**T24 – Bluzelle Integration**

**Bluzelle Cross Border Payments Solution**

**Design Document**

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Section(s) Amended |
| 22-Jun-2016 | 0.1 | Shiva Prasad Y | Initial Draft |
| 01-Aug-2016 | 0.2 | MadhuPriya R | Added technical details |
| 05-Aug-2016 | 0.3 | Dinesh T | Added Techical details & changed sender bank image. |
|  |  |  |  |

Document Sign Off

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Name | Company / Position |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1 Introduction 4](#_Toc458172506)

[2 High Level Design 5](#_Toc458172507)

[2.1 Sender Bank Design 5](#_Toc458172508)

[2.2 Receiver Bank Design 6](#_Toc458172509)

[3 Technical Details 6](#_Toc458172510)

[3.1 Milestone 1 & 2: 6](#_Toc458172511)

[3.1.1 Sender Bank 6](#_Toc458172512)

[3.1.2 Receiver Bank 7](#_Toc458172513)

[4 Assumptions/Exclusions 7](#_Toc458172514)

[4.1 Assumptions: 7](#_Toc458172515)

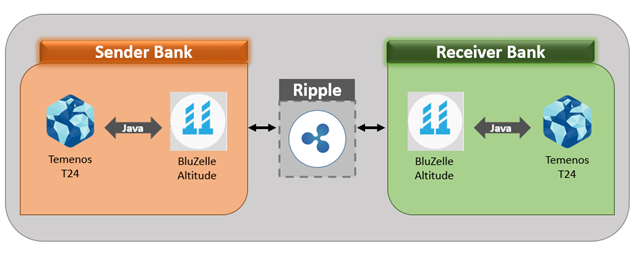
[ Exclusions: 7](#_Toc458172516)

# Introduction

The purpose of this document is to layout the design of T24 – Bluzelle integration with the help of the API provided by Bluzelle’s Altitude technology stack to enable payments from corporate or retail (consumer) customers on two different banks as part of Bluzelle Cross Border Payments Solution.

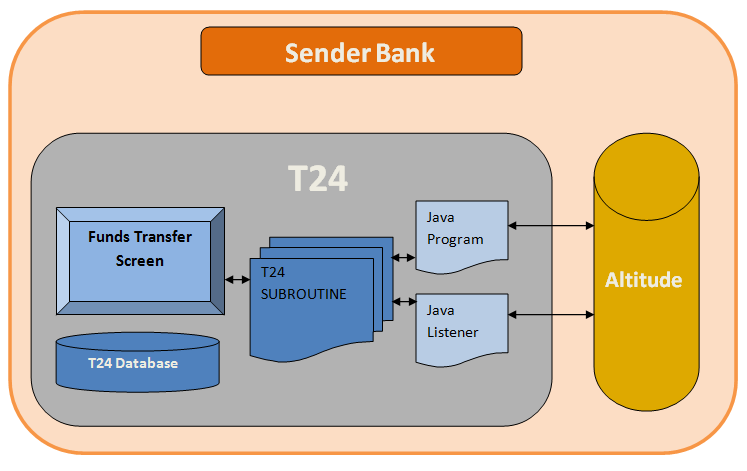
# High Level Design

Two T24 areas, one will act as a sender bank and other will act as a receiver bank. There will be two instances of Bluzelle altitudes tagged to each of the T24 area. The Bluzelle altitudes will be capable of interacting over Ripple.



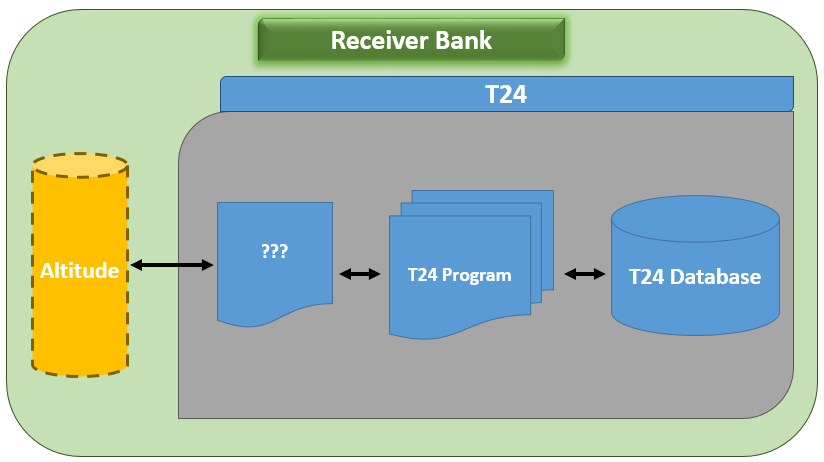
## Sender Bank Design

The sender bank will be making/ triggering requests to its respective altitude from T24 using Java calls to Bluzelle APIs. The response is converted from JSON format to T24 readable format and displayed on the screen or updated in the T24 database accordingly.



## Receiver Bank Design

The receiver bank will have to respond to the requests made by its altitude, the requests have to be consumed within T24 and then the T24 programs will perform the validations and respond with flags and required data. The response should be converted from T24 to JSON format.



**Java Listener to handle Call backs**

# Technical Details

## Milestone 1 & 2:

### Sender Bank

* A new L type table to be created in T24 to store the quotation values with ID of the record as FT ID
* A new FT Screen to be created which will be the starting point in T24
* This FT screen should have all the mandatory fields required to perform a FT in T24, in addition to the mandatory fields it should have the provision for accepting the “Recipient Address” field value from the user.
* A version routine to trigger the API – GetPaymentQuotation using java to the altitude of sender bank
* The response from the altitude to be converted from JSON to T24 format and the FT screen to be reloaded with the response(quotation) and also update the T24 database (on to L type file) for reference
* A version level Input routine to trigger the API – “AcceptPaymentQuotation” using java and based on success response from bluzelle again trigger the API – “ExecutePayment” and update the live table with the responses.
* On error/failure condition the system will throw an application level error to the user.
* API- “GetPaymentStatus” can be triggered from Nofile enquiry to check the status of the quote.
* A sender side Call back PaymentResult is triggered from Sender Bluzelle Altitute, The callback is processed with the incoming data Quote\_id and status by the listener java code and call a T24 subroutine, In turn a OFS message has been built to Authorise the FT and dropped in queue and processed

### Receiver Bank

* The receiver bank T24 has to consume the request which is made from its altitude which is achieved through Jremote call to T24 subroutine.
* The receiver bank altitude will make a request in the form of CALL BACK –“ ValidatePaymentRecipient”
* This request will have one parameter i.e., recipient address comprising of recipient ID and institution name
* The recipient ID is to be validated against the T24 database
  + In case of failure in validations appropriate error messages to be sent back
  + In case of success, the KYC/AML fields as per the API document needs to be responded to the altitude
* The response should be converted from T24 to JSON format and returned to listener call.
* A request to Approve payment details which will return the valid customer details.
* Then a request is made to “Payment received” based on the success response of “Approve PaymentDetails” wherein a FT is posted in T24 for a balance accounting entry.
* KYC/AML fields if missing in T24 should be created locally.

# Assumptions/Exclusions

## Assumptions:

* The communication between sender bank altitude and receiver bank altitude will be handled by Bluzelle.
* The Quote expiration time out will be handled by Bluzelle.
* The sender bank can also act as receiver bank and vice versa, both the Java programs and Listener code must be available in both the banks

## Exclusions: