Cassava PickUp Interface Specification Low Level Design Document

**EIDIKOSYSTEMS INTEGRATORS**



Version: 1.0

Release Date:24-08-2020

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

Document Classification: CONFIDENTIAL

Contents

[1 Introduction 5](#_Toc48249828)

[1.1 General Description 5](#_Toc48249829)

[2 Purpose 5](#_Toc48249830)

[3 Scope 5](#_Toc48249831)

[3.1 In Scope 5](#_Toc48249832)

[4 Channels Involved 5](#_Toc48249833)

[5 Interface Dependencies 6](#_Toc48249834)

[5.1 External Dependencies 6](#_Toc48249835)

[5.2 Internal Dependencies 6](#_Toc48249836)

[6 Business Process Summary 7](#_Toc48249837)

[6.1 Process Overview 7](#_Toc48249838)

[Figure:1 Process flow Approach for Cassava PickUp. 7](#_Toc48249839)

[6.1.1 Cassava PickUp Steps 7](#_Toc48249840)

[7 IIB Implementation Process Flow 8](#_Toc48249841)

[7.1 Cassava PickUp Successfull Request Response Flow 8](#_Toc48249842)

[7.1.1 Process Diagram 8](#_Toc48249843)

[7.1.2 Process Flow Steps 8](#_Toc48249844)

[7.2 Retry Process workflow 9](#_Toc48249845)

[7.2.1 Process Flow Diagram 9](#_Toc48249846)

[7.2.2 Process Flow steps 9](#_Toc48249847)

[8 Interface Definitions 15](#_Toc48249848)

[8.1 Request Message Definition 15](#_Toc48249849)

[8.1.1 ChannelRequest Message Structure /Schema 15](#_Toc48249850)

[8.1.2 ChannelRequest Message Details 15](#_Toc48249851)

[8.1.3 Channel Sample Source Messages 15](#_Toc48249852)

[8.1.4 T24 Request Message Structure /Schema 15](#_Toc48249853)

[8.1.5 T24 Response Message Details 16](#_Toc48249854)

[8.1.6 T24 Sample Source Messages 16](#_Toc48249855)

[8.1.7 Data Base Response Codes 18](#_Toc48249856)

[8.2 Response Message Definition 19](#_Toc48249857)

[8.2.1 Channel Response Message Structure/Schema 19](#_Toc48249858)

[8.2.2 Channel Response Message Details 19](#_Toc48249859)

[8.2.3 Channel Sample Response Messages 20](#_Toc48249860)

[9 Logging Mechanism 22](#_Toc48249861)

[9.1 Insert into RAW\_AUDIT\_TABLE 22](#_Toc48249862)

[9.2 Insert into ERROR\_AUDIT\_TABLE 23](#_Toc48249863)

[9.3 File Based Logging 24](#_Toc48249864)

[9.3.1 Log4j 24](#_Toc48249865)

**Approval**

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

**Revision History**

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

# Introduction

## General Description

This Interface Design document outlines the integration requirements for the Cassava PickUp ***(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 Cassava PickUp ***(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
* Error handling and exception scenarios
* Validation and backup requirements

# Channels Involved

The following table lists Cassava PickUp channels

|  |  |
| --- | --- |
| **Item #** | **Channel Name** |
| 1 | Mobile Application |
| 2 | Self Service |
| 3 | Agent Portal |

# Interface Dependencies

## External Dependencies

The following table lists interface specific External requirements.

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

## Internal Dependencies

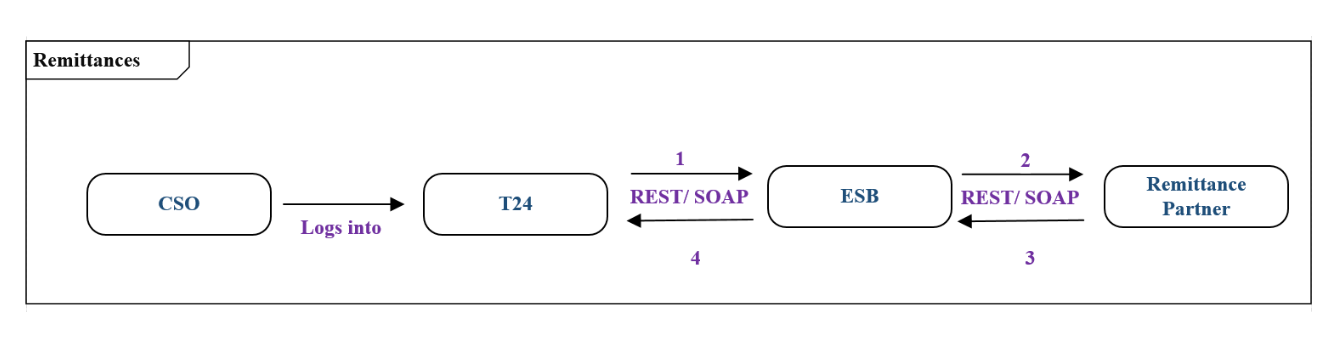
The following table lists interface specific internal requirements.

|  |  |
| --- | --- |
| **Item #** | **Internal Requirements** |
| 1 | HTTP Router Interface (Gateway) |
| 2 | HTTP Retry Interface (Subflow) |
| 3 | DB Logging App (Audit Logging) |
| 4 | StewardBankCommonEsql (Shared Library) |

# Business Process Summary

## Process Overview

The **Cassava PickUp** process primarily involves to



REST

### Figure:1 Process flow Approach for Cassava PickUp.

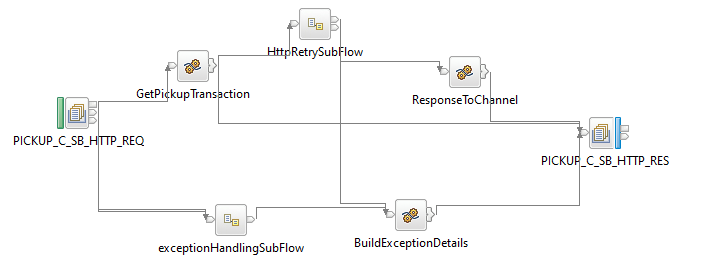
### Cassava PickUp Steps

|  |  |
| --- | --- |
| **S. No** | **Activities** |
| 1 | ESB receives the request from channel (Portal Backend). |
| 2 | Requestchannel is the http request accepts the request from the channel |
| 3 | ESB frames the Request URL According to the T24 requires and send to the T24. |
| 4 | T24 will Respond back to the ESB with Response |
| 5 | The same T24 Response, ESB will send back to the channel. |
| 6 | On Successful call of T24 Response will be placed in theRAW\_LOG\_SB\_REQto logthe response in Data Base(RAW\_AUDIT\_LOGGING |
| 7 | On Unsuccessful call of T24 Response will beplaced in theERROR\_LOG\_SB\_REQto log the response in Data Base(ERR\_AUDIT\_LOGGING |

# IIB Implementation Process Flow

## Cassava PickUp Successfull Request Response Flow

### Process Diagram



**Figure:2 Cassava PickUp Flow**

### Process Flow Steps

|  |  |
| --- | --- |
| **S. No** | **Activities** |
| 1. | ESB receives the request from the channel using the HTTP Router Application based on procesingCode& channel trigs the Business Application(CassavaPickUpFlow) using Business Queue (PICKUP\_C\_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 ESB frames the Request URL to T24 serveri.e T24 server. |
| 4 | After Framing the T24 Request is logged in RAW\_LOG\_SB\_REQ and is insert in Data Base |
| 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 and insert in Data Base (RAW\_AUDIT\_LOGGING) |
| 7 | After 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 Unsuccessfull calls of Mobiquity and T24 Request and Response will be placed in the ERROR\_LOG\_SB\_REQto log the request in Data Base(ERR\_AUDIT\_LOGGING) |

## Retry Process workflow

### Process Flow Diagram



**Figure:3 HTTP Retry for Cassava PickUp**

### Process Flow steps

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

**7.3Raw\_Audit Process Flow:**

**7.3.1Process Flow Diagram:**



**Figure:4Raw\_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 DataBase. |
|  | Final Response is inserted into the DataBase and then Response is logged into the RAW\_LOG\_SB\_REQ. |

**7.4HTTP Router Process Flow**

**7.4.1 Incoming Router Process Flow Diagram:**



**Figure:5Incoming Router Flow**

**7.4.2 Process Flow Steps:**

|  |  |
| --- | --- |
| **S. No** | IIB receives the request from channel. |
|  | ESB receives the request from channel. |
|  | Requestchannel is the http request accepts the request from the channel |
|  | The request message is placed in theRAW\_LOG\_SB\_REQto 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 Cassava PickUp Flow |

**7.4. Integration Solution Details**

**Table Name: INTEGRATION\_SOL\_DETAILS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 |
|  |  |  |  |  |  |  |

**7.4.4 Outgoing Router Process Flow Diagram**



**Figure:6 Outgoing Router Flow**

**7.4.5 Process Flow Steps:**

|  |  |
| --- | --- |
| **S. No** | **Activities** |
|  | TheOutgoingHttpRes 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 DataBase and then Response is logged into the ERR\_LOG\_SB\_REQ. |

# Interface Definitions

## Request Message Definition

### ChannelRequest Message Structure /Schema

**Request Type:** HTTP

**Request Format:**JSON

**Request URL:** https://stag20-za.securemit.com/api/v3/transactions/id/

8.1.2 ChannelRequest Message Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Tag Name/ Field Name** | **Data Type** | **Mandatory(Yes/No)** | **Comments** |
| 1 | Channel | String | Y | ChannelName |
| 2 | ProcessingCode | String | Y | Processing Code |

### 8.1.3 Channel Sample Source Messages

**Headers:**

Content-Type : application/vnd.api+json

Accept: application/vnd.api+json

Authorization : Bearer <Token or access >

app-id : 5adb798a-d809-4822-8e03-6acbaeb80751

|  |
| --- |
| {  "CassavaPickup":{  "header":{  "channel":"Remittance",  "channelsubtype":"Cassava",  "processingCode":"630000"    },  "body":{  "data": {    "type": "transactions",  "id": "92211188",  "attributes": {  "status": "C",  "payout\_amount": "14.00",  "payout\_comment": "My comment"  },  "relationships": {  "payout\_currency": {  "data": {  "id": "153",  "type": "currencies"  }  }  }  }  }  }  }}  }  } |

### 8.1.4 T24 Request Message Structure /Schema

**Request Type:** HTTP

**Request Format:**JSON

**Request URL**https://stag20-za.securemit.com/api/v3/transactions/id/

8.1.5 T24 Response Message Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Tag Name/ Field Name** | **Data Type** | **Mandatory(Yes/No)** | **Comments** |
| 2 | data | String | Y | Data Tag |
| 2 | Type | String | Y | Type |
| 2 | Id | String | Y | Id |
| 3 | Attributes | String | Y | Attributes |
| 3 | Code | String | Y | Code |
| 3 | status | String | Y | Status |
| 3 | status\_date | String | Y | Status\_Data |
| 3 | log\_data | String | Y | Log\_Date |
| 3 | beneficiary\_name | String | Y | Benificiary\_Name |
| 3 | Amount | String | Y | Amount |
| 4 | Relationships | String | Y | Main Tag |
| 4 | Sender | String | Y | Sender |
| 4 | data | String | Y | Data |
| 4 | beneficiary | Stirng | Y | Benificiary |
| 4 | Data | String | Y | Data |
| 5 | Links | String | Y | Main Tag |
| 5 | Self | String | Y | Slef |

### 8.1.6 T24 Sample Source Messages

|  |
| --- |
| {  "data": {  "type": "transactions",  "id": "92211188",  "attributes": {  "code": "92211188",  "status": "C",  "status\_date": "2020-08-17 11:00:58",  "log\_date": "2020-07-08 13:41:07",  "beneficiary\_name": "Thomas Nyagwaya",  "amount": "11.00"  },  "relationships": {  "sender": {  "data": {  "type": "accounts",  "id": "898922011"  }  },  "beneficiary": {  "data": {  "type": "beneficiaries",  "id": "04643875"  }  },  "currency": {  "data": {  "type": "currencies",  "id": "153"  }  }  },  "links": {  "self": "https://stag20-za.securemit.com/api/v3/transactions/92211188/"  }  }  } |

### 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

### Channel Response Message Structure/Schema

**Response Type:** JSON

### 8.2.2 Channel Response Message Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item #** | **Tag Name/ Field Name** | **Data Type** | **Mandatory(Yes/No)** | **Comments** |
| 2 | data | String | Y | Data Tag |
| 2 | Type | String | Y | Type |
| 2 | Id | String | Y | Id |
| 3 | Attributes | String | Y | Attributes |
| 3 | Code | String | Y | Code |
| 3 | status | String | Y | Status |
| 3 | status\_date | String | Y | Status\_Data |
| 3 | log\_data | String | Y | Log\_Date |
| 3 | beneficiary\_name | String | Y | Benificiary\_Name |
| 3 | Amount | String | Y | Amount |
| 4 | Relationships | String | Y | Main Tag |
| 4 | Sender | String | Y | Sender |
| 4 | data | String | Y | Data |
| 4 | beneficiary | Stirng | Y | Benificiary |
| 4 | Data | String | Y | Data |
| 5 | Links | String | Y | Main Tag |
| 5 | Self | String | Y | Slef |

### 8.2.3 Channel Sample Response Messages

|  |
| --- |
| {  "data": {  "type": "transactions",  "id": "92211188",  "attributes": {  "code": "92211188",  "status": "C",  "status\_date": "2020-08-17 11:00:58",  "log\_date": "2020-07-08 13:41:07",  "beneficiary\_name": "Thomas Nyagwaya",  "amount": "11.00"  },  "relationships": {  "sender": {  "data": {  "type": "accounts",  "id": "898922011"  }  },  "beneficiary": {  "data": {  "type": "beneficiaries",  "id": "04643875"  }  },  "currency": {  "data": {  "type": "currencies",  "id": "153"  }  }  },  "links": {  "self": "https://stag20-za.securemit.com/api/v3/transactions/92211188/"  }  }  } |

# Logging Mechanism

## 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\_N | Retrieval Reference Number | Varchar | 20 | Y |  |

## 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 |
|  | TIME\_LOCAL\_TRANSACTION | Transaction Time | Varchar | 20 | Y | Time of the Transaction |
|  | DATE\_LOCAL\_TRANSACTION | Transaction Date | Varchar | 20 | Y | Date of the Transaction |
|  | RETRIEVAL\_REFERENCE\_NUMBER | Retrieval Reference Number | Varchar | 20 | Y | Retrieval Reference Number |
|  | BORKER\_ERROR\_CODE | Broker Error Code | Varchar |  | Y | Error code generate by Broker |
|  | USER\_DEFINE\_ERROR\_CODE | User Defined Error Code | Varchar |  | Y | User Defined Code |

## 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