## Exception Handling

Exceptions are logged and also send to an payment.service.invalid.messages apache camel queue to ensure that the service/application does not crash because of invalid messages. Clarification needs to be obtained from the analyst if there is a need to send an email to the relevant user when an error message is encountered so they can fix and/or resend.

## Dependencies

Dependencies are added to the pom.xml file in the <dependencies></dependencies> tag to ensure that relevant dependencies are included.

## Systems Analyst Questions

The following questions would be asked to ensure requirement clarification:

1. Do the routing rules reside within the database?
2. After handling retrieve payment message status do we immediately call Send MT195 Acknowledgement for both Not Rejected and Rejected statuses or for just Rejected status?
3. Do we need to send a message to anyone who handles error files after we send the messages to an error queue?
4. What is the structure of an MT195 Acknowledgement?
5. What is the message structure of the Swift M101 message?
6. Ask the Systems Analyst to walk me through the payments process in detail so as to understand the process flow.

## Assumptions

The following assumptions were taken in the development of the system:

* Rules for the Routing Service reside in the database which is implemented using the H2 in memory database.
* A custom and simplistic Swift Message validation was implemented to check the basic message fields.
* Swagger was used for Web Service documentation and testing. The link to test on the browser is: http://localhost:8080/payment-service/swagger-ui.html#/
* Apache Camel was used as the messaging system/api and the initial queue for the micro service is: incoming.validated.payment.messages.
* GitHub link to clone the project is: <https://github.com/khumbue/paymentservice.git>

## Sample Valid Swift Message Used

{1:F01SAESVAV0AXXX0466020121}{2:O1011538070522LRLRXXXX4A0700005910650705221739N}{3:{108:MT101 001 OF 019}}{4:

:20:123456789

:28D:1/1

:50H:/GB12SEPA12341234123412

ORDERING CUST NAME

ORDERING CUST ADDR LINE 1

ORDERING CUST ADDR LINE 2

ORDERING CUST ADDR LINE 3

:52A:BANKGB01XXX

:30:160211

:21:11FEB2016INV1

:23E:URGP

:32B:EUR123,45

:57A:BANKGB02XXX

:59:/GB12SEPA12341234123498

JAMES BOND

SUPPLIER ADDR LINE 1

SUPPLIER ADDR LINE 2

SUPPLIER ADDR LINE 3

:70:SUPPLIER-INV-REF1

:77B:/BENEFRES/GB

:71A:SHA

-}{5:{MAC:00000000}{CHK:24857F4599E7}{TNG:}}

## Sample Invalid Swift Message Used (Tag 20 – Sender Reference Removed)

{1:F01SAESVAV0AXXX0466020121}{2:O1011538070522LRLRXXXX4A0700005910650705221739N}{3:{108:MT101 001 OF 019}}{4:

:28D:1/1

:50H:/GB12SEPA12341234123412

ORDERING CUST NAME

ORDERING CUST ADDR LINE 1

ORDERING CUST ADDR LINE 2

ORDERING CUST ADDR LINE 3

:52A:BANKGB01XXX

:30:160211

:21:11FEB2016INV1

:23E:URGP

:32B:EUR123,45

:57A:BANKGB02XXX

:59:/GB12SEPA12341234123498

JAMES BOND

SUPPLIER ADDR LINE 1

SUPPLIER ADDR LINE 2

SUPPLIER ADDR LINE 3

:70:SUPPLIER-INV-REF1

:77B:/BENEFRES/GB

:71A:SHA

-}{5:{MAC:00000000}{CHK:24857F4599E7}{TNG:}}

## Internal XML mapped from the Swift MT101 Message

<?xml version="1.0" encoding="UTF-8"?>

<payment>

<beneficiaryAccountNumber>/GB12SEPA12341234123498</beneficiaryAccountNumber>

<beneficiaryCustomer>JAMES BOND</beneficiaryCustomer>

<currency>EUR</currency>

<detailsOfCharges>SHA</detailsOfCharges>

<orderingCustomer>ORDERING CUST NAME</orderingCustomer>

<orderingCustomerAccountNumber>/GB12SEPA12341234123412</orderingCustomerAccountNumber>

<requestedExecutionDate>2016-02-11T00:00:00+02:00</requestedExecutionDate>

<senderReference>123456789</senderReference>

<status>Rejected</status>

<transactionAmount>123.45</transactionAmount>

<transactionReference>11FEB2016INV1</transactionReference>

</payment>