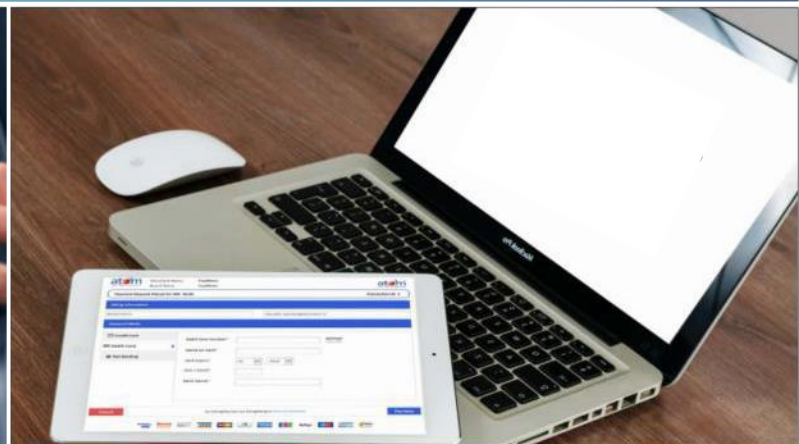# Merchant Integration Document

Net Banking / Internet Payment Gateway

Atom Technologies Limited

FT Tower, CTS No-256 & 257, 1st Floor,
Suren Road, Chakala, Andheri (E),
Mumbai 400093, India

☎ +91 22 6686 4010   🖷 +91 22 6686 4050

✉ info@atomtech.in   🌐 www.atomtech.in

atom
payment solutions for every business

1. Background:

This note briefly describes the mode/ manner of technical integration between Atom Payment Gateway and Merchant website in respect of powering online transactions.

2. Electronic Payment Interface:

Atom offers electronic payment interface services to merchant organizations through its partnership with various banks and card companies.

The Electronic Payment Interface (EPI) is an API provided by atom Technologies on its

Paynetz platform to facilitate electronic commerce transactions.

This facility allows for the end users to make electronic payment through credit card and online net banking accounts.

Atom also provides a Merchant Console that support various features including transaction settlement reports, query module, refund module.

3. Process Flow:

This section briefly details the overall customer transaction flow.

Transaction Process

1. Customer logs-in at the merchant website and selects the product / service to purchase. Based on the purchase amount, appropriate transaction amount would be computed at the merchant website.

2. Customer then decides to make payment at merchant website.

3. Merchant website will log the order by generating a unique Order Number; and establish a connection with the Atom Electronic Payment Interface [**refer the section on Payment Request**].

4. The customer lands on the **Atom Payment Page** where the customer is displayed with various 'payment options' that can be used. These would include–Credit Card; Debit Card; Online Net Banking.

5. Customer chooses the payment option at **Atom Payment Page**, and is further redirected to the page of that respective bank. Customer then enters the relevant authentication details [ie, User ID/ Card Number/ Password] at the bank's website; and then is requested to authorize the payment amount.

6. Customer's account is debited and the Customer is then redirected back to the designated Return URL at Merchant website where the status of the transactions is displayed to the customer.

7. The Atom Electronic Payment Interface will provide the **return response** to the designated Merchant return URL. Merchant can use this response to update its system and display the transaction status to the customer.

8. The Atom Electronic Payment Interface also generates a **unique Transaction ID** against each order number that is received.

Transaction Process Flowchart
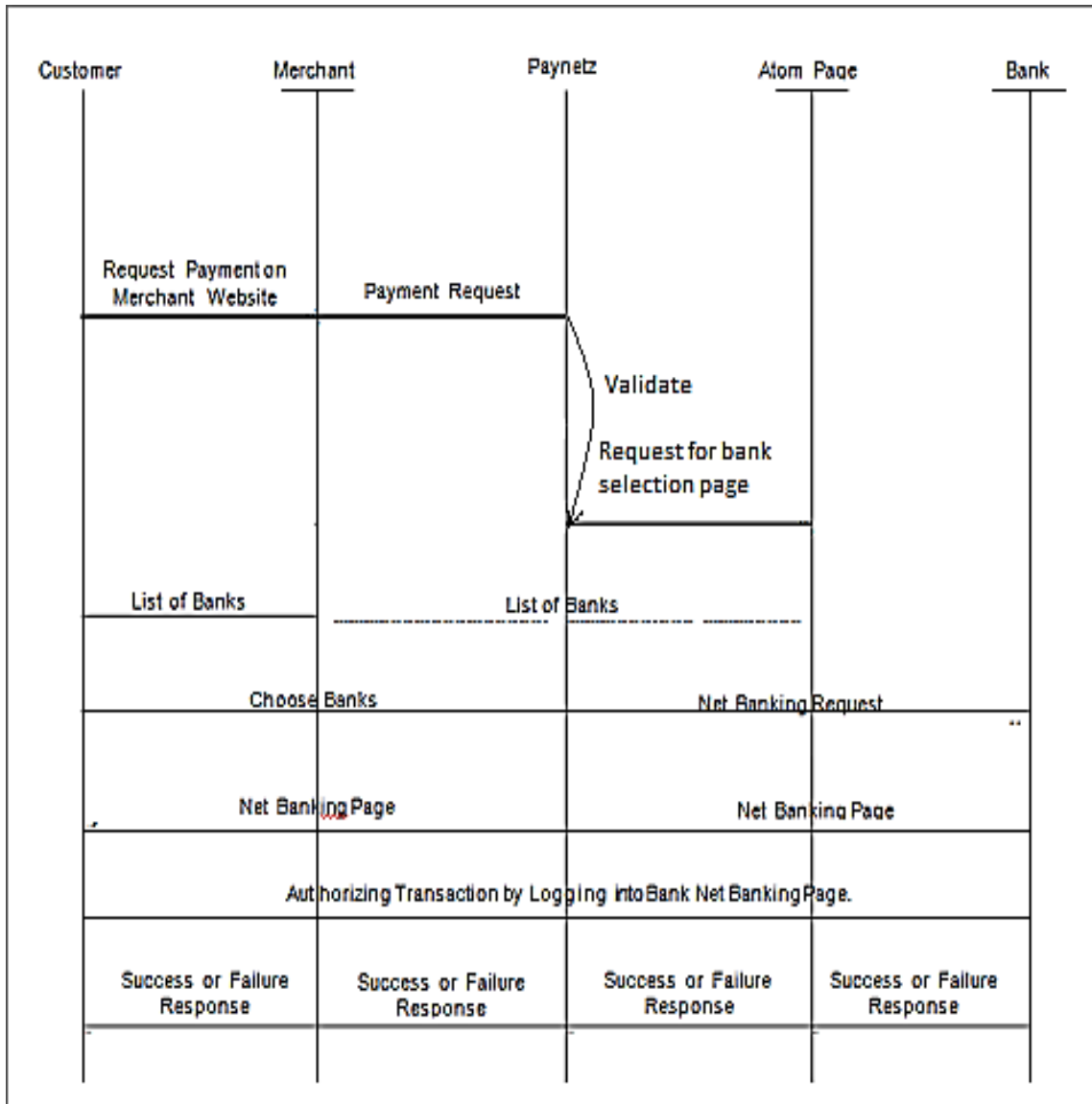
4. Technical Integration Process:

Key aspects of the integration between the Merchant website and Atom are described in the paragraphs below.

After the customer clicks on PAY [within Merchant website], the merchant initiates the payment by generating a request using Atom Electronic Payment Interface (EPI) by sending a POST method request to the URL.

To make the payment request secure, we have implemented parameter posting with signature (i.e. checksum). Signature parameter is created by hashing the value of below mentioned parameters.

      i.   login

      ii.   pass

     iii.   ttype

     iv.   prodid

      v.   txnid

     vi.   amt

    vii.   txncurr

Note: Details of above parameters are provided in table on page 7.

The hash value should be generated using HMACSHA512 algorithm with hash key. Hash key will be unique and shared by atom team. 'signature' value generated will be in byte and needs to be converted into hex. 'signature' parameter is mandatory and need to be added in request URL. After receiving the request Atom Electronic Payment Interface (EPI) will generate signature using parameters present in request and verify created signature with signature which is present in request URL. User will be redirected to payment page If both signature match.

Server Details:

Test Server Details:

1. **URL:** https://paynetzuat.atomtech.in/paynetz/epi/fts

2. **Port for https:** 443 for Test Server

Test Server Credentials:

3. **Login ID: 197**
4. **Password: Test@123**
5. **Product ID: NSE**
6. **ReqHashKey:** KEY123657234
7. **RespHashKey:** KEYRESP123657234

Production Server Details & Credentials

8. https://<Domain Name>/paynetz/epi/fts

9. **Domain Name:** Will be shared at the time of production movement

10. **Credentials:** Will be shared at the time of production movement

11. **Product ID:** Will be shared at the time of production movement
12. **ReqHashKey:** Will be shared at the time of production movement
13. **RespHashKey:** Will be shared at the time of production movement

Changes required moving from test server to production

14. **Port for https:** 443 for Production Server

15. **Server IP**: Domain Name

16. **Login & Password**

17. **Product ID**

Payment Request -

Parameter Details for Payment request

A brief explanation of parameters is below:

Parameter Name	Parameter Description	Field	Field Type	Length	Sample Value	Explanation
login	Login ID obtained on registration of Merchant URL's and IP	Mandatory	Numeric		Provided by atom	Fixed Value
pass	Password obtained on registration of Merchant URL's and IP	Mandatory	Alphanumeric	8-20	Provided by atom	Password as set by merchant
ttype	NB Fund Transfer in this parameter	Mandatory	Alpha		NBFundTransfer	NBFundTransferfor all transactions. If the card details are captured by merchant then 'CCFundTransfer'
prodid	Product ID as decided and approved by the Merchant	Mandatory	Alphanumeric	45	NSE' can be used for Testing Purposes.	Based on merchant integration understanding. The possible values for this field will be shared by Atom at the time on production movement
amt	Transaction Amount	Mandatory	Numeric		Transaction Amount, min should be Rs.50.00	Rs.ps format should be 2 decimal places
txncurr	Currency	Mandatory	Alpha	3	INR	Fixed Value
txnscamt	Transaction Service Charge Amount. Charged by the merchant.	Mandatory	Numeric		"0"	Fixed Value
Clientcode	Client code	Mandatory	Alphanumeric	45	Identifier of the end customer. If identifier is not available then always pass a standard Value. Base64 Encryption needs to be used.	As passed by merchant, encoded with base 64.
txnid	Merchant's	Mandatory	Alphanumeric	50	Unique	As passed by merchant, unique

					Transaction ID generated by the Merchant	transaction id
	Transaction ID				Transaction ID generated by the Merchant	transaction id
date	Date of Transaction	Mandatory	Numeric		Exact date and time stamp	DD/MM/YYYY HH:MM:SS
custacc	Customer's Account No	Mandatory	Numeric	45	Always pass a standard 10-12 digits value. E.g. 1234567890	Used for Broker integration. In normal integration Always pass a standard 10-12 digits value. E.g. 1234567890
mdd	Application Identifier	Optional	Alpha	300	See table Below	If no values passed then all banks /credit card options enabled for merchant will be displayed to the customer on Atom payment option page. See Table below for all Possible values
ru	Return URL	Optional	Alphanumeric	200	URL on which merchants needs the transaction status	If no values passed in this filed than the trx respose will be posted to the default merchant URL configured in atom's system.
signature	signature generated using HMACSHA512 algorithm	Mandatory	Hex	N/A	704e6a78ca61c89127ca5430ddf59a329dacb142ae2790e19d676f5ca8ca80b9c534552e877bfb0ce1c2dddf252ae3d7580d329556213811f828711e9f4ea371	Signature will contain the hash values of the login, pass, ttype, prodid, txnid, amt, txncurr parameters generated using HMACSHA512 algorithm.
udf1	Customer Name	Optional	Alphanumeric	45	Name of end customer	Mandatory in case of Prepaid Wallet
udf2	Customer Email ID	Optional	Alphanumeric	100	Email ID of end customer	Mandatory in case of Prepaid Wallet
udf3	Customer Mobile No	Optional	Numeric	10	Mobile No of end customer	Mandatory in case of Prepaid Wallet
udf4	Billing Address	Optional	Alphanumeric	200	Billing address of end customer	
udf5	Merchant Data	Optional	Numeric	45	Only for EMI merchant	Bank names
udf6	Merchant Data	Optional	Numeric	45	Only for EMI merchant	EMI tenures
udf9	Merchant Data	Optional	Alphanumeric	500	Echo back Parameter	Treated as Echo back parameter for merchant in respose from Atom

Possible Values for Parameter 'mdd' –

Field Value	Effect on Payment option Page
NB	In case the merchant wishes to show only Net Banking as payment option to customer.
CC	In case the merchant wishes to show only Credit Cards as payment option to customer.
DC	In case the merchant wishes to show only Debit Card as payment option to customer.
AMEX	In case the merchant wishes to show only American Express as payment option to customer.
VISADC	In case the merchant wishes to show only VISA Debit Card as payment option to customer.
WALLET	for Atom prepaid wallet

Sample URL for First Payment Request

Merchant website will log the order by generatinga unique Order Number; and establish a connectionwith the Atom Electronic Payment Interface

Sample URL for needs to generate by Merchant integrated module

https://paynetzuat.atomtech.in/paynetz/epi/fts?login=197&pass=Test@123&ttype=NBFundTransfer&prodid=NSE&amt=100.00&txncurr=INR&txnscamt=0&clientcode=001&txnid=M123&date=23/08/2010%2011:57:00&custacc=100000036600&udf1= bobsmith &udf9=ABCD&ru=https://paynetzuat.atomtech.in/paynetzclient/ResponseParam.jsp&udf2=smith.bob@gmail.com&udf3=9999999 999&signature=704e6a78ca61c89127ca5430ddf59a329dacb142ae2790e19d676f5ca8ca80b9c534552e877bfb0ce1c2dddf252ae3d 7580d329556213811f828711e9f4ea371

Note: udf parameters are required to be captured from customer and need to pass in the request else the same will be captured on payment page.

Client Code Logic

While sending request to atom, value sent in client code needs to be converted in to base64 encoding.

https://paynetzuat.atomtech.in/paynetz/epi/fts?login=197&pass=Test@123&ttype=NBFundTransfer&prodid=NSE&amt=100.00&txncurr=INR&txnscamt=0&clientcode= TkFWSU4= &txnid=M123&date=23/08/2010%2011:57:00&custacc=100000036600&udf1= smithbob &udf9=ABCD&ru=https://paynetzuat.atomtech.in/paynetzclient/ResponseParam.jsp&udf2=smith.bob@gmail.com&udf3=9999999999&signature=704e6a78ca61c89127ca5430ddf59a329dacb142ae2790e19d676 f5ca8ca80b9c534552e877bfb0ce1c2dddf252ae3d7580d329556213811f828711e9f4ea371

In the above URL the client code is NAVIN, this is Base64 encoded which gives TkFWSU4= and this in turn is URL Encoded which gives TkFWSU4%3d

5. Payment Response

After the Customer chooses the payment option at Atom Payment Page, he/she is further redirected to the page of that respective bank. Here customer authorizes the payment by giving the relevant authentication details. Post authentication the customer's account is debited & the bank gives the transaction response to Atom Electronic Payment Interface (EPI). The confirmation from the Bank (success / failure) will be posted back to the URL provided by the Merchant. The Merchant needs to design to display appropriate message on the web page.

Response Posting Parameters

The below are the posting parameters, that atom will post back on to return URL of the merchant:

Parameter Name	Parameter Description	Parameter Sample Value
mmp_txn	atom transaction ID. The ID is generated by atom	1111492
mer_txn	Merchant'stransaction ID	1
amt	Amount	100.00
surcharge	Amount of Surcharge	100.00
prod	Product ID. For testing you can use "NSE", for Production Purposesmerchant can suggest the name.	NSE
date	Date & Time Stamp of the Transaction	Fri May 14 12:03:24 IST 2010
bank_txn	Bank Transaction ID. This ID is generated by the Bank	11114921
f_code	OK for Successful transaction F for Failed Transaction C for Cancelled Transaction	Ok
clientcode	Client code needs to be Passed by the merchant. This field is to identify the customer. In case customer logins in to perform transaction, his user id or customer id can be passed otherwise you can send a constant value.	Mt012
bank_name	Bank on which customer performed transaction. In case of credit/debit card transactions, bank name will be received as Atom Bank. For testing it will be a single bank called "Atom Bank".	Atom Bank
discriminator	Paymentchannel	NB (**See Table below for all Possible values)**
CardNumber	Maskedcard number for Credit card / Debit card trx or blank in case of NB / IMPS	401288XXXXXXX1881
signature	Signature will contain the hash values of the mmp_txn, mer_txn, f_code, prod, discriminator, amt, bank_txn parameters generated using HMACSHA512 algorithm.	fab572693f4d546a947b2ccceec1eee5c282414949aa6145f11f884e72cb85b9aa6f430c235fb8eee254e7be7121c2524b987381d243fb27f5f1e2a6bab610a8
udf1	Customer Name	As passed by merchant while initiating transaction

udf2	Customer Email ID	As passed by merchant while initiating transaction
udf3	Customer Mobile No	As passed by merchant while initiating transaction
udf4	Billing Address	As passed by merchant while initiating transaction
udf5	Merchant Data	As passed by merchant while initiating transaction
udf6	Merchant Data	As passed by merchant while initiating transaction
udf9	Merchant Data	As passed by merchant while initiating transaction
desc	This filed contains the transaction failure reason in case of Card transaction. This filed is not mandatory. If no values passed by the bank, this filed will not be posted in response.	Card Expired

Possible Values for Parameter 'discriminator' –

Value	Payment Channel
NB	Net banking
CC	Credit Cards
DC	Debit Card
MX	American Express Cards

Verification of response using 'signature' parameter -

Response posted on return URL contains 'signature' parameter consisting of hashed value of following parameters.

   i.   mmp_txn
   ii.   mer_txn
   iii. f_code
   iv.   prod
   v.   discriminator
   vi.   amt
   vii. bank_txn

Same as 'signature' present in request URL, 'signature' is generated using HMACSHA512 algorithm with hash key. Hash key will be unique and shared by atom team. After receiving the response, 'signature' can be generated using above parameters.
If Signature generated using HMACSHA512 algorithm and signature present in response URL matches, it confirms that response is proper.

6. Transaction Tracking URL:

The transaction Tracking URL is the tracking device provided to merchant to:

1. Reconfirm on the online real time transaction status received

2. Check the status of the transaction for which the online real time transaction response is not received by the merchant due to various reasons as session timed out, dropped transaction etc.

The Merchant can hit the atom Paynetz system through Transaction Tracking URL after 15 minutes of the transaction that was initiated in case transaction are timed out. Success transaction

To verify Success transaction status Transaction tracking URL can be hit immediately.

Parameters details for Tracking URL

Parameter Name	Description
merchantid	Merchant ID provided by atom
merchanttxnid	Unique transaction ID sent by the merchant after the first transactionrequest was initiated
amt	Transactionamount
tdate	Transactiondate in YYYY-MM-DD format

Sample URL for Tracking URL

Request format for tracking URL on staging server

https://paynetzuat.atomtech.in/paynetz/vfts?merchantid=160&merchanttxnid=1015&amt=100.00&tdate=2012-08-11

Response from tracking URL

Parameter Name	Description
MerchantID	Merchant ID provided by atom
MerchantTxnID	Unique transaction ID sent by the merchant after the first transaction request was initiated
AMT	Transaction amount
VERIFIED	**Initiated –** in Case customer is on Payment page and not submitted payment request. **SUCCESS** – in case of success transaction. **FAILED** – in case of failure transaction. **NODATA** – in case of mismatch of value in the request. **Pending from Bank –** in case atom not received response from bank **Invalid date format** – in case of date format not received as YYYY-MM-DD
BID	Reference No. received from bank only for success transactions.
bankname	Bankon which customerperformedtransaction. In case of credit/debit card transactions, bank name will be received as "ATOM PG". In case of IMPS transactions, bank name will be received as IMPS".
atomtxnId	Transaction ID generated by atom.
atomtxnId	Transaction ID generated by atom.
discriminator	Payment Channel
CardNumber	Maskedcard numberor blank in case of NB / IMPS
surcharge	Amount of surcharge (Transaction Fee) in Rs.Ps format

Sample Response URL for Tracking Request

<?xmlversion="1.0"encoding="UTF-8"?> <VerifyOutputMerchantID="160" MerchantTxnID="1015"AMT="100.00"VERIFIED="SUCCESS"BID="123456" bankname="HDFCBank"atomtxnId="665544"discriminitor="NB"CardNumber="401288XXXXXXX1881"surcharge="10.00'/>

7. Atom Payment page Screens:
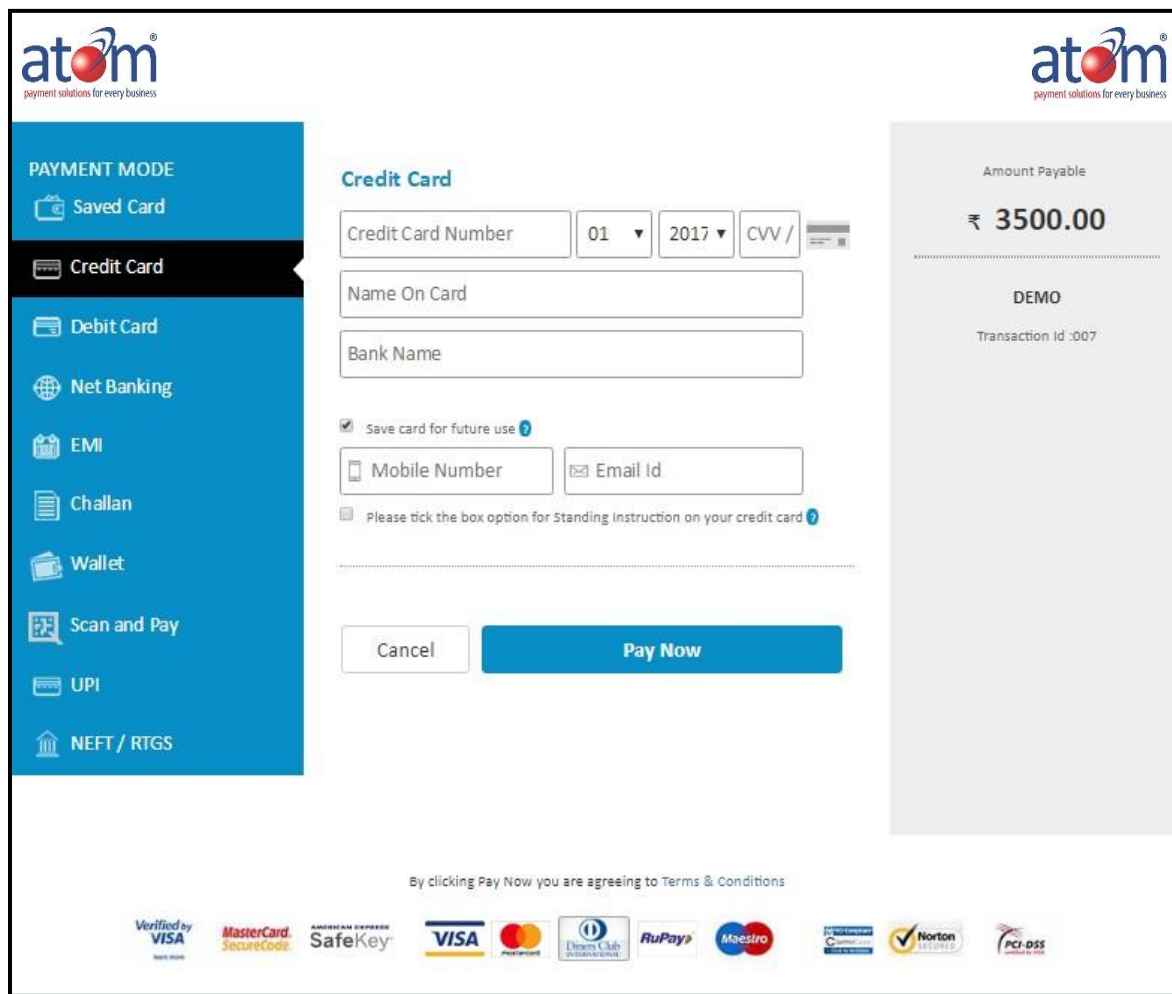
a. Payment Page

b. **Customer Enters mobile number and email Id and chooses mode of payment.**

[Note "It's not possible to test credit/debitcard transaction on staging server, as 3D secure password needs to be entered on bank verificationpage, which is not available on staging server."]

i. Credit Card

ii. Debit Card

iii. Net Banking

iv. Cancel Transaction

V. Transaction Response: Success

vi. Transaction Response: Fail

8. UAT Test Credentials:

1. **Netbanking**: Atom has created a stimulator bank called atom bank You can select atom bank which will redirect you to a page which will have options of success and failure depending upon the selection response will be send to your return Url.

2. **Credit card**:
   Card number: 4012 8888 8888 1881
   Expiry:12/25
   Cvv:456

3. **Debit card**:
   Card Number:5555 5555 5555 4444
   Expiry 12/16
   Cvv:123

9. Sample Java Code

1. Signature generation

```java
public class SignatureGenerate {

    /*
     * Hashing using key with HMACSHA512
     */
    public static byte[] encodeWithHMACSHA2(String text,String keyString)
    {

        java.security.Key sk = new
        javax.crypto.spec.SecretKeySpec(keyString.getBytes("UTF-
        8"),"HMACSHA512");
        javax.crypto.Mac mac =
        javax.crypto.Mac.getInstance(sk.getAlgorithm());
        mac.init(sk);

        byte[] hmac = mac.doFinal(text.getBytes("UTF-8"));

        return hmac;
    }
```

```java
/*
 * Convert from byte array to HexString
 */
public static String byteToHexString(byte byData[])
{
      StringBuilder sb = new StringBuilder(byData.length * 2);

      for(int i = 0; i < byData.length; i++)
      {
            int v = byData[i] & 0xff;
            if(v < 16)
                  sb.append('0');
            sb.append(Integer.toHexString(v));
      }

      return sb.toString();
}


/*
 * Encoded with HMACSHA512 and encoded with utf-8 using url encoder for
   given list of parameter values appended with the key
 */
public static String getEncodedValueWithSha2(String hashKey,String ...param)
{
      String resp = null;

      StringBuilder sb = new StringBuilder();
      for (String s : param) {
            sb.append(s);
      }

      try{
            System.out.println("[getEncodedValueWithSha2]String to Encode ="
            + sb.toString());
            resp = byteToHexString(encodeWithHMACSHA2(sb.toString(),
            hashKey));
            //resp = URLEncoder.encode(resp,"UTF-8");

      }catch(Exception e)
      {
            System.out.println("[getEncodedValueWithSha2]Unable to encocd
            value with key :" + hashKey + " and input :" + sb.toString());
            e.printStackTrace();
      }

      return resp;
}
```

```java
public static void main(String[] args) {

        String login = "7";
        String pass = "Test@123";
        String ttype = "NBFundTransfer";
        String prodid = "NSE";
        String txnid = "Mer123";
        String amt = "3000.00";
        String txncurr = "INR";
        String reqHashKey = "KEY123657234";
        //login,pass,ttype,prodid,txnid,amt,txncurr
        String signature_request = getEncodedValueWithSha2(reqHashKey,
        login,pass,ttype,prodid,txnid,amt,txncurr);
        System.out.println("Request signature ::" + signature_request);


        //Response signature based on parameters

        String mmp_txn = "121212";
        String mer_txn = "Mer123";
        String f_code = "Ok";
        String prod = "NSE";
        String discriminator = "NB";
        amt = "3000.00";
        String bank_txn = "Bank123";
        String respHashKey = "KEYRESP123657234";
        //mmp_txn,mer_txn, f_code, prod,  discriminator, amt, bank_txn
        String signature_response = getEncodedValueWithSha2(respHashKey,
        mmp_txn,mer_txn, f_code,prod, discriminator, amt, bank_txn);
        System.out.println("Response signature ::" + signature_response);
    }
}
```

2. Decryption logic

```java
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;

public class AtomAES
{
  private String password = "8E41C78439831010F81F61C344B7BFC7";
  private String salt = "200000054575202";
  private static int pswdIterations = 65536;
  private static int keySize = 256;
  private final byte[] ivBytes = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14, 15 };

  public String decrypt(String encryptedText, String key, String
merchantTxnId) throws Exception
  {
        this.password = key;
        this.salt = merchantTxnId;
        return decrypt(encryptedText);
  }

  private String decrypt(String encryptedText)
            throws Exception
  {
        byte[] saltBytes = this.salt.getBytes("UTF-8");
        byte[] encryptedTextBytes = hex2ByteArray(encryptedText);

        SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
        PBEKeySpec spec = new PBEKeySpec(
                    this.password.toCharArray(),
                    saltBytes,
                    pswdIterations,
                    keySize);

        SecretKey secretKey = factory.generateSecret(spec);
        SecretKeySpec secret = new SecretKeySpec(secretKey.getEncoded(),
"AES");

        IvParameterSpec localIvParameterSpec = new
IvParameterSpec(this.ivBytes);
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(2, secret, localIvParameterSpec);

        byte[] decryptedTextBytes = (byte[])null;
        decryptedTextBytes = cipher.doFinal(encryptedTextBytes);

        return new String(decryptedTextBytes);
  }
```

```
private byte[] hex2ByteArray(String sHexData)
{
      byte[] rawData = new byte[sHexData.length() / 2];
      for (int i = 0; i < rawData.length; ++i)
      {
            int index = i * 2;
            int v = Integer.parseInt(sHexData.substring(index, index + 2),
16);
            rawData[i] = (byte)v;
      }

      return rawData;
}
}
```