Remittance Bank Deposit Interface Specification Low Level Design Document

**EIDIKOSYSTEMS INTEGRATORS**



Version: 1.0

Release Date:11 -08 -2020

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

Document Classification: CONFIDENTIAL

Contents

[1 Introduction 5](#_Toc49283220)

[1.1 General Description 5](#_Toc49283221)

[2 Purpose 5](#_Toc49283222)

[3 Scope 5](#_Toc49283223)

[3.1 In Scope 5](#_Toc49283224)

[4 Channels Involved 5](#_Toc49283225)

[5 Interface Dependencies 6](#_Toc49283226)

[5.1 External Dependencies 6](#_Toc49283227)

[5.2 Internal Dependencies 6](#_Toc49283228)

[6 Business Process Summary 7](#_Toc49283229)

[6.1 Process Overview 7](#_Toc49283230)

[Figure:1 Process flow Approach for Remittance Bank Deposit. 7](#_Toc49283231)

[6.1.1 Remittance Bank Deposit Out Steps 7](#_Toc49283232)

[7 IIB Implementation Process Flow 8](#_Toc49283233)

[7.1 Remittance Bank Deposit Successful Request Response Flow 8](#_Toc49283234)

[7.1.1 Process Diagram 8](#_Toc49283235)

[7.1.2 Process Flow Steps 8](#_Toc49283236)

[7.2 Retry Process workflow 9](#_Toc49283237)

[7.2.1 Process Flow Diagram 9](#_Toc49283238)

[7.2.2 Process Flow steps 9](#_Toc49283239)

[8 Interface Definitions 15](#_Toc49283240)

[8.1 Request Message Definition 15](#_Toc49283241)

[8.1.1 Channel (Remittance Partner) Request Message Structure /Schema 15](#_Toc49283242)

[8.1.2 Remittance Partner Request Message Details 15](#_Toc49283243)

[8.1.3 Remittance Partner Sample Source Messages 16](#_Toc49283244)

[8.1.4 T24 Request Message Structure /Schema 17](#_Toc49283245)

[8.1.5 T24 Request Message Details 17](#_Toc49283246)

[8.1.6 T24 Sample Source Messages 18](#_Toc49283247)

[8.1.7 Data Base Response Codes 18](#_Toc49283248)

[8.2 Response Message Definition 19](#_Toc49283249)

[8.2.1 T24 Response Message Structure/Schema 19](#_Toc49283250)

[8.2.2 T24 Response Message Details 19](#_Toc49283251)

[8.2.3 T24 Sample Response Messages 20](#_Toc49283252)

[8.2.5 Remittance Partner Response Message Details 21](#_Toc49283253)

[8.2.6 Remittance Partner Sample Response Messages 22](#_Toc49283254)

[9 Logging Mechanism 23](#_Toc49283255)

[9.3 File Based Logging 27](#_Toc49283256)

[9.3.1 Log4j 27](#_Toc49283257)

**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 Remittance Bank Transfer ***(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 focuses on outlining the interface design for the Remittance Bank Transfer ***(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 Remittance Bank Transfer channels

|  |  |
| --- | --- |
| **Item #** | **Channel Name** |
| 1 | Remittance |

# Interface Dependencies

## External Dependencies

The following table lists interface specific External requirements.

|  |  |
| --- | --- |
| **Item #** | **External Requirements** |
| 1 | ORADSN DataBase |
| 2 | Core Banking System |

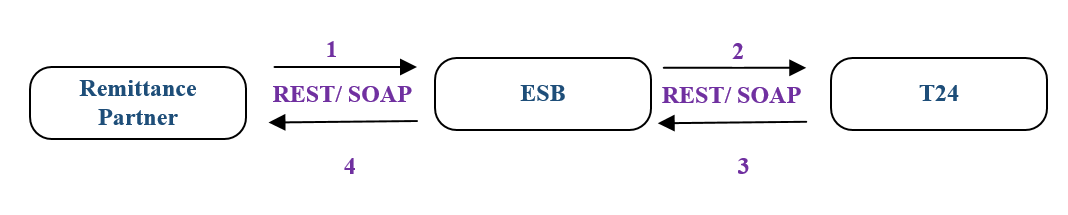
## Internal Dependencies

The following table lists interface specific internal requirements.

|  |  |
| --- | --- |
| **Item #** | **Internal Requirements** |
| 1 | HTTP Router Interface |
| 2 | HTTP Retry Interface |
| 3 | DB Logging App |
|  |  |

# Business Process Summary

## Process Overview



### Figure:1 Process flow Approach for Remittance Bank Deposit.

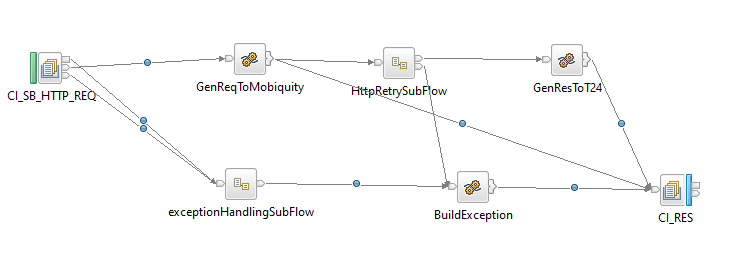
### Remittance Bank Deposit Out Steps

|  |  |
| --- | --- |
| **S. No** | **Activities** |
| 1 | ESB receives the request from channel(Remittance Partner). |
| 2 | Request channel is the http request accepts the request from the channel |
| 3 | ESB frames the Request According to the T24 requires and sends to the T24. |
| 4 | T24 will Respond back to the ESB with Response |
| 5 | The same T24 Response, ESB will send back to the T24 |
| 6 | On Successful calls of T24 and Channel Request and Response will be placed in the RAW\_LOG\_SB\_REQ to log the request in Data Base (RAW\_AUDIT\_LOGGING |
| 7 | On Unsuccessful calls of Channel and T24 Request and Response will be placed in the ERROR\_LOG\_SB\_REQ to log the request in Data Base (ERR\_AUDIT\_LOGGING |

# IIB Implementation Process Flow

## Remittance Bank Deposit Successful Request Response Flow

### Process Diagram



**Figure:2 Remittance Bank Deposit Flow**

### Process Flow Steps

|  |  |
| --- | --- |
| **S. No** | **Activities** |
| 1. | ESB receives the request from the channel(Remittance Partner) using the HTTP Router Application based on processingCode trigs the Business Application (RemittanceBankDeposit) using Business Queue (REM\_SB\_HTTP\_REQ) i.e Request from channel. |
| 2 | Initial request and response are logged in the RAW\_LOG\_SB\_REQ and is insert in Data Base (RAW\_AUDIT\_LOGGING) |
| 3 | After logging of request and response ESB frames the Request to backend server i.e T24 server |
| 4 | After Framing the T24 Request is logged in RAW\_LOG\_SB\_REQ and is insert in Data Base(RAW\_AUDIT\_LOGGING) |
| 5 | If the T24 call is successful, then JSON response is generate. |
| 6 | T24 Response in JSON Format is logged in RAW\_LOG\_SB\_REQ |
| 7 | After ever the Response is generated by T24 call same response is sending back to Channel with the help of HTTP Router Application. Here ESB frames the Channel Response (T24 Response) |
| 8 | T24 Response is logged in RAW\_LOG\_SB\_REQ and is insert in Data Base (RAW\_AUDIT\_LOGGING) |
| 9 | On Unsuccessful calls of channel and T24 Request and Response will be placed in the ERROR\_LOG\_SB\_REQ to log the request in Data Base (ERR\_AUDIT\_LOGGING) |

## Retry Process workflow

### Process Flow Diagram



**Figure:3 HTTP Retry for Remittance Bank Deposit**

### Process Flow steps

|  |  |
| --- | --- |
| **S. No** | **Activities** |
| 1 | HTTP Retry call is used to call the backend service for multiple times |
| 2 | On Successful call of backend service, response will be generated and send back to Business Application For further process |
| 3 | On Unsuccessful call of backend service it Retry for 3 times and sends error message to Business Application for further Process |

**7.3 Raw\_Audit Process Flow:**

**7.3.1 Process Flow Diagram:**



**Figure:4 Raw\_Audit\_Logging Flow**

**7.3.2 Process Flow Steps:**

|  |  |
| --- | --- |
| **S. No** | **Activities** |
|  | The RAW\_AUDIT locks 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. |

**7.4 HTTP Router Process Flow**

**7.4.1 Incoming Router Process Flow Diagram:**



**Figure:5 Incoming Router Flow**

**7.4.2 Process Flow Steps:**

|  |  |
| --- | --- |
| **S. No** | IIB receives the request from channel. |
|  | ESB receives the request from channel. |
|  | Request channel is the http request accepts the request from the channel |
|  | The request message is placed in the RAW\_LOG\_SB\_REQ to log the request in Data Base (RAW\_AUDIT\_LOGGING) |
|  | According to the Channel Request message fields such as channel and processing code it picks the queue name of the business flow from INTEGRATION\_SOL\_DETAILS table and propagates it to the business flow. |
|  | It validates the processingCode, Channel and domain of input from INTEGRATION\_SOL\_DETAILS if it fails to validate the error response is logged in the logs and in Data Base (ERROR\_AUDIT\_LOG) |
|  | After the validation Request message is passed through ESB. If the queue name presents in MQ manager the ESB business flow triggers otherwise the error response is sent to the front end as Unable to open queue. |
|  | If it is success or failure response will store them in RAW\_AUDIT\_LOG table or ERROR\_AUDIT\_LOG and logging into log file. |
|  | The request from channel is sent to ESB Remittance Bank Deposit Flow |

**7.4.3 Integration Solution Details**

**Table Name: INTEGRATION \_AUDIT\_TABLE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item #** | **DB Field Name** | **Field Description** | **Data Type** | **Length** | **Mandatory(Yes/No)** | **Comments** |
| 1 | PROC\_CODE | Processing Code | Varchar | 20 | Y | From input request |
| 2 | REQ\_QUEUE | TCPIP Request Queue | Varchar | 20 | N | This Request Queue Name field is used for ISO Related Application |
| 3 | RES\_QUEUE | TCPIP Response Queue | Varchar | 20 | N | This Response Queue Name field is used for ISO Related Application |
| 4 | HTTP\_REQ | HTTP Request Queue | Varchar | 20 | Y | This Request Queue Name field is used for REST Related Application |
| 5 | HTTP\_RES | HTTP Response Queue | Varchar | 20 | Y | This Response Queue Name field is used for REST Related Application |
| 6 | CHANNEL | Channel | Varchar | 20 | Y | Channel Name |
| 7 | MSGDOMAIN | Message Domain | Varchar | 20 | Y | Request Message Format |

**SQL Query:**

CREATE TABLE "SYSTEM"."INTEGRATION\_SOL\_DETAILS"

( "PROC\_CODE" VARCHAR2(50 BYTE),

"REQ\_QUEUE" VARCHAR2(20 BYTE),

"RES\_QUEUE" VARCHAR2(20 BYTE),

"HTTP\_REQ" VARCHAR2(20 BYTE),

"HTTP\_RES" VARCHAR2(20 BYTE),

"CHANNEL" VARCHAR2(20 BYTE),

"MSGDOMAIN" VARCHAR2(20 BYTE)

);

Insert into SYSTEM.INTEGRATION\_SOL\_DETAILS (PROC\_CODE,REQ\_QUEUE,RES\_QUEUE,HTTP\_REQ,HTTP\_RES,CHANNEL,MSGDOMAIN) values ('23',null,null,'REM\_SB\_HTTP\_REQ','ALL\_SB\_HTTP\_RES','remittance','JSON');

**7.4.4 Outgoing Router Process Flow Diagram**



**Figure:6 Outgoing Router Flow**

**7.4.5 Process Flow Steps:**

|  |  |
| --- | --- |
| **S. No** | **Activities** |
|  | The OutgoingHttpRes brings the Channel Response |
|  | Channel Response will be logged in RAW\_LOG\_SB\_REQ and insert in Data Base. |
|  | Finally Channel Response will be send to the Front End. |

**7.5 Error\_Audit Process Flow**

**7.5.1 Process Flow Diagram:**



**Figure:7** **Exception Subflow**



**Figure:8** **Error\_Audit\_Logging Flow**

**7.5.2 Process Flow Steps:**

|  |  |
| --- | --- |
| **S. No** | **Activities** |
|  | The ERROR\_AUDIT\_Q locks the input Request. |
|  | 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 ERR\_LOG\_SB\_REQ. |

# Interface Definitions

## Request Message Definition

### Channel (Remittance Partner) Request Message Structure /Schema

**Request Type:** HTTP

**Request Format:** JSON

**Request URL:** <http://IPAddress:Port/v1/stewardBank/>

### 8.1.2 Remittance Partner Request Message Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Tag Name/ Field Name** | **Data Type** | **Mandatory(Yes/No)** | **Comments** |
| 1 | RemittanceBankDeposit |  | Y | Root Tag |
| 2 | Header |  | Y | Header Tag |
| 3 | Channel | String | Y | Channel Name |
| 4 | ProcessingCode | String | Y | Processing Code |
| 5 | override |  |  | Override Tag |
| 6 | overrideDetails |  |  | Override Details |
| 7 | Body | String | Y | Body Tag |
| 8 | narratives | String | Y | Narratives Tag |
| 9 | narrative | String | Y | Remitter Name |
| 10 | additionalInformations | Decimal | Y | Additional Informations Tag |
| 11 | additionalInformation | String | Y | Remittance Partner Name |
| 12 | orderingReference | String | Y | Ordering Reference |
| 13 | transactionAmount | String | Y | Transaction Amount |
| 14 | executionDate | String | Y | Execution Date |
|  | orderingCustomer |  |  | Ordering Customer |
| 15 | debitAccountId | String | Y | Debit Account |
| 16 | creditAccountId | String | Y | Credit Account |
| 17 | paymentCurrency |  | Y | Currency |
|  | paymentOrderProductId |  |  | Payment Order Product Id |

### 8.1.3 Remittance Partner Sample Source Messages

|  |
| --- |
| {  "RemittanceBankDeposit": {  "header": {  "processingCode": "230000",  "channel": "remittance",  "override": {  "overrideDetails": [  ]  }  },  "body": {  "narratives": [{  "narrative": "BANK OF LISBON"  }],  "additionalInformations": [{  "additionalInformation": "SADZA PARTNER"  }],  "orderingReference": "123REMIT",  "transactionAmount": 872.87,  "executionDate": " 20200814",  "orderingCustomer": " 100121",  "debitAccountId": "1000000497",  "creditAccountId": "1000000047",  "paymentCurrency": "ZWL",  "paymentOrderProductId": "REMIT.DEP"  }  }  } |

### 8.1.4 T24 Request Message Structure /Schema

**Request Type:** HTTP

**Request Format:** JSON

**Request URL:** http://IPAddress:Port/jigsaw/serviceRequest/CASHIN

### 8.1.5 T24 Request Message Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Tag Name/ Field Name** | **Data Type** | **Mandatory(Yes/No)** | **Comments** |
| 1 | Header |  | Y | Header Tag |
| 2 | override |  |  | Override Tag |
| 3 | overrideDetails |  |  | Override Details |
| 4 | Body | String | Y | Body Tag |
| 5 | narratives | String | Y | Narratives Tag |
| 6 | narrative | String | Y | Remitter Name |
| 7 | additionalInformations | Decimal | Y | Additional Informations Tag |
| 8 | additionalInformation | String | Y | Remittance Partner Name |
| 9 | orderingReference | String | Y | Ordering Reference |
| 10 | transactionAmount | String | Y | Transaction Amount |
| 11 | executionDate | String | Y | Execution Date |
| 12 | orderingCustomer |  |  | Ordering Customer |
| 13 | debitAccountId | String | Y | Debit Account |
| 14 | creditAccountId | String | Y | Credit Account |
| 15 | paymentCurrency |  | Y | Currency |
| 16 | paymentOrderProductId |  |  | Payment Order Product Id |

### 8.1.6 T24 Sample Source Messages

|  |
| --- |
| {  "header": {  "override": {  "overrideDetails": []  }  },  "body": {  "narratives": [  {  "narrative": "BANK OF LISBON"  }  ],  "additionalInformations": [  {  "additionalInformation": "SADZA PARTNER"  }  ],  "orderingReference": "123REMIT",  "transactionAmount": 872.87,  "executionDate": " 20200814",  "debitAccountId": "1000000497",  "orderingCustomer": " 100121",  "creditAccountId": "1000000047",  "paymentCurrency": "ZWL",  "paymentOrderProductId": "REMIT.DEP"  }  } |

### 8.1.7 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";

## Response Message Definition

### T24 Response Message Structure/Schema

**Response Type:** JSON

### T24 Response Message Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Tag Name/ Field Name** | **Data Type** | **Mandatory(Yes/No)** | **Comments** |
| 1 | Header |  | Y | Header Tag |
| 2 | audit | String | Y | Audit Tag |
| 3 | T24\_time | Integer | Y | T24 Time |
| 4 | parse\_time | Integer | Y | Parse Time |
| 5 | responseParse\_time | Integer | Y | Response Parse Time |
| 6 | requestParse\_time | Integer | Y | Request Parse Time |
| 7 | versionNumber | String | Y | Version Number |
| 8 | Id | String | Y | Id |
| 9 | Status | String | Y | Status of Transaction |
| 10 | Body | String | Y | Body Tag |
| 11 | narratives | String | Y | Narratives Tag |
| 12 | narrative | String | Y | Remitter Name |
| 13 | orderingReference | String | Y | Ordering Reference |
| 14 | transactionAmount | String | Y | Transaction Amount |
| 15 | executionDate | String | Y | Execution Date |
| 16 | additionalInformations | String | Y | Additional Informations Tag |
| 17 | additionalInformation | String | Y | Remittance Partner Name |
| 18 | orderingCustomer | String | Y | Ordering Customer |
| 19 | debitAccountId | Integer | Y | Debit Account |
| 20 | creditAccountId | Integer | Y | Credit Account |
| 21 | paymentCurrency | String | Y | Currency |
| 22 | paymentOrderProductId | String | Y | Payment Order Product Id |

### 8.2.3 T24 Sample Response Messages

**It**

|  |
| --- |
| {  "header": {  "transactionStatus": "Live",  "audit": {  "T24\_time": 362,  "parse\_time": 1,  "responseParse\_time": 1,  "requestParse\_time": 4,  "versionNumber": "1"  },  "id": "PI202330L5DW6LHC",  "status": "success"  },  "body": {  "orderingReference": "123REMIT",  "narratives": [  {  "narrative": "BANK OF LISBON"  }  ],  "orderingCustomerId": "100139",  "debitCurrency": "ZWL",  "transactionAmount": 872.87,  "endToEndReference": "NOTPROVIDED",  "executionDate": "20200820",  "additionalInformations": [  {  "additionalInformation": "SADZA PARTNER"  }  ],  "debitAccountId": "1000000497",  "creditAccountId": "1000000047",  "paymentCurrency": "ZWL",  "paymentOrderProductId": "REMIT.DEP"  }  } |

### Remittance Partner Response Message Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Tag Name/ Field Name** | **Data Type** | **Mandatory(Yes/No)** | **Comments** |
| RemittanceBankDeposit |  | Y | Root Tag |
| Header |  | Y | Header Tag |
| Channel | String | Y | Channel Name |
| ProcessingCode | String | Y | Processing Code |
| TransactionStatus | String | Y | Transaction Status |
| Audit | String | Y | Audit Tag |
| T24\_time | Integer | Y | T24 Time |
| parse\_time | Integer | Y | Parse Time |
| responseParse\_time | Integer | Y | Response Parse Time |
| requestParse\_time | Integer | Y | Request Parse Time |
| versionNumber | String | Y | Version Number |
| Id | String | Y | Id |
| Status | String | Y | Status of Transaction |
| Body | String | Y | Body Tag |
| Narratives | String | Y | Narratives Tag |
| Narrative | String | Y | Remitter Name |
| OrderingReference | String | Y | Ordering Reference |
| TransactionAmount | String | Y | Transaction Amount |
| executionDate | String | Y | Execution Date |
| AdditionalInformations | String | Y | Additional Informations Tag |
| AdditionalInformation | String | Y | Remittance Partner Name |
| OrderingCustomer | String | Y | Ordering Customer |
| debitAccountId | Integer | Y | Debit Account |
| creditAccountId | Integer | Y | Credit Account |
| PaymentCurrency | String | Y | Currency |
| PaymentOrderProductId | String | Y | Payment Order Product Id |

### 8.2.6 Remittance Partner Sample Response Messages

|  |
| --- |
| {      "RemittanceBankDeposit": {          "header": {              "channel": "remmitance",              "processingCode": "230000",              "responseCode": "200",              "transactionStatus": "Live",              "audit": {                  "T24\_time": 330,                  "parse\_time": 1,                  "responseParse\_time": 1,                  "requestParse\_time": 5,                  "versionNumber": "1"              },              "id": "PI202330L5DXTLJZ",              "status": "success"          },          "body": {              "orderingReference": "123REMIT",              "narratives": [                  {                      "narrative": "BANK OF LISBON"                  }              ],              "orderingCustomerId": "100139",              "debitCurrency": "ZWL",              "transactionAmount": 8.7287E+2,              "endToEndReference": "NOTPROVIDED",              "executionDate": "20200820",              "additionalInformations": [                  {                      "additionalInformation": "SADZA PARTNER"                  }              ],              "debitAccountId": "1000000497",              "creditAccountId": "1000000047",              "paymentCurrency": "ZWL",              "paymentOrderProductId": "REMIT.DEP"          }      }  } |

# Logging Mechanism

**9.1 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 |  |

**SQL Query:**

CREATE TABLE "SYSTEM"."RAW\_AUDIT\_LOGGING"

( "MSGID" VARCHAR2(100 BYTE),

"LOGGING\_TIME" TIMESTAMP (6),

"MESSAGE" CLOB,

"MESSAGETYPE" VARCHAR2(100 BYTE),

"APPNAME" VARCHAR2(100 BYTE),

"BROKER" VARCHAR2(100 BYTE),

"TIME\_LOCAL\_TRANSACTION" VARCHAR2(20 BYTE),

"DATE\_LOCAL\_TRANSACTION" VARCHAR2(20 BYTE),

"RETRIEVAL\_REFERENCE\_NUMBER" VARCHAR2(20 BYTE)

);

Insert into SYSTEM.RAW\_AUDIT\_LOGGING (MSGID,LOGGING\_TIME,MESSAGETYPE,APPNAME,BROKER,TIME\_LOCAL\_TRANSACTION,DATE\_LOCAL\_TRANSACTION,RETRIEVAL\_REFERENCE\_NUMBER) values ('414d5120514d20202020202020202020b6143e5f24b5672f',to\_timestamp('24-08-20 08:35:55.870821000 PM','DD-MM-RR HH12:MI:SSXFF AM'),'Remmitance BankDeposit Request','StewardBankRemittanceBankDepositApp','BRKR1','203550','2408',null);

**9.2 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 |

**SQL Query:**

CREATE TABLE "SYSTEM"."ERROR\_AUDIT\_LOGGING"

( "MSGID" VARCHAR2(100 BYTE),

"LOGGING\_TIME" TIMESTAMP (6),

"MESSAGE" CLOB,

"MESSAGETYPE" VARCHAR2(100 BYTE),

"APPNAME" VARCHAR2(100 BYTE),

"BROKER" VARCHAR2(100 BYTE),

"ERRORDESCRIPTION" CLOB,

"TIME\_LOCAL\_TRANSACTION" VARCHAR2(50 BYTE),

"DATE\_LOCAL\_TRANSACTION" VARCHAR2(20 BYTE),

"RETRIEVAL\_REFERENCE\_NUMBER" VARCHAR2(20 BYTE),

"BORKER\_ERROR\_CODE" VARCHAR2(20 BYTE),

"USER\_DEFINE\_ERROR\_CODE" VARCHAR2(50 BYTE)

);

Insert into SYSTEM.ERROR\_AUDIT\_LOGGING (MSGID,LOGGING\_TIME,MESSAGETYPE,APPNAME,BROKER,TIME\_LOCAL\_TRANSACTION,DATE\_LOCAL\_TRANSACTION,RETRIEVAL\_REFERENCE\_NUMBER,BORKER\_ERROR\_CODE,USER\_DEFINE\_ERROR\_CODE) values ('414d5120514d202020202020202020206a8e3b5f250b117d',to\_timestamp('18-08-20 08:33:20.990785000 PM','DD-MM-RR HH12:MI:SSXFF AM'),'Application Error','StewardBankRemmitanceBankDepositApp','BRKR1','203238','1808',null,'3150','RemmitanceBankDeposit\_012');

## 9.3 File Based Logging

### 9.3.1 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