



PART 2: Building Applications with Anypoint Studio



Goal



Mule Debugger

Name	Value	Type
Encoding	UTF-8	java.lang.String
Payload (Implements: com.mulesoft.training.Flight)	com.mulesoft.training.Flight@422c7 com.mulesoft.training.Flight@32a4a3d com.mulesoft.training.Flight@22ba4a4d com.mulesoft.training.Flight@7e8a872d	org.mule.runtime.core.internal.message.DefaultMessageBuilder\$MessageImpl<com.mulesoft.training.Flight>

Implementation

```

graph LR
    Source --> SetVariable[Set Variable airline]
    SetVariable --> SetCode[Flow Reference setCode]
    SetCode --> IsValid[Is valid destination]
    IsValid --> ChoiceAirline{Choice airline}
    ChoiceAirline --> GetAmericanFlight[Flow Reference getAmericanFlights]
    ChoiceAirline --> GetDefaultFlight[Flow Reference getDeflights]
    GetAmericanFlight --> Default[Flow Reference Default]
    GetDefaultFlight --> Transform[Flow Reference transform to JSON]
    Default --> Transform
    Transform --> SetPayload[Flow Reference set payload]
    SetPayload --> Logger[Logger]
  
```

Mule Breakpoints

- Logger (Core)
- Request (HTTP)
- Transform Message (Core)
- Set Payload (Core)
- Flow Reference (Core)
- Choice (Core)

Mule Palette

- Search in Exchange
- Add Modules
- Global Variables
- Core
 - American Flights API
 - APIKit
 - HTTP
 - Sockets
 - Validation
 - Web Service Consumer

Message Flow

All contents © MuleSoft Inc.

2

At the end of this part, you should be able to



- Debug Mule applications
- Read and write event payloads, attributes, and variables using the DataWeave Expression Language
- Structure Mule applications using flows, subflows, asynchronous queues, properties files, and configuration files
- Call RESTful and SOAP web services
- Route and validate events and handle messaging errors
- Write DataWeave scripts for transformations



Module 6: Accessing and Modifying Mule Events



Goal

The screenshot shows the Mule Debugger interface. At the top, there's a toolbar with various icons. Below it is a table titled "Mule Debugger" showing variable values:

Name	Value	Type
Attributes	org.mule.extension.http.api.iHttpRequestAttributes	org.mule.extension.http.api.HttpRequestAttributes
Component Path	fundamentalsFlow/processors/2	org.mule.runtime.dsl.api.component.config.DefaultComponentLocation
Data Type	SimpleDataType{type=java.lang.String, mimeType="*/*"}	org.mule.runtime.core.internal.metadata.SimpleDataType
Message		org.mule.runtime.core.internal.message.DefaultMessageBuilder\$...
Payload (mimeType="*/*")	Hello	java.lang.String
Variables	size = 1 code=SFO	java.util.HashMap java.util.HashMap\$Node
O		

Below the debugger is a "fundamentals" flow diagram:

```

graph LR
    Listener[HTTP Listener] --> SetPayload[Set Payload]
    SetPayload --> SetVariable[Set Variable]
    SetVariable --> Request[Request<br/>HTTP Request]
    Request --> Logger[Logger]
    
```

The "Request" node is highlighted with a dashed blue border. The "fundamentals" folder is expanded.

5

At the end of this module, you should be able to



- Log event data
- Debug Mule applications
- Read and write event properties
- Write expressions with the DataWeave expression language
- Create variables

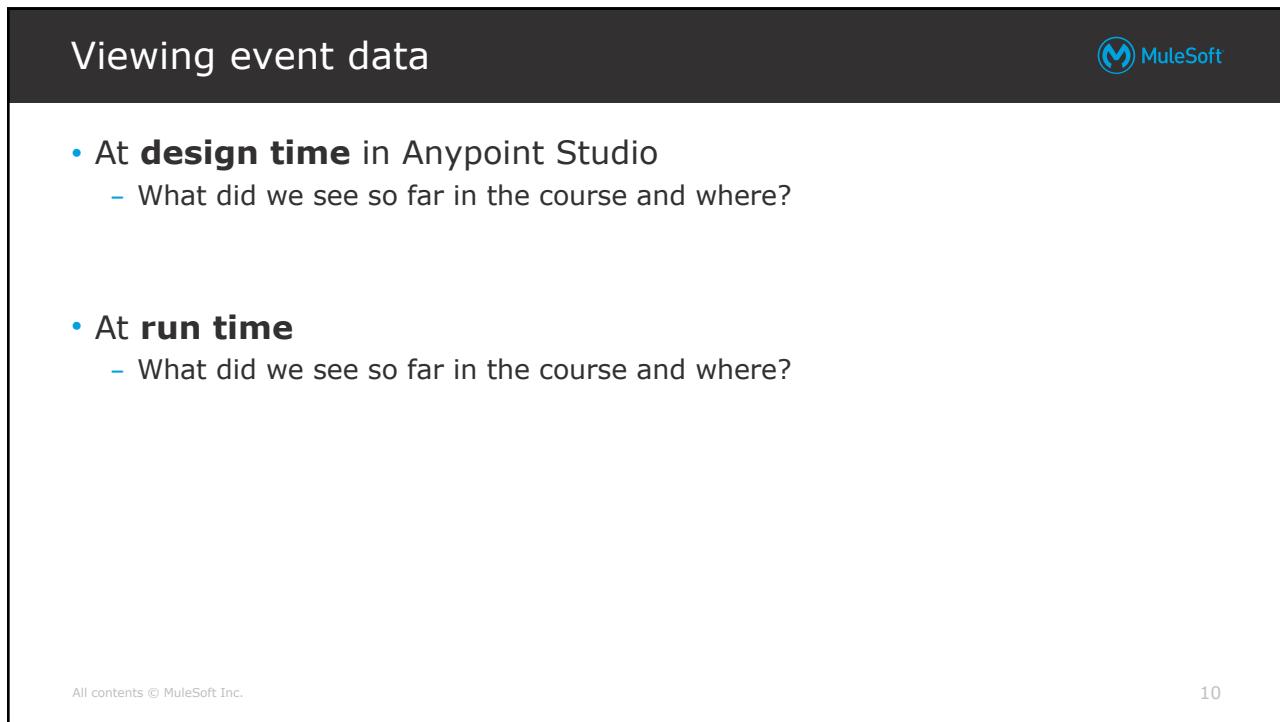
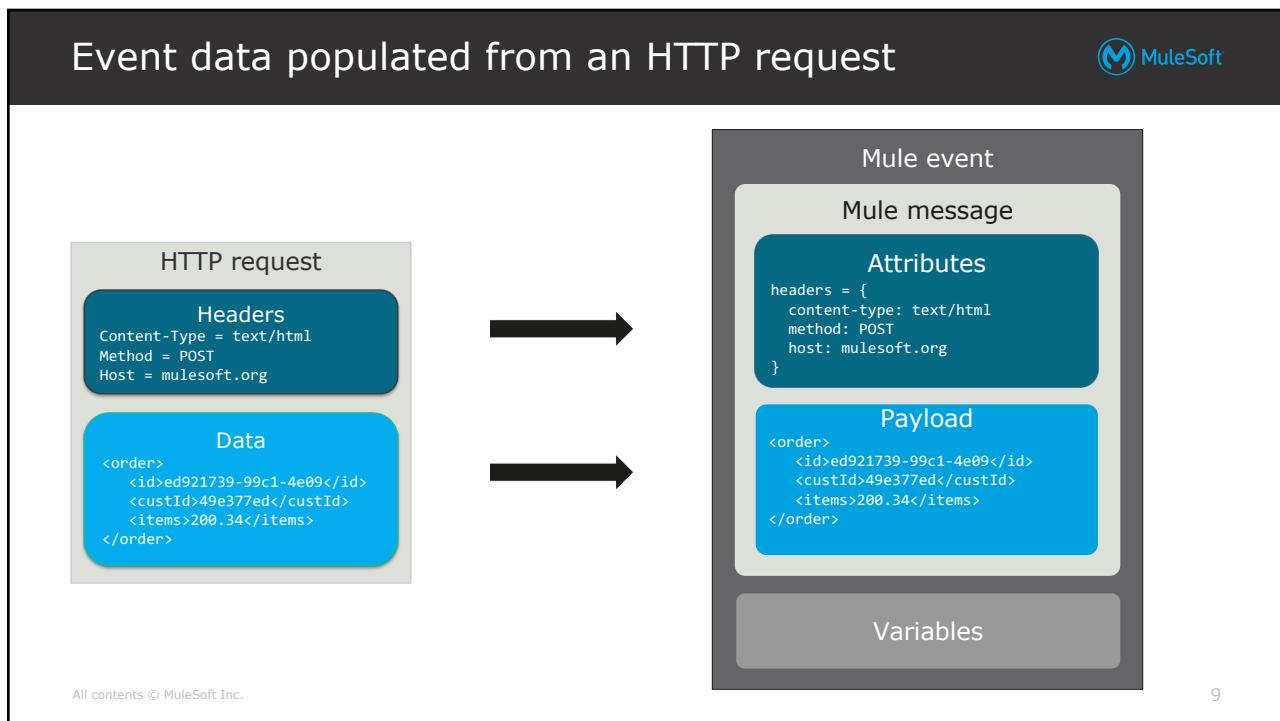
Viewing information about Mule 4 events



Reviewing the structure of Mule 4 events



- The data that passes through flows in the app
- Metadata contained in the message header
- The core info of the message - the data the app processes
- Metadata for the Mule event - can be defined and referenced in the app processing the event



Viewing event data at design time



- At **design time** in Anypoint Studio using DataSense
 - In the Transform Message component
 - In the DataSense Explorer
 - When writing expressions using auto-completion
- **DataSense** is the ability to proactively discover metadata from internal and external resources
 - Keeps you from having to manually discover info about the data
 - Facilitates transformations by providing DataWeave expected input or output

All contents © MuleSoft Inc.

11

Viewing event data at run time



- At **run time**
 - In the client when making a request
 - For deployed applications, in the log files
 - In the Anypoint Studio console by using a Logger
 - In Anypoint Studio using the Visual Debugger
 - Most comprehensive way

Name	Value
method	GET
queryParams	MultiMap[[fullName=[Max Mule]]]
0	fullName=Max Mule
key	Max Mule
value	Max Mule
queryString	fullName=Max Mule
relativePath	/goodbye
remoteAddress	/127.0.0.1:52451
requestPath	/goodbye
requestUri	/goodbye?fullName=Max Mule

All contents © MuleSoft Inc.

12

View event data by logging it



- Add a **Logger** component to a flow and view its output
- Logged values are displayed
 - For an application run from Anypoint Studio with embedded runtime, in the **Console view**
 - For applications deployed to CloudHub or customer-hosted runtimes, in the **log files**



Logger

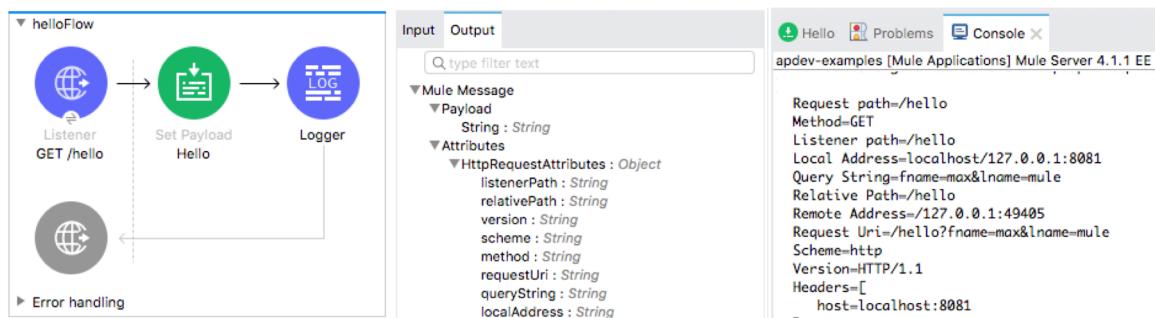
All contents © MuleSoft Inc.

13

Walkthrough 6-1: View event data



- Create a new Mule project with an HTTP Listener and set the payload
- View event data in the DataSense Explorer
- Use a Logger to view event data in the Anypoint Studio console



All contents © MuleSoft Inc.

35

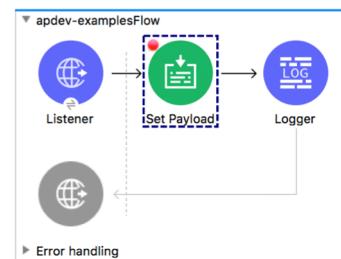
Debugging Mule applications



Debugging applications with the Mule Debugger



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



Walkthrough 6-2: 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 event properties
- Pass query parameters to a request and locate them in the Debugger
- Increase the request timeout for Advanced REST Client

The screenshot shows the Mule Debugger interface with two main panes. The top pane displays event properties for an incoming message, including attributes like 'headers' (MultiMap[host=localhost:8081, content-type=application/json]), 'method' (GET), and 'payload' (MultiMap[theme=mule, themeId=mule]). The bottom pane shows a flow diagram titled 'HelloWorld' with components: 'Get Hello' (HTTP), 'Set Response' (Script), and 'Logger'. A 'Breakpoint' icon is placed over the 'Set Response' component. To the right, there's a 'Watches' table and a 'Dataflow Expression Watcher' tab.

All contents © MuleSoft Inc.

17

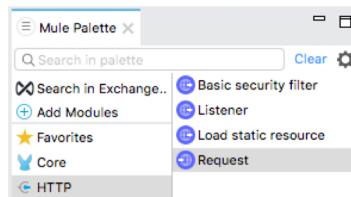
Tracking event data as it moves in and out of Mule applications



Changes to the event object



- We examined changes to an event as it moved through a flow and we set the payload
- What happens to the event object when calls are made to an external resource from a flow?
 - For example, you call a web service, and get return data?



All contents © MuleSoft Inc.

19

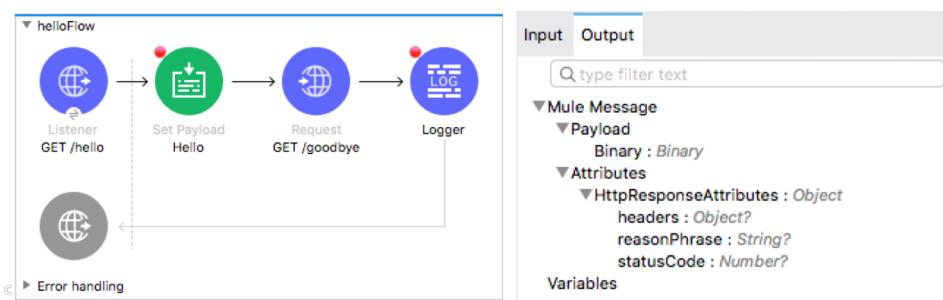
Walkthrough 6-3: Track event data as it moves in and out of a Mule application



- Create a second flow with an HTTP Listener
- Make an HTTP request from the first flow to the new HTTP Listener
- View the event data as it moves through both flows

*Note: You are making an HTTP request from one flow to another in this exercise **only** so you can watch the value of event data as it moves in and out of a Mule application. You will learn how to pass events between flows within and between Mule applications in the next module.*

All contents ©



20

Setting request and response data



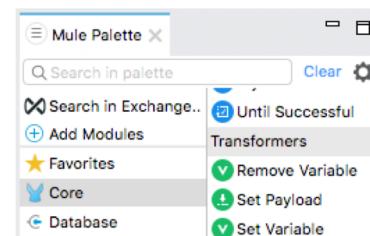
Setting the message payload



- Use the Set Payload transformer to set the payload



Set Payload



Setting data returned from HTTP listeners



- Set in the HTTP Listener properties view > Responses

All contents © MuleSoft Inc.

23

Setting data sent to HTTP requests



- Set in the HTTP Request properties view > General

All contents © MuleSoft Inc.

24

Walkthrough 6-4: Set request and response data



- View the default setting for a response body
- Set a response header
- View the default setting for a request body
- Set a request query parameter

The screenshot shows two API configurations side-by-side:

- GET /hello:** Under the "Responses" tab, the "Body" field contains the expression `#[payload]`. The "Headers" section shows a single header named "name" with the value "Max".
- GET /goodbye:** Under the "General" tab, the "Method" is set to "GET (Default)", "Path" is "/goodbye", and "URL" is empty. The "Query Parameters" tab is selected, showing a query parameter named "fullName" with the value "Max Mule".

35

Using DataWeave expressions to read and write event data



The DataWeave expression language



- A Mule-specific expression and transformation language
- Can be used to access and evaluate the data in the payload, attributes, and variables of a Mule event
- Accessible and usable from all event processors and global elements
 - Is used to modify the way the processors act upon the event such as routing
- Case-sensitive
- Easy to use with auto-complete everywhere

All contents © MuleSoft Inc.

27

Types of DataWeave expressions



• Standalone scripts

- Were generated using the Transform Message graphical editor in Module 4
- We will write these from scratch in Module 11

```
Output Payload ▾ ⌂ Preview
1@%dw 2.0
2 output application/json
3 ---
4@payload map ( payload01 , indexOfPayload01 ) -> {
5@    ID: payload01.ID,
6    code: payload01.code1 default "",
7    departureDate: payload01.takeOffDate as String
8 }
```

- Inline expressions

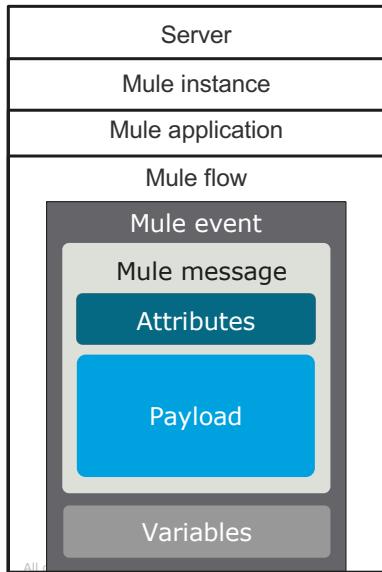
- Are used to dynamically set the value of properties in an event processor or global configuration element
- Are enclosed in #[]

[]

All contents © MuleSoft Inc.

28

Referencing Mule objects in DataWeave expressions



server	#[server.osName]
mule	#[mule.version]
app	#[app.name]
flow	#[flow.name]
message	#[message.payload]
attributes	#[attributes.queryParams.param1]
payload	#[payload]
vars	#[vars.foo]

29

Accessing event attributes



#[message.attributes.method]	POST
#[attributes.method]	POST
#[attributes.headers.host]	mulesoft.org
#[attributes.header['user-agent']]	Mozilla
#[message.payload.id]	ed921739-99c1-4e09
#[payload.id]	ed921739-99c1-4e09
#[payload['id']]	ed921739-99c1-4e09
#[payload.itemsTotal]	200.34

30

Using selectors in DataWeave expressions



Description	Selector	Example
Value of a key:value pair	Single Value selector	#[payload.name] #[attributes.queryParams]
Value at selected array index	Indexed selector	#[payload[0].name]
Array with values of key:value pairs	Multi Value selector	#[payload.*name]
Array with values of key:value pairs	Descendants selector	#[payload..zip]

All contents © MuleSoft Inc.

Note: You will learn additional selectors in Module 11

31

Using operators in DataWeave expressions



Description	Operators	Example
Arithmetic	+, -, /, *	#[payload.age * 2]
Equality	==, !=, ~=	#[payload.name == "max"]
Relational	>, >=, <, <=, is	#[payload.age > 30]
Conditional	and, or, not	#[(payload.name=="max") and (payload.age>30)]
Type coercion	as	#[(payload.age as Number) * 2]
Default value	default	#[payload.type default "student"]

All contents © MuleSoft Inc.

Note: For operator precedence, see

<https://docs.mulesoft.com/mule4-user-guide/v/4.1/dataweave-flow-control-precedence>

32

Using conditional logic statements in DataWeave



- Use if / else if / else statements

```
if (payload.age < 15)
    group: "child"
else if (payload.age < 25)
    group: "youth"
else if (payload.age < 65)
    group: "adult"
else
    group: "senior"
```

Using DataWeave functions



- Functions are packaged in modules
- Functions in the Core module are imported automatically into DataWeave scripts
- For function reference, see
 - <https://docs.mulesoft.com/mule4-user-guide/v/4.1/dataweave>
- For functions with 2 parameters, an alternate syntax can be used
 - #[contains(payload,max)]
 - #[payload contains "max "]

DataWeave Function Reference			
Core (dw::Core)	groupBy	maxBy	scan
++	isBlank	min	sizeOf
--	isDecimal	minBy	splitBy
abs	isEmpty	mod	sqr
avg	isEven	native	startsWith
ceil	isInteger	now	sum
contains	isLeapYear	orderBy	to
daysBetween	isOdd	pluck	trim
distinctBy	joinBy	pow	typeOf
endsWith	log	random	unzip
filter	lower	randomInt	upper
filterObject	map	read	uuid
find	mapObject	readUrl	with
flatMap	match	reduce	write
flatten	matches	replace	zip
floor	max	round	34

Walkthrough 6-5: Get and set event data using DataWeave expressions



- Use expressions to set the payload and a logged value
- Use expressions to set a response header and a request query param
- In expressions, reference values for the event payload and attributes
- Use the DataWeave upper() function and the concatenation, as, and default operators

The screenshot shows the 'fullName' event configuration in the Anypoint Studio interface. The 'General' tab is selected. The 'Display Name' field contains 'fullName'. The 'Message' field contains the expression '#[attributes.queryParams.fullName]'. The 'Level' dropdown is set to 'INFO (Default)'. A status message at the top right indicates 'There are no errors.'

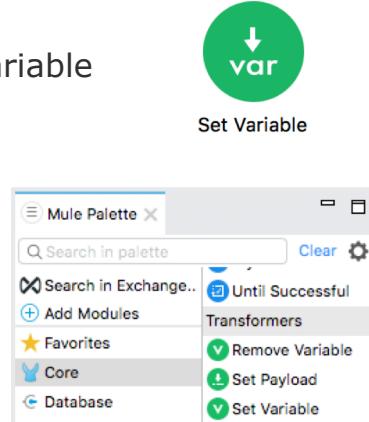
All contents © MuleSoft Inc.

Creating variables



Creating variables

- Create variables to store metadata for the Mule event
- Use the Set Variable transformer to create a variable
- In expressions, reference as vars
 - #[vars.foo]

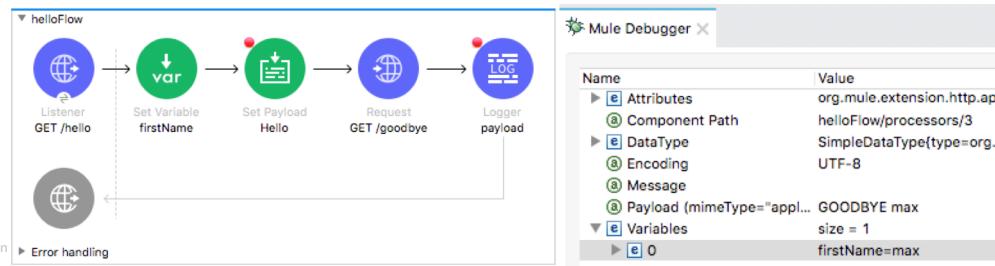


All contents © MuleSoft Inc.

37

Walkthrough 6-5: Set and get variables

- Use the Set Variable transformer to create a variable
- Reference a variable in a DataWeave expression
- Use a variable to dynamically set a response header
- Use the Mule Debugger to see the value of a variable
- Track variables across a transport boundary

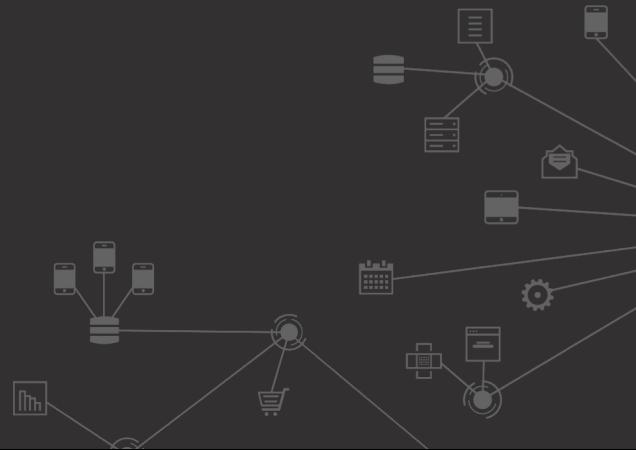


Name	Value
Attributes	org.mule.extension.http.ap
Component Path	helloFlow/processors/3
DataType	SimpleDataType{type=org.i}
Encoding	UTF-8
Message	
Payload (mimeType="appl... GOODBYE max	
Variables	size = 1 firstName=max
0	

All con ▶ Error handling

38

Summary



Summary



- The best way to view event data is to add breakpoints to a flow and use the **Mule Debugger**
- Use the **Logger** component to display data in the console
- Use the **Set Payload** transformer to set the payload
- Use the properties view to set response data for an HTTP Listener and request data for an HTTP Request operation
- Use the **DataWeave language** to write inline expressions in **#[]**
- Use the **Set Variable** transformer to create variables