Store Card Feature

Section I: Brief Description

Store card feature is also known as 1-Click checkout/e-vault feature provided by PayU. The basic concept is that if a customer attempts a transaction via PayU, the card details can be saved in the PayU Database (Based upon the merchant's action). The next time, the same customer can use any of the previously saved cards in PayU Database and complete the transaction without having to enter the card number, expiry date and name on card values again. The customer would only need to select the saved card and enter the CVV number to complete the transaction. Hence, this feature saves customer's transaction time.

Section II: Stored Card Feature for Non-Seamless Mode

To implement Store Card feature in non-seamless mode, please follow the below steps:

1) Along with all the parameters which merchant is sending to PayU (like amount, txnid, productinfo etc) in the transaction POST Request, they need to post one extra parameter – **user_credentials**

2) In this parameter, the merchant needs to populate the value in below format:

   <merchant_key:unique_customer_identifier>

   - Here **merchant_key** would be provided to the merchant by PayU. It is the same key which is used in the transaction request also.

   - **unique_customer_identifier** needs to be a unique value for identifying the customer. It may be an email ID, phone number or a unique numeric/string value.

So, for example, if the **merchant_key** is JBiPgT, then examples of **user_credentials** are as below:

a) **JBiPgT:test@example.com** - [Email ID used here]

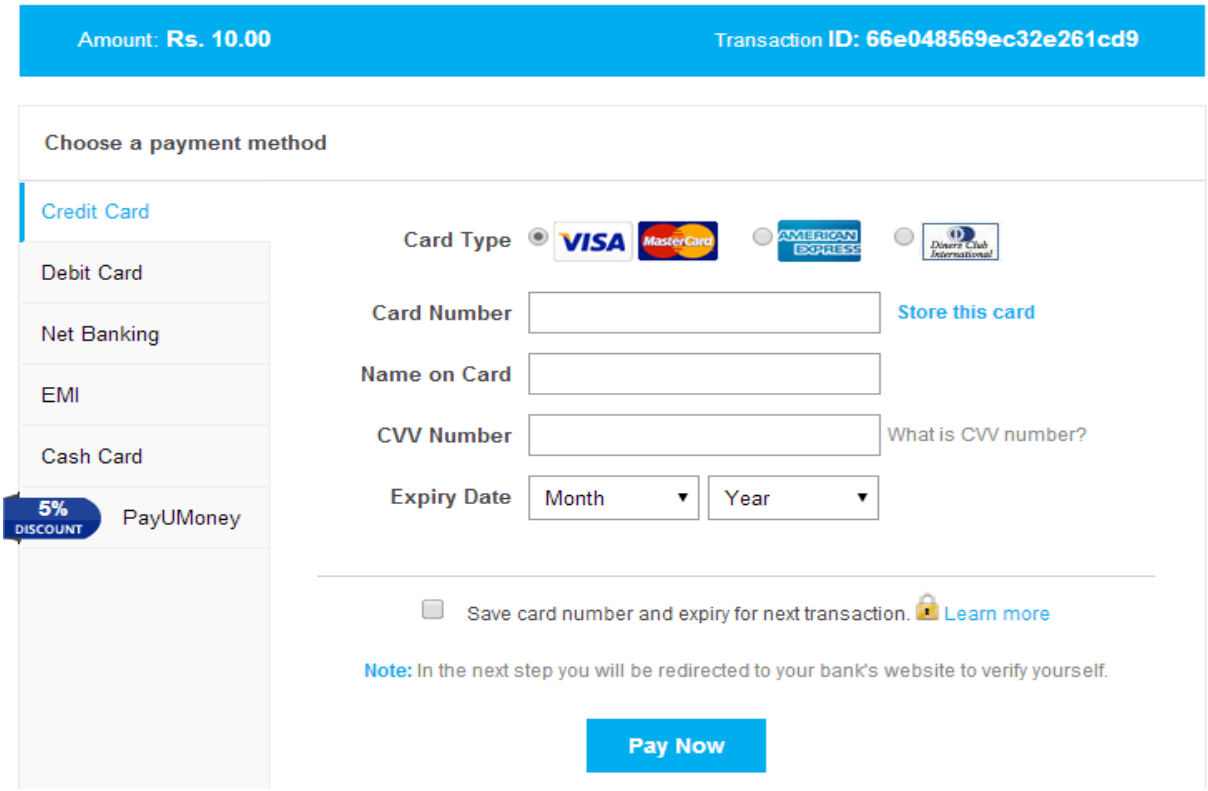
b) **JBiPgT:9999999999** - [Phone number used]

c) **JBiPgT:123456** - [Numeric Value]

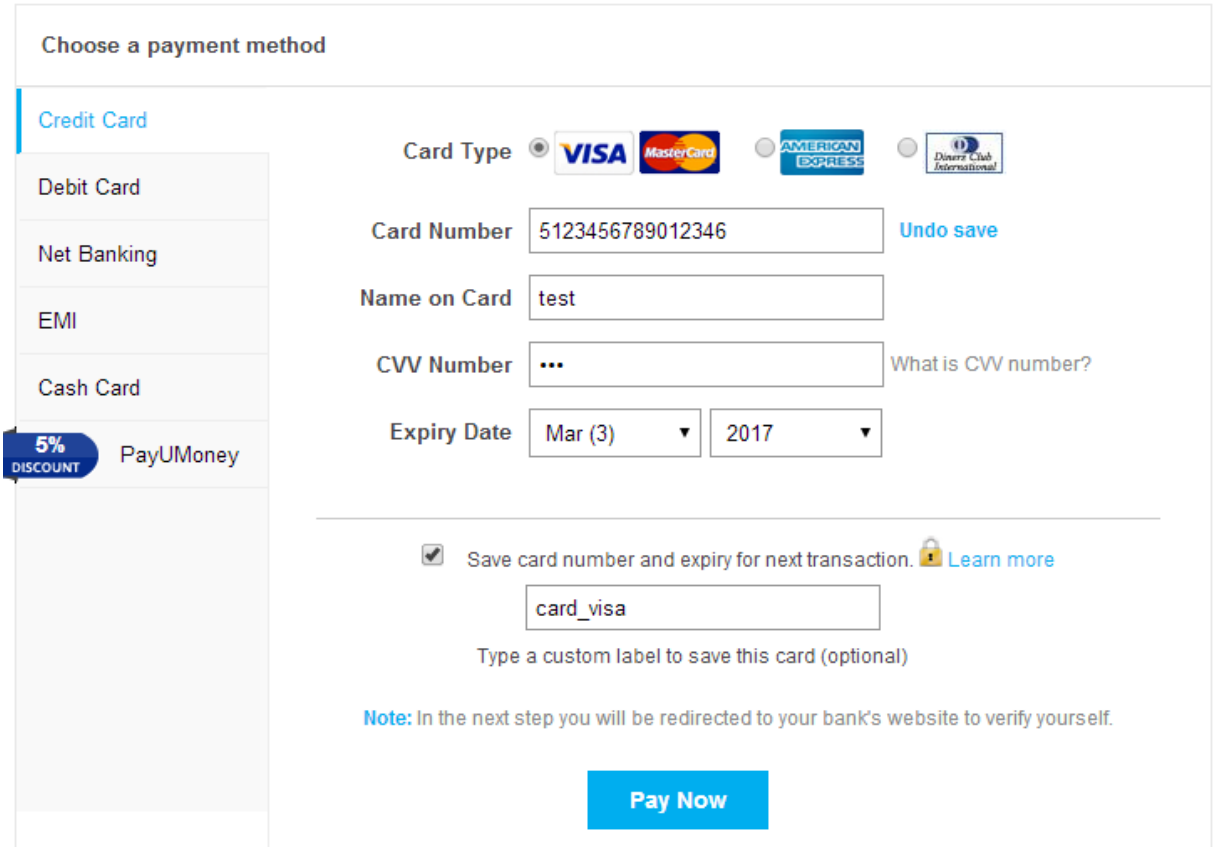
d) **JBiPgT:ertyui12fgh5jasas** - [String used]

Once, user_credentials value is sent from merchant's end to PayU, the payment page would be shown to the customer having the option to save his card. The screenshot is shown below. You can see the checkbox for **'Save the card number and expiry for next transaction'** option, which the customer needs to select after filling up the card data.

As an alternative, the customer can also select the **'Store this card'** option on the right side of card number entry to save the card.

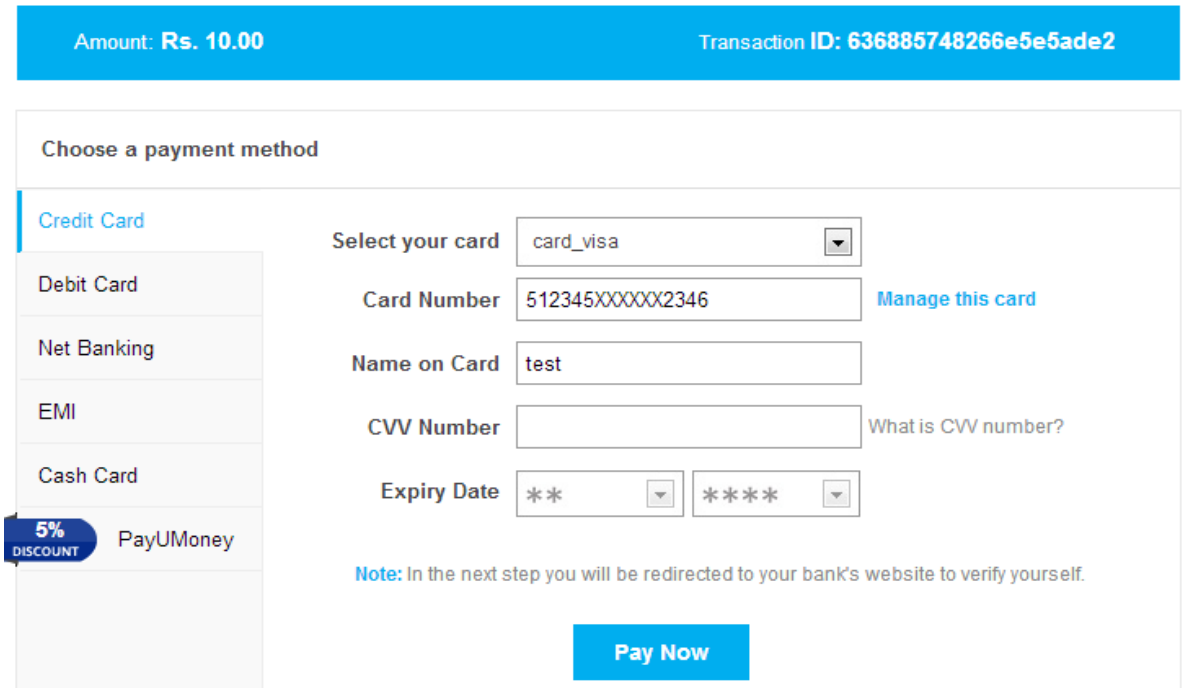


Now the customer needs to enter the card details and click on the save card checkbox. The customer also gets an option to store a custom label (nickname) for the stored card for future reference.



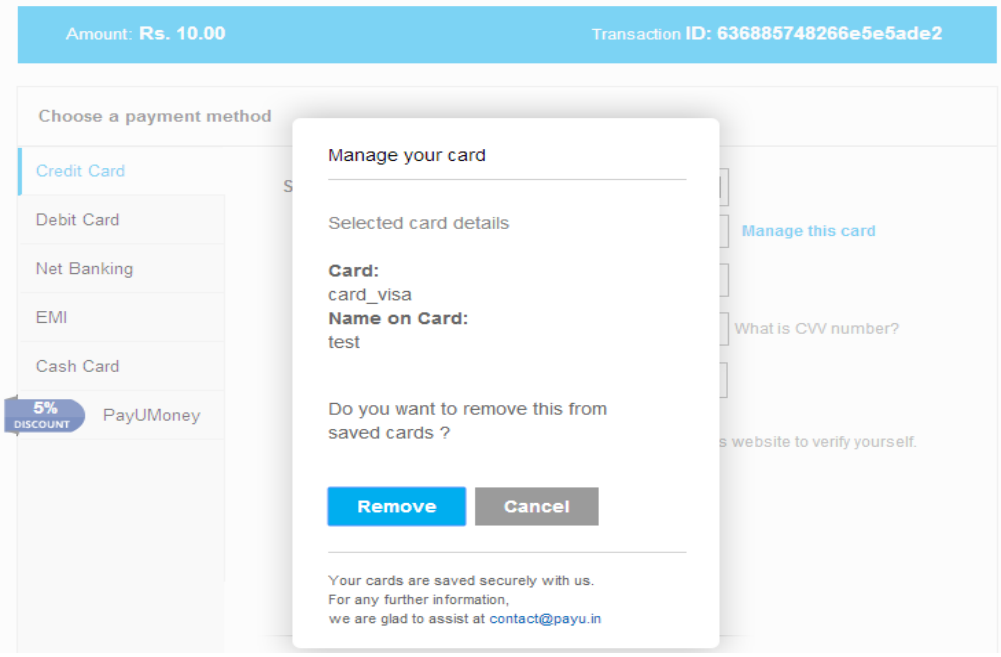
Now once the customer clicks the **'Pay Now'** option, the card details would be saved in PayU Database and the customer would be re-directed to the bank's login page to complete the transaction.

The next time if the same customer visits the merchant website and if the merchant wants the customer to use the stored card, then the same **user_credentials** parameter value must be posted to PayU in the transaction request. The customer would be shown all the previously stored cards as given in the below screenshot:



Here, the customer simply needs to enter the CVV and go ahead with the transaction**. If the customer wants to use a fresh card**, then he needs to click on the dropdown provided in **'Select your card'**.

If the customer wants to delete a card, he need to click on **'Manage this Card'** option on the right hand side. Below screen will appear and the customer can remove the stored card from Database:



Section III: Stored Card Feature for Seamless Mode:

In the seamless mode, the customer enters the card data on the merchant web-page itself and not on PayU page. Hence, the stored card feature has to be implemented slightly differently here. Here, in addition to the normal parameters (like txnid, amount, productinfo etc), below parameters must be posted to PayU in the transaction request

1) Card information related parameters (Card number, Name on card, CVV, Expiry Month, Expiry Year). Please refer to the integration document for the exact parameter names.
2) **user_credentials** in the exact same form as above (non-seamless) –

   <merchant_key:unique_merchant_identifier>

3) A parameter named **store_card** and its value must be 1. This is mandatory here.
4) A parameter named **card_name** for setting the nickname for card – if the customer wants to name it for future reference. It is non-mandatory and hence need not be posted.

Once PayU receives the transaction request from the merchant, the card number would be stored in PayU Database (except the CVV) and the customer would be routed to the desired bank for completing the transaction.

Customer Visiting next time

When the customer visits for the next time and wants to use the already stored cards, then in addition to normal parameters (like txnid, amount, productinfo) the merchant needs to post the below parameters:

1) store_card_token – This parameter must have the **cardToken** for the card. Details for it are mentioned below.
2) ccvv – This parameter must have the CVV Number for the card.

The transaction would now pick up the saved card credentials (like Card number, Expiry etc) and the customer would be re-directed to the bank's login for completing the transaction.

cardToken: Please note that for each new card stored, a unique value is generated in PayU Database to identify the particular card. It is named as **cardToken.** As a single customer may have more than one cards saved in PayU Database, a unique cardToken is used to identify each different card. Hence, there may be more than 1 **cardToken** values for 1 **user_credentials.**

cardToken can be stored at merchant's end using a couple of different ways. In the response from PayU to the merchant, the **cardToken** is sent back in the response parameters. You can either save the **cardToken** from this POST response in transaction real time. Or you can also call our API (server to server call) for retrieving the cardToken. The API for cardToken retrieval is **get_user_cards.** Here, you need to provide the **user_credentials** for the particular customer - as the input and all the cardTokens would be returned as the output.

Sample Output for API:

```
array
(
   [status] => 1
   [msg] => Cards fetched Succesfully
   [user_cards] => Array
          (
         [745d72e2fd9b7e88824fef4e7ed7dac1fe624b74] => Array
            (
               [name_on_card] => {name}
               [card_name] => nickname but if sent empty then (cardType****last 4 digits of card) e.g.
mastercard****2346
               [card_type] => CC(ibibo_code)
               [card_token] =>745d72e2fd9b7e88824fef4e7ed7dac1fe624b74
               [is_expired] => 1(1 when card is expired , 0 when not)
               [card_mode] => CC(card Category)
               [card_no] => 412345xxxxxx2356(masked Card Number)
               [card_brand] => VISA
               [card_bin] => 412345
            )
      )
)
```

Since this API gives the masked card number also, the merchant can use this API to retrieve all the masked card numbers and display them to the customer at the user interface. This way, the customer can select the specific stored card using which he wants to do the transaction.

Important Note: For both seamless and non-seamless mode, maintaining a database of user_credentials is absolutely crucial for the merchant. Rest of the details like (cardToken and masked card number can be retrieved using the API.