



PayU Biz Recurring Platform

Product Overview –

PayU's recurring platform allows merchant to offer standing instruction feature for Credit and selected Debit cards through various integration methods. The feature is intended to automate recurring payments in uses cases of Subscription Business where billing amount and billing cycle is usually fixed and consumer's preferred payment instrument (Credit Card) is used to charge for required service regularly.

Recurring payments don't require consumer's involvement for completing transaction because they are processed without CVV and 2nd factor authentication. This results into frictionless payment experience given the fact that consumer has already provided his/her consent for allowing merchant to charge his card regularly.

Since recurring payments don't have 2nd factor authentication, there are strict guidelines around offering the same to consumers by RBI as stated below -

- The first transaction has to go through a standard 2FA flow (OTP / MasterCard Secure password / Verified by Visa Password) where consumer's consent for further recurring payments needs to be taken either by merchant (seamless flow) or PayU (non-seamless flow)
- Once consent is taken then merchant can use either S2S APIs or File Upload utility of PayU to charge consumer over recurring basis going forward without 2FA.

How Recurring Payment works?

- Consumer lands on merchant website and proceeds for payment.
- Merchant presents an option to sign up for recurring platform where consumer needs to provide his/her consent.
- As per RBI guidelines, the first transaction where consumer is providing his consent needs to be processed with 2nd factor authentication (similar to e-com transaction)
- Hence consumer is redirected to 3DS flow where authentication and authorization process is taken place.
- There are couple of ways to process First transaction or Consent transaction and taking consumer's consent -



- o Consent transaction can be actual subscription for 1st billing cycle so consumer will be charged for whole amount through 3DS (2 factor) flow and sub-sequent transactions will be processed through recurring payments.
- o Consent transaction can be a penny transaction (5rs) where consumer's card is taken on file along with consent and then amount is refunded back by merchant on calling Refund API. This is popular in cases where merchant is offering free services for 1st billing cycle and then charges sub-sequent bills through recurring payments.
- Once consumer's consent is taken then consumer's card details are saved in PayU's secure Vault and a card token is generated.
- This card token is returned back to merchant in the payment response along with payu's id. Merchant is supposed to map this PayU Id it against consumer's profile so that henceforth it can be used charging consumer through recurring platform.
- Please note that the card token is not actual card number and hence merchant is not having any PCI DSS hassles in storing the same at his end.

Supported Card Schemes -

Credit Cards -

- Visa Cards (All Banks)
- Master Cards (All Bank)
- Diners Cards (only HDFC Bank)

Debit Cards -

- Visa (Only ICICI and HDFC Banks)
- Master (Only ICICI and HDFC Banks)



Integration Steps -

To offer recurring platform, merchant needs to integrate two flows as per the standard guidelines -

- Integrating Consent transaction (with 2nd factor authentication)
- Integrating Recurring transaction (without 2nd factor authentication)

Supporting Consent transaction -

Let's look at how Consent transaction can be offered by merchant using PayU APIs. Here it is absolutely important for merchant to present billing cycle which includes transaction amount to be charged over recurring cycle along and frequency of the charging i.e. either every week or every month or every six months etc.

Also consent check-box should be presented which needs to be checked by consumer to proceed further with transaction.

Today there are two different flows with which merchant can take consent of the consumer for SI transaction -

1. Seamless Integration
2. Non-Seamless Integration

In case of Seamless flow merchant needs to present card entry screen on his own page and consumer's consent form also needs to be taken by merchant. This is applicable for PCI DSS compliant merchant so that card details are handled at merchant's site directly.

In case of Non-Seamless flow, merchant needs to present billing cycle and frequency of charge along with consent form however card details will be taken by PayU as shown below –

Amount: Rs. 1000.00

Transaction ID: **f7cc03802f5489e35b4f**

Choose a payment method

PAYU is now PayU**biz**

Credit Card

Debit Card

Card Type  

Card Number

Name on Card

CVV Number [What is CVV number?](#)

Expiry Date

☒ I agree to save card details for Standing Instruction. [Learn more](#)

Type a custom label to save this card (optional)

Note: In the next step you will be redirected to your bank's website to verify yourself.

Pay Now

or Go back to <https://www.delpay.in>



Integrating PayU's API for Consent transaction -

The steps for integrating with PayU can technically be described as below:

1. To start off the integration process, you would be provided a test setup by PayU where you would be given a test merchant account and test credit card credentials to have a first-hand experience of the overall transaction flow. Here, you need to make the transaction request on our **test server (and not the production server)**. Once your testing is complete, **then only** you will be ready to move to the PayU production server.
2. To initiate a transaction, the merchant needs to generate a **POST REQUEST** - which must consist of mandatory and optional parameters mentioned in the **later section**. This POST REQUEST needs to be hit on the below mentioned PayU URLs:

For PayU Test Server:

POST URL: https://test.payu.in/_payment

For PayU Production Server:

POST URL: https://secure.payu.in/_payment

3. In the merchant initiated **POST REQUEST**, one of the mandatory parameters is named as **hash**. The details of this hash parameter have been covered in the later section. But it is **absolutely critical** for the merchant to calculate the hash correctly and post to us in the request.
4. When the transaction **POST REQUEST** hits the PayU server, a new transaction entry is created in the PayU Database. To identify each new transaction in the PayU Database, a unique identifier is created every time at PayU's end. This identifier is known as the **PayU ID (or mihpayid)**.



5. With the POST REQUEST, customer would be re-directed to PayU's payment page. Customer now selects the particular payment option on PayU's page (Credit Card/Debit Card/Net Banking /Sodexo etc.) and clicks on 'Pay Now'. PayU re-directs the customer to the chosen payment method. The customer goes through the necessary authorization/authentication process at bank's login page, and the bank gives the success/failure response back to PayU.
6. PayU marks the transaction status on the basis of response received from Bank. PayU provides the final transaction response string to the merchant through a **POST RESPONSE**. The parameters in this response are covered in the subsequent sections.
7. In the POST RESPONSE sent by PayU, you would receive the final status of the transaction. You will receive the **hash** parameter here also. Similar to step 3, it is **absolutely crucial** to verify this hash value at your end and then only accept/reject the invoice order. This is done to strictly avoid any tampering attempt by the user.

DISCLAIMER:

1. Test URL: The Test URL is provided to PayU merchants to test the integration of their server with that of PayU or Bank. It is understood that since this is merely a Test URL, the Merchant should not treat any transactions done on this Test server as live and should not deliver the products/services with respect to any such test transactions even in the case your server receives a successful transaction confirmation from PayU/Bank.

2. Merchants are herein forth requested to set up required control checks on their (merchant) systems/servers to ensure that only those transactions should get routed to the PayU test server which are initiated with sole intention of test the environment.

Standard Request Parameters to be passed for Consent Transactions

Variable	Description
key (Mandatory)	<p>This parameter is the unique Merchant Key provided by PayU for your merchant account. The Merchant Key acts as the unique identifier (primary key) to identify a particular Merchant Account in our database. While posting the data to us, you need to put this Merchant Key value for you merchant account in this parameter.</p> <p>Also, please note that during integration with PayU, you would need to first integrate with our Test Server. PayU would be providing you the necessary Merchant Key for test server. Please do not use your live account's merchant key here. It would not work.</p> <p>Once testing is done, you are ready to move to live server. Here, you would need to replace the test Merchant Key with Live Merchant Key. This is a critical step for successfully moving to live PayU server.</p> <p>Example: C0Ds8q</p>
txnid (Mandatory)	<p>This parameter is known as Transaction ID (or Order ID). It is the order reference number generated at your (Merchant's) end. It is an identifier which you (merchant) would use to track a particular order. If a transaction using a particular transaction ID has already been successful at PayU, the usage of same Transaction ID again would fail. Hence, it is essential that you post us a unique transaction ID for every new transaction.</p> <p>(Please make sure that the transaction ID being sent to us hasn't been successful earlier. In case of this duplication, the customer would get an error of 'duplicate Order ID').</p> <p>Data Type – Varchar Character Limit – 25 characters Example: fd3e847h2</p>
amount (Mandatory)	<p>This parameter should contain the payment amount of the particular transaction.</p> <p>Note: Please type-cast the amount to float type Example: 10.00</p>
productinfo (Mandatory)	<p>This parameter should contain a brief product description. It should be a string describing the product (The description type is entirely your choice).</p> <p>Data type - Varchar Character Limit – 100 characters Example: tshirt100</p>

firstname (Mandatory)	<p>Self-Explanatory (Must contain the first name of the customer)</p> <p>Data Type – Varchar Character Limit – 60 characters Example: Ankit</p>
email (Mandatory)	<p>Self-explanatory (Must contain the email of the customer)</p> <p>Data type – Varchar Character Limit – 50 Example: ankitverma@gmail.com</p> <p>This information is helpful when it comes to issues related to fraud detection and chargebacks. Hence, it is must to provide the correct information.</p> <p>Also, MIS reporting is shared with few issuing banks where email and mobile number is used to keep track of users using SI transactions</p>
phone (Mandatory)	<p>Self-explanatory (Must contain the phone number of the customer)</p> <p>Data type – Varchar Character Limit – 50 (numeric value only) Example:9843176540</p> <p>This information is helpful when it comes to issues related to fraud detection and chargebacks. Hence, it is must to provide the correct information</p> <p>Also, MIS reporting is shared with few issuing banks where email and mobile number is used to keep track of users using SI transactions</p>
surl (Mandatory)	<p>Success URL - This parameter must contain the URL on which PayU will redirect the final response if the transaction is successful. The response handling can then be done by you after redirection to this URL</p>
furl (Mandatory)	<p>Failure URL - This parameter must contain the URL on which PayU will redirect the final response if the transaction is failed. The response handling can then be done by you after redirection to this URL</p>
hash (Checksum) (Mandatory)	<p>Hash is a crucial parameter – used specifically to avoid any tampering during the transaction. There are two different methods to calculate hash. Please</p>

follow method 1 only. Method 2 is just there for the documentation and is not to be used.

Method 1 - This is the simplest way of calculating the hash value. Here, please make sure that the api_version parameter is NOT POSTED from your end.

For hash calculation, you need to generate a string using certain parameters and apply the sha512 algorithm on this string. Please note that you have to use pipe (|) character in between these parameters as mentioned below.

The

parameter order is mentioned below:

```
sha512(key|txnid|amount|productinfo|firstname|email|udf1|udf2|udf3|udf4|udf5|||||SALT)
```

All these parameters (and their descriptions) have already been mentioned earlier in this table. Here, SALT (to be provided by PayU), key, txnid, amount, productinfo, firstname, email are mandatory parameters and hence can't be empty in hash calculation above. But, udf1-udf5 are optional and hence you need to calculate the hash based upon the fact that whether you are posting a particular udf or not. For example, if you are NOT posting udf1. Then, in the hash calculation, udf1 field will be left empty. Following examples will clarify various scenarios of hash calculation:

Case 1: If all the udf parameters (udf1-udf5) are posted by the merchant. Then,

```
hash=sha512(key|txnid|amount|productinfo|firstname|email|udf1|udf2|udf3|udf4|udf5|||||SALT)
```

Case 2: If only some of the udf parameters are posted and others are not. For example, if udf2 and udf4 are posted and udf1, udf3, udf5 are not. Then,

```
hash=sha512(key|txnid|amount|productinfo|firstname|email||udf2||udf4|||||SALT)
```

Case 3: If NONE of the udf parameters (udf1-udf5) are posted. Then,

```
hash=sha512(key|txnid|amount|productinfo|firstname|email|||||||SALT)
```

Example: If key=C0Dr8m, txnid=12345, amount=10, productinfo=Shopping, firstname=Test, email=test@test.com, udf2=abc, udf4=15, SALT=3sf0jURk and

udf1, udf3, udf5 are not posted. Then, hash would be calculated as Case 2 above:

```
sha512(C0Dr8m|12345|10|Shopping|Test|test@test.com||abc||15|||||3sf0jURk)
```


(This value comes out to be
ffcdbf04fa5beefdcc2dd476c18bc410f02b3968e7f4f54e8f43f1e1a310bb32e3
b4dec9305232bb89db5b1d0c009a53bcace6f4bd8ec2f695baf3d43ba730ce)
IMPORTANT: For details related to hash at the time of post back from PayU
to the merchant, please refer to later section. This is also absolutely
mandatory to avoid any tampering.

Method 2- Second method for hash calculation (Don't use this method. It is
only for internal documentation).

Here, parameter api_version should be equal to 2.

hash = sha512(key|txnid|amount|offer_key|api_version|SALT)

For your reference, please find sample code below which shows the basic set of
parameters being posted. Please execute this piece of code in browser to observe
the POST request being re-directed to PayU page and then you can form the
complete transaction request in your code base (with the mandatory and optional
parameters)

```
<html>
```

```
<body>
```

```
<form action='https://test.payu.in/_payment' method='post'>
```

```
<input type="hidden" name="firstname" value="Vikas Kumar" />
```

```
<input type="hidden" name="lastname" value="" />
```

```
<input type="hidden" name="surl" value="https://www.google.com" />
```

```
<input type="hidden" name="phone" value="9999999999" />
```

```
<input type="hidden" name="key" value="C0Dr8m" />
```

```
<input type="hidden" name="hash" value =  
"c2522a8d561e7c52f7d6b2d46c96b924afac8554313af4b80edef3e237e179bd6e2020e8  
c548060306d9fa2cf5c75c35205bcc4b09bcf5b9a9bec8de2952d0" />
```

```
<input type="hidden" name="curl" value="http://www.google.com" />
```

```
<input type="hidden" name="furl" value="https://www.yahoo.in" />
```

```
<input type="hidden" name="txnid" value="PLS-10061-3" />
```

```
<input type="hidden" name="productinfo" value="SAU Admission 2014" />
```



```
<input type="hidden" name="amount" value="600.000" />
<input type="hidden" name="email" value="chota.bheem@gmail.com"
/>
<input type="submit" value="submit"> </form>
</body>
</html>
```

Extra Parameters required especially for Seamless flow in Consent transaction

Apart from above standard parameters, below are the extra mandatory parameters required in the payment request to perform Consent transaction.

Variable	Description
pg (Mandatory)	<p>In case of SI, value will be CC in case of Credit Cards and DC in case of selected Debit cards which support SI</p> <p>Please note that NOT all Debit Cards support SI. Hence while passing this value, merchant needs to confirm what issuers are supporting Debit Card SI with PayU's team and then only pass only applicable Debit Cards for SI.</p>
bankcode (Mandatory)	<p>This parameter would contain the code indicating the payment option used for the transaction. For example, in Debit Card mode, there are different options like Visa Debit Card, Mastercard, Maestro etc. For each option, a unique bankcode exists. It would be returned in this bankcode parameter.</p> <p>For example, Visa Debit Card – VISA, Master Debit Card – MAST</p>

ccnum (Mandatory)	This parameter must contain the card (credit/debit) number entered by the customer for the transaction.
ccname (Mandatory)	This parameter must contain the name on card – as entered by the customer for the transaction.
ccvv (Mandatory)	This parameter must contain the cvv number of the card – as entered by the customer for the transaction.
ccexpmon (Mandatory)	This parameter must contain the card's expiry month - as entered by the customer for the transaction. Please make sure that this is always in 2 digits. For months 1-9, this parameter must be appended with 0 – like 01, 02...09. For months 10-12, this parameter must not be appended – It should be 10, 11 and 12 respectively.
ccexpyr (Mandatory)	The customer must contain the card's expiry year – as entered by the customer for the transaction. It must be of 4 digits. For example - 2017, 2029 etc.
store_card (Mandatory)	Should be passed as always 1
si (Mandatory)	Should be passed as always 1
user_credentials (Mandatory)	<p>Parameter to define the unique customer ID. Its format is <key: unique_idenfier></p> <p>Example – gtKFFx: vikas@test.com, gtKFFx:9999999999</p> <p>PayU will store consumer's card details against this user identifier and return an encrypted card token in the payment response as explained in document further.</p>

Extra Parameters required especially for NON - Seamless flow in Consent transaction



Since in case of non-seamless flow, PayU accepts card details on checkout page, only below parameters are required to be sent in the payment request which are extra from Standard parameters.

si (Mandatory)	Should be passed as always 1
user_credentials (Mandatory)	<p>Parameter to define the unique customer ID. It's format is <key:unique_idenfier></p> <p>Example – gtKFFx:vikas@test.com, gtKFFx:9999999999</p> <p>PayU will store consumer's card details against this user identifier and return an encrypted card token in the payment response as explained in document further.</p>

Here is how PayU's page gets rendered when merchant passes above extra parameters for SI transactions –



PayU**biz**

Amount: Rs. 1000.00

Transaction ID: **f7cc03802f5489e35b4f**

Choose a payment method

PayU India is now PayU**biz**

Credit Card

Debit Card

Card Type

VISA

MasterCard

Card Number

4111 1111 1111 1111

Name on Card

Swaroop Kularni

CVV Number

...


What is CVV number?

Expiry Date

Feb (2)

2019

☒ I agree to save card details for Standing Instruction.

 [Learn more](#)

Citi Bank Card

Type a custom label to save this card (optional)

Note: In the next step you will be redirected to your bank's website to verify yourself.

Pay Now

or Go back to <https://www.delpay.in>

Important Things to remember: Characters allowed for parameters

- For parameters address1, address2, city, state, country, product info, email, and phone following characters are allowed:
- Characters: A to Z, a to z, 0 to 9
- -(Minus)
- _ (Underscore)
- @ (At the Rate)
- / (Slash)
- (Space)
- . (Dot)

If the merchant sends any other special characters then they will be automatically removed. The address parameter will consider only first 100 characters.

Formula for hash (checksum) before transaction

This has already been covered in the description of **hash** in the table containing the POST Parameters above.

Formula for hash (checksum) after transaction

This time the variables are in reverse order and status variable is added between salt and udf1.

sha512(SALT|status| || || |udf5|udf4|udf3|udf2|udf1|email|firstname|productinfo|amount|txnid|key)

It is absolutely mandatory that the hash (or checksum) is computed again after you receive response from PayU and compare it with post back parameters below. This will protect you from any tampering by the user and help in ensuring safe and secure transaction experience.

Hash (Checksum) Algorithm Example codes

The Checksum algorithm used is SHA512 which is globally well-known algorithm. To need help with implementation, feel free to call us, mail us or use Google to find the desired function library for your implementation. Some example codes are also mentioned below:

For PHP

Example code:

```
$output = hash ("sha512", $text);
```

For .NET

Link: <http://msdn.microsoft.com/en-us/library/system.security.cryptography.sha512.aspx>

Example code:

```
byte[] data = new byte[DATA_SIZE];
byte[] result;
SHA512 shaM = new SHA512Managed();

result = shaM.ComputeHash(data);
```

For JSP

Example code:

```
import java.io.FileInputStream;
import java.security.MessageDigest;

public class SHACheckSumExample {

    public static void main(String[] args) throws Exception {

        MessageDigest md = MessageDigest.getInstance("SHA512");

        FileInputStream fis = new FileInputStream("c:\\\\logging.log");

        byte[] dataBytes = new byte[1024];

        int nread = 0;

        while ((nread = fis.read(dataBytes)) != -1) {

            md.update(dataBytes, 0, nread);

        };

        byte[] mdbytes = md.digest();

        //convert the byte to hex format method
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < mdbytes.length; i++)

        {
            sb.append(Integer.toString((mdbytes[i] & 0xff) +
            0x100, 16).substring(1));
        }
    }
}
```



```
System.out.println("Hex format : " +  
sb.toString()); //convert the byte to hex format  
method 2 StringBuffer hexString = new  
StringBuffer();  
for (int i=0;i<mdbytes.length;i++)  
hexString.append(Integer.toHexString(0xFF &  
mdbytes[i]));  
  
}  
  
System.out.println("Hex format : " + hexString.toString());  
  
}
```

Response Parameters posted by PayU to Merchant

Variable Name	Description						
mihpayid	<p>It is a unique reference number created for each transaction at PayU's end. For every new transaction request that hits PayU's server (coming from any of our merchants), a unique reference ID is created and it is known as mihpayid (or PayU ID)</p> <p>In case of recurring transaction, this value will be further used as a reference point always against given card holder. So, it is imperative to map this value with user profile at merchant end so that same will be used in the recurring transaction platform.</p>						
mode	<p>This parameter describes the payment category by which the transaction was completed/attempted by the customer. The values are mentioned below:</p> <table border="1"> <thead> <tr> <th>Category used by Customer</th><th>Value of Mode Parameter</th></tr> </thead> <tbody> <tr> <td>Credit Card</td><td>CC</td></tr> <tr> <td>Debit Card</td><td>DC</td></tr> </tbody> </table>	Category used by Customer	Value of Mode Parameter	Credit Card	CC	Debit Card	DC
Category used by Customer	Value of Mode Parameter						
Credit Card	CC						
Debit Card	DC						
status	<p>This parameter gives the status of the transaction. Hence, the value of this parameter depends on whether the transaction was successful or not. You must map the order status using this parameter only. The values are as below:</p> <p>If the transaction is successful, the value of 'status' parameter would be 'success'.</p> <p>The value of 'status' as 'failure' or 'pending' must be treated as a failed transaction only.</p> <p>For charging card over recurring platform, the status of Consent transaction should be returned as success. Then only merchant</p>						

	should consider that recurring platform can be used for given card holder going forward.
key	This parameter would contain the merchant key for the merchant's account at PayU. It would be the same as the key used while the transaction request is being posted from merchant's end to PayU.
txnid	This parameter would contain the transaction ID value posted by the merchant during the transaction request.
amount	This parameter would contain the original amount which was sent in the transaction request by the merchant.
productinfo	This parameter would contain the same value of productinfo which was sent in the transaction request from merchant's end to PayU
email	This parameter would contain the same value of email which was sent in the transaction request from merchant's end to PayU
phone	This parameter would contain the same value of phone which was sent in the transaction request from merchant's end to PayU
hash	<p>This parameter is absolutely crucial and is similar to the hash parameter used in the transaction request send by the merchant to PayU. PayU calculates the hash using a string of other parameters and returns to the merchant. The merchant must verify the hash and then only mark a transaction as success/failure. This is to make sure that the transaction hasn't been tampered with. The calculation is as below:</p> <p>sha512(SALT status udf5 udf4 udf3 udf2 udf1 email firstname productinfo amount txnid key)</p> <p>The handling of udf1 – udf5 parameters remains similar to the hash calculation when the merchant sends it in the transaction request to PayU. If any of the udf (udf1-udf5) was posted in the transaction request, it must be taken in hash calculation also.</p> <p>If none of the udf parameters were posted in the transaction request, they should be left empty in the hash calculation too.</p>

error	For the failed transactions, this parameter provides the reason of failure. Please note that the reason of failure depends upon the error codes provided by different banks and hence the detailing of error reason may differ from one transaction to another. The merchant can use this parameter to retrieve the reason of failure for a particular transaction.
bankcode	This parameter would contain the code indicating the payment option used for the transaction. For example, in Debit Card mode, there are different options like Visa Debit Card, Mastercard, Maestro etc. For each option, a unique bankcode exists. It would be returned in this bankcode parameter. For example, Visa Debit Card – VISA , Master Debit Card – MAST .
PG_TYPE	This parameter gives information on the payment gateway used for the transaction. For example, if SBI PG was used, it would contain the value SBIPG . If SBI Netbanking was used for the transaction, the value of PG_TYPE would be SBINB . Similarly, it would have a unique value for all different type of payment gateways.
bank_ref_num	For each successful transaction – this parameter would contain the bank reference number generated by the bank.
unmappedstatus	This parameter contains the status of a transaction as per the internal database of PayU. PayU's system has several intermediate status which are used for tracking various activities internal to the system. Hence, this status contains intermediate states of a transaction also -

	<p>and hence is known as unmappedstatus.</p> <p>For example: dropped/bounced/captured/auth/failed/usercancelled/pending</p>
cardToken	<p>This is the secure card token generated by PayU against user_credentials parameter sent by merchant and returned back in the payment response. This value will be used by merchant going forward for charging consumer over recurring payments so it is imperative for merchant to map this against right consumer profile.</p> <p>Sample value - f59f6vc6f2e7d18c3d476</p>

payment_source	<p>This value can be used to distinguish normal ecom transaction and SI consent transaction.</p> <p>Value returned will be always sist against this parmaeter for SI consent transaction</p>
-----------------------	---

Please note that this transaction **must be successful** in order to make it eligible for Recurring platform. Also, merchant must verify the below 4 values received in the response from PayU before setting up recurring plan / subscription for given merchant against given card details -

- **status = success** (Indicates that transaction is successful)
- **card_token = <Must be a non-empty string value>** (Indicates that card details are saved correctly in PayU Database)
- **payment_source = sist** (Indicates that card details have been marked correctly for Standing Instruction)
- **mihpayid = <Must be non-empty value>** (Indicates PayU's transaction acknowledgement for Consent transaction)

If any of the above 4 checks are not satisfied, that means the transaction hasn't been correctly authorized for Standing Instruction. The merchant must not consider this transaction eligible for Recurring platform.

It is important for merchant to map below parameters against user identity so that merchant can use recurring platform

- **user_credentials**
- **card_token**
- **mihpayid**

Integrating PayU's API for Recurring transaction

After consent transaction is completed successfully, merchant can start charging consumer over recurring basis through either of the below approaches -

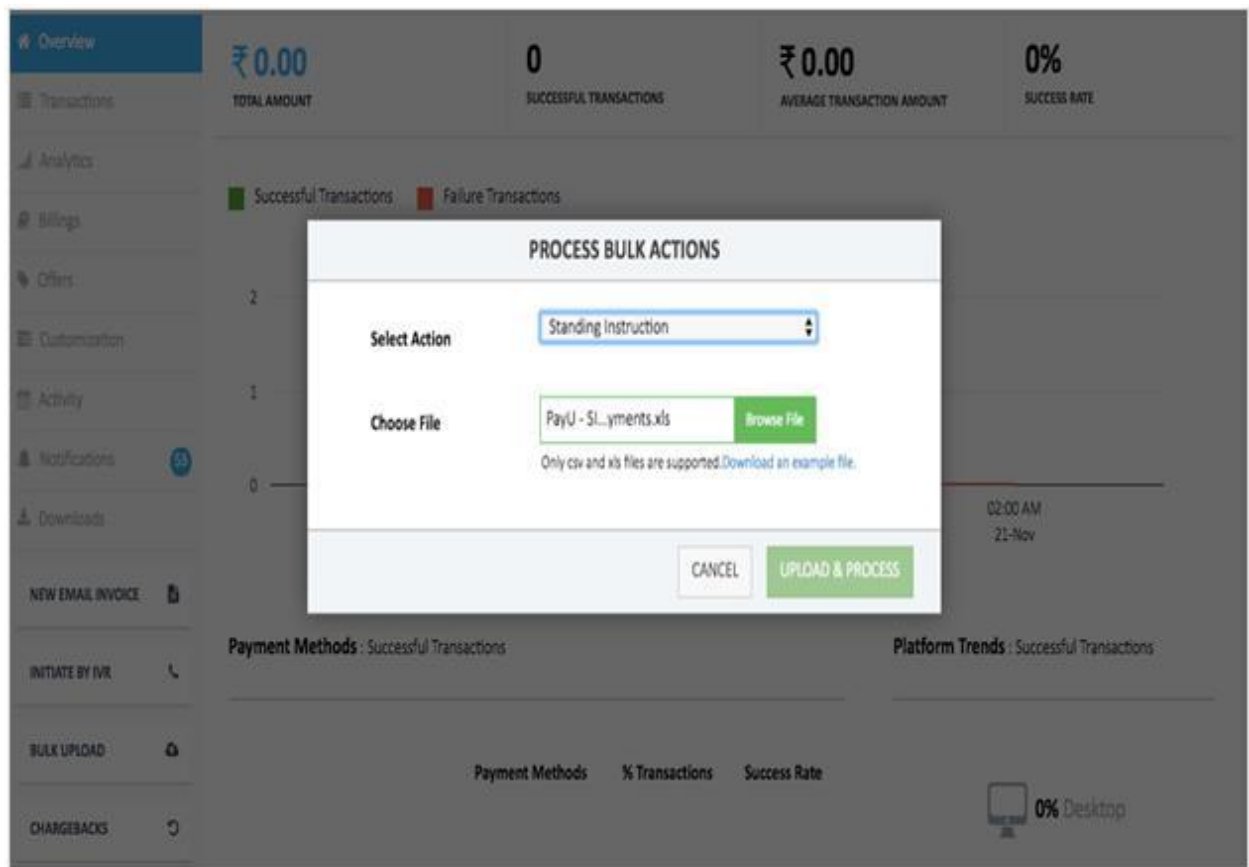
- **File Upload Utility from PayU panel**
- **PayU's Recurring API Integration**

Please note that there is a time period of 6 hours required to call Recurring transaction after successful Consent transaction is processed.

Using File Upload Utility -

File upload utility helps merchants who wish to charge consumer for recurring payments over simple file upload rather than integrating API.

In this case, the recurring payment details like transaction amount, transaction id and card token is filled in the csv file and same is uploaded through PayU panel as shown in the image below -



Below columns are mandatory for file upload utility –

Variable Name	Description
authpayuid (mandatory)	It is a unique reference number created for each transaction at PayU's end. For every new transaction request that hits PayU's server (coming from any of our merchants), a unique reference ID is created and it is known as mihpayid (or PayU ID)
amount (mandatory)	This parameter should contain the payment amount of the particular transaction. Note: Please type-cast the amount to float type Example: 10.00
txnid (mandatory)	This parameter is known as Transaction ID (or Order ID). It is the order reference number generated at your (Merchant's) end. It is an identifier which you (merchant) would use to track a particular order. If a transaction using a particular transaction ID has already been successful at PayU, the usage of same Transaction ID again would fail. Hence, it is essential that you post us a unique transaction ID for every new transaction.

	(Please make sure that the transaction ID being sent to us hasn't been successful earlier. In case of this duplication, the customer would get an error of 'duplicate Order ID')
user_credentials (optional)	<p>Parameter defined while performing the unique customer ID. It's format is <key:unique_idenfifer> Example – gtKFFx:vikas@test.com, gtKFFx:9999999999</p> <p>Please make sure that this is the same user_credentials sent by merchant during Consent transaction when consumer's card was taken first time and card token was generated post successful authentication.</p>
card_token (optional)	<p>Encrypted card token returned by PayU post successful authentication of Consent transaction</p> <p>Sample value - f59f6vc6f2e7d18c3d476</p>
email (Mandatory)	Email Id of the consumer.
phone (Mandatory)	10-digit Mobile number of the consumer

Offering Recurring Platform Using Recurring APIs –

With PayU's Server to Server API, merchant can create seamless and automated payment experience for consumer for recurring payments. Below section will detail out necessary steps to achieve the same.

Base URLs

Test Server:

- Returns XML response -
<https://test.payu.in/merchant/postservice.php?form=1>
- Returns JSON response -
<https://test.payu.in/merchant/postservice.php?form=2>

Production Server:

- Returns XML response -
<https://info.payu.in/merchant/postservice.php?form=1>
- Returns JSON response -
<https://info.payu.in/merchant/postservice.php?form=2>

Request Parameters for Recurring Payment API

Method	POST
Path	https://{Base_URL}/merchant/postservice.php?form=2
Content Type	application/x-www-form-urlencoded

Variable	Description
key (Mandatory)	This parameter is the unique Merchant Key provided by PayU for your merchant account. The Merchant Key acts as the unique identifier (primary key) to identify a particular Merchant Account in our database.

	Sample value – YbfVda		
command	Value of the parameter will be always passed as - si_transaction		
hash (Mandatory)	<p>512 SHA hash strings generated by encrypting request parameters so that any tampering can be avoided.</p> <p>Sample Value -</p> <pre>a6105d0e5c8726fd37713d02ace8f63652bfe1337f14e4a3652084ba9787e4ee601a503bcc42ba101fb1c378b910d5637e9806b92fc89322db0180697ff906b3</pre> <p>hash = sha512(key command var1 SALT)</p>		
var1 (Mandatory)	<p>var1 is constructed by combination of some mandatory and non-mandatory fields under JSON object.</p> <p>Please note that sequence of this parameter needs to be maintained as it is given in the below JSON object. i.e. authpayuid, amount, txnid, phone and email</p> <p>Sample Value -</p> <pre>{"authpayuid": "6611192557", "amount": 3, "txnid": "REC15113506209", "phone": "9999999999", "email": "chota.bheem@gmail.com"}</pre> <p>Let's look at each of every field in detail.</p> <table border="1"> <tr> <td>authpayuid (Mandatory)</td><td> <p>The value of mihpayid returned in the payment response of Consent transaction when transaction is successfully completed and card is stored securely vault.</p> <p>As explained earlier in the document, merchant needs to map this value against consumer profile at his end</p> </td></tr> </table>	authpayuid (Mandatory)	<p>The value of mihpayid returned in the payment response of Consent transaction when transaction is successfully completed and card is stored securely vault.</p> <p>As explained earlier in the document, merchant needs to map this value against consumer profile at his end</p>
authpayuid (Mandatory)	<p>The value of mihpayid returned in the payment response of Consent transaction when transaction is successfully completed and card is stored securely vault.</p> <p>As explained earlier in the document, merchant needs to map this value against consumer profile at his end</p>		

		<p>so that correct authpayuid will be passed in the request.</p> <p>It is important for merchant to send this value against correct consumer because PayU will pull out card details against this id from dB and put charge on the card. By passing wrong or different value against this field will result into charging different consumer's card</p>	
	amount (Mandatory)	<p>Recurring amount which needs to be charged for given consumer. Example 2.25</p> <p>Please note that this is the same amount for which consumer has given Consent for charging his card recurring.</p>	
	txnid (Mandatory)	<p>This parameter is known as Transaction ID (or Order ID). It is the order reference number generated at your (Merchant's) end. It is an identifier which you (merchant) would use to track a particular order. If a transaction using a particular transaction ID has already been successful at PayU, the usage of same Transaction ID again would fail. Hence, it is essential that you post us a unique transaction ID for every new transaction.</p> <p>(Please make sure that the transaction ID being sent to us hasn't been successful earlier. In case of this duplication, the customer would get an error of 'duplicate Order ID').</p>	
	email (Mandatory)	Email Id of the consumer.	
	phone (Mandatory)	10-digit mobile number of the consumer.	



Calculating hash value for Recurring API -

Hash value is integral part of PayU's API since it provides highest security standards as well as avoid any kind of tampering in the payment request.

For recurring API, hash is calculated by concatenating key, command, var1 and salt and generating 512 HASH for the resultant string.

hash = sha512(<key> + "|" + <command> + "|" + <var1> + "|" + <salt>)

For example,

- Merchant key - YBfVda
- Merchant salt - 2b1b1
- Generated var1 -

```
{"authpayuid": "6611192557", "amount": 3, "txnid": "REC15113506209", "phone": "9999999999", "email": "chota.bheem@gmail.com"}
```

Please note the sequence of the var1 is maintained as per the guidelines given above.

Then hash value will be calculated as below,

```
Hash = sha512 {YBfVda|si_transaction|{"authpayuid": "6611192557", "amount": 3, "txnid": "REC15113506209", "phone": "9999999999", "email": "chota.bheem@gmail.com"}|2b1b1}
```

Hash =

```
bf4a0c610737c9c0acf45f40c1d914b1b4d45c447f877ed23cff53b5fbb4f9cbc3a8793c5bef2485c46cd9d03d143a0060fc2832bcc5802c1ff5260b35a68da0
```

Response Parameters returned back in Recurring Payment API

Here is the sample response returned in the payment API when transaction is successfully charged through recurring API.

```
{
  "status": 1,
  "message": "Transaction Processed successfully",
  "details": {
    "REC15113506209": {
```



```
"transactionid": "REC15113506209",  
"amount": "3",  
"payuid": "6611427463",  
"status": "captured",  
"field9": "Transaction Completed Successfully",  
"phone": "9999999999",  
"email": "chota.bheem@gmail.com"  
}  
}  
}
```

Variable Name	Description
status	Status defines acknowledgement from PayU. When it is returned as 1 it means that the request posted by merchant is valid.
message	Message defines acknowledgement from PayU and returned as Transaction Processed Successfully when valid request is posted by merchant
transactionid	Value of txnid parameter which is echoed back in the response. This is unique transaction id generated by merchant during calling recurring API
amount	Requested transaction amount is echoed back in the payment response
payuid	PayU's transaction id for processed recurring transaction. Merchant can use this field for reference point in the settlement report
status	This parameter gives the status of the transaction. Hence, the value of this parameter depends on whether the transaction was successful or

	<p>not. You must map the order status using this parameter only. The values are as below:</p> <p>If the transaction is successful, the value of 'status' parameter would be 'captured'.</p> <p>The value of 'status' as 'failed' or 'pending' or blank must be treated as a failed transaction only.</p>
field9	Description of the transaction response status
phone	Mobile number of the consumer echoed back in the payment response
email	Email address of the consumer echoed back in the payment response

Few Sample Error response scenarios are mentioned here –

- Invalid signature generation will throw below error. This can happen if key or salt or var1 is having incorrect values

```
{
  "status": 0,
  "msg": "Invalid Hash."
}
```

- If incorrect authpayuid is passed for the transaction or authpayuid passed doesn't represent successful consent transaction then below error is thrown

```
{
  "status": 1,
  "message": "Transaction Processed successfully",
  "details": {
    "REC9812123123": {
      "authpayuid": "6611192559",
      "transactionid": "REC9812123123",
      "amount": "1",
      "user_credentials": " ",
      "card_token": " ",
      "payuid": "",
      "status": "failed",
      "field9": "Basic authentication check failed",
      "phone": "",
      "email": ""
    }
  }
}
```

```
}  
}  
}
```

- If any of the mandatory parameter is missing then below error is thrown by the system

```
{  
    "status": 0,  
    "msg": "Invalid characters or empty data in one or more input parameters"  
}
```

Supporting Query and Refund APIs -

For Query and Refund transaction, please refer SECTION II: WEB SERVICES – APIs from PayU's standard integration guide since there is no change in the request and response format for these functions either for Consent transaction or for Recurring transaction.