Swift coming Interface Specification Low Level Design Document

**EIDIKO SYSTEMS INTEGRATORS**



Version: 1.0

Release Date: 01-09-2020

|  |  |  |
| --- | --- | --- |
| Prepared By | Reviewed By | Approved By |
| EIDIKO |  |  |

Document Classification: CONFIDENTIAL

Contents

[1 Introduction 5](#_Toc50411631)

[1.1 General Description 5](#_Toc50411632)

[2 Purpose 5](#_Toc50411633)

[3 Scope 5](#_Toc50411634)

[3.1 In Scope 5](#_Toc50411635)

[4 Channels Involved 5](#_Toc50411636)

[5 Interface Dependencies 5](#_Toc50411637)

[5.1 External Dependencies 5](#_Toc50411638)

[5.2 Internal Dependencies 6](#_Toc50411639)

[6 Business Process Summary 6](#_Toc50411640)

[6.1 Process Overview 6](#_Toc50411641)

[Figure: 1 Process flow Approach for Swift Outgoing 7](#_Toc50411642)

[6.2 Swift Incoming Steps 7](#_Toc50411643)

[7 ESB Implementation Process Flow 7](#_Toc50411644)

[7.1 Swift Outgoing Request Flow 7](#_Toc50411645)

[7.1.1 Process Diagram 7](#_Toc50411646)

[Figure: 3 Swift Outgoing Flow. 7](#_Toc50411647)

[7.1.2 Process Flow Steps 7](#_Toc50411648)

[7.2 ERROR\_AUDIT Process Flow 8](#_Toc50411649)

[7.2.1 Process Flow Diagram 8](#_Toc50411650)

[7.2.2 Process Flow Steps 9](#_Toc50411651)

[7.3 Raw\_Audit Process Flow 9](#_Toc50411652)

[7.3.1 Process Flow Diagram 9](#_Toc50411653)

[7.3.2 Process Flow Steps 9](#_Toc50411654)

[7.3.3 Data Base Response Codes 10](#_Toc50411655)

[8 Interface Definitions 10](#_Toc50411656)

[8.1 Request Message Definition 10](#_Toc50411657)

[8.1.1 MT Request Message Structure /Schema 10](#_Toc50411658)

[9 Queue Details 11](#_Toc50411659)

[9.1 Queue Details Table 11](#_Toc50411660)

[10 Logging Mechanism 12](#_Toc50411661)

[10.1 Insert into Error\_Audit\_Table 12](#_Toc50411662)

[10.2 Insert into Raw\_Audit\_Table 13](#_Toc50411663)

[10.3 File Based Logging 14](#_Toc50411664)

[10.3.1 Log4j: 14](#_Toc50411665)

**Approval**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Project Role | Approver | Company | E-mail | Sign-Off Date |
| Delivery Manager |  | Eidiko |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Description | Version | Author |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## General Description

This Interface Design document outlines the integration requirements for the Swift Outgoing ***(Request and Response)***interface used as banking operations. It summarizes the business processes, which use this interface. It also covers error handling and exception scenarios.

# Purpose

The purpose of this document is to capture events that trigger the interface, main steps within the interface and the integration architecture. This document is intended for use by the developers of the applications identified, the integration development team, and by the test organizations responsible for the testing of these applications.

# Scope

## In Scope

This document focuseson outlining the interface design for the Swift Outgoing***(Request and Response)***interface. Central to this document are the following:

* Overview of the business process that drives the need for the interface
* Proposed integration approach
* Trigger events and business dependencies on this interface
* Sequence /Flow Chart diagram of the interface
* Data mapping between the source and the target
* Error handling and exception scenarios
* Validation and backup requirements

# Channels Involved

The following table lists Swift Outgoing channels

|  |  |
| --- | --- |
| **Item #** | **Channel Name** |
| 1 | Batch-Swift Allience |

# Interface Dependencies

## External Dependencies

The following table lists interface specific External requirements.

|  |  |
| --- | --- |
| **Item #** | **External Requirements** |
| 1 | MS SQL Database - 2017 |
| 2 | Core Banking System(Temenos-T24) |
| 3 | FTP |

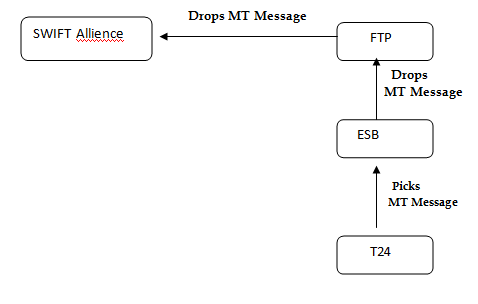
## Internal Dependencies

The following table lists interface specific internal requirements.

|  |  |
| --- | --- |
| **Item #** | **Internal Requirements** |
| 1 | ISO8583\_Lib (Shared Library) |
| 2 | StewardBankCommonEsql (Shared Library) |
| 3 | DB Logging App (Audit Logging) |
| 4 | StewardBankExceptionHandler (Subflow) |

# Business Process Summary

## Process Overview



### Figure: 1 Process flow Approach for Swift Outgoing

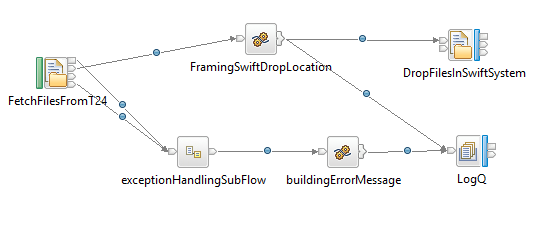
## Swift Incoming Steps

|  |  |
| --- | --- |
| **S. No** |  |
| 1 | ESB picks Mt message from SFTP location (T24 ) . |
| 2 | Then ESB drops MT103 message in FTP location (Swift Alliance). |

# ESB Implementation Process Flow

## Swift Outgoing Request Flow

### Process Diagram



### Figure: 3 Swift Outgoing Flow.

### Process Flow Steps

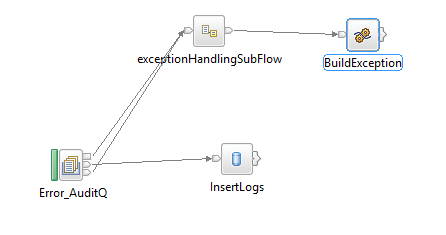
|  |  |
| --- | --- |
| **S. No** |  |
| 1 | ESB picks Mt message from SFTP location and reads as BLOB. |
| 2 | After that, the request is logged in RAW\_LOG\_SB\_REQ queue and in RAW\_AUDIT\_TABLE database table. |
| 3 | Then ESB drops MT103 message in FTP location at SWIFT Alliance. |
| 4 | On Unsuccessful transaction, those details are logged in ERROR\_LOG\_SB\_REQ Queue and ERR\_AUDIT\_LOGGING database table. |

## ERROR\_AUDIT Process Flow

### Process Flow Diagram



**Figure: 4 Exception Subflow**



**Figure: 5 Error\_Audit\_Logging Flow**

### Process Flow Steps

|  |  |
| --- | --- |
| **S. No** | **Activities** |
|  | The ERROR\_AUDIT\_Q logs the input Request. |
|  | After the query is executed the record is inserted into the Data Base. |
|  | Final Response is inserted into the DataBase and then Response is logged into the ERR\_LOG\_SB\_REQ. |

## Raw\_Audit Process Flow

### Process Flow Diagram



**Figure: 6 Raw\_Audit\_Logging Flow**

### Process Flow Steps

|  |  |
| --- | --- |
| **S. No** | **Activities** |
|  | The RAW\_AUDIT logs the input Request and Response |
|  | After the query is executed the record is inserted into the Data Base. |
|  | Final Response is inserted into the Data Base and then Response is logged into the RAW\_LOG\_SB\_REQ. |

### Data Base Response Codes

001 = "Fatal Exception";

002 = “Recoverable Exception ";

003 = "Configuration Exception";

004 = "Security Exception";

005 = "Parser Exception";

006 = "Conversion Exception";

007 = "Data Base Exception";

008 = "User Exception";

009 = "Cast Exception";

010 = "Message Exception";

011 = "SQL Exception";

012 = "Socket Exception";

013 = "Socket Timeout Exception";

014 = "Unknown Exception";

015 = "Failure";

# Interface Definitions

## Request Message Definition

### MT Request Message Structure /Schema

**Request Type:** Message Model(DFDL)

**Request Format:** MT

**Request MT103 Sample File:**

{1:F01STBLZWHXAXXX2069700773}{2:O1031434200305COBZZWHAAXXX78060642442003051437N}{3:{103:ZET}{113:PROP}{115:CBZBP20065A0508020200305143014}}{4:

:20:029FTIB200650184

:23B:CRED

:32A:200305ZWL2500,

:33B:ZWL5000,

:50K:/23550950010

ZIMBIRU SECONDARY SCHOOL P O BOX BW

1120 BORROWDALE HARARE

:52A:COBZZWHA

:53A:COBZZWHA

:54A:STBLZWHX

:56A:COBZZWHA

:57A:STBLZWHX

:59:/1000000163

JOHNSON G GABARINOCHEKA harare harare

:71A:OUR

:72:/REC/CTP /RRF/JOHNSON G GABARINOCHEK

-}{5:{CHK:65D52019DAEC}}{S:{SAC:}{FAC:}{COP:P}}

# Queue Details

## Queue Details Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Queue Objects** | **Script** | | |
| Queues | Name | Purpose | Script Combined |
| RAW\_LOG\_SB\_REQ | This queue is used to DB Log the successful transactions into the database. | <Final Script Will be placed, after complete Unit Testing> |
|  | 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. |

# Logging Mechanism

## Insert into Error\_Audit\_Table

**Table Name: ERROR\_AUDIT\_TABLE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item #** | **DB Field Name** | **Field Description** | **Data Type** | **Length** | **Mandatory(Yes/No)** | **Comments** |
| 1 | MSGID | MessageID | Varchar | 50 | Y | From input request |
| 2 | LOGGING\_TIME | Application Name | Timestamp | 50 | Y | Time of the logging |
| 3 | MESSAGE | Message Type(Request/Response) | Clob | 50 | Y | Request of the Message |
| 4 | MESSAGETYPE | Message Type | Varchar | 50 | Y | Type of message either Request or Response |
| 5 | APPNAME | Application Name | Varchar | 4000 | Y | Name of the Application |
| 6 | BROKER | Broker Name | Varchar | - | Y | Name of the Broker |
| 7 | ERRORDESCRIPTION | Exception | Clob | 4000 | Y | Exception Information |
| 8 | TIME\_LOCAL\_TRANSACTION | Transaction Time | Varchar | 20 | Y | Time of the Transaction |
| 9 | DATE\_LOCAL\_TRANSACTION | Transaction Date | Varchar | 20 | Y | Date of the Transaction |
| 9 | RETRIEVAL\_REFERENCE\_NUMBER | Retrieval Reference Number | Varchar | 20 | Y | Retrieval Reference Number |
| 10 | BORKER\_ERROR\_CODE | Broker Error Code | Varchar |  | Y | Error code generate by Broker |
| 11 | USER\_DEFINE\_ERROR\_CODE | User Defined Error Code | Varchar |  | Y | User Defined Code |

## Insert into Raw\_Audit\_Table

**Table Name: RAW\_AUDIT\_TABLE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item #** | **DB Field Name** | **Field Description** | **Data Type** | **Length** | **Mandatory(Yes/No)** | **Comments** |
| 1 | MSGID | MessageID | Varchar | 100 | Y | From input request |
| 2 | LOGGING\_TIME | Logging Time | TimeStamp | 6 | Y | Name |
| 3 | MESSAGE | Message(Request/Response) | Clob | - | Y | Request from channel |
| 4 | MESSAGETYPE | Type of Message(Request/Response) | Varchar | 100 | Y | Type of message either Request or Response |
| 5 | APPNAME | Application Name | Varchar | 100 | Y | Name of the Application |
| 6 | BROKER | Broker Name | Varchar | 100 | Y | Broker Name |
| 7 | TIME\_LOCAL\_TRANSACTION | Transaction Time | Varchar | 20 | Y | Time of the Transaction |
| 8 | DATE\_LOCAL\_TRANSACTION | Transaction Date | Varchar | 20 | Y | Date of the Transaction |
| 9 | RETRIEVAL\_REFERENCE\_NUMBER | Retrieval Reference Number | Varchar | 20 | Y |  |

## File Based Logging

### Log4j:

As part of auditing the request we have to use Log4j mechanism in the application to log the request in the file.

File Path: To be specified, while deploying the application