



Module 2: Building Integration Applications with Anypoint Studio

Goal

The screenshot displays the Anypoint Studio interface. The main canvas shows a Mule flow with the following components: HTTP (Inbound), Set Payload, Property, Variable, Logger, and HTTP (Outbound). The flow is titled 'apressentialsFlow'. The Package Explorer on the left shows the project structure, including 'apressentials', 'src/main/app', 'src/main/resources', and 'target'. The Mule Debugger at the bottom shows the current message with the following details:

Name	Value	Type
Message		org.mule.DefaultMuleMessage
Session Variable	HELLO WORLD	java.lang.String
payload		

A browser window in the foreground shows the URL 'localhost:8081/?name=mule' and the response 'HELLO WORLD'.

Objectives

- In this module, you will learn:
 - About Mule applications, flows, messages, and message processors
 - To use Anypoint Studio to create flows graphically using connectors, transformers, components, scopes, and flow control elements
 - To build, run, test, and debug Mule applications
 - To read and write message properties
 - To write expressions with Mule Expression Language (MEL)
 - To create variables

3 | All contents Copyright © 2015, MuleSoft Inc.



Introducing Mule applications

4 | All contents Copyright © 2015, MuleSoft Inc.

Mule applications

- Accept and process **messages** through a series of **message processors** plugged together in a **flow**
 - A message can be initiated by an events like
 - A consumer request from a mobile device
 - A change to data in a database
 - The creation of a new customer ID in a SaaS application

5 | All contents Copyright © 2015, MuleSoft Inc.



Mule applications

- Are written in XML (primarily)
 - Under the hood, are Java applications using Spring
- Can be created and tested visually with Anypoint Studio
 - Available as a stand-alone or as an Eclipse plug-in
- Are deployed to a Mule Server Runtime
 - A standalone app to a Mule runtime (typically)
 - A WAR file with an embedded Mule instance to an app server

6 | All contents Copyright © 2015, MuleSoft Inc.



Mule Server Runtime

- A JVM server that
 - Can handle many concurrent requests for different Java (Mule) applications in a single JVM
 - Decouples point-to-point integrations by having all (non-Mule) applications talk to the bus (to a Mule app) instead of directly to each other
 - Decouples applications by using
 - Protocol adaption
 - Communication can be over different protocols, like HTTP or FTP
 - A canonical data format
 - A common format all messages are transformed to

7 | All contents Copyright © 2015, MuleSoft Inc.



Mule runtime editions

- Enterprise and community editions
 - Mule ESB Server runtime EE
 - Mule ESB Server runtime CE
- CE is open-source
- EE is a hardened code line with support and additional capabilities
- By default, Anypoint Studio uses EE
 - You can install other versions and select which one to use
- <http://www.mulesoft.com/platform/soa/mule-esb-enterprise>

8 | All contents Copyright © 2015, MuleSoft Inc.



Mule runtime EE

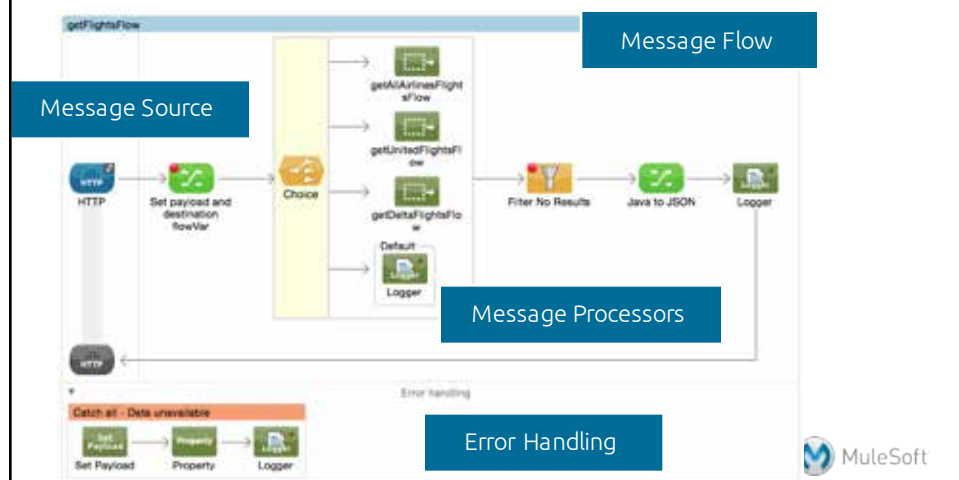
- 24/7 global support
- Additional connectors
- ✧ Visual debugging
- ✧ DataMapper and DataSense
- ✧ Batch module
- Caching and transaction support
- Performance monitoring
- Security module
- Templates
- ✧ Deployment and performance management

9 | All contents Copyright © 2015, MuleSoft Inc.



Mule applications and flows

- Mule applications accept and process **messages** through a series of **message processors** plugged together in a **flow**



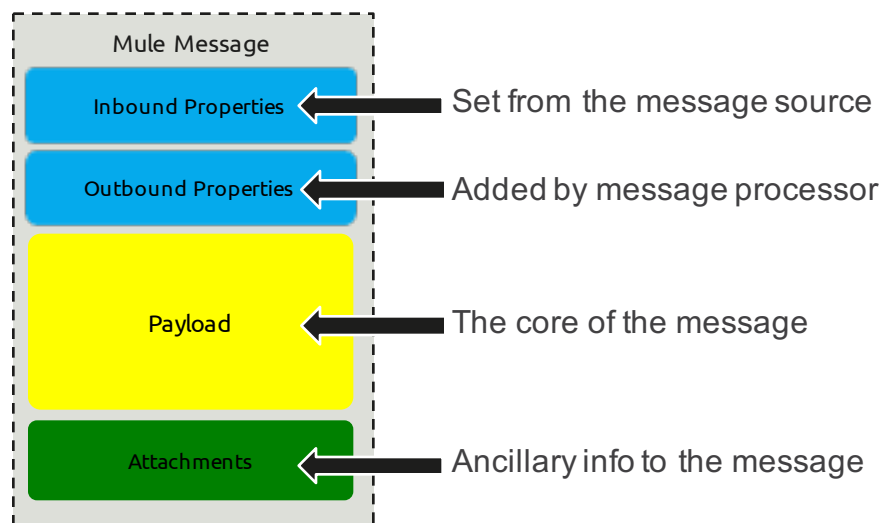
Mule flows

- A typical flow has
 - A message source
 - Accepts a message from an external source triggering the execution of the flow
 - Message processors
 - Transform, filter, enrich, and process the message
- An application can consist of
 - A single flow
 - Multiple flows
 - Multiple flows connected together

11 | All contents Copyright © 2015, MuleSoft Inc.



Mule messages



12 | All contents Copyright © 2015, MuleSoft Inc.



Creating Mule applications with Anypoint Studio

13 | All contents Copyright © 2015, MuleSoft Inc.

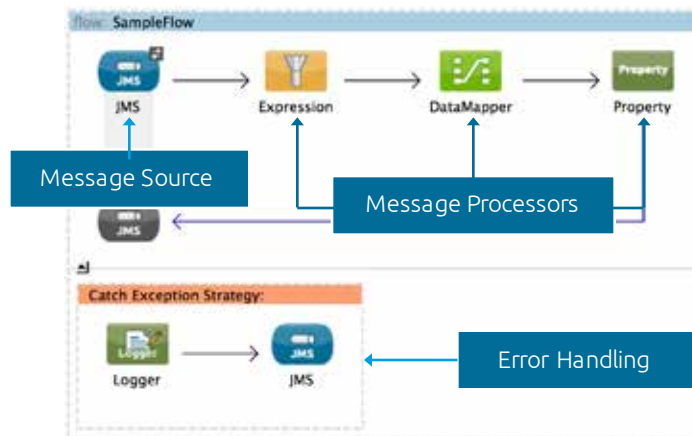
Creating Mule applications with Anypoint Studio

- Anypoint Studio is an Eclipse-based integration development environment
 - Two-way editing between graphical and XML views
 - Visual debugging (EE)
 - Pre-built tooling to connect to
 - Many popular services (Salesforce, Workday, Facebook, more!)
 - Many standard protocols (HTTP, HTTPS, FTP, SMTP, more!)
 - Any SOAP or RESTful API
 - A data transformation framework and language (EE)
 - One-click deployment of applications
 - Templates for common integration patterns (EE)
 - Integration with Maven for continuous build processes

14 | All contents Copyright © 2015, MuleSoft Inc.



Anatomy of a flow: Visual

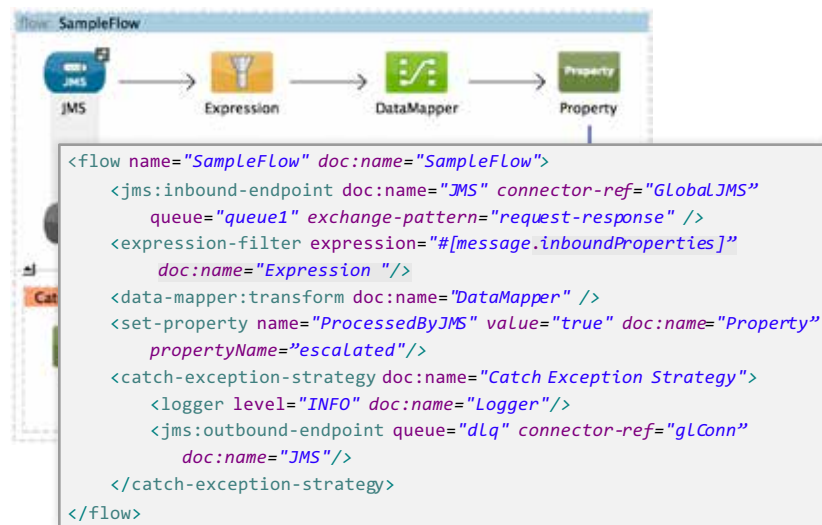


* Each flow operates on its own thread pool

15 | All contents Copyright © 2015, MuleSoft Inc.



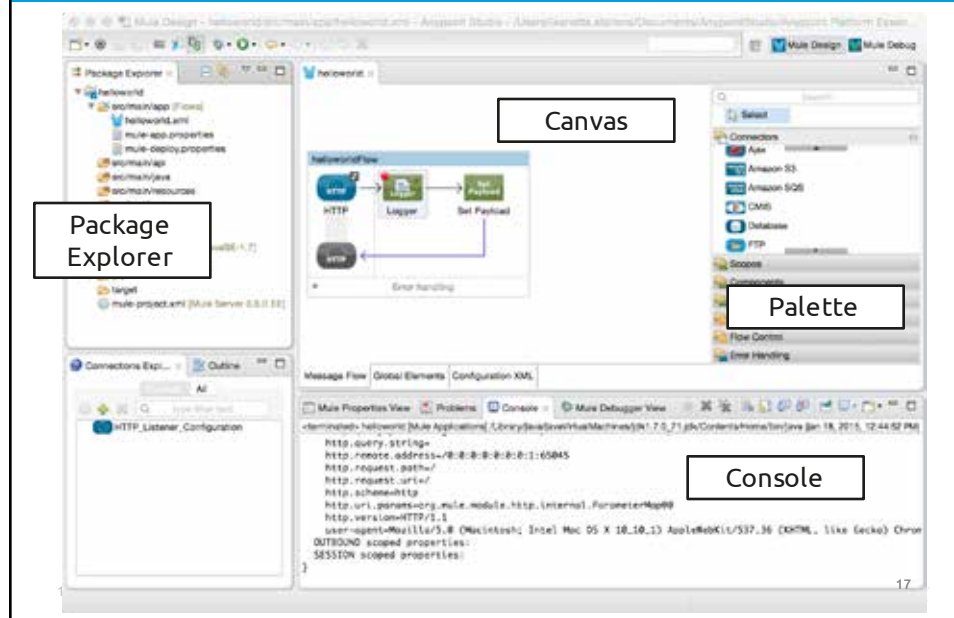
Anatomy of a flow: XML



16 | All contents Copyright © 2015, MuleSoft Inc.



Anypoint Studio anatomy



Walkthrough 2-1: Create your first Mule application

- Create a new Mule project with Anypoint Studio
- Add a connector to receive requests at an endpoint
- Display a message in the Anypoint Studio console
- Set the message payload



Understanding Mule application building blocks

19 | All contents Copyright © 2015, MuleSoft Inc.

Message sources



- Mule applications accept and process **messages** through a series of **message processors** plugged together in a **flow**
- Message sources are usually Anypoint Connectors
- Connectors provide connectivity to external resources
 - Such as databases, protocols, or APIs
 - Standard protocols like HTTP, FTP, SMTP, AMQP
 - Third-party APIs like Salesforce.com, Twitter, or MongoDB
- The first building block of most flows is a receiver that receives new messages and places them in the queue for processing

20 | All contents Copyright © 2015, MuleSoft Inc.



Anypoint Connectors

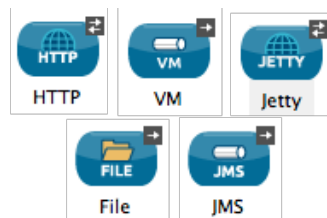
- Anypoint Platform has over 120 pre-built connectors
 - 30 bundled with Anypoint Studio
 - Full library at <https://www.mulesoft.com/exchange>
- There are 2 main types
 - Endpoint-based connectors
 - Operation-based connectors

21 | All contents Copyright © 2015, MuleSoft Inc.



Endpoint-based connectors

- Are either inbound or outbound endpoints in a flow
- Inbound endpoints serve as a message source for a flow
- Outbound endpoints send information to external systems
 - Can occur mid-flow or at the end of flows (or at the beginning in a poll)

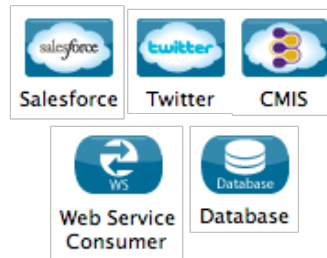


22 | All contents Copyright © 2015, MuleSoft Inc.



Operation-based connectors

- Require the specification of an operation for that connector to perform
- Includes most connectors not based on a standard communication protocol



23 | All contents Copyright © 2015, MuleSoft Inc.



Terminology: Connector vs endpoint vs transport

- A **connector** is a Mule-specific connection to an external resource of any kind
- An **endpoint** is a flow-level element that is configured to receive and/or send messages from and/or to external resources
- **Transports** provide connectivity to a data source or message channel by implementing one of a variety of standard connection protocols that are bundled with Studio

24 | All contents Copyright © 2015, MuleSoft Inc.



Connectors

- A **connector** is a Mule-specific connection to an external resource of any kind
- Resources may be
 - A generic protocol like HTTP or FTP
 - A Java-based database
 - A specific third-party API like Salesforce or Workday
- Connectors may be operation-based or endpoint-based
 - This distinction refers to a difference in configuration details, not in functionality
 - Endpoint-based connectors are often constructed around a transport

25 | All contents Copyright © 2015, MuleSoft Inc.



Endpoints

- An **endpoint** is a flow-level element that is configured to receive and/or send messages from and/or to external resources
- Any connector (whether endpoint-based or operation-based) can function as an endpoint
- But some connectors can only act as inbound endpoints, others can only act as outbound endpoints

26 | All contents Copyright © 2015, MuleSoft Inc.



Connectors and endpoints and global elements

- When you drag a connector from the Studio palette, an endpoint is created
- For most endpoints, a lot of the configuration is encapsulated in a separate global element
 - A reusable object that can be used by many endpoints
 - Defines a connection to a network resource
 - These are referred to as
 - Connectors
 - Global Mule configuration elements



27 | All contents Copyright © 2015, MuleSoft Inc.

HTTP connector

- Can send and receive HTTP and HTTPS requests over a selected host, port, and address
- Can be either a listener or a requester depending upon where you add it in a flow
- HTTP Listener connector (inbound)
 - Listens for HTTP requests that arrive at a certain address and provides an HTTP response to these
 - By default, host is set to 0.0.0.0
 - A shortcut to simultaneously listen on all active IP addresses (including localhost)
- HTTP Request connector (outbound)
 - Sends HTTP requests to a certain address and receives the returned response

28 | All contents Copyright © 2015, MuleSoft Inc.



Message processors

- Components
 - Execute specific logic upon a message, including custom-logic in Java, JavaScript, Groovy, Python or Ruby
 - **Logger** component
- Transformers
 - Convert data types and formats so as to "translate" messages between applications or systems
 - **Set Payload** transformer

29 | All contents Copyright © 2015, MuleSoft Inc.



Message processors

- Scopes
 - Wrap snippets of code to define fine-grained behavior within a flow
- Filters
 - Limit processing of messages based on set criteria
- Flow control
 - Direct messages through different pathways in an application depending upon content or other criteria
- Error handling
 - Handle any errors that occur during message processing

30 | All contents Copyright © 2015, MuleSoft Inc.



- Anypoint Studio comes complete with an embedded Mule runtime to test applications without leaving it
- The console outputs application logs and information

The screenshot shows the 'Console' tab in Anypoint Studio. The log output includes the following information:

```

apessentials [Mule Applications] /Library/Java/JavaVirtualMachines/jdk1.8.0_45-jdk/Contents/Home/bin/java (Aug 19, 2015, 9:59:51 AM)
+ Started app 'apessentials'
INFO 2015-08-19 09:59:55,368 [main] org.mule.module.launcher.DeploymentDirectoryWatcher:
+ Mule is up and kicking (every 5000ms)
INFO 2015-08-19 09:59:55,407 [main] org.mule.module.launcher.StartupSummaryDeploymentListener:

```

DOMAIN	STATUS
default	DEPLOYED

APPLICATION	DOMAIN	STATUS
apessentials	default	DEPLOYED

Testing applications

- There are several options for making requests to an endpoint
 - A browser
 - A cURL command-line utility
 - A browser extension like [Postman](#) (for Google Chrome)

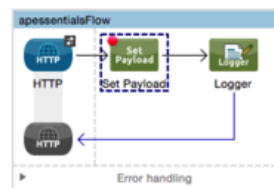
33 | All contents Copyright © 2015, MuleSoft Inc.



Debugging applications with the Mule Debugger



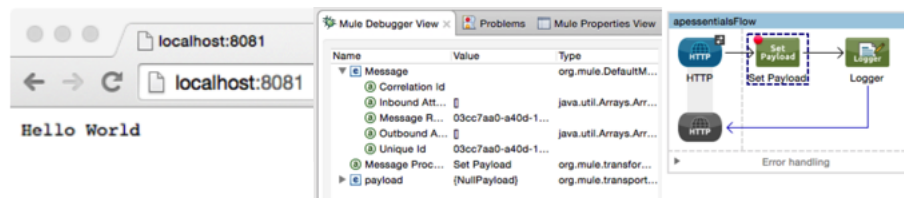
- Can add breakpoints to processors and step through the application
 - Watch message and variable values
 - Watch and evaluate expressions
- Debugger listens for incoming TCP connections on localhost port 6666



Name	Value	Type
accept	text/html,application/xhtml+xml,application/javascript;q=0.9,*/*;q=0.8	java.lang.String
accept-encoding	gzip, deflate, sdch	java.lang.String
accept-language	en-US,en;q=0.8	java.lang.String
cache-control	max-age=0	java.lang.String
connection	keep-alive	java.lang.String
host	localhost:8081	java.lang.String
isHttps	false	java.lang.Boolean

Walkthrough 2-2: Run, test, and debug a Mule application

- Run a Mule application using the embedded Mule runtime
- Make an HTTP request to the endpoint via a web browser or a tool like cURL or Postman
- Receive a response with the text Hello World
- Redeploy an application
- Use the Mule Debugger to debug an application



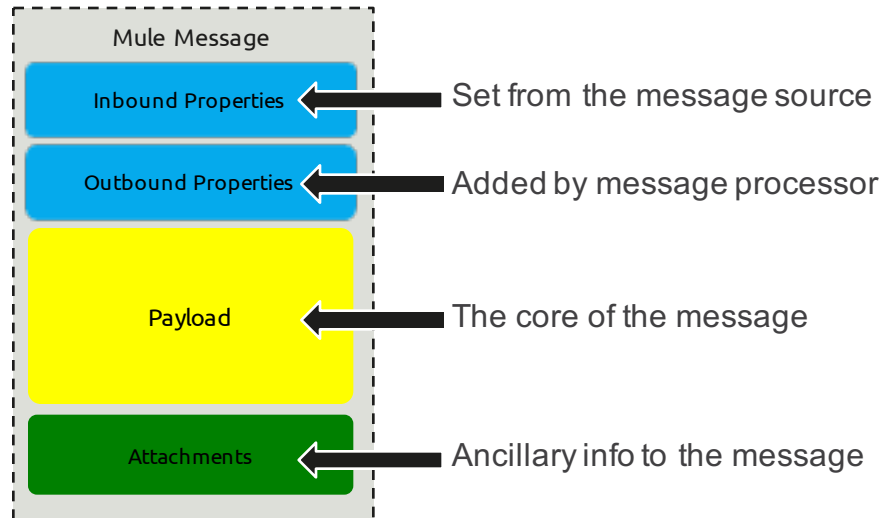
35 | All contents Copyright © 2015, MuleSoft Inc.



Reading and writing Mule messages

36 | All contents Copyright © 2015, MuleSoft Inc.

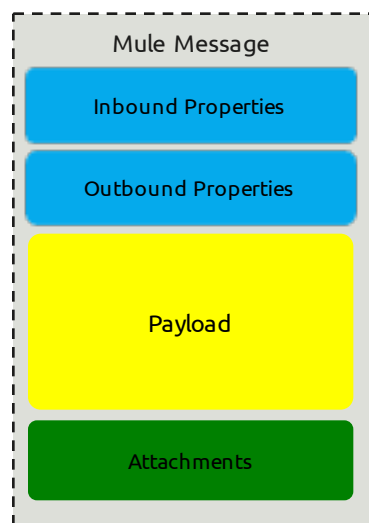
Mule messages



37 | All contents Copyright © 2015, MuleSoft Inc.



Message properties

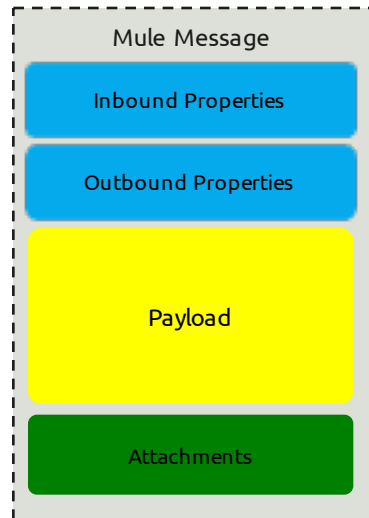


- Inbound properties
 - Set from the message source
 - Read-only access
 - Persist throughout the flow
- Outbound properties
 - Added by message processor
 - Read/write access
 - Can set, remove, copy

38 | All contents Copyright © 2015, MuleSoft Inc.



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

39 | All contents Copyright © 2015, MuleSoft Inc.



Payload representation

Raw Data

Payload

```
<order>
  <id>ed921739-99c1-4e09</id>
  <custId>49e377ed-bc72-4523</custId>
  <itemsTotal>200.34</itemsTotal>
</order>
```

org.java.lang.String

- Raw data often of type
 - String
 - InputStream
 - Byte[] (Byte array)
- Structured data often of type
 - Map
 - Structured Java Object
 - Order, Account, etc.

Structured Data

Payload

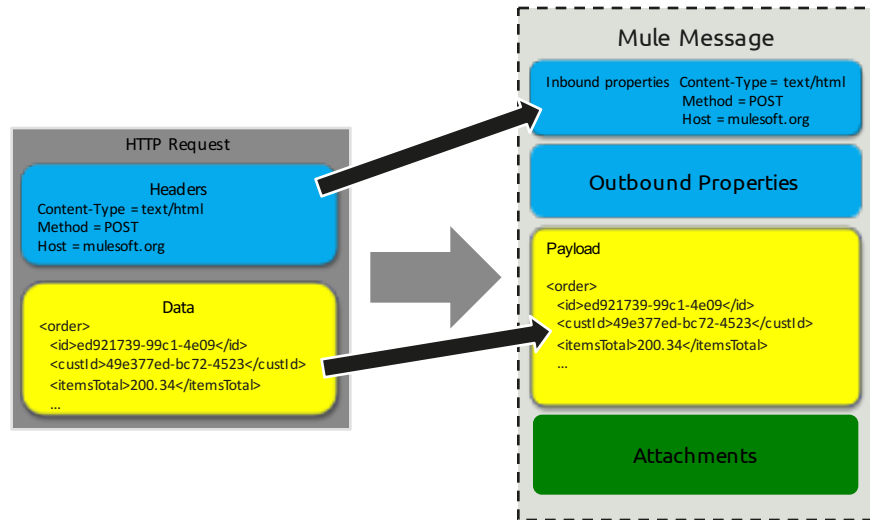
```
id: ed921739-99c1-4e09
custId: 49e377ed-bc72-4523
itemsTotal: 200.34
```

org.java.util.HashMap

40 | All contents Copyright © 2015, MuleSoft Inc.



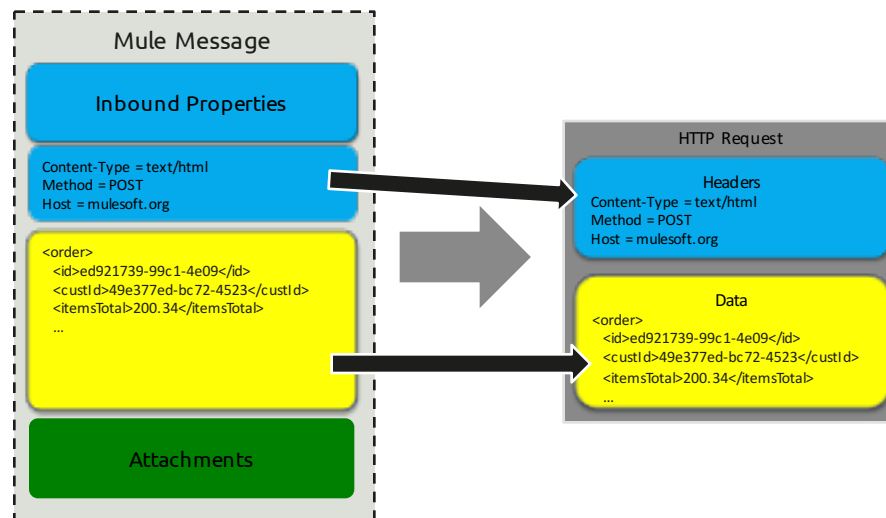
Inbound message properties



41 | All contents Copyright © 2015, MuleSoft Inc.



Outbound message properties



42 | All contents Copyright © 2015, MuleSoft Inc.



Setting message properties

Set Payload

- Sets the value of the message payload

Property

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

Writing Mule expressions

The Mule Expression Language (MEL)

- MEL is a lightweight, Mule-specific expression language
- Use it to access and evaluate the data in the payload, properties, and variables of a Mule message
- 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 auto-complete everywhere

45 | All contents Copyright © 2015, MuleSoft Inc.



Basic MEL syntax

#[] Encapsulates all Mule expressions

#[message] Holds a context object

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

46 | All contents Copyright © 2015, MuleSoft Inc.



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

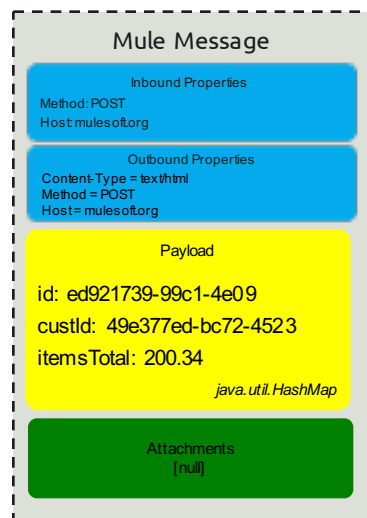
message

The Mule message that the message processor is processing

47 | All contents Copyright © 2015, MuleSoft Inc.



Accessing message data



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

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

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

48 | All contents Copyright © 2015, MuleSoft Inc.



Accessing message payload data

Mule Message

Inbound Properties

Method: POST
Host: mulesoft.org

Outbound Properties

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

Payload

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

java.util.HashMap

Attachments
[null]

`#[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 | All contents Copyright © 2015, MuleSoft Inc.



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 | All contents Copyright © 2015, MuleSoft Inc.



Writing expressions

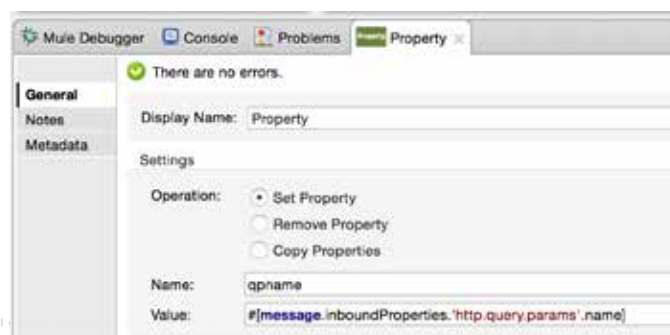
- Operators
 - Arithmetic: +, -, /, *, %
 - Evaluation: ==, !=, >, <, >=, <=, contains, is
#[message.inboundProperties.'http.query.params'.lastname != null]
- Testing for emptiness
 - The literal **empty** tests the emptiness of a value
 - Null, boolean false, "", " ", zero, empty collections
- Data extraction
 - XPath: #[xpath('expression')]
 - RegEx: #[regex('expression')]

51 | All contents Copyright © 2015, MuleSoft Inc.



Walkthrough 2-3: Read and write message properties

- Write MEL expressions
- Use the Debugger to read inbound and outbound message properties
- Use the Property transformer to set outbound message properties



52 | All contents



MEL references

- MEL expression reference
 - <http://www.mulesoft.org/documentation/display/current/Mule+Expression+Language+Reference>
- MEL language tips
 - <http://www.mulesoft.org/documentation/display/current/Mule+Expression+Language+Tips>

Creating variables

Context variables

flowVars
sessionVars
recordVars

`#[flowVars.ticketNum]`

55 | All contents Copyright © 2015, MuleSoft Inc.



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

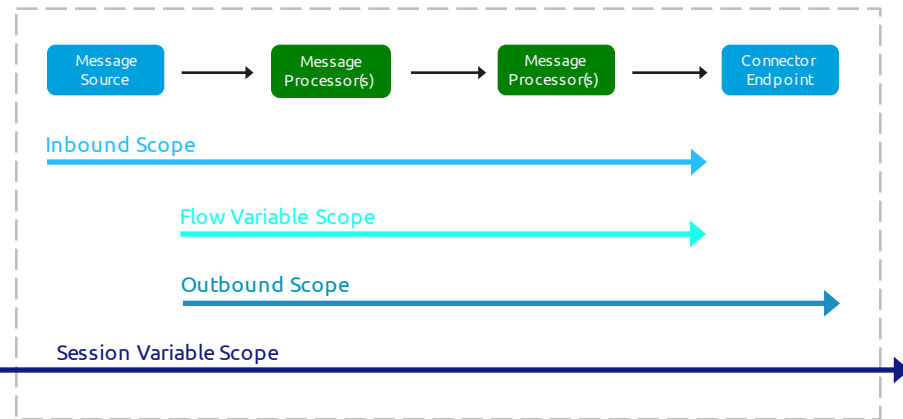
- Sets or removes **session variables**
 - Variables tied to a message for its lifecycle across flows, applications, and servers
 - They are persisted across some but **not all** transport barriers
 - Reference as sessionVars
 - `#[sessionVars.foobar]`

56 | All contents Copyright © 2015, MuleSoft Inc.



Variable persistence

Flow

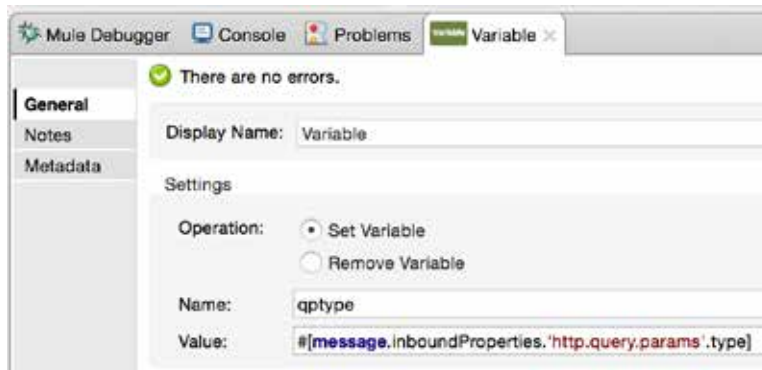


57 | All contents Copyright © 2015, MuleSoft Inc.



Walkthrough 2-4: Read and write variables

- Use the Variable transformer to create flow variables
- Use the Session transformer to create session variables



58 | All contents Copyright © 2015, MuleSoft Inc.



Summary

59 | All contents Copyright © 2015, MuleSoft Inc.

Summary

- In this module, you learned to build integration applications graphically with Anypoint Studio
 - Two-way editing between graphical and XML views
 - An embedded Mule runtime for testing applications
 - A Visual Debugger (EE) for debugging applications
- Mule applications accept and process messages through a series of message processors plugged together in a flow
- Mule messages have inbound properties, outbound properties, a payload, and attachments

60 | All contents Copyright © 2015, MuleSoft Inc.



Summary

- Message processors include connectors, transformers, components, scopes, and flow control elements
- Connectors are inbound or outbound and endpoint or operation based
- When you drag out a connector, an endpoint is created
- For most endpoints, a lot of the configuration is encapsulated in a separate, reusable global element

61 | All contents Copyright © 2015, MuleSoft Inc.



Summary

- Use the HTTP Listener connector as an inbound endpoint to trigger a flow with an HTTP request
- Use the Property transformer to set, remove, or copy message outbound properties
- Use the Set Payload transformer to set the payload
- 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

62 | All contents Copyright © 2015, MuleSoft Inc.

