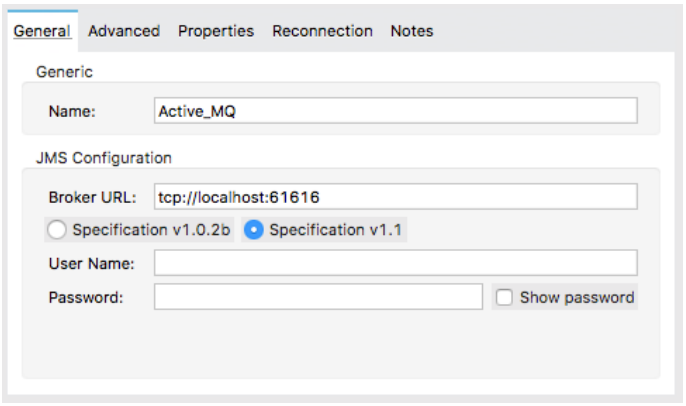
17. Make sure to do this for all Java files in package com.mulesoft.training.model.

Refactor the flows to use ActiveMQ instead of VM queues

18. Open flow configuration file transferservice-refactored.xml.
19. Replace all VM transport elements with JMS connectors.
    a. Delete all VM transport elements
    b. Drag JMS connectors from the Palette to the flow that used to contain VM transport elements.
20. Click one of the JMS endpoints.
21. Create a new ActiveMQ Connector Configuration from the properties view.



22. Select “ActiveMQ” as JMS connector configuration type.
23. Set the following settings:
    a. Name: Active_MQ
    b. Broker URL: tcp://localhost:61616
    c. Specification v1.1
24. Save your changes and apply this Connector Configuration to all JMS endpoints.



25. Set the following Queue names for these JMS endpoints:
    a. transferServiceAquisitionFlow Request-scope endpoint: split
    b. splitTransfersFlow inbound endpoint: split
    c. splitTransfersFlow outbound endpoint: process

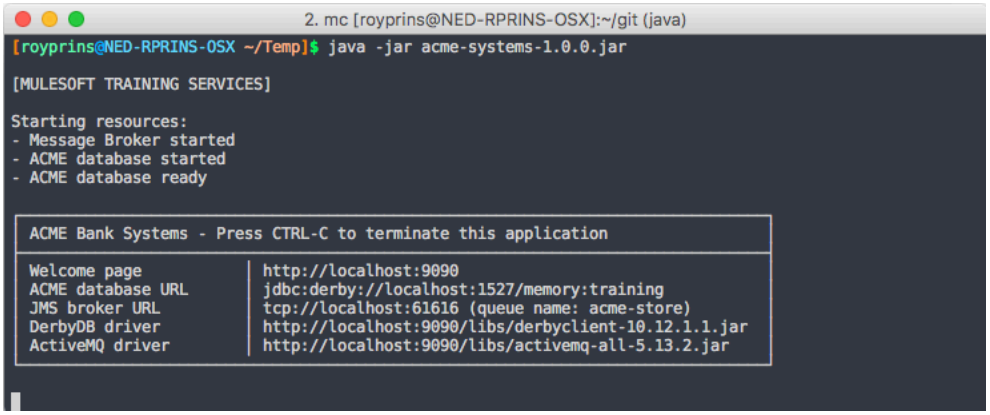
d. transferServiceProcessingFlow inbound endpoint: process
e. transferServiceProcessingFlow outbound endpoint: return
f. transferServiceAquisitionFlow reply-scope: return

26. Add a Byte Array to Object transformer after each JMS inbound endpoint.
27. Add an Object to Byte Array transformer before each JMS outbound endpoint. These transformers are needed for facilitating the serialization of objects to and from the JMS queues.
28. Save your changes. Do not run the application at this point.

Run the application

29. Copy file “acme-systems-1.0.0.jar” from the student files, folder Exercises/Module03 to your local file system.
30. Open a terminal/command window.
31. Navigate to the folder where you copied the acme-systems jar file.
32. Execute the command “java –jar acme-systems-1.0.0.jar”.

    Note: This executable jar file contains an embedded ActiveMQ message broker and Derby database.



33. Return to Anypoint Studio.
34. Run the application.

Test the application using Apache JMeter

35. Open Apache JMeter.
36. Open file “ACME Bank Performance Test Plan.jmx” you created in Walkthrough 3-1.
37. Make sure thread group NF002 is **dis**abled (right-click > Disable).
38. Expand thread group NF001.
39. Make sure HTTP Request “Transfer service – 100 transfers” is **dis**abled.

40. Make sure HTTP Request “Transfer service – 2000 transfers” is **en**abled.
41. Run thread group NF001 (right-click > Start).
42. Observe the console in Anypoint Studio and note how the performance is affected. Compare to results to Walkthrough 3-1.

    Note: When running the thread group multiple times, you may experience caching issues. In real life scenarios, when an actual message broker is used, the mentioned issues will not arise.

Stop the application and close the project in Anypoint Studio

43. Stop the application.
44. Close project “AcmeBankingServices-reliable”.

Module 4: Tuning Performance

Objectives:

- Understand system design considerations and tradeoffs.
- Fine-tuning Mule applications for performance.

Prerequisites

- Having completed the previous walkthroughs.
- If you have not completed it successfully:
    - Import the solution (AcmeBankingServices-WT3-1.zip) from the student files, folder Solutions/Module03/ in Anypoint Studio.
    - Open the solution (ACME Bank Performance Test Plan.jmx) from the student files, folder Solutions/Module03/ in Apache JMeter.

Walkthrough 4-1: Tuning threading profiles

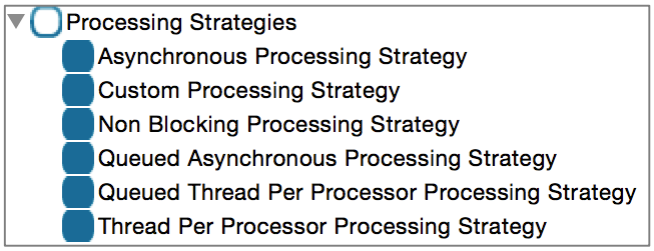
Objectives

In this walkthrough, you will:

- Define a custom threading profile.
- Reference a threading profile from a flow.
- Analyze the profile's impact on the application.

Create a processing strategy

1. Open Anypoint Studio
2. Open project AcmeBankingServices.
3. Open flow configuration file global.xml.
4. Select the Global Elements tab.
5. Click Create.
6. Expand Processing Strategies.



7. Choose Queued Asynchronous Processing Strategy.
8. Name the strategy “highThroughputProcessing”.
9. Set the following threading settings:
    - Max Threads: 80
    - Min Threads: 16
    - Thread TTL: 10000

```
<queued-asynchronous-processing-strategy name="highThroughputProcessing"
    maxThreads="100" minThreads="16" threadTTL="10000" />
```

10. Click OK.

Reference the strategy in your application

11. Select the transferServiceProcessingFlow.
12. Select Processing Strategy Ref.
13. From the drop-down list, select highThroughputProcessing.
14. Save your changes
15. Run the application.

Test the tuned application

16. Return to JMeter.
17. Open file “ACME Bank Performance Test Plan.jmx” your created in Walkthrough 3-2.
18. Make sure thread group NF001 is **en**abled (right-click > Enable).
19. Make sure thread group NF002 is **dis**abled (right-click > Disable).
20. Expand thread group NF001.
21. Make sure HTTP request “Transfer service – 100 transfers” is **dis**abled.
22. Make sure HTTP request “Transfer service – 2000 transfers” is **en**abled.
23. Right-click thread group NF001. Select “Start”.
24. Check the results in “View Results in Table” and compare to the previous test run.

Question 6: How long does it take to process 2000 transfers after tuning? Does the transfer service meet its non-functional requirements? How can you tell?

Answer: _____

Stop the application

25. Return to Anypoint Studio and stop the application.

Walkthrough 4-2: Optimizing HTTP for high concurrency

Objectives

In this walkthrough, you will:

- Configure HTTP connectors for high volumes/concurrency

Prerequisites

- Having completed Walkthrough 4-1.
- If you have not completed it successfully:
    - Import the solution (AcmeBankingServices-WT4-1.zip) from the student files, folder Solutions/Module04/ in Anypoint Studio.
    - Open the solution (ACME Bank Performance Test Plan.jmx) from the student files, folder Solutions/Module03/ in Apache JMeter.

Tune HTTP connector configuration

1. Open Anypoint Studio.
2. Open project AcmeBankingServices.
3. Open flow configuration file global.xml.
4. Select tab Global Elements.
5. Select the existing HTTP Listener Configuration.
6. Click “Edit”.
7. Select “Use custom worker threading profile”.
8. Set the following settings:
    a. Max Active Threads: 200
    b. Max Idle Threads: 16
9. Click “OK”.
10. Save your changes.

Start the tuned application

11. Run the application.

Test the impact of the changes

12. Start VisualVM.

13. Start Apache JMeter.
14. Open file “ACME Bank Performance Test Plan.jmx” your created in Walkthrough 4-1.
15. Make sure thread group NF001 is **dis**abled (right-click > Disable).
16. Make sure thread group NF002 is **en**abled (right-click > Enable).
17. Right-click thread group NF002 and select “Start”.
18. Return to VisualVM.
19. Observe the Threads tab.

Question 7: Does the customer service meet its non-functional requirements? How do you know? Answer: _____

Stop all running applications

20. Stop Apache JMeter.
21. Stop VisualVM.
22. Return to Anypoint Studio.
23. Stop the running application.