

Functional design of E-Pay Transaction Service

August 2024

|  |  |
| --- | --- |
| **Document Control Sheet** | |
| **Current Version** | 0.4 |
| **Project Code** | NS\_SBI\_04\_08\_2024 |
| **Project Name** | SBI ePay Implementation |
| **Document Type** | ePay Functional Transaction Service |
| **Author** | Sagar |
| **First Reviewed By** | Neeraj Durgapal |
| **Final Reviewed By** | Anand Sharma |
| **Frequency of Document Review** | Next Revision of Standard Documents |
| **Document Creation Date** | 12-08-2024 |
| **Last Updated Date** | 21-09-2024 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Version** | **Date** | **Name** | **Comments** |
| 0.1 | 12-08-2024 | Ranu Jain | Draft Version |
| 0.2 | 16-08-2024 | Subhash Dharade | Transaction Flow |
| 0.3 | 21-08-2024 | Ranu Jain | Address Review Comments |
| 0.4 | 21-09-2024 | Sagar Thakare | Addressed review comments |

Table of Content

[Document Purpose 3](#_Toc1614458146)

[Scope 4](#_Toc2139056523)

[Transaction Service 4](#_Toc2033088198)

[Payment Service 4](#_Toc655013316)

[Rule Service 5](#_Toc1473836150)

[Business Requirement Mapping 5](#_Toc474832118)

[Actors 6](#_Toc756877045)

[Functional Specifications 7](#_Toc431631884)

[1. FS\_TX 1 – Token Generation 7](#_Toc962741392)

[1.1 - Objective 7](#_Toc1520001075)

[1.1.1 - Merchant Customer Creation 7](#_Toc1450206863)

[1.1.2 - Merchant Order Creation 7](#_Toc1917379882)

[1.1.3 - Merchant Order Transaction 8](#_Toc1598641474)

[1.2 - Functional Flow 8](#_Toc2032640138)

[1.3 - Functional Specification 8](#_Toc1376672355)

[1.3.1 - Validation 8](#_Toc2023335549)

[1.3.2 - Token Generation Logic 9](#_Toc850935352)

[1.3.2.1 - Active Token Check 9](#_Toc1325338002)

[1.3.2.2 - Token Invalidation Process 9](#_Toc1355785251)

[1.3.2.3 - Generate New JWT Token 9](#_Toc1543566086)

[1.3.3 - Token Uniqueness 10](#_Toc1116073427)

[1.3.4 - Token Revocation 10](#_Toc848147661)

[1.3.5 - Audit 10](#_Toc1228903909)

[1.3.6 - Notification 10](#_Toc1323885257)

[1.3.7- Error Code 10](#_Toc1912290569)

[1.3.8 - Token Authentication 10](#_Toc2110033597)

[1.3.9 API Mapping 10](#_Toc1647195877)

[2. FS\_TX 2 – Merchant Customer 11](#_Toc16856863)

[2.1 - Objective 11](#_Toc276256623)

[2.1.1 - Storage 11](#_Toc154132499)

[2.1.2 - Feature Type 11](#_Toc1596794177)

[2.1.4 - Uniqueness 11](#_Toc1964472939)

[2.1.5 - Operations 11](#_Toc1272601894)

[2.2 - Functional Flow 11](#_Toc1733446881)

[2.3 Functional Specification 11](#_Toc376088475)

[2.3.1- Basic Validation Check 12](#_Toc1289146463)

[2.3.1.1 - JWT Token 12](#_Toc1220103124)

[2.3.1.2 - Merchant 12](#_Toc1595137171)

[2.3.2 - Merchant Customer Creation 12](#_Toc1863227933)

[2.3.3 - Merchant Customer Retrieval Validation Check 13](#_Toc1565522927)

[2.3.3.1 - Get All 13](#_Toc62689049)

[2.3.3.2 - Get by Customer Id/Name/Email/Phone Number 13](#_Toc609007623)

[2.3.5 - Uniqueness Check for Merchant Customer Creation 13](#_Toc635922277)

[2.3.6 - Merchant Customer ID Generation Logic 13](#_Toc1272693489)

[2.3.6.1 - Uniqueness 13](#_Toc1331533123)

[2.3.6.2 - Format of Customer ID 14](#_Toc1004050427)

[2.3.7 - Auditing 14](#_Toc1118333632)

[2.3.8 - Notification 14](#_Toc120576210)

[3. FS\_TX 3 – Merchant Order 14](#_Toc1099529560)

[3.1 - Objective 14](#_Toc537903249)

[3.1.1 - Order ID Generation 15](#_Toc1805756533)

[3.1.2 - Order Lifecycle Management 15](#_Toc508259382)

[3.1.2.1 Created 15](#_Toc1465712688)

[3.1.2.2 Attempted 15](#_Toc1890629204)

[3.1.2.3 Paid 16](#_Toc1316114658)

[3.1.2.4 Expired 16](#_Toc1194288577)

[3.1.2.5 Failed 16](#_Toc935656313)

[3.1.3 – Multiple Redirection Methods (Merchant page to SBIePay payment page) 16](#_Toc1863164489)

[3.1.3.1 - Form Posting 16](#_Toc1796981229)

[3.1.3.2 - Link Based Approach 17](#_Toc544074021)

[3.1.4 – Merchant Order Retry Functionality 17](#_Toc446361745)

[3.1.5 - Operations 17](#_Toc2109800214)

[3.2 - Functional Flow 17](#_Toc321589216)

[3.3 - Functional Specification 18](#_Toc1681334631)

[3.3.1- Basic Validation Check 19](#_Toc1450571407)

[3.3.1.1 - JWT Token 19](#_Toc1323379009)

[3.3.1.2 - Merchant 19](#_Toc830708115)

[3.3.3 - Business validation 20](#_Toc1835634718)

[3.3.3.1 Data Type Checks 20](#_Toc463896692)

[3.3.3.2 Merchant Order Number Checks 20](#_Toc1224232947)

[3.3.3.3 Amount Check 20](#_Toc847691201)

[3.3.3.4 Currency Checks 20](#_Toc2135001334)

[3.3.3.5 Pay Mode Mapping Checks 20](#_Toc890156530)

[3.3.3.6 Multi Accounts Checks 21](#_Toc110479871)

[3.3.3.7 Threshold Checks 21](#_Toc462841947)

[3.3.4 - Merchant Order Retrieval Validation Check 21](#_Toc1858004001)

[3.3.4.1 - Get All for getting all the order details for that Merchant 21](#_Toc258628274)

[At a single request 10,000 records will be provided with pagination option (till records are available) 21](#_Toc1309363018)

[3.3.4.2 - Get by Merchant Order Number/Customer Id/ Order Status 21](#_Toc1482788804)

[3.3.5 - Auditing 22](#_Toc586870475)

[3.3.7 - Error code 22](#_Toc122108669)

[4. FS\_TX 4 – Merchant Payment Transaction 22](#_Toc1586519878)

[4.1 - Objective 22](#_Toc1755999172)

[4.1.1 - Transaction Ref Number Creation 22](#_Toc64513525)

[4.1.2 - Transaction Life Cycle Managment 23](#_Toc1875023708)

[4.1.2.2 - Paid 23](#_Toc2091703598)

[4.1.2.3 - Expired 24](#_Toc529669443)

[4.1.2.4 - Refund 24](#_Toc647744221)

[4.1.2.5 - Settled 24](#_Toc1302047918)

[4.1.2.6 - Failed 24](#_Toc802926552)

[4.1.3 - Transaction Payment Integration 27](#_Toc283400224)

[4.1.3.1 - Credit/Debit Card 27](#_Toc245719291)

[4.1.3.2 - Net Banking – SBI / Channel Banks 27](#_Toc1005738550)

[4.1.3.3 UPI 27](#_Toc1618240720)

[4.1.4 - Transaction Processing 27](#_Toc503465011)

[UX Flow- Transaction UX flow is available on below figma link. 27](#_Toc1647704853)

[Link - 28](#_Toc2009072121)

[4.1.4.1 - Form Posting 28](#_Toc1621078623)

[4.1.4.2 - Link Based Approach 28](#_Toc1526194684)

[4.1.5 - Merchant Payment Mode 28](#_Toc33867850)

[4.1.6 - Payment Mode Downtime 28](#_Toc226024252)

[4.1.7 - Transaction Processing Amount 28](#_Toc694995687)

[4.1.8 - Retry Functionality 29](#_Toc89739378)

[4.1.9 - Refund / Cancellation Functionality 29](#_Toc1251576697)

[4.1.10 - Operations 31](#_Toc2131396869)

[4.1.11 - Settlement / Reconciliation Processes 31](#_Toc147067727)

[4.3 - Functional Specification 32](#_Toc897078754)

[4.3.1- Basic Validation Check 32](#_Toc599600294)

[4.3.1.1 - Token 32](#_Toc553389351)

[4.3.1.2 - Validity 32](#_Toc955474990)

[4.3.1.3 - Merchant 32](#_Toc235866323)

[4.3.1.4 - Base64Encode Check 32](#_Toc1302453572)

[4.3.1.5 - Transaction Creation Request (ATRN Generation Request) Should have below parameter 32](#_Toc1295695788)

[4.3.1.6 - Transaction Retrieval Validation Check (Transaction Status Inquiry) 33](#_Toc1781272096)

[4.3.1.6.1 - Get All for getting all the transaction details for that Merchant 33](#_Toc2100429430)

[4.3.2 - Business validation 33](#_Toc1835128470)

[4.3.2.1 Merchant Order Number Checks 33](#_Toc1205972948)

[4.3.2.2 ATRN Number 33](#_Toc1982429273)

[4.3.2.3 RFC Checks 33](#_Toc788925296)

[4.3.2.4 -Payment Method Validation 34](#_Toc207481230)

[4.3.2.5 - Bank Double Verification 36](#_Toc399244154)

[5. NFR-1: Non Function Requirements: 37](#_Toc1516215033)

# Document Purpose

This document refers to implementing Transaction and Payment Microservices for SBIePay transaction portal for SBIePay application by defining business rule in Rule Microservice.

# Scope

The scope of this document is to avail secure and robust merchant Transaction and payment handling with end-to-end transaction flow and SBIePay other services (Refund and enquiry) for merchant.

Transaction Service

The core responsibility of Transaction Microservices

* **Token Creation:**
  + Generate tokens for Merchants, Customers, Orders, and Transactions to ensure secure and unique identifiers.
* **Merchant-Customer Mapping:**
  + Manage and maintain the mapping between Merchants and their Customers, ensuring accurate associations.
* **SBIePay Order Management:**
  + **Initiation:** Start the order process within the SBIePay system.
  + **Validation:** Verify order details to ensure they meet business and system requirements.
  + **Creation:** Generate and store order records in the system.
* **SBIePay Transaction Management:**
  + **Initiation:** Begin the transaction process.
  + **Validation:** Confirm transaction details for accuracy and compliance.
  + **Creation:** Establish and record the transaction in the system.
* **Order and Transaction Status Management:**
  + Maintain and update the status of Orders and Transactions, including auditing changes for traceability.
* **Notification Management:**
  + Trigger notifications for changes in Order and Transaction statuses to keep relevant parties informed.
* **Payment Service Initiation:**
  + Initiate the Payment Service as required by the Transaction flow.

Payment Service

The core responsibility of Payment Microservices

* **Restricted Access:**
  + Transaction Microservices exclusively makes this payment service accessible to maintain security and integrity.
* **Payment Method Configuration:**
  + Configure and manage various payment methods at the service level.
* **Supported Payment Methods:**
  + Provide support for multiple payment methods including:
    - QR Code
    - UPI (Unified Payments Interface)
    - Cards
    - Net Banking (NB)
    - Cash
* **Channel Bank Interface Management for Net Banking:**
  + Manage and interface with banks for Net Banking transactions.
* **Payment Gateway Interface Management:**
  + Handle interfaces with payment gateways for UPI, Cards transactions.

Rule Service

The core responsibility of Rule Microservices for Transaction microservice

* **Business Rule Definition:**
  + Define and manage the business rules related to transactions within the service, ensuring they are applied consistently and accurately across the Transaction microservice.
  + To be integrated with Admin Dashboard. Currently, it will be set with Liquibase script.

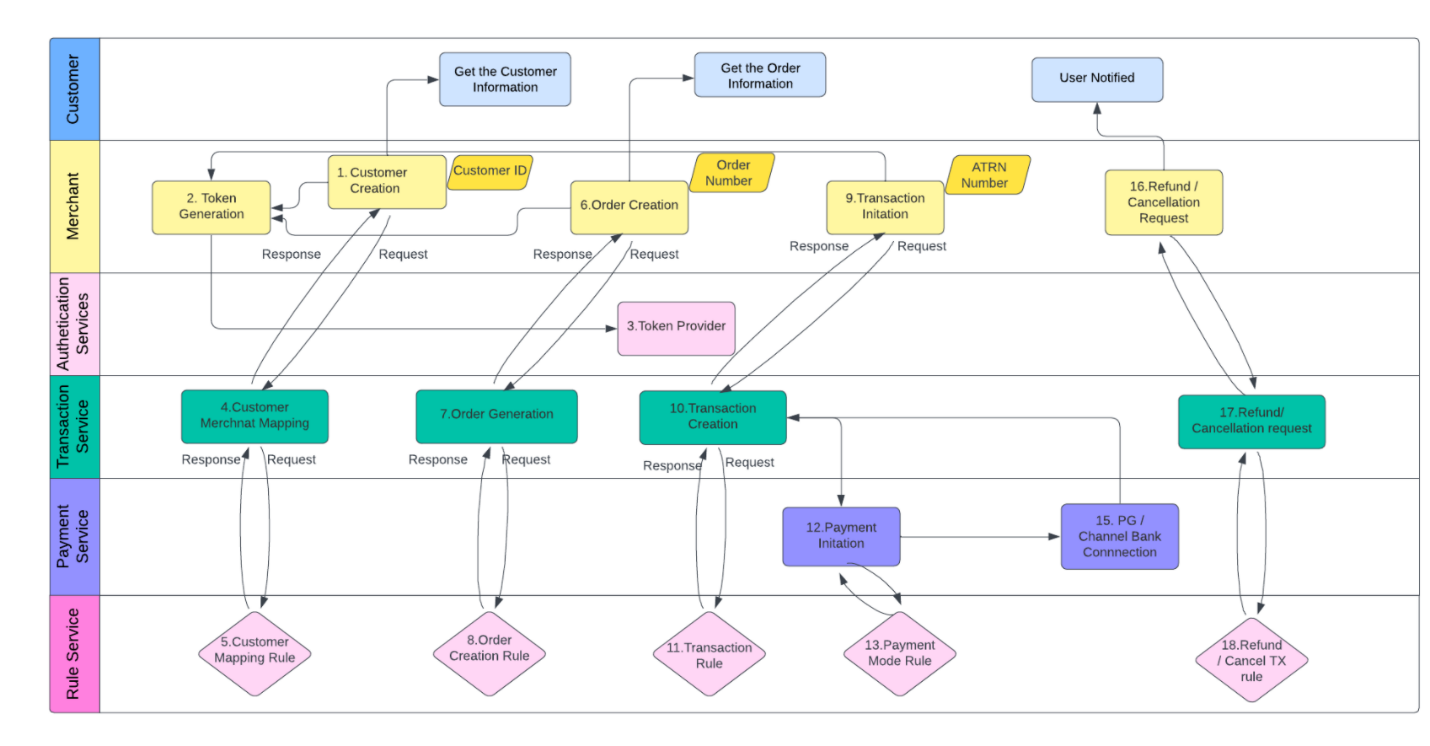
Business Requirement Mapping

|  |  |  |
| --- | --- | --- |
| FS | BRD | BRD Details |
| FS\_TX 1 | BRD 1 | Token Generation |
| FS\_TX 2 |
| FS\_TX 3 | Merchant Customer, Order and Transaction initiation |
| FS\_TX 4 | BRD 2 | Merchant Payment Flow |

Actors

|  |  |
| --- | --- |
| **Actors** | **Functions** |
| Customer | The individual initiates the payment process. |
| Merchant | The entity responsible for creating customer profiles, generating orders, and enabling the checkout page for customers, integrating with SBIePay. |
| SBI ePay System | The payment aggregator facilitates transactions for merchants. |
| Payment Channels | Various payment options integrated with the SBIePay platform, including UPI, net banking, credit cards, and debit cards. |

Functional Flow



# Functional Specifications

## FS\_TX 1 – Token Generation

### 1.1 - Objective

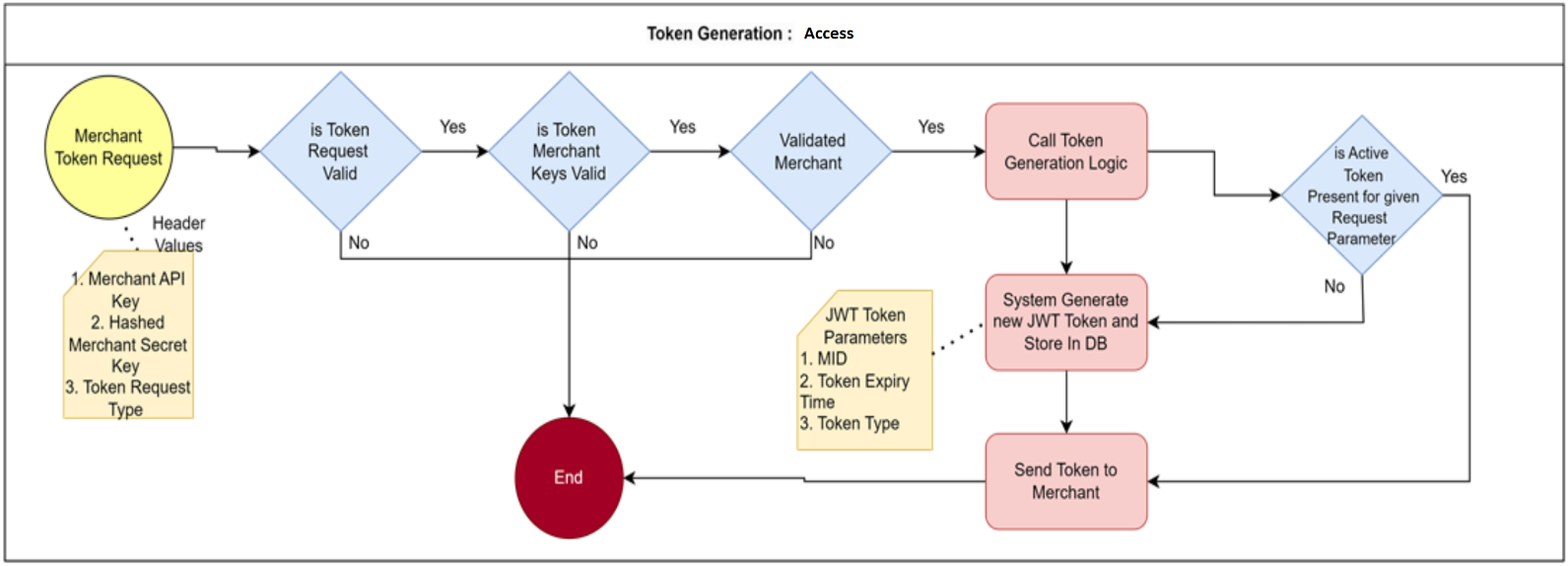
Tokens will be generated with the help of Authentication Java Service to grant the merchant access to the following functionalities within the Transaction Service.

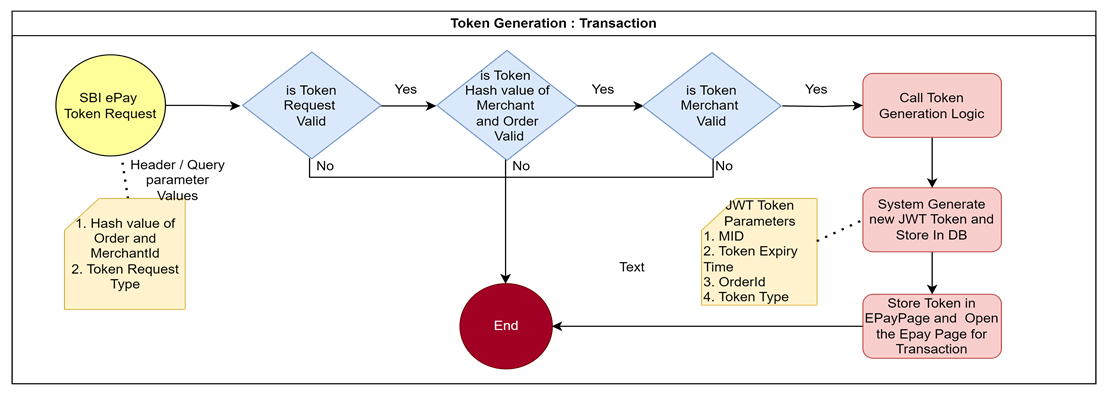
#### 1.1.1 - Merchant Customer Creation

#### 1.1.2 - Merchant Order Creation

#### 1.1.3 - Merchant Order Transaction.

### 1.2 - Functional Flow





### 1.3 - Functional Specification

#### 1.3.1 - Validation

* + - For Access Token request should have the below parameters in headers

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Merchant API Key | Yes | Merchant API Key which was generated during onboarding and shared with Merchant |
| Merchant Secret Key | Yes | Hash Value of Merchant API Key Secret which was generated during onboarding and shared with Merchant |
| Function Type | Yes | Access |

* + - For Transaction token request should have the below parameters in headers or as a Query Parameter

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Order ID + MId | Yes | Hash Value of combination of Order ID and Mid |
| Function Type | Yes | Transaction |

* + - Merchant should be Active, Merchant allowed for given Request Type etc.

#### 1.3.2 - Token Generation Logic

##### 1.3.2.1 - Active Token Check

* + - * Before generating a new JWT token, check if an active JWT token already exists for the given request parameters
      * An active JWT token is one that has not expired and matches the request parameters.

##### 1.3.2.2 - Token Invalidation Process

* + - * If an active JWT token is found, invalidate it by updating its status in the database (e.g., mark it as expired or invalid).
      * Optionally, you can notify the merchant or log this event for auditing purposes.

##### 1.3.2.3 - Generate New JWT Token

* + - * After invalidating any existing tokens, proceed with generating a new JWT token as per the merchant's configuration.

1.3.2.4 - Token Expiry

|  |  |  |  |
| --- | --- | --- | --- |
| Token Type | Function | Default Expiry Time | Configurable by Merchant |
| Customer and Order Creation | Customer Creation and Unique Order creation | AccessTokenExpiryTime | No |
| Transaction | Payment Initiation | TransactionTokenExpiryTime  mins | No |

#### 1.3.3 - Token Uniqueness

* + - Generate **Token Should include** Tokens need to be generated as per Merchant Configuration during onboarding parameter for given request type.
    - **Functionality Scope:** Limits what the token can be used for (e.g. Customer Mapping, Order Creation, Transaction initiation and Payment Request).
    - **Expiration Time:** Define the validity period for the token based on the merchant's configuration at SBI ePay end.
    - **Unique Identifier:** A secure, unique identifier to ensure that the token is distinct and traceable.

#### 1.3.4 - Token Revocation

* + - Implement a mechanism to revoke tokens, if necessary, especially if a security breach is suspected or the merchant requests a new token for the same functionality.

#### 1.3.5 - Audit

* + - All Token Requests and Responses must be securely logged and stored in the Database Layer.
    - Payment and Transaction token should have the mapping with order ID in DB level.

#### 1.3.6 - Notification

* + - Merchant will be notified over Email and SMS for Token generation or failure if opted by the Merchant

1.3.7- Error Code

* + - Maintain the proper Individual Error Code for each failure

#### 1.3.8 - Token Authentication

* + - Implement a mechanism to validate the tokens, based on merchant info, Expiry time, transaction type, order info.

#### 1.3.9 API Mapping

* An Interface will be provided to Business / required team, wherein access to exposed APIs can be granted/restricted for merchants.
* This will be added to the current merchant onboarding journey.

## FS\_TX 2 – Merchant Customer

### 2.1 - Objective

To create and store the Merchant Customer ID during the Order Booking process and enable analytics in SBI ePay.

#### 2.1.1 - Storage

* + The Merchant Customer module will store details of end users (customers) who will use SBI ePay for payments.

#### 2.1.2 - Feature Type

* + **Feature Type** - This is an optional feature, meaning merchants may choose to provide customer details or not.

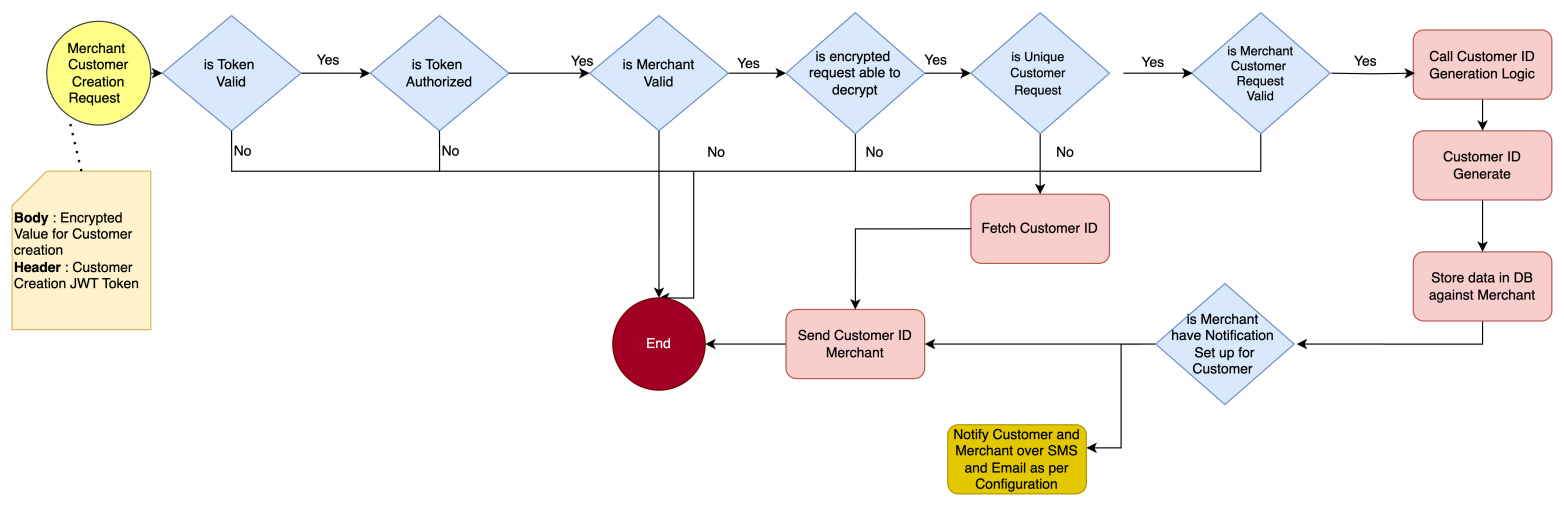
#### 2.1.4 - Uniqueness

* + **Uniqueness** - Merchant Customer ID will be unique through the system, and Customer will be unique for a Merchant by Name, Email and Phone Number

#### 2.1.5 - Operations

* + **Operations -** Create, Get, Active/In Active,Deletion, Get All Customers for a Merchant, Get Customer by Id/Name/Phone Number/ Email

### 2.2 - Functional Flow

l

### 2.3 Functional Specification

#### 2.3.1- Basic Validation Check

##### 2.3.1.1 - JWT Token

* + - * Verify the identity and authorization of the requester to ensure that only valid and authorized entities can request tokens.

##### 2.3.1.2 - Merchant

* + - * Check if the merchant is active and eligible to perform the requested operation.

#### 2.3.2 - Merchant Customer Creation

* + - **Request Check** - Customer Creation Request, the data should be decrypted using the corresponding decryption algorithm defined in the Encryption-Decryption service.(AES256-GCM)
    - **Validation Check** - Customer Creation Request Should have below parameters in encrypted request

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Token | Yes | Valid Request Token for Customer Creation |
| MID | Yes | Merchant Id |
| Name | Yes | Customer Name |
| Email | Yes, If Phone Number Parameter is not provided | Customer Email |
| Phone Number | Yes, If Email parameter is not provided | Customer Phone Number |
| GST In | No | Customer GST Number |
| Address Line 1 | No | Customer Full Address |
| Address Line 2 | No |
| City | No |
| State | No |
| Country | No |
| Pin Code | No |

#### 2.3.3 - Merchant Customer Retrieval Validation Check

##### 2.3.3.1 - Get All

* + - At a single request 10,000 records will be provided with pagination option (till records are available)

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Token | Yes | Valid Request Token for Customer Creation |
| Period | Yes | Date and Time |

2.3.3.2 - Get by Customer Id/Name/Email/Phone Number

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Token | Yes | Valid Request Token for Customer Creation |
| Request Type | Yes | Customer Id/Name/Email/Phone Number |
| Request Value | Yes | Value of Customer Id/Name/Email/Phone Number |

*2.3.4 - Merchant Customer Active/Inactive*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Mandatory |  |  | Description |
| Token | Yes |  |  | Valid Request Token for Customer Creation |
| Customer Id | Yes |  |  | Customer Id on Which Action Need to perform |

#### 2.3.5 - Uniqueness Check for Merchant Customer Creation

* + - Validate that there is no duplicate record for a given merchant customer based on **Name**, **Email**, and **Phone Number** for a specific merchant in the database
    - If a Record is found, then the system should send the Customer ID to Merchant with Error Code, otherwise Customer ID generation Logic need to call

#### 2.3.6 - Merchant Customer ID Generation Logic

* + - To implement a unique Customer ID generation logic here is a detailed approach:

##### 2.3.6.1 - Uniqueness

* + - * The Customer ID must be unique and not follow a sequential pattern.
      * Before finalizing the Customer ID, check the database to ensure it does not already exist.
      * If a duplicate is found, generate a new ID until a unique one is created.

##### 2.3.6.2 - Format of Customer ID

* + - * **Structure:**
        + **Prefix:** Cust\_
        + **Value**

Use a combination of random alphanumeric characters and follow with the prefix.

Use a secure random number generator to create a 16-character alphanumeric string.

Ensure the generated string does not follow a predictable pattern.

#### 2.3.7 - Auditing

* + - All Requests and Responses must be securely logged and stored in the Database Layer.

2.3.8 - Notification

* Customer and Merchant will be notified over email and SMS for successful Customer Id Creation as per configuration set up during Merchant Onboarding New if opted by Merchant

*2.3.9 - Merchant Customer Deletion*

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Token | Yes | Valid Request Token for Customer Creation |
| Customer Id | Yes | Customer Id on Which Action Need to perform |

## FS\_TX 3 – Merchant Order

### 3.1 - Objective

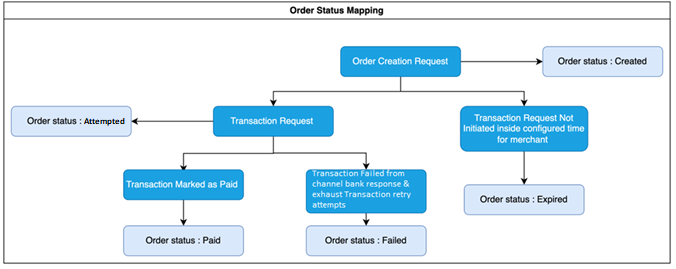
* **Order Creation:** Captures and validates orders from merchants, based on merchant-specific configurations.
* **Order Processing:** Handles order execution
* **Retry Mechanism:** Retries order processing upon failure, based on merchant-specific configurations.

3.1.1 - Order ID Generation

* Provide a unique "Merchant Order Number" for every Order by Merchant but still SBI Epay will also generate the Order ID with unique constraint.
* The Order ID number's length will be of 30 characters. (ORD-<MerchantID>-<Timestamp>-<Sequence Number>)

#### 3.1.2 - Order Lifecycle Management

* Introduction of defined order statuses (Created, Attempted, Paid, Expired, Failed) with clear conditions for status transitions.



3.1.2.1 Created

* + - **Description:** The initial state of an order after it is successfully booked.
    - **Conditions:**
      * An order remains in the **Created** state until a payment is attempted.
      * **Max Validity:** Order Id will be validated till as per configuration (MerchantOrderExpiryTime) at SBIePay end. If not paid within this period, the order will be marked as **Expired**.

3.1.2.2 Attempted

* + - **Description:** The state of an order when a payment attempt has been made.
    - **Transition:** Moves from **Created** to **Attempted** when a payment attempt is initiated.
    - **Conditions:**
      * The order remains in the **Attempted** state until a payment associated with it is successfully captured.
      * **Max Retry Validity:** 3 attempts and MerchantOrderExpiryTime time validity at the system level (configurable at the merchant level). If the maximum number of retries is reached without a successful payment, the order will be marked as **Failed**.

3.1.2.3 Paid

* + - **Description:** The final state indicating that the payment has been successfully processed.
    - **Conditions:**
      * The order is marked as **Paid** once the payment is successfully captured. This is the last stage of the order.

3.1.2.4 Expired

* + - **Description:** The state indicating that the order is no longer valid due to inactivity.
    - **Conditions:**
      * An order is marked as **Expired** if it remains in the **Created** or **Attempted** state beyond the allowed time (configured at the system or merchant level).

3.1.2.5 Failed

* + - **Description:** The state indicating that the payment process was unsuccessful after multiple attempts.
    - **Conditions:**
      * An order is marked as **Failed** if it reaches the maximum number of payments retry attempts (configured at the system or merchant level) without a successful payment.

#### 3.1.3 – Multiple Redirection Methods (Merchant page to SBIePay payment page)

3.1.3.1 - Form Posting

* The merchant passes the SBIePay-generated Order Id.
* SBIePay validates the Order Id and, if valid, allows redirection to Transaction Page

3.1.3.2 - Link Based Approach

* After successful Order Creation, The payment link generated by SBI ePay will be validated, and redirection will be allowed if the link is valid.

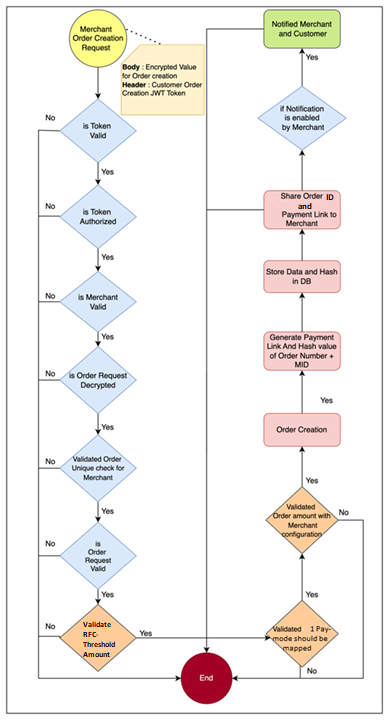
#### 3.1.4 – Merchant Order Retry Functionality

* Merchant can retry the Order Creation for validation failures, as per configure time during Merchant onboarding otherwise system will define the common retry numbers.

#### 3.1.5 - Operations

* Create, Get, Active/In Active, Get All Order for a Merchant, Get Order by Order Number/Customer Id/ Customer Name/ Customer Phone Number/ Customer Email/ Order Status

### 3.2 - Functional Flow



### 3.3 - Functional Specification

#### 3.3.1- Basic Validation Check

##### 3.3.1.1 - JWT Token

* + - * Verify the identity and authorization of the requester to ensure that only valid and authorized entities can request tokens.

##### 3.3.1.2 - Merchant

* + - * Check if the merchant is active and eligible to perform the requested operation.
      * Verified fraud and risk associated with Order, as per Merchant Configuration.
      * Merchant Payment link will be generated if merchant has passed the payment link parameter. ExpiredDateTime –DDMMYYYY HH:MM:SS
      * Payment link will be activated as merchant is sending in parameter.
      * Payment link DateTime cannot be more than 48 hours' time of initiation of order request.
      * Merchant order expiry time would be same for Payment Link shared Expiry time.

*3.3.2 - Merchant Order Creation*

* + - **Request Check** - Merchant Order Creation Request, the data should be decrypted using the corresponding decryption algorithm defined in the Encryption-Decryption service (AES256-GCM)
    - **Validation Check** - Order Creation Request Should have below parameter

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Token | Yes | Valid Request AccessToken for Order Creation |
| Amount | Yes | Order Amount |
| Merchant Order Number | Yes | Merchant Need to send the unique Number for new request |
| Customer Id | No | Customer ID for which order is getting created |
| Currency | Yes | Order amount Currency |
| Return URL | Yes | Requests need to redirect to this URL after Successfully Order Creation |
| Payment mode | No | Payment mode (NB, CC, DC & UPI) for paymode locking |
| Gateway Map ID | No | Payment mode channel ID |
| Other Details: {} | No | Max 10 Other Json Objects value |
| Multi Account: [  {  Account Identifier: XXXX,  Amount: XXXX,  },  {  Account Identifier: XXXX,  Amount: XXXX,  }  ] | No | Its mandatory if Merchant is opting for Multi Account feature  This Account Number and Amount used during settlement |
| Third Party Validation: {  IFSC: XXX,  Account Number: XXXX,  Name: XXXX  } | No | It will be mandatory if Merchant opted for Third Party Validation |
| Payment Link : {  ExpiryDateTime: XXXXXXXX  } | No | It will be non-mandatory if a Merchant needs a payment link with expiry date and time.  Date & Time format – DDMMYYYY HH:MM: SS |

#### 3.3.3 - Business validation

##### 3.3.3.1 Data Type Checks

* Check all fields data type like Amount should be double, Other Details should be array with Json object, Multi Account should be array with having property Account Number and Amount

##### 3.3.3.2 Merchant Order Number Checks

* Check Merchant order number is unique or in use to prevent duplicate orders
* If a duplicate merchant order number is detected and the order status is "Failed," the request will enter a retry flow based on the retry count configured for the merchant.

##### 3.3.3.3 Amount Check

* The amount should be a positive number and must be validated against the merchant-configured cumulative value.

##### 3.3.3.4 Currency Checks

* The valid transaction mode is international or domestic.
* The valid currencies are USD (International) and INR (Domestic).

##### 3.3.3.5 Pay Mode Mapping Checks

* Merchants can process orders only for the payment modes those are configured during merchant on boarding process

##### 3.3.3.6 Multi Accounts Checks

* Merchants can perform multi-account settlements only if they have opted for the multi-account settlement option.
* The order amount must match the total multi-account order amount.

##### 3.3.3.7 Threshold Checks

* Validate if the merchant transaction amount is more than the configured amount with the status
  + Block – The transaction request will be blocked or marked as failed and will not proceed further.
  + Review - The order booking process will be allowed, and the transaction will proceed. However, the settlement will occur only after the RFC team reviews the order.
  + Not Applicable - The order booking process will be allowed to proceed without any restrictions.

#### 3.3.4 - Merchant Order Retrieval Validation Check

##### 3.3.4.1 - Get All for getting all the order details for that Merchant

##### At a single request 10,000 records will be provided with pagination option (till records are available)

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Token | Yes | Valid Request Token for Customer Creation |
| Period | Yes | Particular Period creation for orders – DD/MM/YYYY |

3.3.4.2 - Get by Merchant Order Number/Customer Id/ Order Status

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Token | Yes | Valid Request Token for Customer Creation |
| Request Type | Yes | Order Number/Customer Id/Name/Email/Phone Number/ Order Status |
| Request Value | Yes | Value of Order Number/Customer Id/Name/Email/Phone Number/ Order Status |
| Period | Yes | Time period |

#### 3.3.5 - Auditing

* All Requests and Responses must be securely logged and stored in the Database Layer.

3.3.6 - Notification

* Customer and Merchant will be notified over email and SMS for successful Order Creation as per configuration has been setup during Merchant Onboarding

#### 3.3.7 - Error code

* Maintain the proper Individual Error Code for each failure

## FS\_TX 4 – Merchant Payment Transaction

### 4.1 - Objective

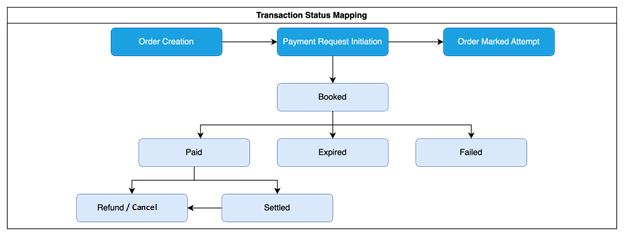
To implement a Merchant Payment Transaction system, we need to design a process that handles the following key aspects:

* **Payment Request Creation**: Initiating a payment transaction from the merchant.
* **Payment Processing**: Validating, processing, and completing the payment.
* **Retry Mechanism**: Handling failed payment attempts with retry logic based on merchant configuration
* **Security and Compliance**: Ensuring the transaction process is secure and compliant with relevant regulations.
* **Integration with Channel Bank**: Establish a reliable connection with the Channel Bank for handling transaction settlements and reconciliation, ensuring that funds are accurately credited to the merchant's account.
* **Customer Experience Enhancement**: Improve the overall customer experience by providing a reliable, fast, and secure payment process, with clear communication and support in case of payment issues.
* **Scalability and Performance Optimization**: Design the system to handle varying transaction volumes, with the ability to scale as needed while optimizing performance to reduce processing times and improve system responsiveness.

4.1.1 - Transaction Ref Number Creation

* An ATRN number will be generated for every transaction linked to an order, ensuring that each transaction can be uniquely identified across the merchant’s system.
* The ATRN number's length will be a minimum of **20 characters** to a maximum of **30 characters**. This length will be changed in incremental way once ATRN is generated.
* The ATRN number will be an **alphanumeric value,** which may include a combination of letters (A-Z, a-z) and numbers (0-9).
* This ATRN will serve as a unique identifier for tracking and reconciling transactions within the system, ensuring that each transaction is distinct and easily traceable.

4.1.2 - Transaction Life Cycle Managment

* This process involves managing transactions through clearly defined statuses: **Booked, Paid, Expired, Refund, Settled,** and **Failed.** Each status is associated with specific conditions that govern when and how a transaction transitions between these states.
*  4.1.2.1 - Booked
  + - **Description:** This is the initial status of a transaction once it has been successfully initiated.
    - **Conditions:**
      * The transaction remains in the **Booked** state until the transaction token is expired, or payment is successful or failed.
      * **Max Validity:** The transaction can stay in this state for up to TransactionTokenExpiryTime at the system level (configurable at the merchant level). If no payment is made within this period, the transaction will transition to the **Expired** state.

4.1.2.2 - Paid

* + - **Description:** This status indicates that the payment for the transaction has been successfully processed and captured.
    - **Conditions:**
      * The transaction is marked as **Paid** once the payment is successfully captured, representing the second stage of the transaction.

4.1.2.3 - Expired

* + - **Description:** This status signifies that the transaction is no longer valid due to inactivity or non-payment within the allowed time frame.
    - **Conditions:**
      * A transaction is marked as **Expired** if it remains in the **Booked** state beyond the allowed time, as configured at the system or merchant level.

4.1.2.4 - Refund

* + - **Description:** The state indicating that the transaction is initiated for Refund
    - **Conditions:**
      * A transaction moves to the **Refund** state if it was previously in the **Paid/Settled** state or if it failed during the settlement process.
      * The transition to **Refund** may also depend on specific merchant configurations.

4.1.2.5 - Settled

* + - **Description:** This status shows that the transaction has been successfully reconciled and settled.
    - **Conditions:**
      * A transaction is marked as **Settled** if it remains in the **Paid** state and passes the reconciliation process successfully.

4.1.2.6 - Failed

* + - **Description:** The state indicating that the payment process was unsuccessful after multiple attempts.
    - **Conditions:** Transaction will be marked Failed
      * Transaction reaches the maximum number of payments retry attempts (configured at the system or merchant level) without a successful payment.
      * Payment has been failed by the configured PG or Channel Banks.
      * Any Business validation error
      * **Transaction Status and sub status code and description are as follows-**

Transaction Flow - [Transaction\_Status\_Flow.html](https://sbionline.sharepoint.com/:u:/s/EPAY_VKG/EedzgfgarwZEj_pE7NSyVmMBKcSv_2-Z73TXF9wSyBb2dQ?e=Id4zWz)

|  |  |  |  |
| --- | --- | --- | --- |
| STATUSCATEGORY | STATUSCODE | STATUSDESCRIPTION | USERSTATUSDESCRIPTION |
| CHARGEBACK | 105 | CB Booked | CB Booked |
| CHARGEBACK | 106 | CB Admin Doc Uploaded | CB Admin Doc Uploaded |
| CHARGEBACK | 107 | CB Doc Uploaded | CB Doc Uploaded |
| CHARGEBACK | 108 | CB Doc Approved | CB Doc Approved |
| CHARGEBACK | 109 | CB Doc Rejected | CB Doc Rejected |
| CHARGEBACK | 110 | CB Hold | Chargeback Hold |
| CHARGEBACK | 111 | CB Debit Initiated | Chargeback Debit Adjustment Initiated |
| CHARGEBACK | 112 | CB Debit Approved | Chargeback Debit Adjustment Approved |
| CHARGEBACK | 113 | CB Debit Adjusted | Chargeback Debit Adjusted |
| CHARGEBACK | 114 | CB Closed | Chargeback Closed |
| CHARGEBACK | 115 | CB Accepted | Chargeback Accepted |
| CHARGEBACK | 116 | CB Deem Accepted | Chargeback Deem Accepted / Merchant Non-Revert |
| TXN | 21 | Transaction Booked | Transaction Booked |
| TXN | 22 | Payment Success | Payment Success |
| TXN | 23 | Payment Failed | Payment Failed |
| TXN | 24 | Payment Abort | Payment Abort |
| TXN | 25 | Payment In Clearing | Payment In Clearing |
| TXN | 26 | Transaction Hold | RFC On Hold |
| TXN | 27 | Transaction Reject | RFC Reject |
| TXN | 28 | Payment Sighted | Payment Sighted |
| TXN | 29 | Transaction Closed | Transaction Closed |
| TXN | 30 | Transaction Expired | Transaction Expired |
| TXN | 31 | Transaction Settled | Transaction Settled |
| TXN | 32 | Transaction Refund | Transaction Refund |
| TXN | 33 | Transaction Paid Out | Transaction Paid Out |
| TXN | 34 | Transaction Cancelled | Transaction Cancelled |
| TXN | 35 | Pending Authorization | Pending Authorization |
| TXN | 38 | Refund Against Failed Transaction | Transaction DVP Refund |
| TXNRFC | 46 | Pending RFC Clearing | Pending for Clearing |
| TXNRFC | 47 | WIP RFC Clearing | WIP |
| TXNRFC | 48 | Release RFC Clearing | Approve |
| TXNRFC | 49 | Reject RFC Clearing | Reject |
| GTWSTMT | 60 | Pending Reconciliation | Pending Reconciliation |
| GTWSTMT | 61 | Reconciled | Reconciled |
| GTWSTMT | 62 | Offline Refund | Offline Refund |
| GTWSTMT | 63 | DVP Refunded | DVP Refunded |
| GTWSTMT | 64 | Online Refund | Online Refund |
| REFUND | 65 | Refund Authorization Pending | Refund Authorization Pending |
| REFUND | 66 | Refund Booked | Refund Initiated |
| REFUND | 67 | Refund Adjusted | Refund Pending |
| REFUND | 68 | Refund In Process | Refund In Clearing |
| REFUND | 69 | Refund File Generated | Refund File Generated |
| REFUND | 70 | Refund Dispatched | Refund Processed |
| REFUND | 71 | Refund Failed | Refund Failed |
| REFUND | 72 | Refund Cancelled | Refund Cancelled |

4.1.3 - Transaction Payment Integration

* The Transaction Payment can be done by different methods through which a customer can make a payment for a transaction on the merchant's platform. Each payment mode represents a different channel or method of payment that the system can process.

##### 4.1.3.1 - Credit/Debit Card

* Payment is made using a credit card issued by a bank or financial institution.
* Requires entry of card details like card number, expiry date, and CVV.
* Payment Gateway need to be used PayU (Wibmo)

##### 4.1.3.2 - Net Banking – SBI / Channel Banks

* Payment is processed through the customer’s online banking portal.
* The customer is redirected to their bank's online banking page to complete the payment.
* For a successful corporate transaction status will be in the Pending state, the checker user has to complete the transaction within 24Hours, otherwise transaction will be marked as expired at SBIePay end. Post which if Checker will authorize the transaction it will treat as expired only.
* SBIePay application will run the scheduler for offline pulling with corporate bank which will fetch the status till 24 hours and update accordingly the transaction if response will received from bank as Success/Failed.

##### 4.1.3.3 UPI

* Payment is initiated using a UPI ID or Mobile number or QR code.
* Payment Gateway need to be used as BHIM-UPI

4.1.4 - Transaction Processing

UX Flow- Transaction UX flow is available on below figma link.

Link - [Page 1 - PaymentLatestScreen (figma.com)](https://www.figma.com/proto/87u5s50rTAUq1wEGDiBf9Y/PaymentLatestScreen?node-id=435-2043&t=hSyEv2o1vm54GJmr-0&scaling=contain&content-scaling=fixed&page-id=0%3A1)

4.1.4.1 - Form Posting

* The merchant passes the SBIePay-generated Order Id, via API call or URL call
* SBIePay validates the Order Id and, if valid, allows redirection to Transaction Page

4.1.4.2 - Link Based Approach

* Merchant Payment link will be generated if merchant has passed the payment link parameter. ExpiredDateTime –DDMMYYYY HH:MM: SS.
* Payment link will be activated as merchant is sending in parameter.
* Payment Link Expiry time cannot be set for more than 48 hrs., a check will be introduced in configuration page to handle the same.
* Merchant order expiry time would be same for Payment Link shared Expiry time.

4.1.5 - Merchant Payment Mode

* The Merchant Payment Mode must be validated against the payment methods pre-configured for the merchant. This ensures that only the payment modes that the merchant has enabled or supports are available for processing transactions

4.1.6 - Payment Mode Downtime

* The **Downtime API** is a critical component that ensures seamless transaction processing by checking whether any scheduled downtime is planned for payment channels. If a downtime is detected, the affected payment channels are temporarily inactive for customers for that period and info icon will be displayed with the information.
* This functionality helps prevent payment failures and enhances the overall user experience.
* The downtime schedule is managed and configured through the **SBIePay Admin** interface.

4.1.7 - Transaction Processing Amount

* To ensure accurate transaction processing, the system must calculate the transaction (TX) amount by considering various components such as the Order Amount, GST (Goods and Services Tax), and any additional configured charges and fees according to the Merchant's configuration. This ensures that the final transaction amount reflects all applicable costs.
* Transaction Amount Calculation Process
  + **Order Amount:**
    - **Definition:** The base amount for the transaction, representing the total cost of goods or services before any taxes or additional fees.
    - **Source:** Derived directly from the order placed by the customer.
  + **GST Amount:**
    - **Definition:** The tax amount calculated based on the applicable GST rate.
    - **Calculation:**
      * The GST amount is typically a percentage of the SBIePay Commission Amount.
      * The specific rate may vary depending on the product or service category and regional tax laws.
      * **Formula:** GST Amount= SBIePay Commission Amount × GST Rate (%)
      * GST Rate is configurable in the system as per prevailing rates.
  + **Configured Charges:**
    - **Definition:** These are additional charges configured by SBIePay Admin Interface, which could include handling fees, convenience fees, or other service charges.
    - **Source:** Configured within the SBIePay Admin Interface

#### 4.1.8 - Retry Functionality

* Retry is a new functionality, which will allow customers to retry transactions using other paymodes, if previous transaction is failed and transaction token time is not expired.
* For Each transaction attempt a unique ATRN will be generated.
* Maximum Five retry attempts will be allowed to customer as default.
* Retry attempt will be configurable merchant to merchant.
* Each retry attempt generates a new **ATRN if the** previous attempt is genuine failure from channel bank / payment channel.
* This new ATRN uniquely identifies each retry attempt, allowing for clear tracking and differentiation from previous attempts.
* No retry functionality will be available for Offline payment modes

Scenarios

* If transaction token is expired and no successful **payment** attempt is captured, all the ATRN(s) status will be marked as failed, and orderID status should be changed to ‘expired’ from ‘attempted’ status.
* If the Transaction token is valid and number of payments to be attempted count has been exhausted, then orderID status will be moved from ‘attempted’ to ‘failed’.

#### 4.1.9 - Refund / Cancellation Functionality

* The system allows merchants to initiate refund and cancellation requests for transactions identified by a valid **ATRN (Aggregator Transaction Reference Number)**. The eligibility for initiating a refund or cancellation is determined by the status of the ATRN, ensuring that only appropriate transactions are processed for these actions.
* **Refund Request**
  + Refund Request can be raised from Merchant panel or through API Call.
  + **Eligibility Criteria:**
    - A merchant can initiate a refund request only if the ATRN is in the **Paid** and **Settled** status.
    - This ensures that the transaction has been successfully processed and funds have been captured before any refund is initiated.
  + **Process:**
    - **Refund Initiation:** The merchant selects the transaction ATRN that is in the Paid and Settled status and submits a refund request through API & Merchant Panel.
      * Full Refund
        + Merchant raises refund request for the complete amount at once.
      * Partial Refund
        + Merchant initiates a partial refund from the currently available amount.
      * Bulk Refund
        + Merchant uploads a CSV file for Bulk refund via Merchant Panel.
    - **Validation:** The system validates that the ATRN meets the refund eligibility criteria. If it’s a partial refund, the system will validate the amount available for the refund.
    - **Refund Processing:** Once validated, the system processes the refund request according to the merchant’s configured refund policies and workflows.
    - **Status Update:** The transaction status is updated to **Refund** once the refund process is initiated.
* **Cancellation Request**
  + Cancellation Request can be raised from Admin panel or through API Call
  + Eligibility Criteria:
    - A merchant can initiate a cancellation request if the ATRN is in the **Paid** status.
    - **Paid Status:** This indicates that the transaction has been processed and the payment captured, but it hasn't moved to settlement.
  + **Process:**
    - **Cancellation Initiation:** The merchant selects the transaction ATRN that is in the Paid status and submits a cancellation request.
    - **Validation:** The system checks if the ATRN meets the criteria for cancellation.
    - **Cancellation Processing:** Upon validation, the system processes the cancellation request.
    - **Status Update:** The transaction status is updated to **Cancelled** upon successful processing of the request.

#### 4.1.10 - Operations

* Create, Get, Get All TX for a Merchant, Get TX by ATRN Number/Order Number/ Customer Id/ Customer Name/ Customer Phone Number/ Customer Email/Tx Status/ Refund/Cancellation Status/ Payout Status
* The requirement of All MIS related inquiries will be fulfilled with above APIs

#### 4.1.11 - Settlement / Reconciliation Processes

* Settlement and Reconciliation will happen from Legacy system only.

4.2 - Functional Flow

Please refer to the attached image. the attached image.



### 4.3 - Functional Specification

#### 4.3.1- Basic Validation Check

##### 4.3.1.1 - Token

* Verify the transaction token is valid to proceed to the creation of payment transactions

##### 4.3.1.2 - Validity

* Validate the merchant transaction time as per Configuration.

##### 4.3.1.3 - Merchant

* Check if the merchant is active and eligible to perform the requested operation.
* Verified fraud and risk associated with Transaction, as per Configuration

##### 4.3.1.4 - Base64Encode Check

* To ensure the security and integrity of transaction data, the system needs to validate that the transaction data which is encoded with Base64 encoded has been successfully decoded.

##### 4.3.1.5 - Transaction Creation Request (ATRN Generation Request) Should have below parameter

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Token | Yes | Valid Request Transaction Token for ATRN generation |
| OrderAmount | Yes | Order Amount |
| GatewayPosting Amount | Yes | Gateway Posting Amount |
| Order Id | Yes | Merchant Need to send the unique Number for which is generated during order Creation |
| Paymode | Yes | UPI/CC/DD/NB any applicable payment mode for a Merchant |
| PayGtwID | Yes | Payment Gateway Id |
| Paygtw | Yes | PayGateway (NB/CC/DC) |
| PayProc | Yes | Payproc (Self/Onus/offus) |
| Currency | Yes | Txn amount Currency |

#### 4.3.1.6 - Transaction Retrieval Validation Check (Transaction Status Inquiry)

##### 4.3.1.6.1 - Get All for getting all the transaction details for that Merchant

* At a single request 10,000 records will be provided with pagination option (till records are available)

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Token | Yes | Valid Request Token |
| Period | Yes | Date and Time |

4.3.1.6.2 - Get by ATRN Number/Order Number/Customer Id/ /Txn Status

|  |  |  |
| --- | --- | --- |
| Parameter | Mandatory | Description |
| Token | Yes | Valid Request Token |
| Request Type | Yes | ATRN Number/Txn Status |
| Request Value | Yes | ATRN Number/Txn Status |

#### 4.3.2 - Business validation

##### 4.3.2.1 Merchant Order Number Checks

* Check Merchant order number is valid or not to initiate the Transaction request

##### 4.3.2.2 ATRN Number

* The system should generate a unique ATRN number for each Transaction.

##### 4.3.2.3 RFC Checks

* **Card Txn Limit** - Based on the order Amount following limit set in daily card limit per day per card.
* **Volume Velocity Check** - Based on the order Amount following limit set in volume velocity should be checked for Daily/ Weekly/ Monthly / Quarterly / Half-Yearly Limit is being breached.
* **Payment Mode check** – Only Merchant level configured payment mode should be available for transaction.
* Name/Email/Mobile Number with the status
  + Block – The request will be blocked or marked as failed and will not proceed further.
  + However, the settlement will occur only after the RFC team reviews the order.
  + Not Applicable - The order booking process will be allowed to proceed without any restrictions.
* SDN - Specially Designated Nationals List
* OFAC - Office of Foreign Assets Control
* PEP - Politically Exposed Person
* Max Day Card Transaction Count: For Card Payments, System will check whether the Card Transaction Count.
* Dormancy: This will check whether merchant is dormant ot not.
* BIN Mode
* Threshold Amount: This will check whether the threshold amount is not configured or not.

#### 4.3.2.4 -Payment Method Validation

* UPI
  + **UPI-QR**:
    - "UPI-QR" is available only for web-based application users (desktop/laptop).
    - Mobile app users won't see this option.
    - Customers can generate a QR code for UPI payment by clicking "Generate QR."
    - The QR code must be scanned and used within the given expiration time.
    - If payment isn't completed in time, the transaction fails, and the QR code blurs.
    - A new QR code can be generated if needed.
    - Payments to expired QR codes result in a failed transaction and a refund.
  + **UPI ID**
    - Ensure that the UPI ID provided by the customer is in the correct format (e.g., username@bankname).
    - Validate the existence and accuracy of the UPI ID by sending a request to the UPI network.
  + **Merchant UPI Configuration**
    - Confirm that the merchant has enabled UPI as a payment method.
  + **Transaction Limits**
    - Validate that the transaction amount does not exceed the limits set by UPI (e.g., daily limits, per transaction limits).
  + **Customer UPI Verification**
    - Verify the customer’s UPI account and ensure it is active and capable of processing payments.
* Card

Checks at SBIePay end:

* + **Identification of Card:** To identify card processor.
  + **Card Validation:** To check Validity of card
  + **Bin Validation :** Validate the bin details
  + **DCMS API for only DC-Onus cards**
  + **Wibmo Auth API call**
  + **Card Number**
    - Validate the card number using the Luhn algorithm to ensure it’s a valid credit or debit card number which will be checked by the system.

Checks at WIBMO end:

* + **Card Number:**
    - Check that the card number belongs to a recognized card network (e.g., Visa, MasterCard, Amex).
    - The second level validation is done via the card processor (WIBMO) through API call.
  + **Card Expiry Date:**
    - Ensure the card is not expired by checking the provided expiry date. (API CALL with WIBMO)
  + **CVV (Card Verification Value):**
    - Validate that the CVV is a three or four-digit number, depending on the card network. (API CALL with WIBMO)
* Net Banking
  + **Bank Selection:**
    - Ensure that the customer is only able to view the list of banks which is configured for the merchant.
  + **Bank Availability:**
    - Check if the selected bank’s net banking service is currently available and not undergoing maintenance or downtime.
  + **Transaction Limits:**
    - Validate that the transaction amount adheres to the limits set.

#### 4.3.2.5 - Bank Double Verification

Once the web response of payment is successful, a direct server-to-server call happens between SBIePay server and the customer bank to validate the payment status, in a positive scenario the Order status is marked as Paid.

If SBIePay if not received the response from payment channel, for those transactions SBIePay will run offline pulling (DV) to get the status. Once response received from payment channel it will mark as success or failure for the same.

If no response received from bank end, it will be in queue to get the status till 5 attempts for Booked to fail (Merchant time interval-Gatewaytimeinterval to EpayTransactionDVExpiryTime) scheduler.

Once transaction is marked as success/fail from DV and failed from Book to fail scheduler, PUSH response will send to merchant if URL is configured at SBIePay end.

Expiry Time descriptions -

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Expiry Time** | **Time in minute** |
| 1 | AccessTokenExpiryTime | 30 |
| 2 | TransactionTokenExpiryTime | Merchant time interval as Configured in SBIePay |
| 3 | TransactionDVExpiryTime  (Book to Fail scheduler) | TransactionExpiryTime - Gatewaytimeinterval to = TransactionDVExpiryTime |
| 4 | OrderExpiryTime | OrderExpiryTime will be as per configration |

TRANSACTION STATUS & Description -

## NFR-1: Non Function Requirements:

5.1.**Active-Active Setup**: The system will be designed to operate in an Active-Active configuration. This means that multiple instances of the application will be running simultaneously on different servers.

5.2. **Horizontally Scalable:** The application will be capable of horizontal scaling. This means that the system should automatically add more instances or reduce instances based on the transaction volume.

5.3 **Multi-Lingual Application:** The Payments page will be provisioned for Multiple language view & user will be able to choose language; view the payments page in multi language.

5.4 **Customized Branding:** The Payments page will be provisioned as per Merchants choice. Customization will include: Color coding of the payments page and Logo of the merchant.