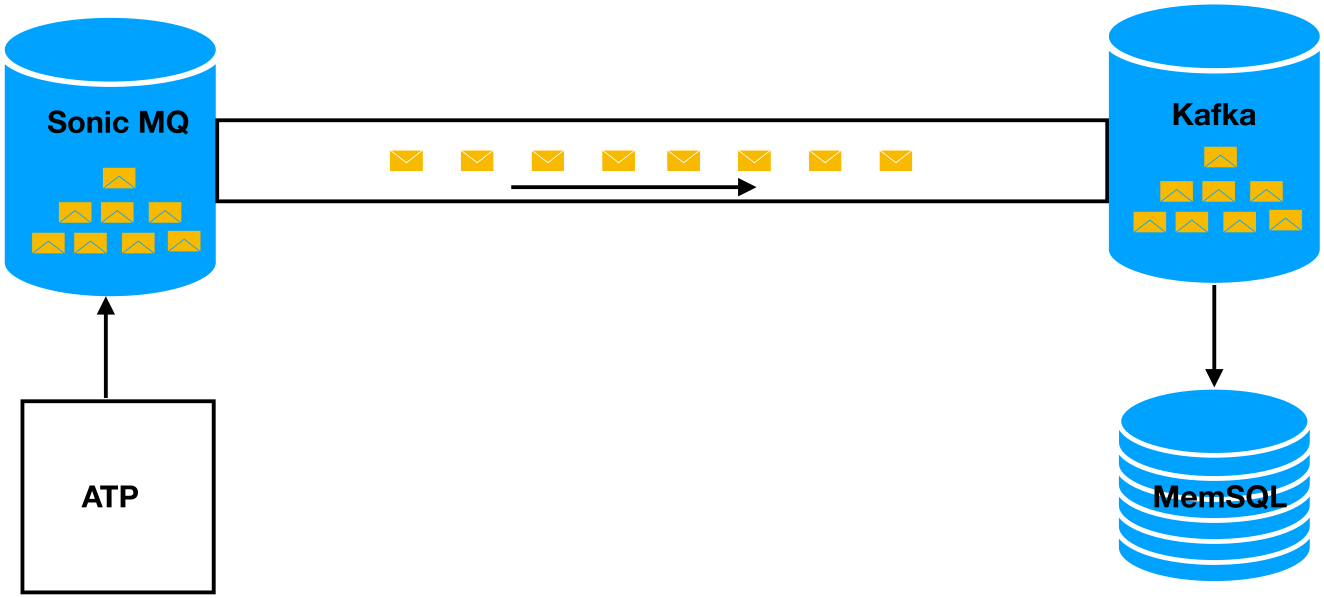
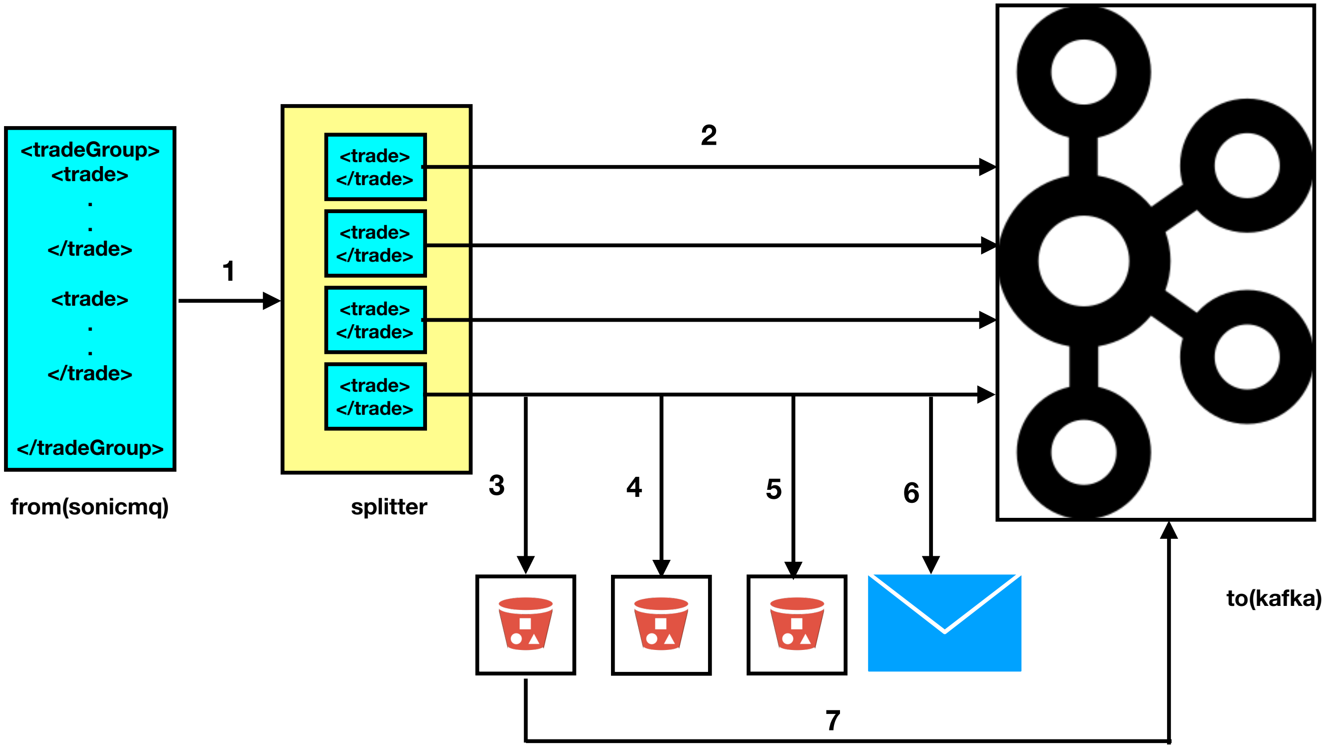
Sonic Kafka Messaging Bridge



Technology Stacks

1. Spring Boot
2. Apache Camel
3. Jolokia (JMX -> HTTP)
4. Hawtio (Monitoring Interface)

Application Architecture



1. Tokenize on trade element (offers a faster, more efficient tokenization for XML documents over xpath)
2. Concurrently publishes all the sub trades to Kafka
3. Sends to offline storage when Kafka brokers are offline
4. Stores trade xml > 2 MB
5. Stores trade xml for all other exceptions
6. Sends failure notifications to IMRS Support
7. Publish from off line storage to Kafka

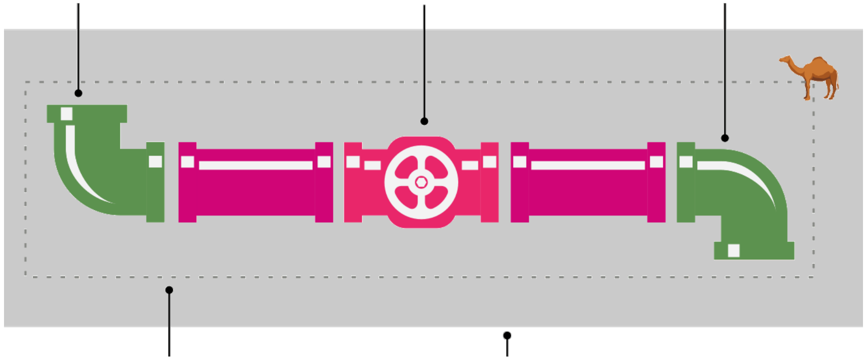
Apache Camel

An integration library for Java which makes integration easier and more accessible to developers

Endpoint
Receives the message

EIP
Processes the message

Endpoint
Sends the message



Route
A flow in Camel

Camel Context
Container where your routes are run

Route

route is a pipe that moves data from point A to point B.

Routes can be defined using DSLs in programming languages such as Java, Scala and Groovy.
It also allows routing rules to be specified in XML.

Java DSL:

```
from ("file:/tmp").to("jms:aQueue");
```

Scala DSL:

```
from "file:/tmp" -> "jms:aQueue"
```

Spring DSL:

```
<route>
  <from uri="file:/tmp"/>
  <to uri="jms:aQueue"/>
</route>
```

Endpoints

Endpoint represents any other external system to Camel. At the start of a route, Camel **receives** a message from ant endpoint.
Then the message might be processed in some way — perhaps by an EIP — before being sent to another destination **endpoint**.

Components

Component connects camel route to endpoint

Component	Purpose	Endpoint URI
HTTP	for creating or consuming web sites	http:
File	for reading and writing files	file:
JMS	for reading and writing to messaging queues	jms:
Direct	for joining your Camel routes together	direct:

Enterprise Integration Patterns (EIPs)

Camel implemented all the patterns defined in the book EIP

When you want to perform some common activities on a message, such as **transformation, splitting and logging**, you'll use an EIP. Here are some common EIPs in Camel:

EIP name	What it does	Java syntax
Splitter	Splits a message into multiple parts	.split()
Aggregator	Combines several messages into one message	.aggregate()
Log	Writes a simple log message	.log()
Marshal	Converts an object into a text or binary format	.marshal()
From*	Receives a message from an endpoint	.from()
To*	Sends a message to an endpoint	.to()

Camel Context

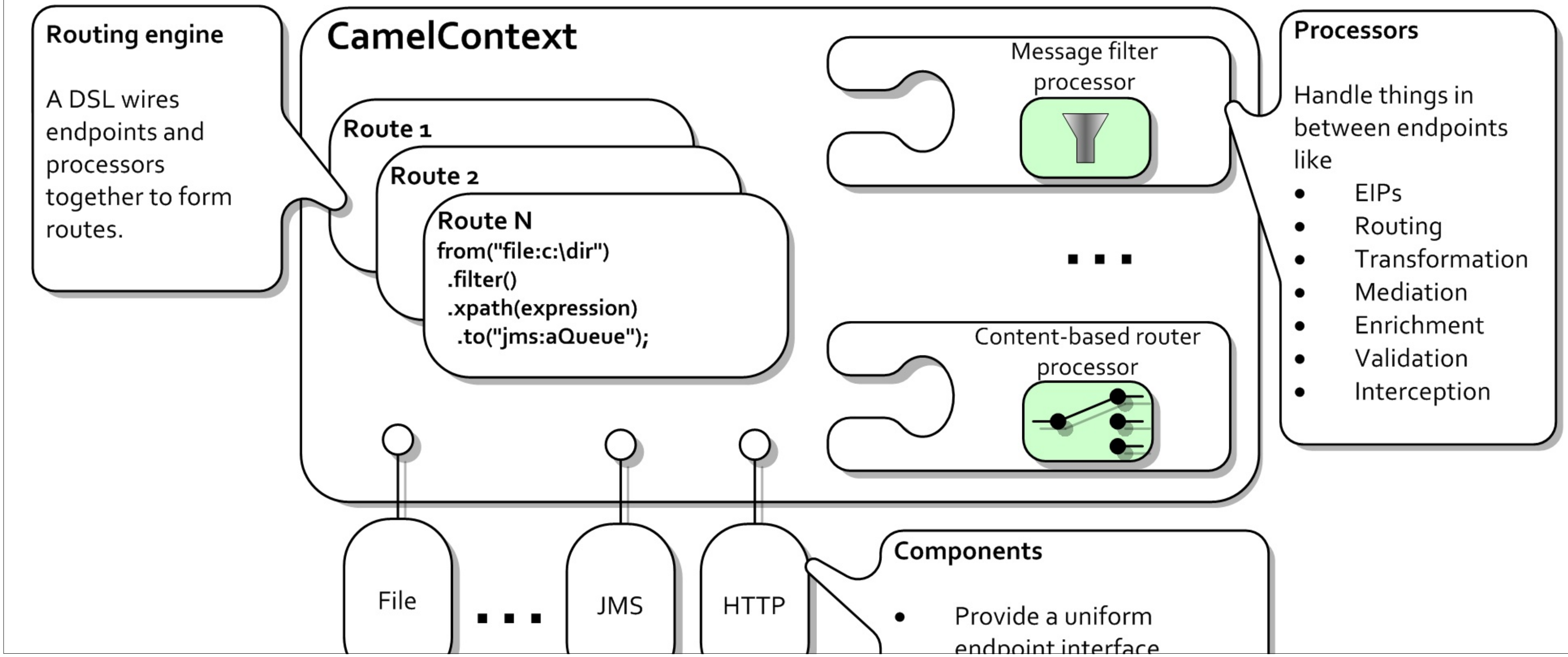
Camel has a container called the Camel Context.
Your routes run inside this engine.
When Camel **starts**, it reads your route definitions (in Java or XML), creates the routes, adds them to a Camel Context, and starts the Camel Context.
When Camel **terminates**, it shuts down your routes, and closes the Camel Context.

Apache Camel

An integration library for Java which makes integration easier and more accessible to developers

There are 3 core concepts in

1. concrete implementations of all the widely used EIPs
2. connectivity to a great variety of transports and APIs
3. easy to use Domain Specific Languages (DSLs) to wire EIPs and transports together



Routing engine

A DSL wires endpoints and processors together to form routes.

Processors

Handle things in between endpoints like

- EIPs
- Routing
- Transformation
- Mediation
- Enrichment
- Validation
- Interception

Components

- Provide a uniform endpoint interface

```
from("file:c:\dir")
  .filter()
  .xpath(expression)
   .to("jms:aQueue");
```