



PART 2: Building Applications with Anypoint Studio

Goal

The screenshot displays the MuleSoft Anypoint Studio interface. The main workspace shows a Mule flow diagram with the following components: HTTP, Set airline variable, setCodeSubflow, Choice, getUnitedFlightsFlow, Validation, Transform Message, and Logger. The 'Choice' component has two paths: one for 'getUnitedFlightsFlow' and another for 'getAmericanFlightFlow'. The 'Set airline variable' component is highlighted with a red box. Below the main flow, an 'Error handling' section shows a 'getAirlineFlightsFlow' sub-flow with components: Scatter-Gather, getDeltaFlightsFlow, Filter Reference, getUnitedFlightsFlow, Filter Reference, Transform Message, and Logger. The bottom of the screen features a 'Mule Debugger' window with two tabs: 'Variables' and 'Record'. The 'Variables' tab is active, showing a table of variables.

Name	Value	Type
cache-control	no-cache	java.lang.String
connection	keep-alive	java.lang.String
host	localhost:8081	java.lang.String
http.listener.path	/flights	java.lang.String
http.method	POST	java.lang.String

Objectives




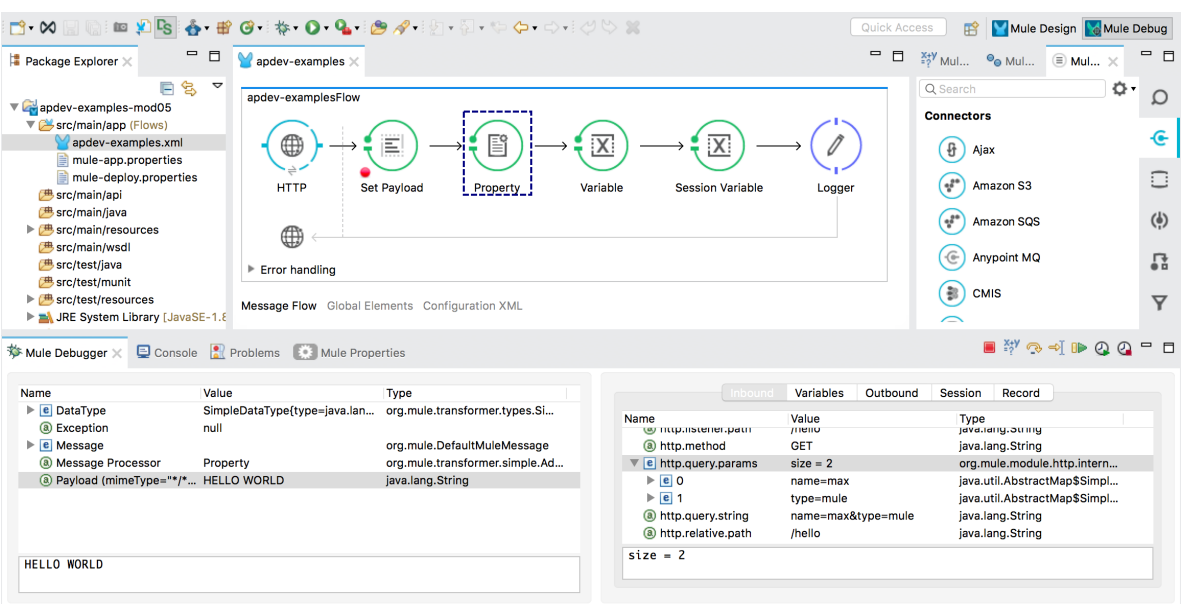
- Debug Mule applications
- Read and write message payloads, properties, and variables using the Mule Expression Language
- Structure Mule applications using flows, subflows, in-memory message queues, properties files, and configuration files
- Connect to web services, SaaS applications, files, polled resources, JMS queues, and more
- Route, filter, and validate messages and handle message exceptions
- Write DataWeave expressions for more complicated transformations
- Process individual records in a collection and synchronize data in databases to SaaS applications

All contents © MuleSoft Inc.



Module 5: Accessing and Modifying Mule Messages

Goal




The screenshot displays the MuleSoft IDE interface. The top section shows a message flow diagram for 'apdev-examplesFlow' with steps: HTTP, Set Payload, Property (highlighted with a dashed box), Variable, Session Variable, and Logger. The bottom section shows the Mule Debugger console with a table of message properties.

Name	Value	Type
DataType	SimpleDataType{type=java.lan...	org.mule.transformer.types.Si...
Exception	null	
Message		org.mule.DefaultMuleMessage
Message Processor	Property	org.mule.transformer.simple.Ad...
Payload (mimeType="*/..."	HELLO WORLD	java.lang.String

The console also shows the raw message content: HELLO WORLD.

Objectives



- Log message data
- Debug Mule applications
- Read and write message properties
- Write expressions with Mule Expression Language (MEL)
- Create variables

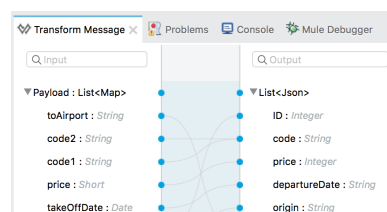
Accessing information about Mule messages

View message info using DataSense

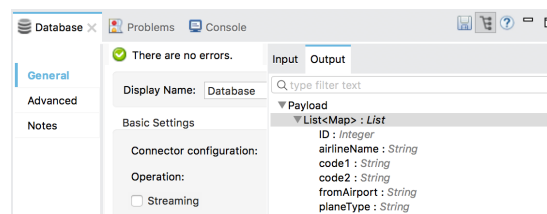


- **DataSense** is Anypoint Studio's ability to proactively discover metadata from internal and external resources

- Keeps you from having to manually discover information about the data
- Facilitates transformations by providing DataWeave expected input or output
 - We saw this with Transform Message component



- There is also a DataSense Explorer in the Properties view
 - Lets you see message data structure throughout a flow at design time



Other ways to view message information



- Add a Logger component to a flow and view its output in the Anypoint Studio console
- Use the Anypoint Studio Visual Debugger
 - Most comprehensive way
 - Also has a Mule Expression Evaluator
- Use autocomplete when writing expressions in the Anypoint Studio Visual Editor



Logger

Name	Value	Type
▶ DataType	SimpleDataType(type=...	org.mule.transformer.t...
▶ Exception	null	
▶ Message		org.mule.DefaultMule...
▶ Message Proces...	Logger	org.mule.api.processor...
▶ Payload (mimeT...	Hello	java.lang.String

```
#[message.inboundProperties.
```

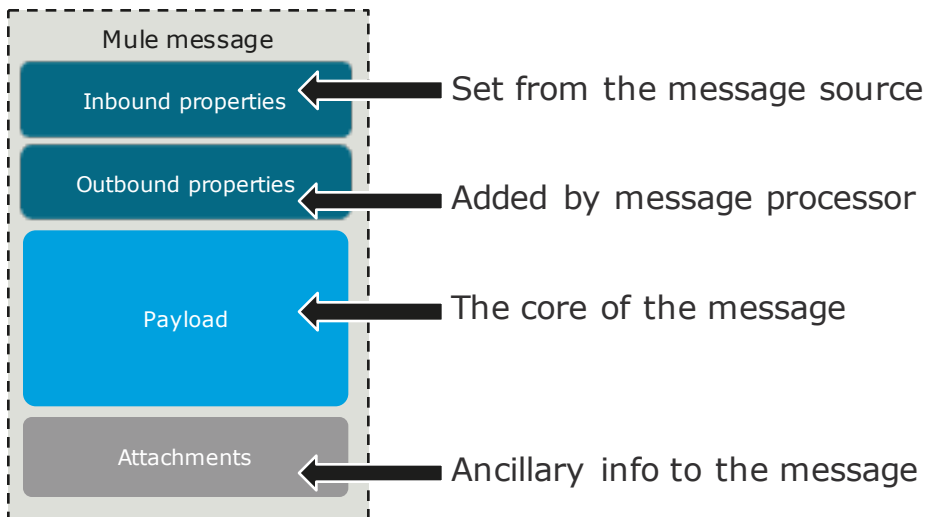
- http.listener.path
- http.method
- http.query.params**
- http.query.string
- http.remote.address
- http.request.path

All contents © MuleSoft Inc.

9

Setting message data

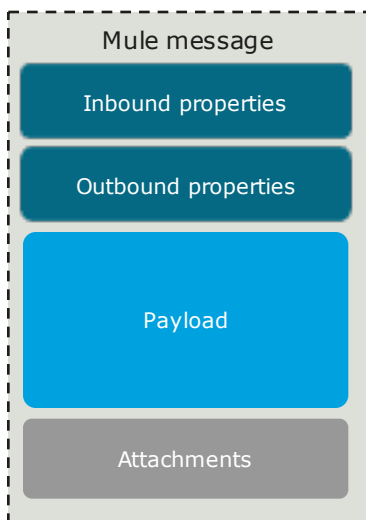
Reviewing the structure of Mule messages



All contents © MuleSoft Inc.

11

Message properties

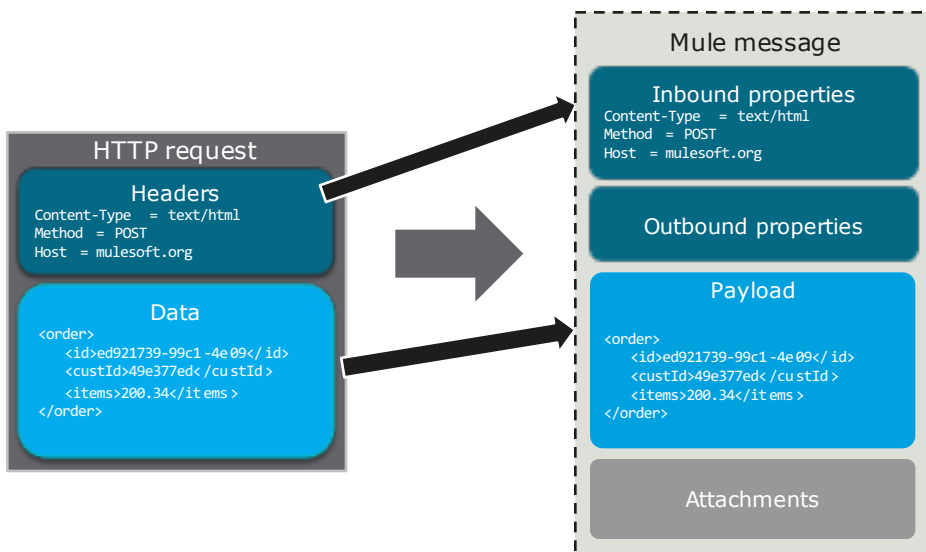


- Inbound properties
 - Set from the message source
 - Read-only access
 - Persist throughout the flow
- Outbound properties
 - Added by message processor
 - Read/write access
 - Can set, remove, copy

All contents © MuleSoft Inc.

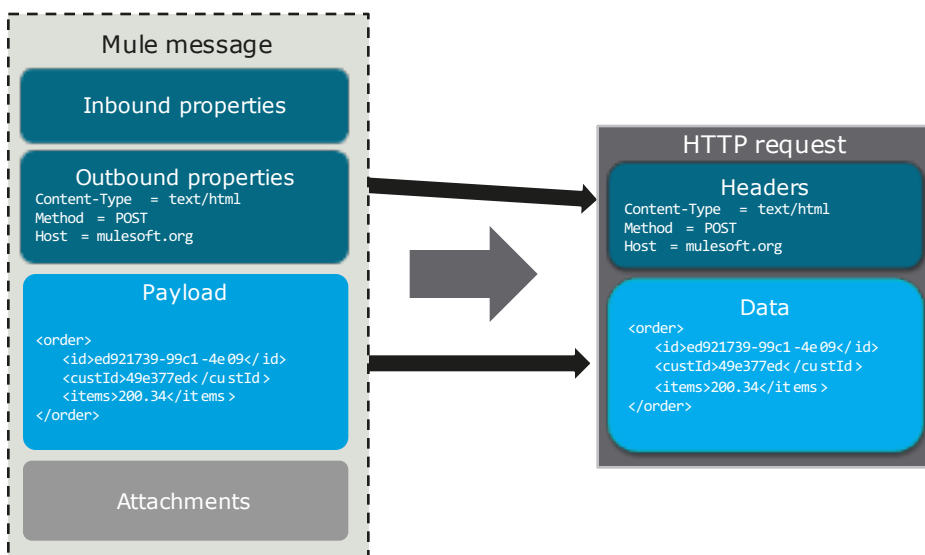
38

Inbound message properties



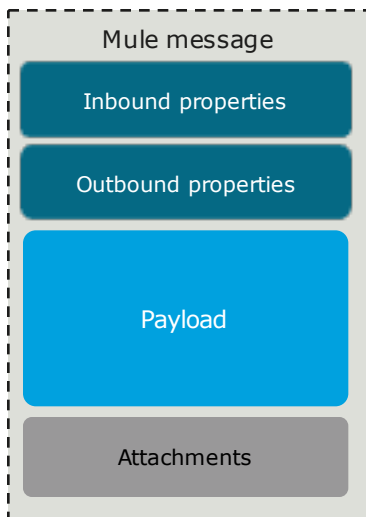
41

Outbound message properties



42

Message payload and attachments



- **Payload**
 - The core of the message
 - Contains primary info to be processed
 - Contains a Java Object
- **Attachments**
 - Ancillary info to the message
 - Similar to an email attachment

All contents © MuleSoft Inc.

39

Payload representation



Payload

```
<order>
  <id>ed921739-99c1-4e09</id>
  <custId>49e377ed</custId>
  <items>200.34</items>
</order>
```

org.java.Lang.String

- **Raw data often of type**
 - String
 - InputStream
 - Byte[] (Byte array)

Payload

```
id: ed921739-99c1-4e09
custId: 49e377ed-bc72-4523
itemsTotal: 200.34
```

org.java.util.HashMap

- **Structured data often of type**
 - Map
 - Structured Java object
 - Order, Account, etc.

All contents © MuleSoft Inc.

40

Setting message properties



Set Payload

- Sets the value of the message payload
 - message.payload



Property

- Sets, removes, or copies properties on the outbound scope of a message
 - message.outboundProperties

- These are transformers in the Mule Palette in Studio

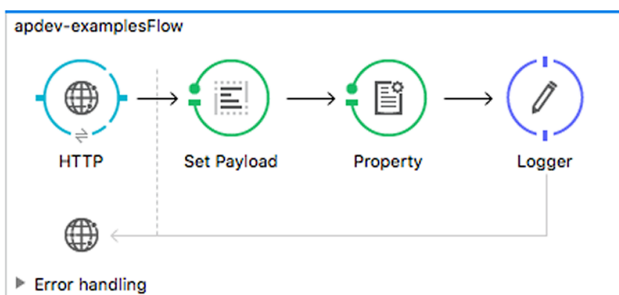
All contents © MuleSoft Inc.

43

Walkthrough 5-1: Set and log message data



- Create a new project
- Set the message payload
- Set message outbound properties
- Log the message to the console



```

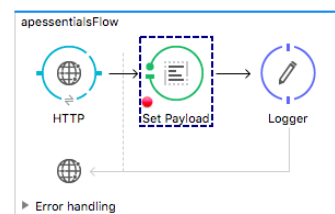
http.request.path=/hello
http.request.uri=/hello
http.scheme=http
http.uri.params=ParameterMap{[]}
http.version=HTTP/1.1
postman-token=675b8e19-012d-f66e-f796-aa3f
user-agent=Mozilla/5.0 (Macintosh; Intel M
OUTBOUND scoped properties:
  qpname=max
SESSION scoped properties:
  }
  
```

Debugging Mule applications

Debugging applications with the Mule Debugger



- Can add breakpoints to processors and step through the application
 - Watch message and variable values
 - Watch and evaluate expressions
- By default, Debugger listens for incoming TCP connections on localhost port 6666
 - Can change this in a project's run configuration



Mule Debugger interface showing the message details and inbound variables.

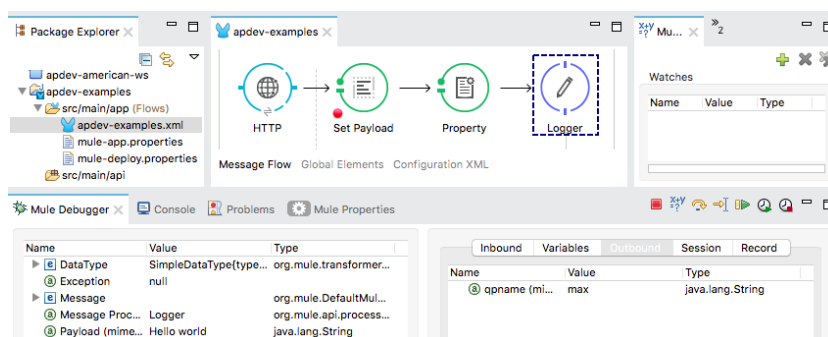
Name	Value	Type
▶ DataType	SimpleDataType{type=...	org.mule.transformer.t...
▶ Exception	null	
▶ Message		org.mule.DefaultMule...
▶ Message Proces...	Logger	org.mule.api.processor...
▶ Payload (mimeT...	Hello	java.lang.String

Inbound		
Name	Value	Type
host	localhost:8081	java.lang.String
http.listener.path	/	java.lang.String
http.method	POST	java.lang.String
http.query.params	size = 0	org.mule.module.htt...

Walkthrough 5-2: Debug a Mule application



- Locate the port used by the Mule Debugger
- Add a breakpoint, debug an application, and step through the code
- Use the Mule Debugger to view message properties
- Pass query parameters to a request and locate them in the Debugger



35

Using expressions to read and write message data

The Mule Expression Language (MEL)



- Use MEL to access and evaluate the data in the payload, properties, and variables of a Mule message
- MEL is a lightweight, Mule-specific expression language
- Accessible and usable from within virtually every message processor in Mule
 - Is used to modify the way the processors act upon the message such as routing or filtering
- Makes use of Mule-specific context objects
- Case-sensitive
- Easy to use with autocomplete everywhere

All contents © MuleSoft Inc.

45

Basic MEL syntax



[] Encapsulates all Mule expressions

[message] Holds a context object

[message.payload] Dot notation to access fields or methods

24 All contents © MuleSoft Inc.

46

Context objects



server

Operating system that message processor is running

mule

The Mule instance that the application is running

application

User application the current flow is deployed in

message

The Mule message that the message processor is processing

Accessing message data



Mule message

Inbound properties

Method = POST
Host = mulesoft.org

```
#[message.inboundProperties.host]
mulesoft.org
```

Outbound properties

Content-Type = text/html
Method = POST
Host = mulesoft.org

```
#[message.inboundProperties['http.method']]
POST
```

Payload

id: ed921739-99c1-4e09
custId: 49e377ed-bc72-4523
itemsTotal: 200.34

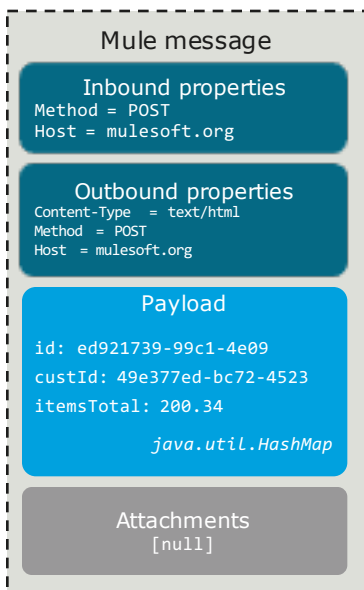
java.util.HashMap

```
#[message.outboundProperties['content-type']]
text/html
```

Attachments

[null]

Accessing message payload data



```
#[message.payload.id]
```

```
#[message.payload['id']]
```

```
ed921739-99c1-4e09
```

```
#[message.payload.itemsTotal]
```

```
200.34
```

```
#[message.payload.toString()]
```

```
#[payload.id] is a shortcut for #[message.payload]
```

This shortcut only works with payload

49

Accessing relational map data



FirstName	LastName	City	State
John	Muley	Boston	Ohio
Mark	Dailer	Cleveland	Ohio
Bill	Muley	Avon	Ohio

```
#[message.payload[1]['LastName']]
```

```
Dailer
```

```
#[message.payload[0].City]
```

```
Boston
```

All content © MuleSoft, Inc.

50

Accessing relational map data



- Operators

- Arithmetic: +, -, /, *, %
- Evaluation: ==, !=, >, <, >=, <=, contains, is
#[message.inboundProperties.'http.query.params'.lastname != null]

- Testing for emptiness

- The literal **empty** tests the emptiness of a value
 - Null, boolean false, "", " ", zero, empty collections

- Data extraction

- XPath: #[xpath('expression')]
- RegEx: #[regex('expression')]

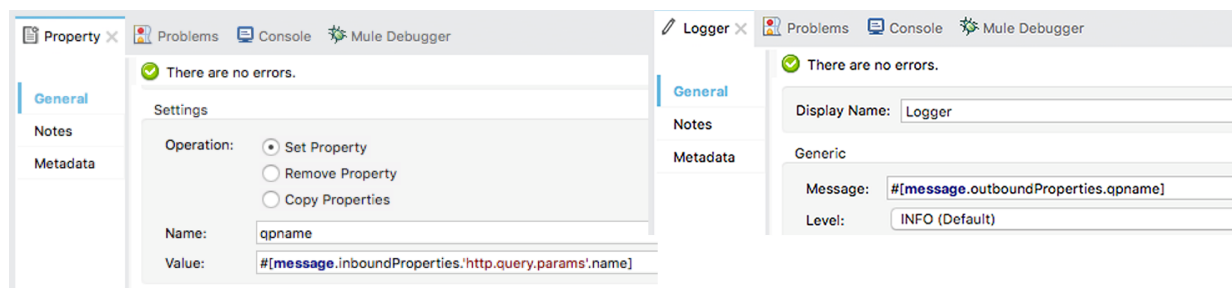
All contents © MuleSoft Inc.

51

Walkthrough 5-3: Read and write message properties using MEL expressions



- Use an expression to set the payload
- Use an expression to display specific info to the console.
- Use an expression to set an outbound property
- Use an expression to read an outbound property



MEL references



- MEL expression reference
 - <https://docs.mulesoft.com/mule-user-guide/v/3.8/mule-expression-language-reference>
- MEL language tips
 - <https://docs.mulesoft.com/mule-user-guide/v/3.8/mule-expression-language-tips>

▼ Mule Expression Language MEL

MEL Cheat Sheet

Mule Expression Language Basic Syntax

Mule Expression Language Examples

▼ Mule Expression Language Reference

Mule Expression Language Date and Time Functions

MEL DataWeave Functions

Mule Expression Language Tips

All contents © MuleSoft Inc.

53

Creating variables



Context variables



flowVars

sessionVars

recordVars

`#[flowVars.ticketNum]`

All contents © MuleSoft Inc.

55

Setting variables



Variable

- Sets or removes **flow variables**
 - Variables on the message tied to the current
 - Reference as flowVars
 - The flowVars reference is optional
 - `#[flowVars.foo]` or `#[foo]`



Session Variable

- Sets or removes **session variables**
 - Variables tied to a message for its lifecycle across flows, applications, and servers
 - They are persisted across some but **not all** transport barriers
 - Reference as sessionVars
 - `#[sessionVars.foo]`

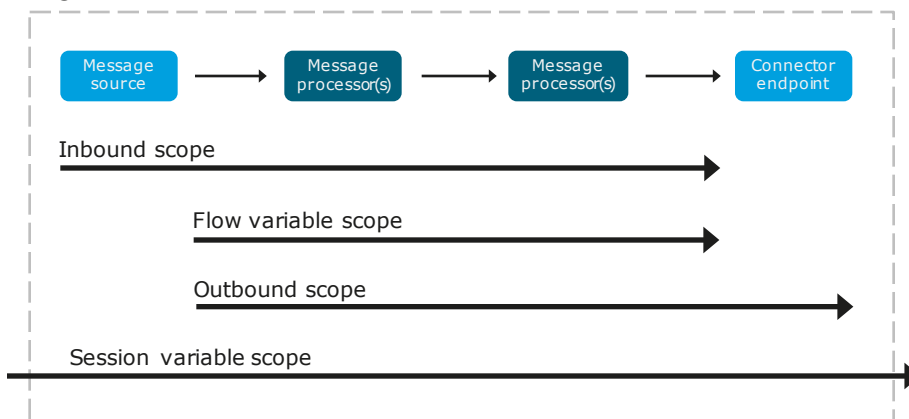
All contents © MuleSoft Inc.

56

Variable persistence



Flow



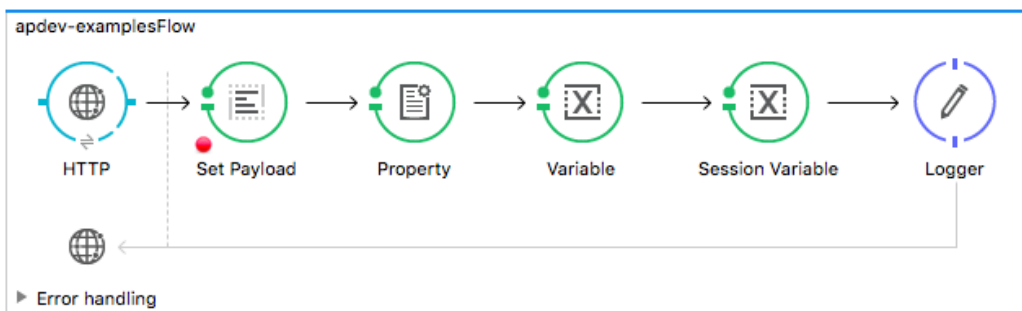
All contents © MuleSoft Inc.

57

Walkthrough 5-4: Read and write variables



- Use the Variable transformer to create a flow variable
- Use the Session transformer to create a session variable
- Use the Mule Debugger to see their values



All contents © MuleSoft Inc.

58

Summary



Summary



- The best way to view message data is to add breakpoints to a flow and use the Mule Debugger
- Use the Set Payload transformer to set the payload
- Use the Property transformer to set, remove, or copy message outbound properties
- Use the Logger component to display data in the console
- Use the Mule Expression Language (MEL) to write expressions `#[]`
- Use the Variable transformer to create flow variables