Business Requirement Mapping

|  |  |  |
| --- | --- | --- |
| Epic 1 | FS\_TX 4 | Details |
| 1 | [3.1.1](https://neosoftmail1-my.sharepoint.com/personal/harshal_kirtane_neosoftmail_in/Documents/Microsoft%20Teams%20Chat%20Files/Transaction_Epic_Epay.docx#Transaction token Generation) | Transaction token Generation |
| 2 | [3.1.2](#_Transaction_State_Management) | Transaction State Management |
| 4 | [3.1.5](#_Operations) | Operations |

## Epic Name: Transaction Creation Module for SBI ePay

### **Description**

This Epic covers the implementation of the Transaction token creation, ID creation and Processing module for SBI ePay. It handles Transaction lifecycle management, including transaction booking, processing and status transitions (Booked, Paid, Expired, Failed, Refund, cancel, Settled).

### Actors

1. Merchant
2. Aggregator
3. Customer

### Objective

* **Transaction initiation:** Create transaction token and share with Merchant to process transaction securely, ensuring the integrity and uniqueness of each transaction.
* **Transaction Processing:** Manage the execution of transaction by considering various components of transaction amount such as Order amount, GST, and any additional configured charges and fees according to the Merchant's configuration

### Functional Details

#### Transaction token Generation

* **Description:**
  + - **Transaction token generation:** Once Order is created, Merchant requests token for transaction to receive payment against the same order.
    - Aggregator checks all required details to create transaction token for the requested order.
* **Technical Considerations:**
  + Ensure uniqueness at the database level (apply unique constraints).
  + Design a sequence generation mechanism that can handle a high volume of transactions without collisions.
  + All mandatory parameters should be received from Merchant in correct format
  + Merchant should be active and Customer details if any should be validated.

#### Transaction State Management

* Defines and manages the states of an transaction:
  + **Booked**
  + **Expired**
* Final transaction state should be maintained in DB

#### Transaction processing

* **Description** 
  + There are 2 payment redirection methods that can be selected by Merchant to initiate transaction
    - **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
    - **Link Based Approach**
      1. Merchant Payment link will be generated if merchant has passed the payment link parameter. ExpiredDateTime –DDMMYYYY HH:MM: SS.
      2. Payment link will be activated as merchant is sending in parameter.
      3. Payment Link Expiry time cannot be set for more than 48 hrs., a check will be introduced in configuration page to handle the same.
      4. Merchant order expiry time would be same for Payment Link shared Expiry time.
  + **Technical Consideration:**

Hash value is required in Transaction booking, it consists of Order ID+MID.

When Merchant initiates Form posting, Aggregator shares Hash value to authorize the transaction processing or when Merchant requests for Payment link, Hash value is used to authorize the Merchant and Order for the same transactions.

##### **2.1. Booked**

* **Description:** Initial state after a successful booking of the Transaction.
* **Transition Conditions:**
  + Transaction remain in the "Booked" state until payment is initiated.
  + If no payment is attempted within the merchant-specific expiry time (TransactionTokenExpiryTime), the transaction status will transition to "Expired."
* **Technical Considerations:**
  + Implement logic for expiry handling based on configurable time.
  + Batch processes or Scheduler Job may be used to automatically transition transaction to "Expired."

##### **2.2. Expired**

* **Description:** The order has become invalid due to no action has been performed in the configured time window.
* **Transition Conditions:**
  + Transition moves from "Booked" to "Expired" if the order exceeds its expiry time (c) without a successful payment.
* **Technical Considerations:**
  + Expiry management via background jobs or Scheduler tasks.
  + Token expiry time based on merchant's configuration at SBI ePay end.

#### Operations

##### **4.1. Book Transaction**

* + **Description:** For secure transaction booking, Merchant request token. After validating all required parameters related Merchant, order and customer, aggregator shares token with Merchant.
  + **Process Flow**
    1. **Receive transaction booking request:** Merchant sends encrypted request, to generate transaction token, which includes sensitive data like customer details and order information, paymode and transaction parameters (including downtime)
    2. **Encrypt Sensitive Data**
       1. Sensitive information (related to order, customer, transaction) is encrypted using **AES-256 GCM** before storing or transmitting.
       2. Encryption keys are securely retrieved from a **Key Management Service (KMS)** or equivalent system.
       3. Merchant ask for token using Order ID + Mid to generate Hash Value
       4. Aggregator decrypts the message from Merchant using secure key and validates the token request before responding, based on mandatory parameters and expiry time.
       5. Background jobs or Scheduler tasks keeps running to monitor Token expiry
       6. Merchant can revoke token in case of security breach and ask for new token for the same transaction.
       7. All token requests/response should be logged into DB for audit purpose
    3. **Generate transaction token**
       1. A unique Transaction token generated by Aggregator
    4. **Persist transaction token in Database:** Store the encrypted sensitive data, along with non-encrypted data required for transaction booking.
    5. **Return Response:** Post validation**,** Aggregator responds with token key and Hash value consist of Order ID and MID.

This response also has Merchant redirection method defined by Merchant.

* + **Technical Considerations**
    1. **Encryption:** Implement field-level encryption for sensitive data.
    2. **Logging:** Ensure sensitive data is not logged in any raw form., DB Logging is also happening for traceability
    3. **Validation:** Ensure merchant-specific validations are applied before transaction booking and encrypting and persisting data.
    4. **Token validity:** Check if token is valid to call the service else Merchant will requests for new token
    5. **Error Handling:** Below list of scenarios to be handled giving proper Error Message.
       1. Decryption fails
       2. Token generation fails
       3. Wrong token request received
       4. Security violation
       5. Any other validation fails during RFC check

##### **4.1.1 Transaction Payment Integration**

Payment integration can be done on SBI ePay using following Pay modes

* + - 1. Credit/Debit Card
      2. Net Banking – SBI / Channel Banks
      3. UPI

**4.1.2 Payment Mode Downtime**

* + **DownTime API** - Scheduled downtime as per payment channels that need to be displayed on SBIePay page.

The downtime schedule is managed and configured through the **SBIePay Admin** interface.

* + 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
* When Merchant redirects customer using redirection links provided by aggregator, it will have Hash value, Customer information, Order information, Payment downtime, Charges, GST, fee amount, Merchant Paymode etc.
* GST, Other charges, fees are calculated at SBIePay admin interface, as per merchant configuration.

**4.1.3 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
  + **Technical Consideration:**

Error Handling: -

* + - 1. When Paymode doesn’t match with Merchant Paymode.
      2. When charges/fees not calculated or wrongly calculated

1. When downtime does not reflect on user interface.

##### **Notification:**

* **Description:** Notification services configured/activated based Merchant’s requirement.

E.g.:- SMS & Email notification service for Order related activities

* + - 1. Transaction Booking
      2. Transaction Expired

### Technical Implementation Notes

1. **Database Design:**
   * Tables for transaction Information, Status History, Retry Attempts, and Expiry Management.
   * Unique constraints for transaction ID and Merchant Order Number.
   * Indexes on frequently queried fields such as Transaction ID
2. **API Design:**
   * RESTful APIs for transaction booking, retrieval and updates.
   * Secure endpoints for handling sensitive transaction and order data.
3. **Logging & Monitoring:**
   * Implement detailed logging for all transaction state transitions, payment attempts, and expiries.
   * Real-time monitoring for expired and booked transactions
4. **Encryption & Decryption:** 
   * The entire transaction request or specific sensitive fields (e.g. transaction Amount”, order Amount, GST Amount, fees Amount, charges Amount, total Amount, Customer ID) will be encrypted before being transmitted to the server or external systems.
   * Use **Encryption and Decryption Service** for strong security with authenticated encryption.
5. **Token Validation:** transactiontokenexpirytime configured at merchant level, for every API call the token validity will be checked. If token expires, new token request should be initiated by Merchant to continue using the services.