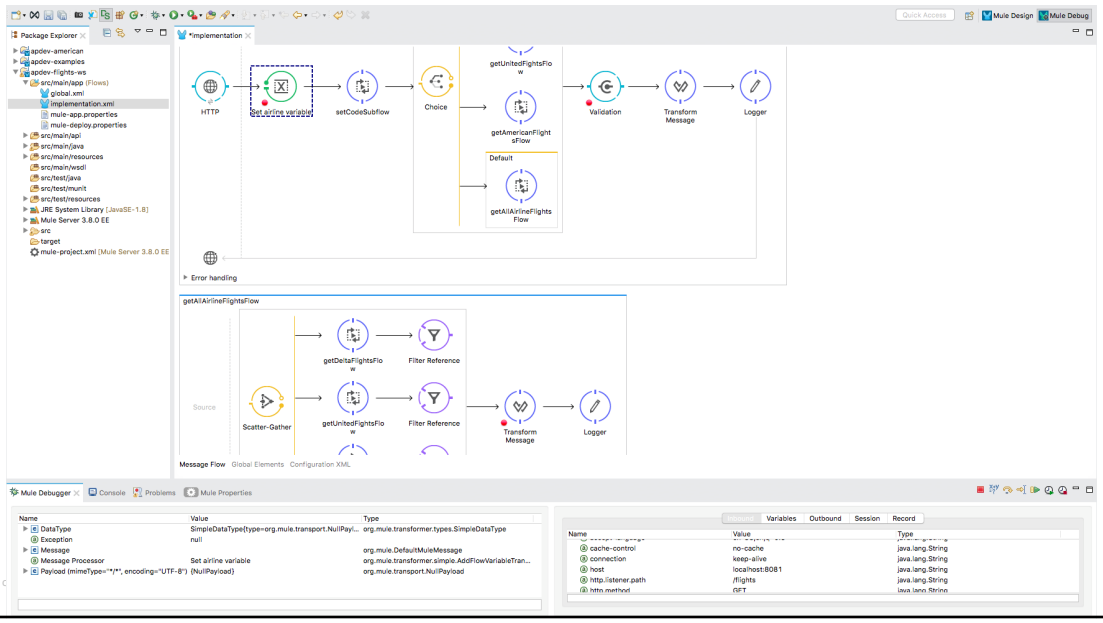
MuleSoft

PART 2: Building Applications with Anypoint Studio

Goal

MuleSoft

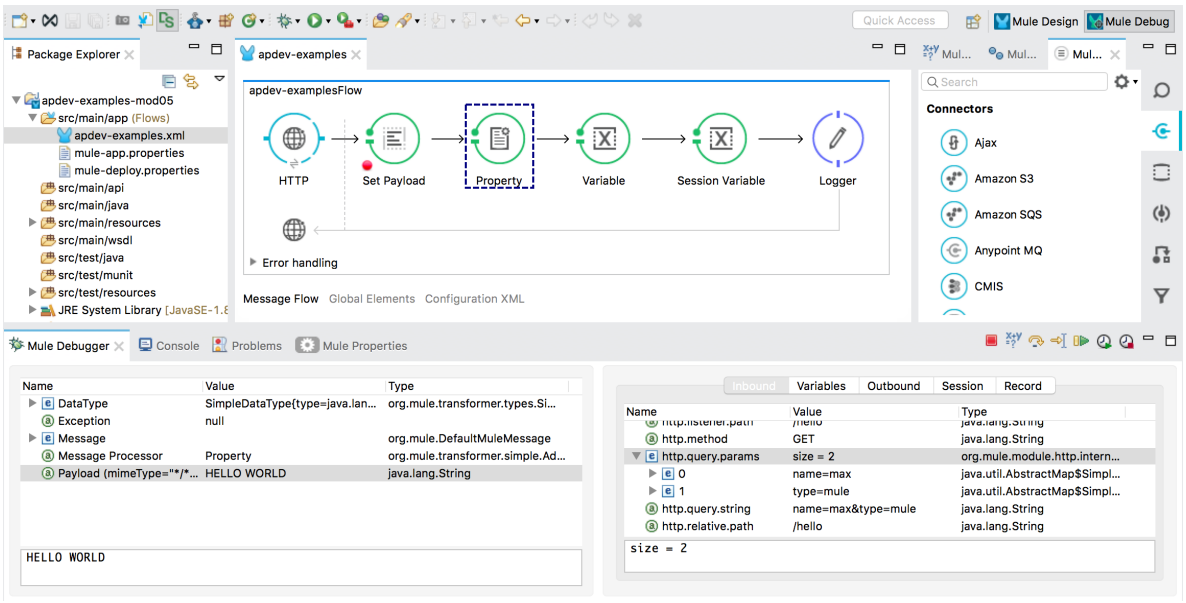


Objectives

- 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

Module 5: Accessing and Modifying Mule Messages

Goal



Objectives

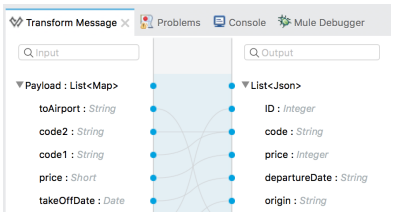
- Log message data
- Debug Mule applications
- Read and write message properties
- Write expressions with Mule Expression Language (MEL)
- Create variables

Accessing information about Mule messages

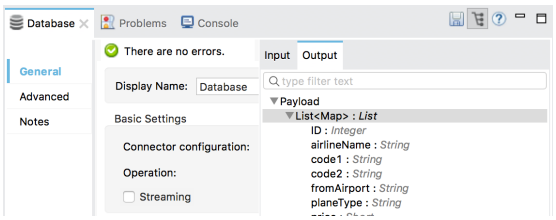
View message info using DataSense

MuleSoft

- **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

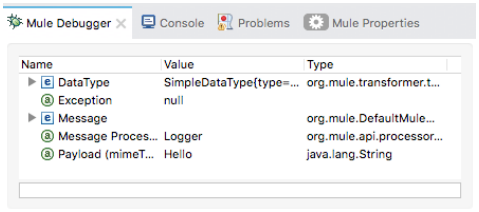


Other ways to view message information

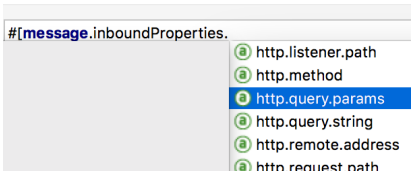
MuleSoft

- 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

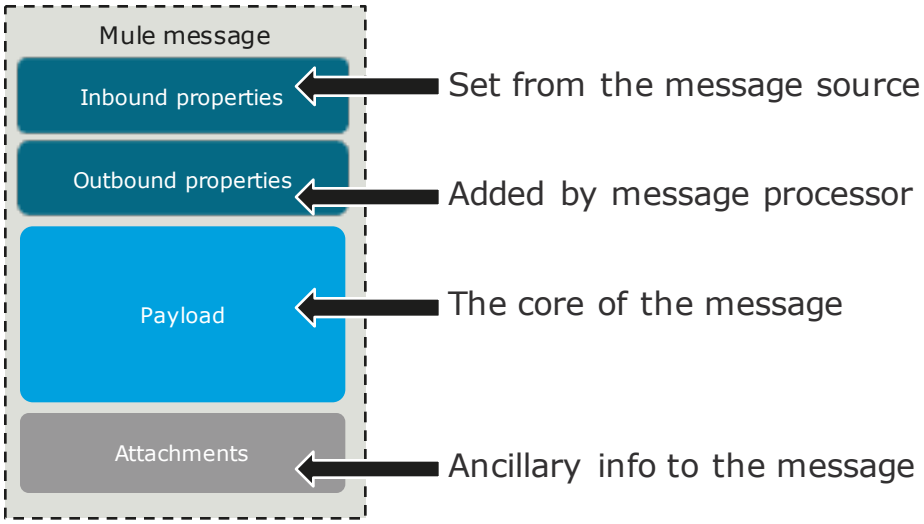


- Use autocomplete when writing expressions in the Anypoint Studio Visual Editor

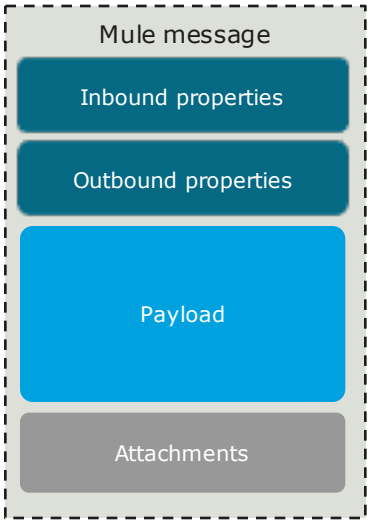


Setting message data

Reviewing the structure of Mule messages

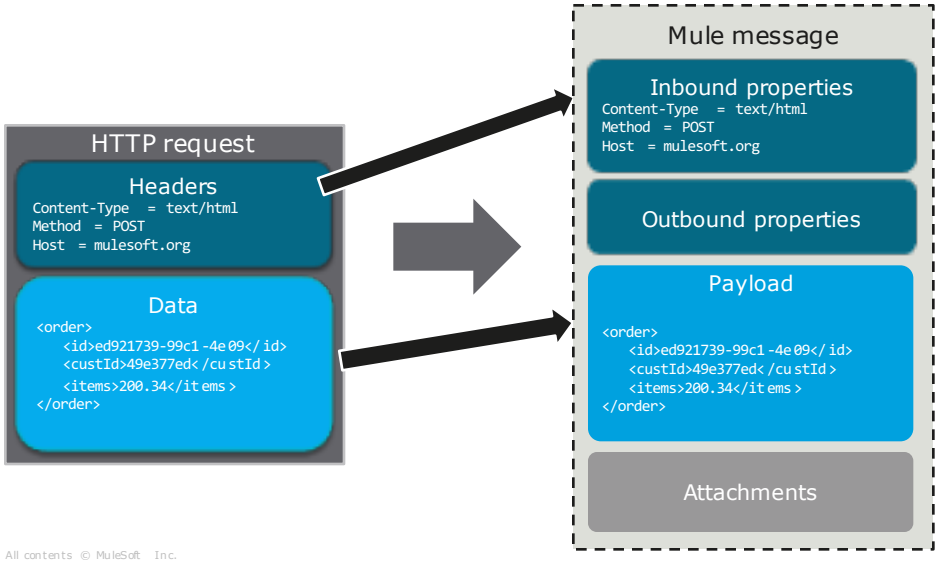


Message properties

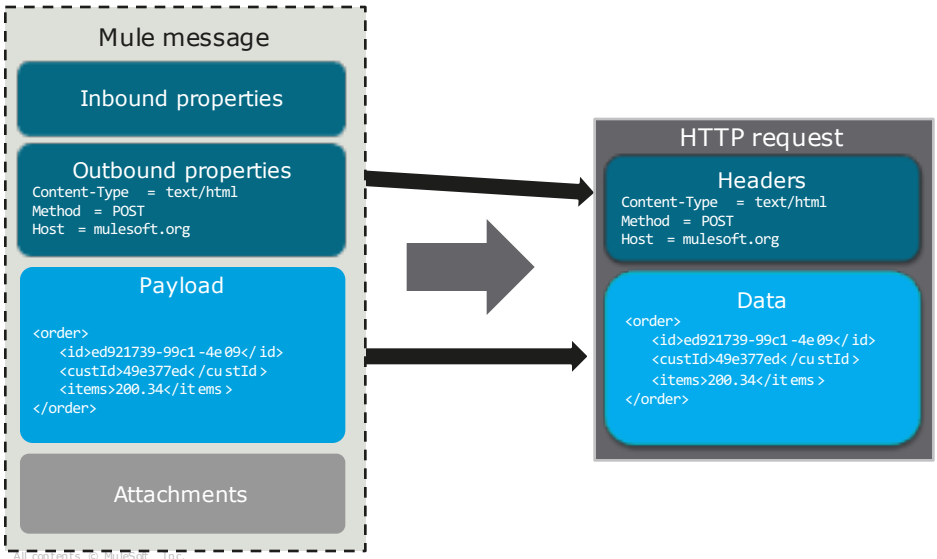


- Inbound properties
  - Set from the message source
  - Read-only access
  - Persist throughout the flow
- Outbound properties
  - Added by message processor
  - Read/write access
  - Can set, remove, copy

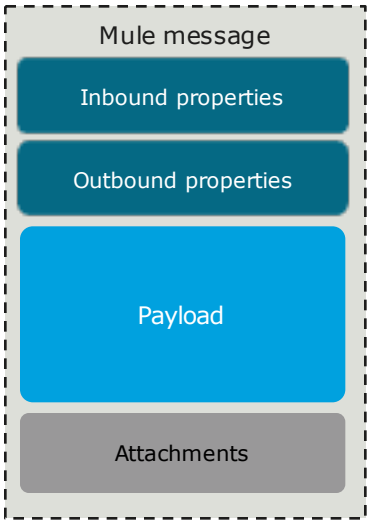
Inbound message properties



Outbound message properties



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

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.

Setting message properties

MuleSoft

Set Payload

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

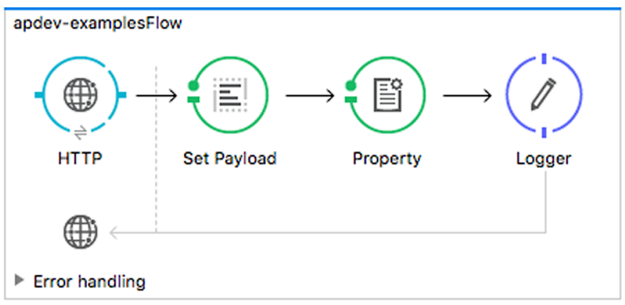
Property

- Sets, removes, or copies properties on the outbound scope of a message
  - message.outboundProperties
- These are transformers in the Mule Palette in Studio

Walkthrough 5-1: Set and log message data

MuleSoft

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



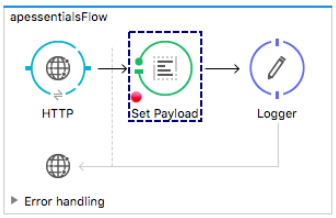

```
  http.request.path=/hello
  http.request.uri=/hello
  http.scheme=http
  http.uri.params=ParameterMap{[]}
  http.version=HTTP/1.1
  postman-token=675b8e19-012d-f66e-f796-aa3f
  user-agent=Mozilla/5.0 (Macintosh; Intel M
OUTBOUND scoped properties:
  qpname=max
SESSION scoped properties:
}
```

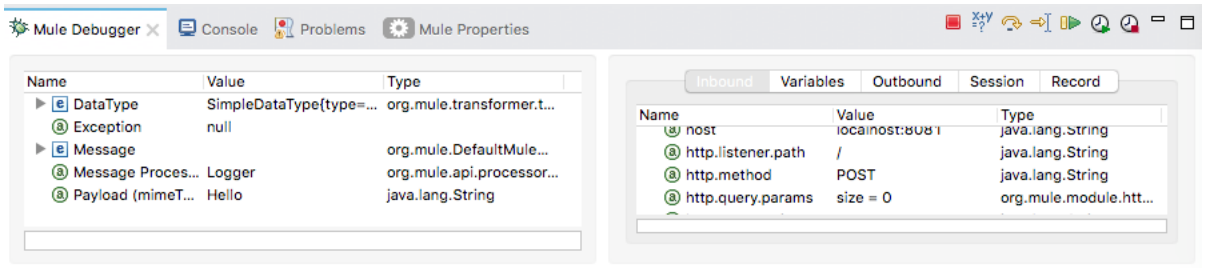
Debugging Mule applications

Debugging applications with the Mule Debugger

MuleSoft

- 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

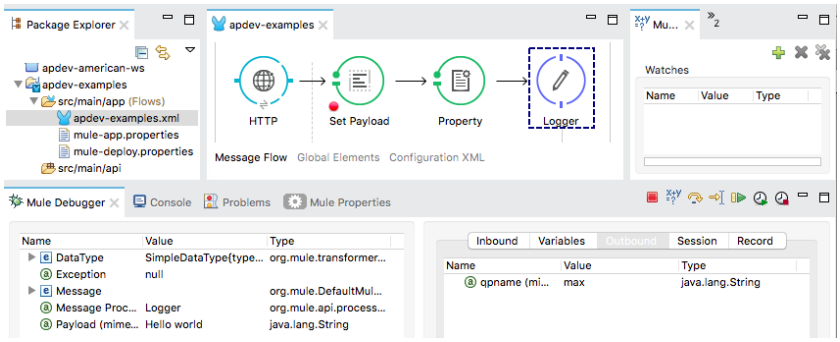




Walkthrough 5-2: Debug a Mule application

MuleSoft

- 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)

MuleSoft

- 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

Basic MEL syntax

MuleSoft

`#[]` Encapsulates all Mule expressions

`#[message]` Holds a context object

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

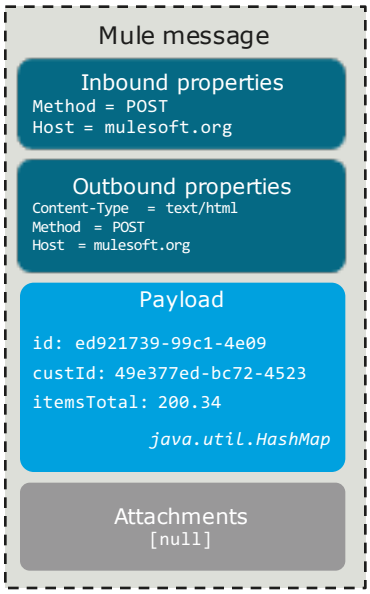
Context objects

MuleSoft

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 processer is processing

Accessing message data

MuleSoft




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

mulesoft.org

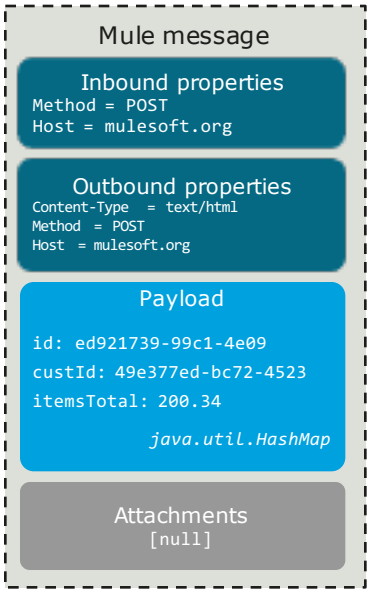
```
#[message.inboundProperties['http.method']]
```

POST

```
#[message.outboundProperties['content-type']]
```

text/html

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

50

Accessing relational map data

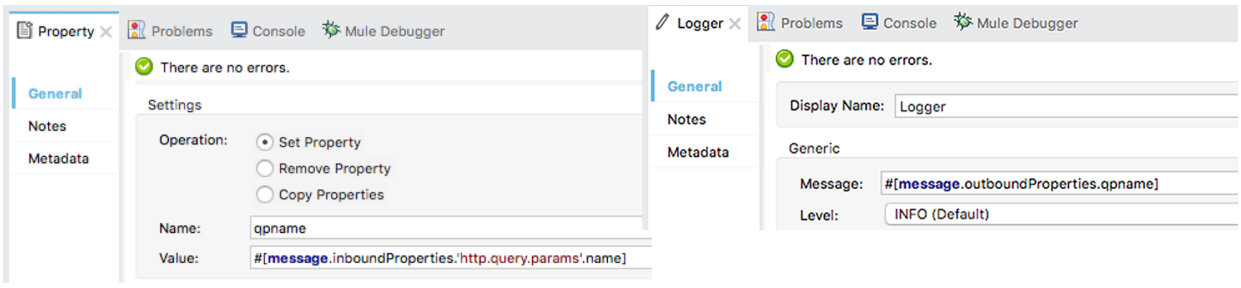
MuleSoft

- 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')]

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

MuleSoft

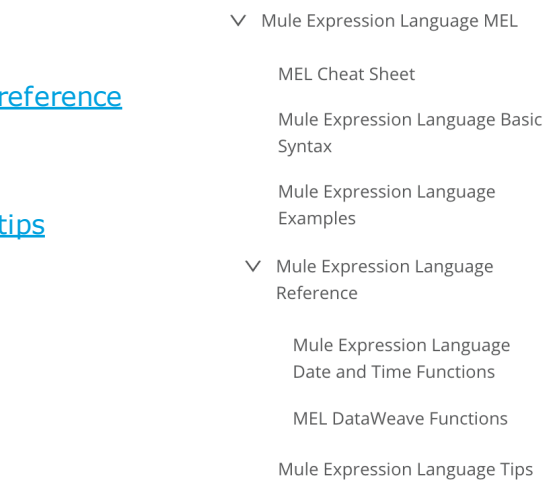
- Use an expression to set the payload
- Use an expression to display specific info to the console.
- Use an expression to set an outbound property
- Use an expression to read an outbound property



MEL references

MuleSoft

- 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



Creating variables

Context variables

MuleSoft

flowVars

sessionVars

recordVars

#[flowVars.ticketNum]

Setting variables

MuleSoft

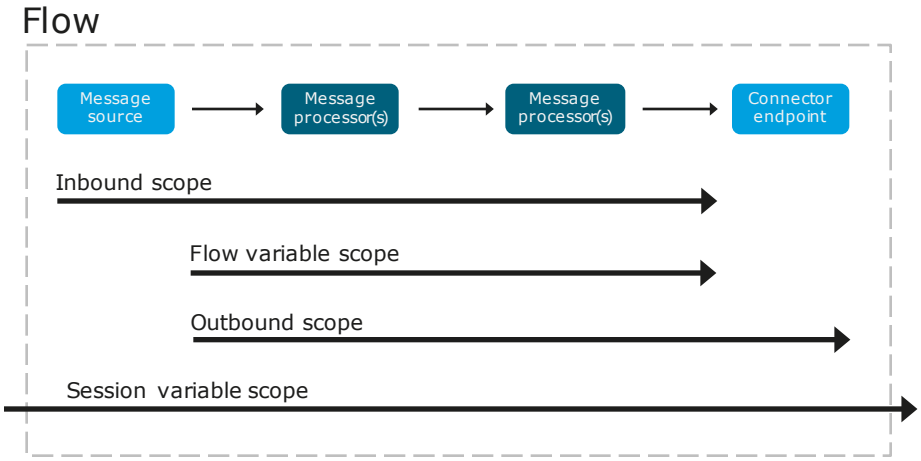
Variable

- 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

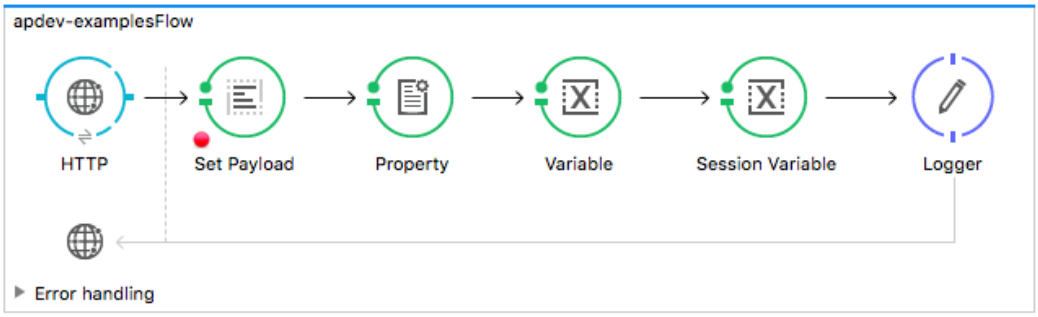
- Sets or removes **session variables**
  - Variables tied to a message for its lifecycle across flows, applications, and servers
  - They are persisted across some but **not all** transport barriers
  - Reference as sessionVars
    - #[sessionVars.foobar]

Variable persistence



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



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