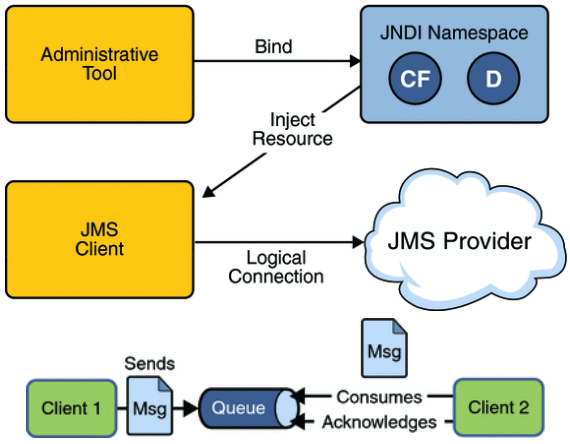
7. Java Messaging Service (JMS)

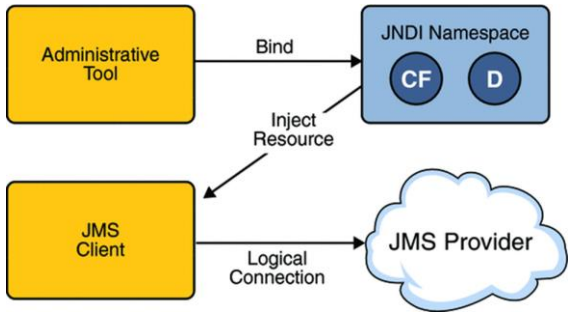
DGP Solutions

JMS High Level Architecture



Administered Objects

- A **connection factory** is the object a client uses to create a connection to a provider.
- A **destination** is the object a client uses to specify the target of messages it produces and the source of messages it consumes i.e. a Queue



19920612 Copyright DGP Solutions 7-3

Other Objects

- A **connection** encapsulates a virtual connection with a JMS provider. Created from the **connection factory**
- A **session** is a single-threaded context for producing and consuming messages. Created from the **connection**
- A **message producer** is an object that is created from a **session** and used for sending messages to a destination
- A **message consumer** is an object that is created by a session and used for receiving messages sent to a destination
- A **message listener** is an object that acts as an asynchronous event handler to poll for messages on a destination
- **JMS Message** is an object that represents the message itself whose payload could be Test or a Serializable Object

JmsTemplate

- Spring Boot hides all these complexities of object creation and interaction behind the class **org.springframework.jms.core.JmsTemplate** and with one maven dependency create an embedded ActiveMQ broker

- JmsTemplate simplifies the use of Java Messaging service (JMS) and gets rid of boilerplate code. It handles the creation and release of JMS resources when sending or receiving messages.


```
<!-- Embedded ActiveMQ -->
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-activemq</artifactId>
</dependency>
```

Spring Boot and ActiveMQ - Producer

```
@Autowired private JmsTemplate jmsTemplate;

private static Logger = Logger.getLogger("HealthPlanController");

@PostMapping(path="/domestic", consumes=MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Long> addToQueue(@RequestBody HealthPlan plan)  {
       logger.log(Level.INFO, "Received message");
       jmsTemplate.convertAndSend("myQueue", plan);
       logger.log(Level.INFO, "Sent message");
       return ResponseEntity.ok().build();
}
```

- ActiveMQ must be told what objects to safely deserialize (String OK)
- In application.properties add the key value pair of;
  - spring.activemq.packages.trust-all=true
  - OR spring.activemq.packages.trust=package-name

Spring Boot and ActiveMQ - Consumer

```
@Autowired private HealthPlanRepository repo;
private static Logger = Logger.getLogger("MessageConsumer");

@JmsListener(destination = "myQueue")
public void receiveMessage(Message msg) throws JMSException {
    ObjectMessage objMsg = (ObjectMessage) msg;
    HealthPlan plan = (HealthPlan) objMsg.getObject() ;
    logger.log(Level.INFO, "Received Message for Plan " + plan);
    HealthPlan pl = repo.save(plan);
    logger.log(Level.INFO, "Repository updated with Plan " + pl.getId());
}
```

- Note the destination name matches where the JmsTemplate put the message

Client is not kept waiting

- Upon invoking our Rest Endpoint, our client is notified of a successful call immediately

```
<200,[Content-Length:"0", Date:"Fri, 31 Jan 2020 18:34:21 GMT", Keep-Alive:"timeout=60", Connection:"keep-alive"]>
```

- We can then observe the logs to see the production and consumption of the message

```
HealthPlanController        : Received message
HealthPlanController        : Sent message
MessageConsumer             : Received Message for Plan HealthPlan [id=0, zip=77777, name=HMO, deductableIndividual=2500,
deductableFamily=3000, outOfPocketIndividual=5000, outOfPocketFamily=5500, copay=30.0]
MessageConsumer             : Repository updated with Plan 22
```