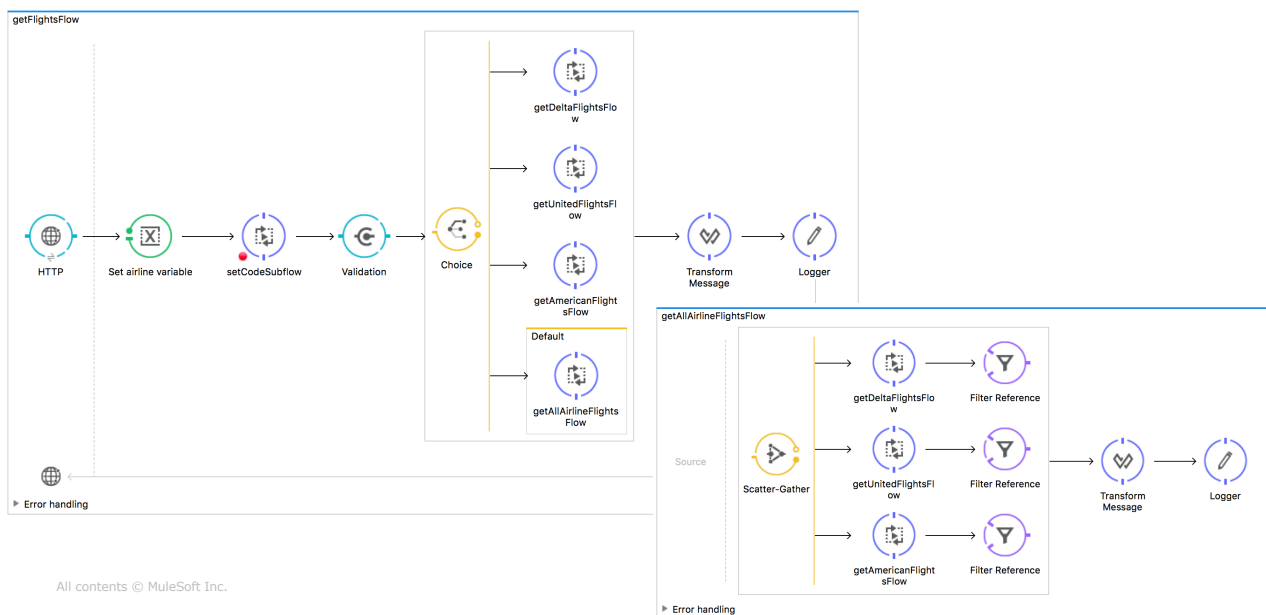




Module 10: Controlling Message Flow

Goal



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



- Route messages based on conditions
- Multicast messages
- Filter messages
- Validate messages

All contents © MuleSoft Inc.

3

Routing messages



Routers



- Routers route messages to various destinations in a Mule flow
- Some incorporate logic to analyze and possibly transform messages before routing takes place
- Some change the payload, some don't

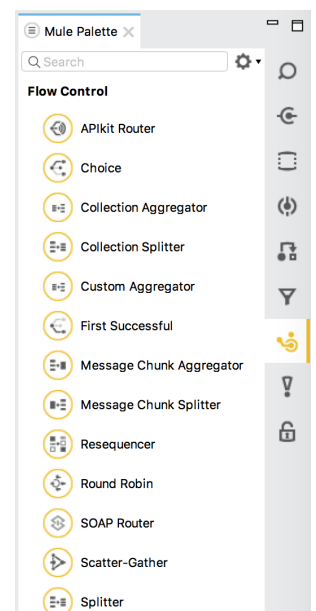
All contents © MuleSoft Inc.

5

Available flow controls



- Three main types, those that
 - Check logic and route
 - Choice
 - Multicast and aggregate
 - Scatter-Gather
 - Split and/or aggregate



All contents © MuleSoft Inc.

Routing messages based on conditions



The Choice router

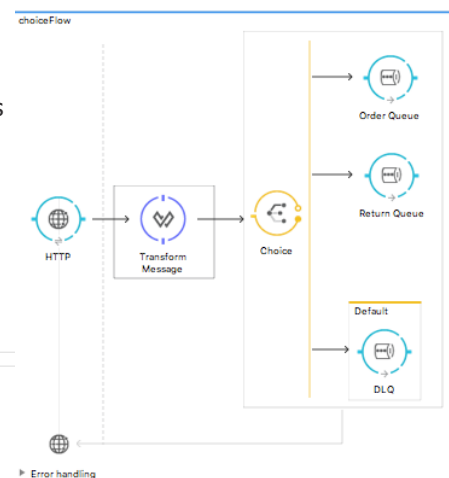


- Sends the message to one route based on conditions
 - Each path can include multiple message processors

- The conditions are written with MEL

```
When
#[message.inboundProperties['requestType']] == 'order'
Default
#[message.inboundProperties['requestType']] == 'return'
```

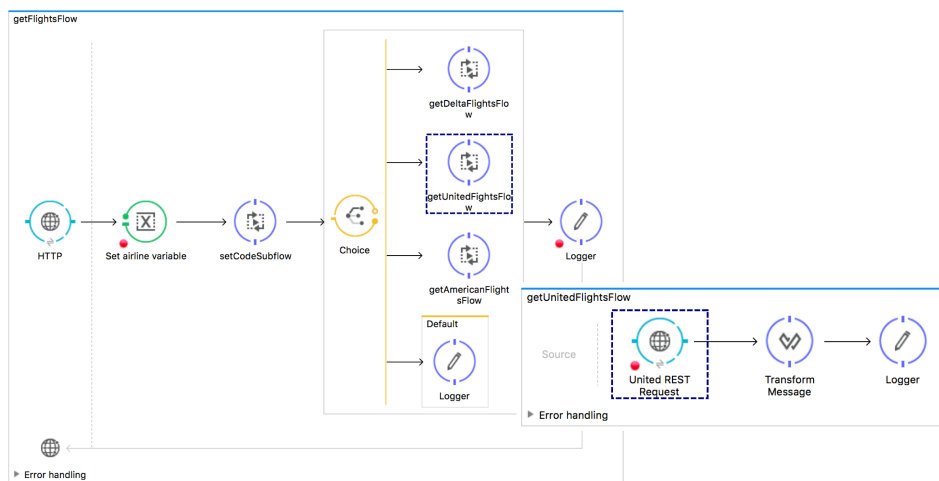
```
Route Message to
Order Queue
DLQ
Return Queue
```



Walkthrough 10-1: Route messages based on conditions



- Use a Choice router
- Set the router paths



9

Multicasting messages



The Scatter-Gather router



- Scatter-Gather sends the message to each route concurrently and returns a collection of all results

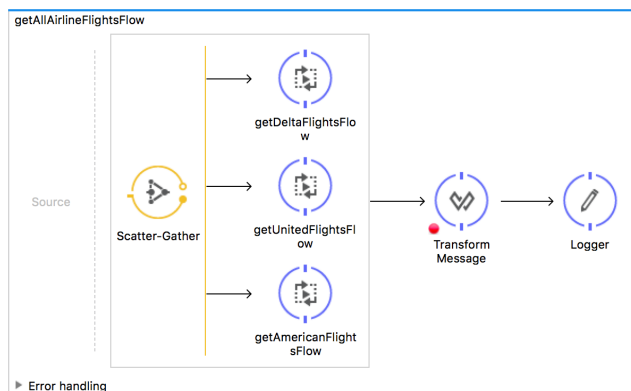
All contents © MuleSoft Inc.

11

Walkthrough 10-2: Multicast a message



- Use a Scatter-Gather router to concurrently call all three flight services
- Use DataWeave to flatten multiple collections into one collection
- Use DataWeave to sort the flights by price and return them as JSON



12

Filtering messages



Filters



- Determine whether a message can proceed in a Mule flow
- By default, filtered messages are dropped and processing of the message ends
 - Keeps subsequent processors from receiving irrelevant or incomprehensible messages
 - Filters can be configured to throw an exception

Available filters



- There are 12 bundled filters
 - Expression and Payload are often used
 - And, Or, Not apply Boolean logic
 - Message filter nests other filters for more complex logical conditions
 - Idempotent ensures a message is not delivered more than once
- You can create custom filters
- You can define global filters and reference them for reuse

Filters

- And
- Custom
- Exception
- Expression
- Filter Reference
- Idempotent Message
- Message
- Message Property
- Not
- Or
- Payload
- Regex
- Schema Validation
- Wildcard

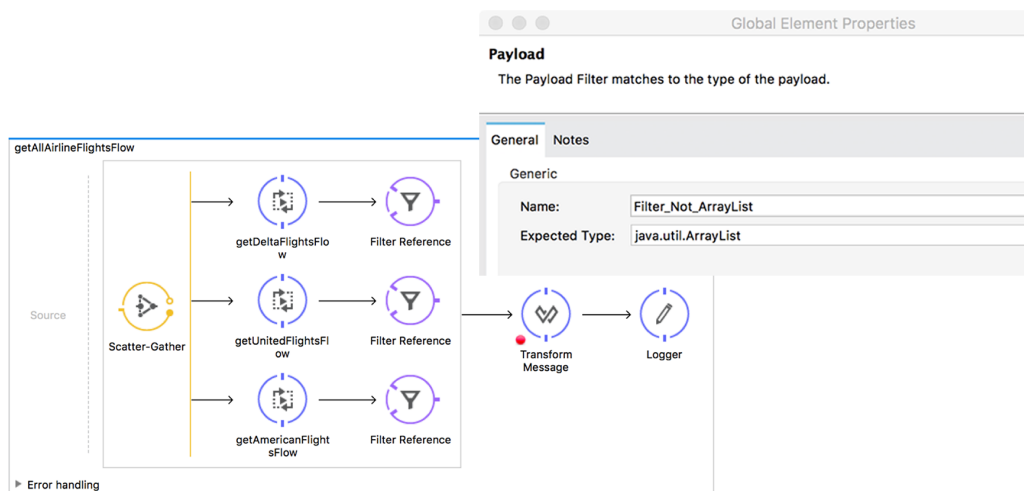
All contents © MuleSoft Inc.

15

Walkthrough 10-3: Filter messages



- Use the Payload filter
- Create and use a global filter



16

Validating messages



Validators

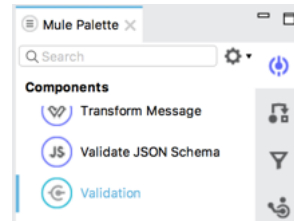


- Provide an easy out-of-the-box way to test some conditions are met and throw an exception if the validation fails
- The main advantage over using filters is traceability
 - Filters all raise identical exceptions, making it hard for you to know where the exception was caused
 - Validators, on the other hand, raise an exception with a meaningful message attached

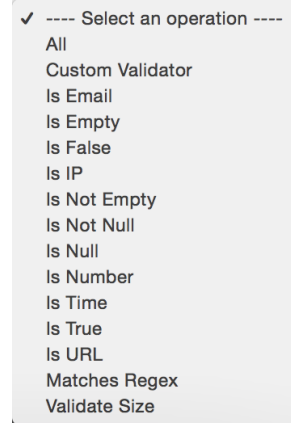
Using validators



- Use the Validation component



- Select the type of validation
 - Select All to create a list of validations to execute

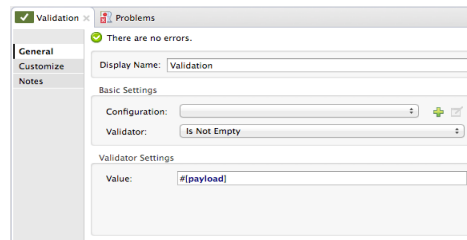


All contents © MuleSoft Inc.

Using validators (cont)



- Set the value to validate



- Handle the ValidationException exception
 - By default, this exception will have a meaningful message attached

Name	Value
▶ DataType	CollectionDataType{type=java.util.LinkedList, itemType=java.lang.Object,...
▼ exceptionThrown	org.mule.api.MessagingException: Collection is empty. Message payload i...
▶ cause	org.mule.extension.validation.api.ValidationException: Collection is empty
▶ CAUSE_CAPTION	Caused by:

All contents © MuleSoft Inc.

20

There are actually two ways to use validators



- Using the Validation component message processor

```
<validation:is-email email="mule@mulesoft.com" />
```

- Through MEL

```
#[validator.validateEmail('mule@mulesoft.com')]
```

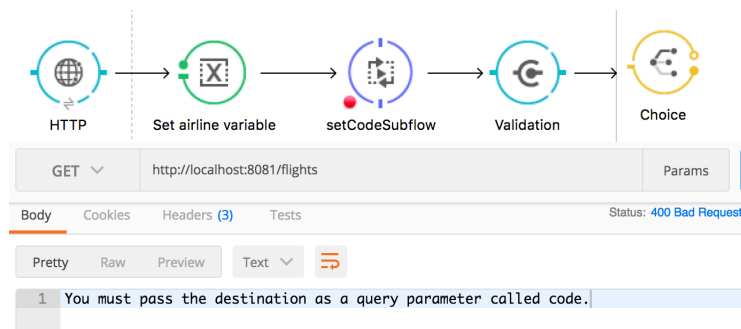
All contents © MuleSoft Inc.

21

Walkthrough 10-4: Validate messages



- Require a destination code to be passed to the app as a query param
- Use the Validation component to throw an exception if the query param is not set
- Catch the exception in the global exception strategy and return an appropriate message



All contents © MuleSoft Inc.

22

Summary



Summary



- Use different, routers, filters, and validators to control message flow
- Use the Choice router to send a message to one route based on conditions
- Use the Scatter-Gather router to send a message concurrently to multiple routes
 - A collection of all results is returned
 - Use DataWeave to flatten the collection
- Use filters and validators to determine whether a message can proceed in a Mule flow
 - Filters all throw identical exceptions
 - Validators can throw a ValidationException with a custom message or a custom exception object