ESB Implementation Architecture

### Account Opening

|  |  |
| --- | --- |
| **Version** | **1.0** |
| **Created on** | **28 Apr, 2020** |
| **Last Updated on** |  |
| **Prepared by** | **Eidiko Systems Integrators** |
|  |  |
| Document Classification | CONFIDENTIAL |
| Distribution | Eidiko, Steward Bank |

Table of Contents

[Account Opening 1](file:///C:\Users\bandaru\Downloads\Direct%20Debit%20Outward_V4.docx#_Toc472606341)

[Table of Contents 2](#_Toc472606342)

[1 Approval 4](#_Toc472606343)

[2 Revision History 4](#_Toc472606344)

[3 Introduction 5](#_Toc472606345)

[3.1 General Description 5](#_Toc472606346)

[4 Integration Overview 5](#_Toc472606347)

[4.1 Business Process 5](#_Toc472606348)

[4.1.1 Withdrawal 5](#_Toc472606349)

[4.2 High Level Architecture 8](#_Toc472606350)

[4.3 Detailed Architecture 8](#_Toc472606351)

[4.4 Dependencies 9](#_Toc472606352)

[4.4.1 External Dependencies 9](#_Toc472606353)

[4.4.2 Internal Dependencies 9](#_Toc472606354)

[4.4.3 Request from Postilion 10](#_Toc472606355)

[4.4.4 Response to Postilion 12](#_Toc472606356)

[4.4.5 Error Handling 37](#_Toc472606358)

[4.4.6 Retry Mechanism 37](#_Toc472606359)

[4.4.7 Reference Data 37](#_Toc472606360)

[5 Database Definitions 37](#_Toc472606361)

[5.1 Scripts 37](#_Toc472606362)

[6 Queue Definitions 41](#_Toc472606363)

[6.1 Scripts 41](#_Toc472606364)

[7 Integration Definition 45](#_Toc472606368)

[7.1 Account Opening 45](#_Toc472606369)

[7.1.1 IntegrationDetails 45](#_Toc472606370)

[8 Requirements 46](#_Toc472606371)

[8.1 Integration Requirements 46](#_Toc472606372)

[8.2 Business Requirements 47](#_Toc472606373)

[8.3 Non-functional Requirements 47](#_Toc472606374)

[8.3.1 Volume 47](#_Toc472606375)

[8.3.2 Performance 47](#_Toc472606376)

[8.3.3 Security 47](#_Toc472606377)

# Approval

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Project Role | Approver | Company | Email | Email |
|  |  |  |  |  |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Description | Version | Author |
| **30 July, 2020** | Initial Version | 1.0 | Mir Mohsin Ali |

# Introduction

## General Description

An Account Opening interface is to open an account with Steward Bank. The process includes 3 steps :

1. KYC\_CHECK (Know your customer).
2. FCM\_CHECK (Fraud customer check).
3. Zss / Postilion Update (updating customer details in ZSS and Postilion CBS).

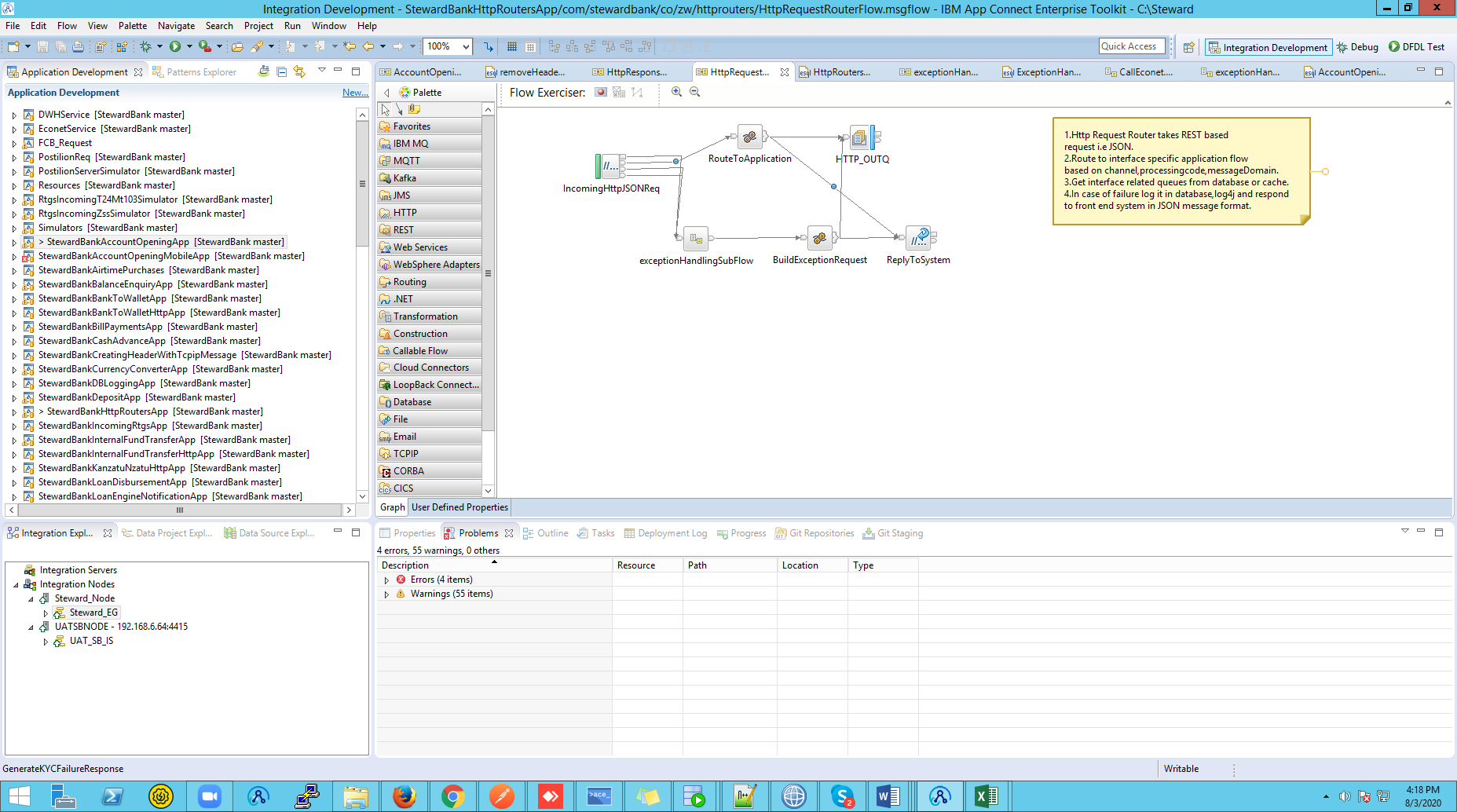
# Integration Overview

## Business Process

The process could be initiated from USSD code (\*236#) from any mobile.

### Withdrawal

**Router Flow : HttpRequestRouterFlow.msgflow**

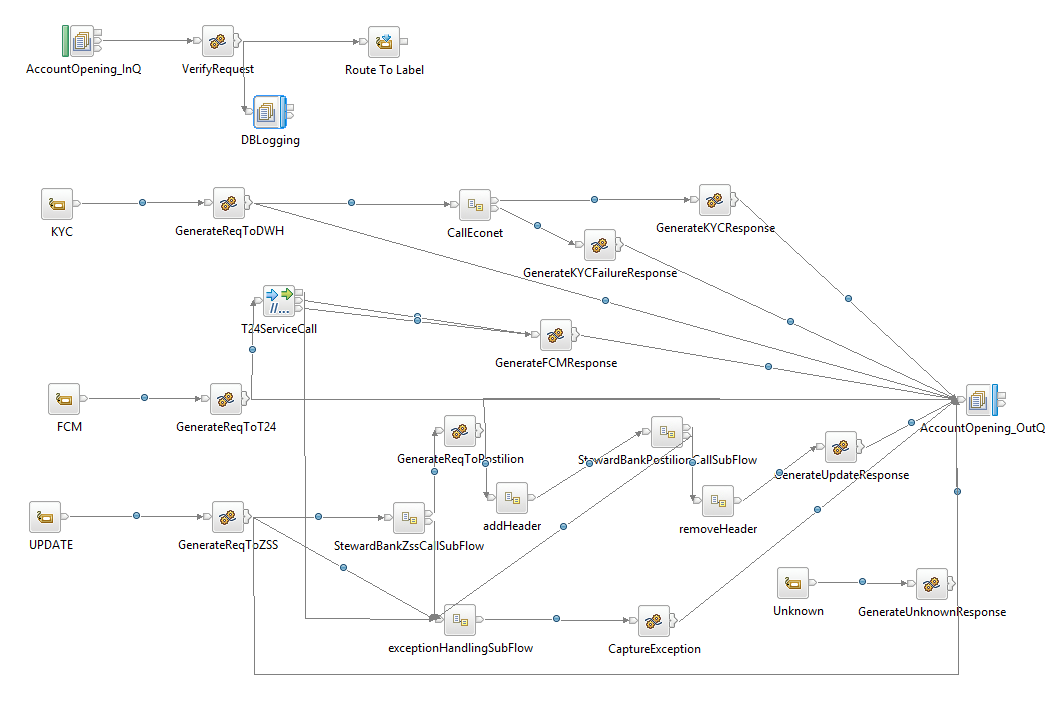


**Process Flow:**

1. The Transactions will initiate by the front end channel by sending a request to the ESB Router on the below URL (http://serverIP:port/v1/stewardBank/).
2. The request message should be either in XML or JSON format.
3. The Router accepts the request as BLOB then it will be converted to its specific format.
4. Once the request is converted to its own format then the request message will be logged in Database by constructing a JSON message with the details of Message ID, Request Message, Application Name, Broker Name, Application Type and sends the request to RAW\_LOG\_SB\_REQ Queue which then trigs the RAW\_AUDIT\_LOGGING application, and the same message will be logged in a log file as well.
5. We will fetch the records from the database based on domain “JSON” in Shared memory and will extract a specific records based on channel name and processing code to select queue details.
6. For 236 Account Opening the request queue which is recorded in the database is **ACOP\_236\_SB\_HTTP\_REQ**. And the common response queue for HTTP flows are **ALL\_SB\_HTTP\_RES**.
7. In this flow the Incoming message will be validated based on the proper structure of the message, channel and processing code.
8. If there is no processing code and channel in the request message the request could not reach the Business Application, and a failure message will be send back to the channel saying “key [processing code & channel] is missing”.
9. If there is no processing code then a failure message will be sent back to the channel saying “key [processing code] is missing”.
10. The same will be delivered if there is no channel or if the provided processing code or channel is not available in the database.
11. If there is any parsing error in the provided request then message cannot be logged in the database but an exceptional message will be sent back to channel.

8. For success case the message will be propagated to the selected queue.

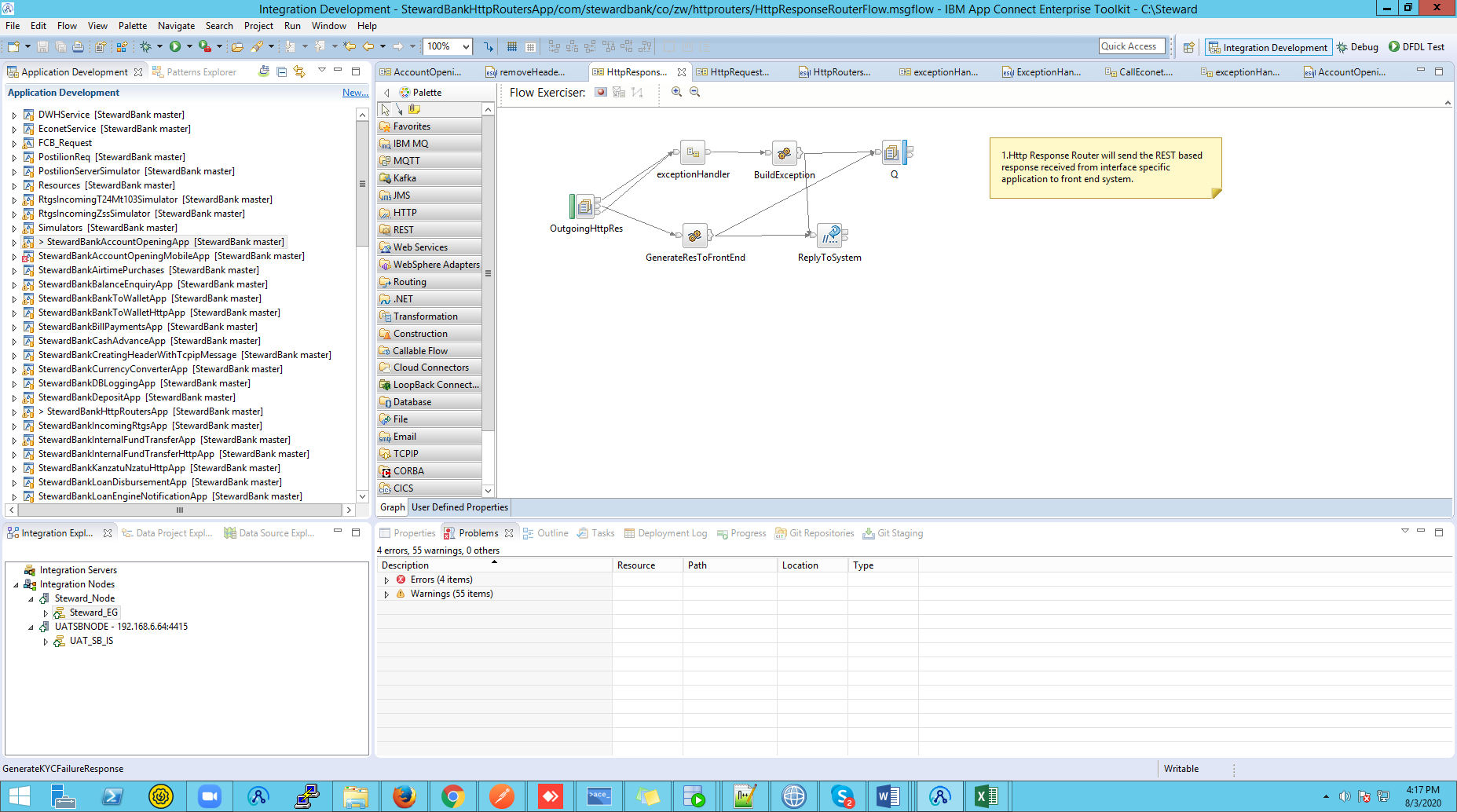
**ESB Flow : AccountOpening.msgflow**



**Process Flow:**

1. This application will get trigs when a message is propagated to **ACOP\_236\_SB\_HTTP\_REQ** queue.
2. Here we are performing routing mechanism based on the account opening check property of the incoming message, which is again includes 3 different checks :
3. KYC\_CHECK
4. FCM\_CHECK
5. Zss / Postilion CHECK
6. Once a message is reached here, it will be logged first into a log file and then in Database, Then the same input message will be propagated to the label.
7. If the request is for **KYC** then we first check with the Data warehouse based on the national ID by hitting a select query, if we found the data about the customer then we will respond the same data to the channel else we will check with Econet and send the responded data as it is.
8. If we doesn’t found data of the customer neither in data warehouse nor in Econet then we will send a failure response to output queue (**ALL\_SB\_HTTP\_RES**) by saying “no data found for the given customer”.
9. If the request is for **FCM** then we will frame a request for T24 and will send the response as per the T24 response to the output queue (**ALL\_SB\_HTTP\_RES**).
10. If the request is for **Update** then will convert the given JSON request to ISO and will call ZSS server to update the Customer details, then again the same request in ISO will be sent to Postilion Server to update the customer details then based on the response code from ZSS and Postilion will map the JSON and send the response to the output queue (**ALL\_SB\_HTTP\_RES**).

**Router Flow : OutgoingRouterFlow.msgflow**



**Process Flow:**

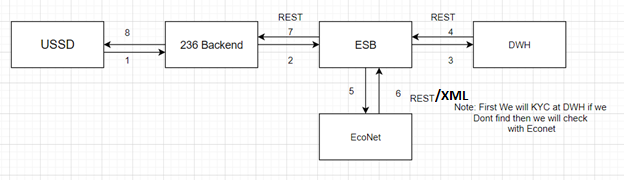
1. Messages propagated to ALL\_SB\_HTTP\_RES queue which is an input queue to the “**HttpResponseRouterFlow**”.
2. The Compute node “**GenerateResToFrontEnd**” is used to route the message to the frontend without any change by mapping RequestIdentifier in the LocalEnvironment.
3. If any exception occurs will be handled by exception handling subflow.

## High Level Architecture

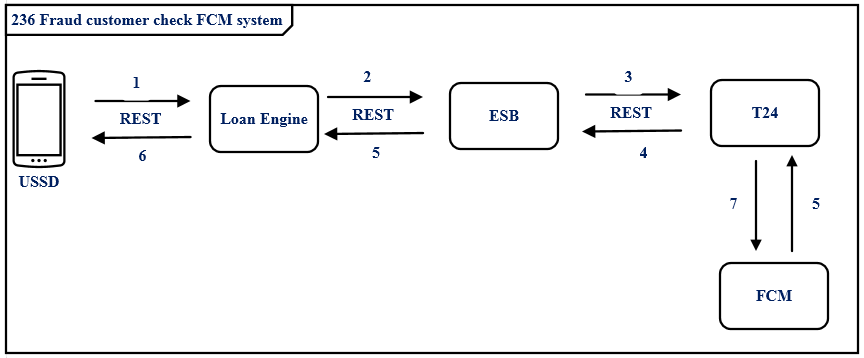
This process includes 3 steps

1. Confirming the KYC of the customer
2. Fraud Customer Check with FCM System
3. Updating the Account Details in ZSS/Postilion
4. **KYC Customer:** This flow is to check   
   whether the customer has KYC in DWH or ECONET.

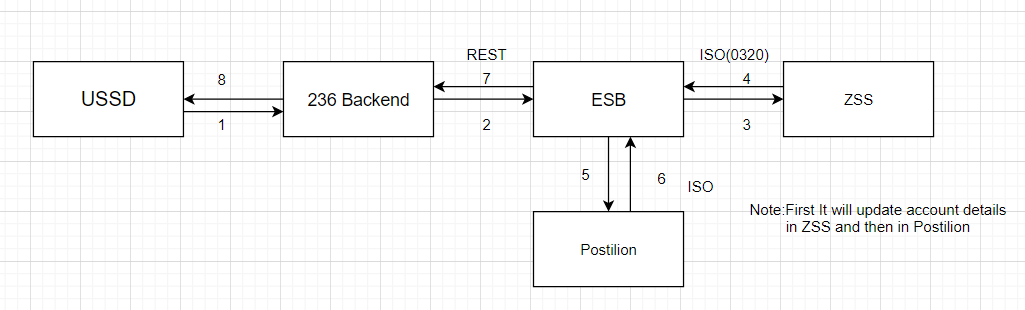
If it can find KYC in the DWH then no need of calling ECONET



1. **FCM:** Once we send the KYC found in either ECONET or DWH then Customer will confirm whether that KYC belongs to him or not, If he confirm then FCM call will takes place to check whether customer is fraud or not.



3. **Update ZSS and Postilion**: If the Customer is not fraud then need to update the Account Details assigned for the customer in ZSS and Postilion.



## Detailed Architecture

## Dependencies

### External Dependencies

The following table lists external dependencies in the ESB solution perspective.

|  |  |
| --- | --- |
| # | External Dependency |
|  | Data warehouse |
|  | Eco net |
|  | Temenos(T24) |
|  | ZSS |
| 1. P | Postilion server |

### Internal Dependencies

The following table lists internal dependencies in the ESB solution perspective.

|  |  |
| --- | --- |
| # | Internal Dependency |
|  | MQ Server |
|  | DB Server |

### Request from Steward Bank

1. KYC Request :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | XML Element | Data Type | Mandatory | Length of the Field |
|  | channel | String | M |  |
| 2 | P processingCode | String | M | 6 |
| 3 | accountOpeningCheck | String | M |  |
| 4 | National\_Id | String | M |  |
| 5 | requestId | String | O |  |
| 6 | transactionId | String | O |  |
| 7 | featureId | String | O |  |
| 8 | timeStamp | String | O |  |
| 9 | channelId | String | O |  |
| 10 | languageId | String | O |  |
| 11 | username | String | M |  |
| 12 | password | String | M |  |
| 13 | name | String | O |  |
| 14 | value | String | O |  |

1. FCM Request :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | JSON Element | Data Type | Mandatory | Length of the Field |
|  | channel | String | M |  |
| 2 | P processingCode | String | M | 6 |
| 3 | accountOpeningCheck | String | M |  |
| 4 | customerId | String | M |  |
| 5 | productId | String | O |  |
| 6 | currencyId | String | O |  |
| 7 | activityId | String | O |  |

1. Zss / Postilion Update Request :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | JSON Element | Data Type | Mandatory | Length of the Field |
|  | channel | String | M |  |
| 2 | P processingCode | String | M | 6 |
| 3 | accountOpeningCheck | String | M |  |
| 4 | mobileNumber | String | M |  |
| 5 | idNumber | String | O |  |
| 6 | accountNumber | String | O |  |
| 7 | accountType | String | O |  |
| 8 | expiryDate | String | O |  |
| 9 | actionType | String | O |  |
| 10 | title | String | O |  |
| 11 | firstname | String | O |  |
| 12 | lastname | String | O |  |
| 13 | middlename | String | O |  |
| 14 | birthDate | String | O |  |
| 15 | emailAddress | String | O |  |
| 16 | maritalStatus | String | O |  |
| 17 | gender | String | O |  |
| 18 | identificationType | String | O |  |
| 19 | nationality | String | O |  |
| 20 | residence | String | O |  |
| 21 | city | String | O |  |
| 22 | address1 | String | O |  |
| 23 | address2 | String | O |  |
| 24 | country | String | O |  |
| 25 | branch | String | O |  |
| 26 | currency | String | O |  |
| 27 | bankingServicesStatus | String | O |  |
| 28 | squareReqStatus | String | O |  |
| 29 | t24CustomerId | String | O |  |
| 30 | detailsAmended | String | O |  |
| 31 | sbPostilionStatus | String | O |  |
| 32 | zssPostilionStatus | String | O |  |

### Response

1. KYC Response :
2. When we found data in data warehouse (DWH).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | XML Element | Data Type | Mandatory | Length of the Field |
|  | channel | String | M |  |
| 2 | P processingCode | String | M | 6 |
| 3 | accountOpeningCode | String | M |  |
| 4 | status | String | M |  |
| 5 | responseCode | String | M | 3 |
| 6 | firstname | String | O |  |
| 7 | lastname | String | O |  |
| 8 | dob | String | O |  |
| 9 | gender | String | O |  |
| 10 | address | String | O |  |

1. When we do not find data in data warehouse (DWH).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | XML Element | Data Type | Mandatory | Length of the Field |
|  | code | String | M |  |
| 2 | P desc | String | M | 6 |
| 3 | name | String | M |  |
| 4 | value | String | M |  |

1. FCM Response :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | JSON Element | Data Type | Mandatory | Length of the Field |
|  | transactionStatus | String | M |  |
| 2 | P aaaId | String | M |  |
| 3 | status | String | M |  |
| 4 | type | String | O |  |
| 5 | fieldName | String | O |  |
| 6 | code | String | O |  |
| 7 | Message | String | O |  |

1. Zss / Postilion Update Response :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | JSON Element | Data Type | Mandatory | Length of the Field |
|  | status | Stirng | M |  |
| 2 | P rresponseCode | String | M | 3 |
| 3 | Account | String | M |  |
| 4 | mobileNumber | String | M |  |
| 5 | sbPostilionResponseCode | String | M |  |
| 6 | zssPostilionResponseCode | String | M |  |

### Error Handling

A Common flow by name "**StewardBankExceptionHandling**" is designed so as to receive any exception and parse it. Upon parsing a corresponding database entry is made against the particular transaction. Subsequent to that an error log is updated with error description and with the interface specific name.

### Retry Mechanism

### Reference Data

|  |  |
| --- | --- |
| **Domain** | **Reference Data Table** |
| XML | <request>  <channel>236</channel>  <processingCode>292000</processingCode>  <accountOpeningCheck>KYC\_CHECK</accountOpeningCheck>  <national\_Id>59029719M32</national\_Id>  <requestId>2</requestId>  <transactionId>5437654328765435</transactionId>  <featureId>search\_msisdn</featureId>  <timeStamp>1535638968307</timeStamp>  <channelId>6</channelId>  <languageId>1</languageId>  <username>Steward</username>  <password>@St3ward</password>  <data>  <param>  <name>msisdn</name>  <value>777991604</value>  </param>  </data>  </request> |
| JSON | {  "channel": "236",  "processingCode": "292000",  "accountOpeningCheck": "Zss/Postilion Update",  "mobileNumber": "sample",  "idNumber": "59029719M32",  "accountNumber": "sample",  "accountType": "sample",  "expiryDate": "2707",  "actionType": "sample",  "title;": "sample",  "firstname": "sample",  "lastname": "sample",  "middlename": "sample",  "birthDate": "sample",  "emailAddress": "sample",  "maritalStatus": "sample",  "gender": "sample",  "identificationType": "sample",  "nationality": "sample",  "residence": "sample",  "city": "sample",  "address1": "sample",  "address2": "sample",  "country": "sample",  "branch": "sample",  "currency": "ZMW",  "bankingServicesStatus": "sample",  "squareRegStatus": "sample",  "t24CustomerId": "sample",  "detailsAmended": "N",  "sbPostilionStatus": "sample",  "zssPostilionStatus": "sample"  } |

# Database Definitions

## Scripts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TableName** | **Sl.No** | **Column Name** | **Data**  **Type** | **Purpose** |
| **INTEGRATION\_SOL\_DETAILS** | **1** | PROC\_CODE | VARCHAR2(20 BYTE) | Contains the processing code of the each transaction. Which is to be mapped with the input field named processingCode |
| **2** | HTTP\_REQ | VARCHAR2(20 BYTE) | Contains the queue name of the input queue (request queue) of each interface. |
| **3** | HTTP\_RES | VARCHAR2(20 BYTE) | Contains the queue name of the output queue (response queue) of each interface. |
| **4** | MSGDOMAIN | VARCHAR2(20 BYTE) | Contains the domain of the message. |
| **5** | CHANNEL | VARCHAR2(20 BYTE) | Contains channel name. |
| **TableName** | **Sl.No** | **Column Name** | **Data**  **Type** | **Purpose** |
| **RAW\_AUDIT\_LOGGING** | **1** | MSGID | VARCHAR2(100 BYTE) | Contains the request identifier for each transaction details. |
| **2** | LOGGING\_TIME | TIMESTAMP(6) | Contains the timestamp, to know when the message is logged. |
| **3** | MESSAGE | CLOB | Contains the received request from the channel. |
| **4** | MESSAGETYPE | VARCHAR2(100 BYTE) | Contains the type of message received (example: T24 Request, Application Request, Application Response and T24 Response). |
| **5** | APPNAME | VARCHAR2(100 BYTE) | Contains the name of the application. |
| **6** | BROKER | VARCHAR2(100 BYTE) | Contains the name of the broker on which the request has been landed. |
| **TableName** | **Sl.No** | **Column Name** | **Data**  **Type** | **Purpose** |
| ERROR\_AUDIT\_LOGGING | **1** | MSGID | VARCHAR2(100 BYTE) | Contains the unique number for each transaction details. |
| **2** | LOGGING\_TIME | TIMESTAMP(6) | Contains the timestamp, to know when the message was received. |
| **3** | MESSAGE | CLOB | Contains the received request from the channel. |
| **4** | MESSAGETYPE | VARCHAR2(100 BYTE | Contains the type of message is received (example: T24 Request, Application Request, Application Response and T24 Response). |
| **5** | APPNAME | VARCHAR2(100 BYTE | Contains the name of the application. |
| **6** | BROKER | VARCHAR2(100 BYTE | Contains the name of the broker on which the request has been landed. |
| **7** | ERRORDESCRIPTION | CLOB | Contains the details of the error and its description. |

# Queue Definitions

## Scripts

|  |  |  |  |
| --- | --- | --- | --- |
| **QM Objects** | **Script** | | |
| Queues | Name | Purpose | Script Combined |
| ACOP\_236\_SB\_HTTP\_REQ | Receives the Messages from Channel via Router to ESB AccountOpeningFlow. | <Final Script Will be placed, after complete Unit Testing> |
| ALL\_SB\_HTTP\_RES | This is the Queue used to send to response back to the channel via router. |
| RAW\_LOG\_SB\_REQ | This queue is used to DB Log the successful transactions into the database. |
|  | ERR\_LOG\_SB\_REQ | This queue is used to DB Log the error/unsuccessful transactions into the database. |  |
|  | SB\_EXCQ | The common flow ‘StewardBankExceptionHandling’ uses this queue to store the exception details. |  |
|  | EXC\_Q | This queue is used to store the exceptions occurred in the flows. |  |

|  |  |
| --- | --- |
| **Nr of retries** | **Comments/Details** |
| No of Retries | 3 in Configurable. |
| Time gap between retries | 5000 in Configurable. |

# Integration Definition

### IntegrationDetails

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Postilion** | | **Temenos** | |
| **Protocol/Format** | ISO8583/WMQ/DFDL | | ISO8583/WMQ/DFDL | |
| **Mode** | Sync | | Sync | |
| **Endpoint Info** | Localhost:7080 | | Localhost:7080 | |
| **Samples Messages** | XML File | | XML File | |
| Channel Request |  | Request to Econet |  |
| Outward Response to Channel |  | Response from Econet |  |
| **Resources**  **(XSD)** | POSTILION Request |  | Request to T24 |  | |
| Outward Response to POSTILION |  | Response from T24 |  | |

# Requirements

## Integration Requirements

This interface will perform these levels of validation:

|  |  |  |
| --- | --- | --- |
| # | Integration Requirement | Comments / Details |
| 1 | Schema Validation Requirements | Inherent with XSDs and message model of ISO8583\_1987. |
| 2 | Records Validation Requirements | According to the requirements. |
| 3 | Transformation Requirements | There are no transformations, the message is from ISO to ISO throughout the flow. |
| 4 | Routing Requirements | Handled by the IncomingRouter and OutgoingRouter message flows. |
| 5 | Enrichment Requirements | According to the requirements. |
| 6 | Logging Requirements | Database Centric |
| 7 | Exception Handling Requirements | See “Error Handling” section above. |
| 8 | Retry Mechanism | Configurable |

## Business Requirements

|  |  |  |
| --- | --- | --- |
| # | Business Requirement | Comments / Details |
|  | N/A |  |

## 

## Non-functional Requirements

The following are service specific non-functional requirements. Non-functional requirements already provided by the overall solution, as described in the Technical Design document, should not be referred here.

### Volume

|  |  |  |
| --- | --- | --- |
| # | Requirement | Detail |
| 1 | Records / Transaction / Files |  |
| 2 | Message Load size |  |

### Performance

|  |  |  |
| --- | --- | --- |
| # | Requirement | Detail |
| 1 | Frequency | Daily |

### Security

|  |  |  |
| --- | --- | --- |
| # | Requirement | Detail |
| 1 | Message security | All outgoing messages are digitally signed and sent. |
| 2 | Authentication | None |
| 3 | Authorization | None |