PAYMENT GATEWAY

APIs for integration

Contact

Email: info@zenpay.biz
Website: https://zenpay.biz

Document version 1.8.0

Contents

1. OVERVIEW

This document describes the steps for technical integration process between merchant website / application and ZenPay.

Through ZenPay, your customers can make electronic payments through various payment modes such as:

- Credit cards
- Debit cards
- Net banking
- EMI
- Cash Cards/Wallets
- Mobile/web invoicing
- Integrated NEFT/RTGS
- Bank deposits
- Standing instruction on cards
- Customer account direct debit (e-NACH)
- UPI
- BharatQR

ZenPay also offers you a business UI (https://mrm.zenpay.biz) where you have access to all your prior transaction/payment details, settlement details, analytics, etc.

You can also use this UI to create invoices singly or in bulk, set reminders, recurring billing, and many more features, manage your payables, vendor payments, set split ratios for vendor payments, process refunds, etc. This online interface can be accessed through https://mrm.zenpay.biz

2. PAYMENT REQUEST API

When you integrate with ZenPay, the customer will be re -directed from your merchant website to the ZenPay payment page. After completion of the transaction, ZenPay will direct the customer back to the merchant website

2.1. Steps for Integration

- You need to submit a **POST REQUEST** to our server, at the below mentioned URL https://pay.zenpay.biz/v2/paymentrequest

  Note: hash is a mandatory parameter. If your hash is not properly calculated or does not match for whatever reason, we will not be able to process the payment. The usage of hash is explained in subsequent sections.

- When you call this API, the customer is necessarily re-directed to ZenPay's payment page. After the customer makes the payment through ZenPay (entering his card details or netbanking details etc.), we direct the customer back to your merchant site.

  Note: If you need the customer to enter credit card details on your (merchant) website and would NOT want us to redirect to the ZenPay page, we can get that done, provided you are PCI-DSS certified. If you are not certified and would like to get certified, let us know. We will guide you appropriately on how to get it done.

- We recommend that you check the hash at your end again, after we send back the response to you. This is essential to prevent user data tampering fraud.

- Transaction ID and order ID:
  - When you submit your transaction request to ZenPay, you need to submit an order ID as part of the request. This order ID can be used by you as a universal reference number for all transaction requests submitted by you.
  - When your customer clicks the "Pay" button on the payment page, a unique transaction ID is assigned to the transaction.
  - Order ID acts as a "merchant reference number". You must maintain uniqueness of your order IDs.

2.2. Parameters to be POSTed in Payment Request

URL: https://pay.zenpay.biz/v2/paymentrequest

Parameter Name	Description	Data type	Optional / Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your	varchar(36)	mandatory

	business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.		
order_id	This is your (merchant) reference number. It must be unique for every transaction. We do perform a validation at our end and do not allow duplicate order_ids for the same merchant.	varchar(30)	mandatory
mode	This is the payment mode (TEST or LIVE are valid values)	varchar(4)	optional
amount	This is the payment amount.	decimal(12,2)	mandatory
currency	This is the 3-digit currency code (INR)	varchar(3)	mandatory
description	Brief description of product or service that the customer is being charged for.	varchar(255)	mandatory
name	Name of customer.	varchar(255)	mandatory
email	Customer email address.	varchar(255)	mandatory
phone	Customer phone number	varchar(30)	mandatory
address_line_1	Customer address	varchar(255)	optional
address_line_2	Customer address 2	varchar(255)	optional
city	Customer city	varchar(255)	mandatory
state	Customer State	varchar(255)	optional
country	Customer country	varchar(100)	mandatory
zip_code	Customer zip code	varchar(20)	mandatory
timeout_duration	Timeout duration (in seconds)	varchar(10)	optional
udf1	User defined field	varchar(255)	optional
udf2	User defined field 2	varchar(255)	optional
udf3	User defined field 3	varchar(255)	optional
udf4	User defined field 4	varchar(255)	optional
udf5	User defined field 5	varchar(255)	optional
return_url	Return URL success - ZenPay will make a POST request to this URL after successful transaction, with a set of parameters, which you can process as you want to.	varchar(255)	mandatory
return_url_failure	Return URL failure - ZenPay will make a POST request to this URL after a FAILED transaction, with a set of parameters, which you can process as you want to.	varchar(255)	optional
return_url_cancel	Return URL success - ZenPay will make a POST request to this URL in case of transaction cancellation, with a set of parameters, which you can process as you want to.	varchar(255)	optional
percent_tdr_by_user	Percent of tdr amount paid by user (optional) (max value:100)	decimal(5,2)	optional
flatfee_tdr_by_user	fixed fee paid by user (optional)	decimal(10,2)	optional
show_convenience_fee	Controls whether the convenience fee amount (for surcharge merchants) is displayed to the customer (on the payment page) or not	varchar(1)	optional

split_enforce_strict	Controls whether payment is required to be split before settlement. By default, it is set to 'n', If this is set to 'y' then settlement will be on HOLD until splitsettlement api is called to provide split information.	varchar(1)	optional
split_info	Split info is for splitting the payment between vendor and themselves. In this field one must provide vendor code and what percentage of the payment to be split. (Note: Currently this accepts single vendor split amount percentage only) Following is an example how it will look *{"**vendors**":**[**{"**vendor_code**":"2VEN449","**split_amount_percentage**":"20"}**]**}* All field in this JSON are mandatory.	varchar(500)	optional
payment_options	payment options to be displayed such credit card (cc), net banking (nb), wallet (w), ATM card (atm) and debit card with pin (dp). Tabs will be displayed by order in which values are sent. Values accepted are: *cc,nb,w,atm,upi,dp* (comma separated string), sequence of values will also determine the tab sequence on payment page.	varchar(50)	optional
payment_page_display_text	This text will be displayed below the logo on payment page.	varchar(100)	optional
allowed_bank_codes	Bank codes sent in this filed will be allowed in payment page, other bank codes will not be allowed to proceed with payment. Refer appendix 3 for the list of bank codes. To send multiple bank codes send a comma separated list. E.g. to allow only credit cards: *MACC,VICC,DINC,VISC,RUPC,MASC, AMXC*	varchar(250)	optional
allowed_emi_tenure	This will be a comma separated integer list depending upon the tenure (in months) of loan allowed to show in EMI payment method. Ex(3 , 6, 9 etc.)	varchar(50)	optional
allowed_bins	BIN is Bank Identification Number, on a card it is first 6 digits. BINs passed here will only be allowed to transact, multiple BINs can be sent as comma separated list. Refer appendix 3 for the list if the payment mode is card	varchar(250)	optional
offer_code	If there is any discount / offer provided by merchant on EMIs, then predefined codes must be mentioned in this field. (This is for specific use case; more information can be provided on demand)	varchar(100)	optional
emi_info	This is an optional param which is to be posted to issuer end in case emi detail are required.the format will be as mentioned below "emi_info": { "subvention": "0.0",	varchar(100)	optional

	"aggregator_name": "BENOW", "bank_merchant_id": null, "bank_term_id": null, "bank_sku_code": null } all the fields are optional and have varchar as data type.		
product_details	Contains information regarding the goods/product for which the payment (emi) is being made. Values in this field should be sent in JSON format, for example: {"**manufacturer**": "*Samsung*","**category**": "*Phone*","**sub_category_1**": "*Smart Phone*","**sub_category_2**": "*High-end*","**model_name**": "*Samsung Galaxy S10 Pro*"} Felds such as manufacturer, category, model_name are self-explanatory; sub_category_1 and sub_category_2 further describe the variants/types of that product. All fields in this JSON are optional.	varchar(2048)	optional
enable_auto_refund	Payment request is auto refunded in case of delay success depending upon the value present in the field is 'y' or 'n'. If this filed is not sent default set for your account at the time of setup will take effect.	varchar(1)	optional
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package	varchar(255)	mandatory

2.3. Response Parameters returned

Parameter name	Description
transaction_id	A unique ID that can be used to trace the transaction uniquely within ZenPay. Transaction IDs are alphanumeric. An example transaction ID is HDVISC1299876438
payment_mode	This tells the payment mode used by customer - example: "credit card", "debit card", "netbanking", etc.
payment_channel	This tells the payment channel used by customer - example: "Visa", "HDFC Bank", "Paytm", etc.
payment_datetime	Date and Time of this payment in "DD-MM-YYYY HH:MM:SS" format

response_code	Status of the transaction (return code). 0 signifies successful transaction. Non-zero values signify error. Response Code returned is documented in Appendix 4
response_message	The response message associated with the transaction.
error_desc	The detailed error description, if any
order_id	The same order_id that was originally posted by the merchant in the request.
amount	The same original amount that was sent by the merchant in the transaction request. In case of customer surcharge model this will be the amount paid by customer **.
currency	This is the 3digit currency code (INR), it will be same value that was originally sent by merchant.
description	The same description that was originally sent by the merchant in the transaction request.
name	The same value that was originally sent by merchant
email	The same value that was originally sent by merchant
phone	The same value that was originally sent by merchant
address_line_1	The same value that was originally sent by merchant
address_line_2	The same value that was originally sent by merchant
city	The same value that was originally sent by merchant
state	The same value that was originally sent by merchant
country	The same value that was originally sent by merchant
zip_code	The same value that was originally sent by merchant
udf1	The same value that was originally sent by merchant
udf2	The same value that was originally sent by merchant
udf3	The same value that was originally sent by merchant
udf4	The same value that was originally sent by merchant
udf5	The same value that was originally sent by merchant
tdr_amount	This is the TDR charged on the transaction **
tax_on_tdr_amount	This is the Tax (GST) charged on the TDR Amount **
amount_orig	This is the amount requested by merchant **. Typically, this will be same as the amount field, but in case of customer surcharge model this will be a different value.
cardmasked	Masked card number which was used to make the transaction **. For example, 437748******0069
'emi_tenure'	If "send_emi_details" merchant param is enabled and emi_info request is received.
'emi_rate_of_interest	If "send_emi_details" merchant param is enabled and emi_info request is received.
hash	ZenPay calculates the hash using the same algorithm which was outlined earlier. Hence, the merchant needs to check whether this returned hash matches the calculated hash.

** Note: This parameter will be returned as part of the response **only** if the merchant's account has been enabled for the same. Please speak to your ZenPay relationship manager if you would like this information to be returned in response.

Note: It is important to validate the hash after you receive the response from ZenPay. A failed response sent from ZenPay server to your server via browser could be tampered by a malicious end-

user and turned into “success”. To make sure the transaction response is the same as what ZenPay server sent please check the hash before considering the transaction response as final.

Note: Format of transaction ID is as follows: HDVISC1299876438". The 3rd to 6th digits (both inclusive) in the transaction ID signify the "bankcode". This information is enough to obtain the payment method and payment channel. A list of bankcodes and corresponding payment mode/channel is available in Appendix 3 of this document.

3. GET PAYMENT REQUEST URL (Two Step Integration)

ZenPay provides an API which returns a unique payment page URL on your merchant server in response which can be used in any browser to show the payment selection page and complete the transaction. This process gets complete in two steps.

3.1 Steps for Integration

- First step is, you need to submit a **POST REQUEST** to API URL https://pay.zenpay.biz/v2/getpaymentrequesturl

  Note: hash is a mandatory parameter. If your hash is not properly calculated or does not match for whatever reason, we will not be able to process the payment. The usage of hash is explained in subsequent sections.

- In response you will get a payment execution URL. Following is the sample response message in case of success.


```
{
    "data": {
    "url": "https://pay.zenpay.biz/v2/executepaymentrequesturl/3c1943aa-13be-4866-925e-d56c32c62d47",
     "uuid": "3c1943aa-13be-4866-925e-d56c32c62d47", "expiry_datetime": "2019-06-14 16:38:36", "order_id": "T103"
  }
}
```


- The response message apart from payment URL will contain a UUID (unique identification number for this transaction request), expiry date/time (this url will not work after the given expiry date/time) and order id (the one sent in the request by merchant). For every get payment URL request a unique UUID is generated in response.

- Second step is after getting the URL, open it in any browser or app webview for showing the payment method selection page to customer to complete the payment.

It is best to use this API if payments are going to be done through mobile apps for preventing frauds or hash / data tampering. The request can be built on your server and the unique URL can be sent to Client app, from where the payment process would continue.

3.2 Parameters to be posted in request

URL: https://pay.zenpay.biz/v2/getpaymentrequesturl

Parameters to be posted for this API are exactly that same as in v2/paymentrequest API (see Section 2.2), except for one additional optional parameter for defining the url expiry in minutes

Parameter Name	Description	Data type	Optional / Mandatory
expiry_in_minutes	This field is to define the response url expiry in minutes. This field excepts integer value minimum of 15 (15 minutes) and maximum of 10080 (7 days)	varchar(10)	optional

3.3 Successful Response Parameters Returned

Parameter Name	Description
url	Payment URL, which can open in any browser
expiry_datetime	The Payment URL expiration time, by default set for 15 mins
uuid	Unique ID generated for every request
order_id	This is your (merchant) reference number which you submitted while making the original transaction.

4. GET PAYMENT REQUEST INTENT URL (Two Step Integration)

ZenPay provides an API which returns a unique payment page URL on your merchant server in response which can be used in any browser to show the payment selection page and complete the transaction. This process gets complete in two steps.

4.1 Steps for Integration

4.1.1 First step is, you need to submit a **POST REQUEST** to API URL https://pay.zenpay.biz/v2/getpaymentrequestintenturl

Note: hash is a mandatory parameter. If your hash is not properly calculated or does not match for whatever reason, we will not be able to process the payment. The usage of hash is explained in subsequent sections.

4.1.2 In response you will get a payment intent URL. Following is the sample response message in case of success.


```
{
  "data": {
          "upi_intent_url":
"upi:\/\/pay?pa=ZENPAYUAT@ybl&pn=Karmendra+Biz&am=13.45&mam=13.45&tr=6478789&tn=Payment+for+6478789&mc=5021&mode=04&purpose=00&utm_campaign=DEBIT&utm_medium=ZENPAYUAT&utm_source=6478789",
          "payment_request_id": 6478789,
          “order_id” : 123456
  }
}
```


4.1.3 The response message contains the intent url, transaction id and order id sent in the payment request.

4.1.4 Second step is after getting the URL is to use trigger an app from the respective mobile device.

It is best to use this API if payments are going to be done through mobile payment apps for preventing frauds or hash / data tampering.

4.2 Parameters to be posted in request

URL: https://pay.zenpay.biz/v2/getpaymentrequestintenturl

Parameters to be posted for this API are exactly that same as in v2/paymentrequest API (see Section 2.2).

4.3 Successful response parameters returned

Parameter Name	**Description**
upi_intent_url	Intent URL, which can open in any mobile payment apps
payment_request_id	It will be transaction id associated with request
order_id	This is your (merchant) reference number which you submitted while making the original transaction.

5. PAYMENT STATUS API

ZenPay provides an API which you can use to check the status of any prior transaction. You can use this to reconcile transactions. We strongly recommend that you make it a practice to use this for every transaction that was made. This serves two purposes:

4.3.1 The response might not reach you due to network issues or other problems such as user clicking refresh button on their browser, etc.

4.3.2 This also protects against any tampering, since you have a second fallback check here.

ZenPay offers a sophisticated API wherein you can apply "filters" on the resultset you want to retrieve. You can search our system by the transaction ID, or the order ID, or even by parameters such as date range, customer phone number, etc. You can also pass in various combinations of these parameters to get the resultset of your choice.

Note: Your designated server IP will need to be whitelisted by ZenPay for this API to work. If you receive errors such as "Unauthorized" while accessing this API, please contact your ZenPay relationship manager to get this fixed.

URL: *https://pay.zenpay.biz/v2/paymentstatus*

4.1. Parameters to be POSTed

Parameter Name	Description	Data type	Optional / Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	Mandatory
order_id	This is your (merchant) reference number which you submitted while making the original transaction. You can send multiple order ids in this field as comma (,) separated list	varchar(30)	Optional
transaction_id	This is the transaction ID generated by ZenPay for the given transaction	varchar(30)	Optional
bank_code	This is the 4-letter bankcode which denotes the payment mode/channel of the payment.	varchar(4)	Optional
response_code	The numeric response code returned by ZenPay during the original transaction	number(4)	Optional

customer_phone	Phone number of the customer, as provided during the original paymentrequest API	varchar(30)	Optional
customer_email	Email address of the customer, as provided during the original paymentrequest API	varchar(255)	Optional
customer_name	Name of the customer, as provided during the original paymentrequest API	varchar(255)	Optional
date_from	Start date of date range to retrieve transactions, in DD-MM-YYYY or YYYY-MM-DD HH:MM:SS format	varchar(20)	Optional
date_to	End date of date range to retrieve transactions, in DD-MM-YYYY or YYYY-MM-DD HH:MM:SS format	varchar(20)	Optional
page_number	Page number you need to retrieve, its value is limited by information received in the first response that is received	integer	Optional
per_page	Number to records need to see per page, this value should be between 1 and 50	integer	Optional
Hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package	varchar(255)	Mandatory

4.2. Response Parameters

On successful call to this API you will receive JSON response. You can read the JSON response and process it at your end. If your result set is greater than 50 transactions, you would need to use pagination.

Note: few parameters in response will be visible only if it is enabled for your account, for example: refund_details will be available if it is enabled for your merchant account.

A few sample responses for given requests are provided below:

In case of success,


```
{
  "data": [
    {
      "transaction_id": "SFSBIN2783912661",
```

```
    "bank_code": "SBIN",
    "payment_mode": "Netbanking",
    "payment_channel": "State Bank of India",
    "payment_datetime": "2018-06-13 16:44:03",
    "response_code": 1000,
    "response_message": "FAILED",
    "authorization_staus": null,
    "order_id": "427641",
    "amount": "27.36",
    "amount_orig": "2.00",
    "tdr_amount": 21.49,
    "tax_on_tdr_amount":   3.87,
    "description": "Web Payment for 433487",
    "error_desc": "FAILED",
    "customer_phone": "9900990099",
    "customer_name": "sharathkumar hegde",
    "customer_email": "sharathkumar@example.com"
    "currency": "INR",
    "cardmasked": null,
    "udf1": null,
    "udf2": null,
    "udf3": null,
    "udf4": null,
    "udf5": null,
    "refund_details": {
       "refund_amount": 0
    }
},
{
    "transaction_id": "HDVISC4291974106",
    "bank_code": "VISC",
    "payment_mode": "Credit Card",
    "payment_channel": "Visa",
    "payment_datetime": "2018-06-13 16:45:39",
    "response_code": 0,
    "response_message": "SUCCESS",
    "authorization_staus": "captured",
    "order_id": "427643",
    "amount": "1.93",
    "amount_orig": "1.90",
    "tdr_amount": 0.03,
    "tax_on_tdr_amount": 0,
```

```
        "description": "Web Payment for 433489",
        "error_desc": null,
        "customer_phone": "9900990099",
        "customer_name": "sharathkumar hegde",
        "customer_email": "sharathkumar@example.com"
        "currency": "INR",
        "cardmasked": null,
        "udf1": null,
        "udf2": null,
        "udf3": null,
        "udf4": null,
        "udf5": null,
        "refund_details": {
           "refund_amount": 0
        }
      }
   ],
   "page": {
      "total": 175,
      "per_page": 10,
      "current_page": 1,
      "last_page": 18,
      "from": 1,
      "to": 10
   },
   "hash":
"30FAAD865191B4064576F063177F0A4692C3DBBBF35D1A20463EAA449269C4715FD13528EA069B3A8
D5C25C62637ED825C297C2337CDC1CFB7FCD0D60DCFEB9D"
}
```


In case of error,


```
{
  "error": {
    "code": 1001,
    "message": "The api key field is incorrect"
  }
}
```


In case there is no record present in our system for the combination of input, following error is returned

```
{
   "error": {
      "code": 1050,
      "message": "No data record found for the given input"
   }
}
```


In case there is no transaction id in our system for the order_id, merchant_order_id or transaction_id, following error is returned


```
{
   "error": {
      "code": 1028,
      "message": "No Transaction found"
   }
}
```


If there are more than 50 transactions for which the status is requested, you would see following error


```
{
   "error": {
      "code": 1086,
      "message": "More than 50 records, refine your search criteria or use pagination"
   }
}
```


In such cases where result set is expected to have status of more than 50 transactions, it is required to use pagination.

Pagination allows to access data in smaller chunks making it easier for server to return data quickly.

To allow for pagination one need to pass following additional parameter in request

Parameter Name	Description	Data type	Optional / Mandatory
page_number	Page number you need to retrieve, its value is limited by information received in the first response that is received	integer	optional
per_page	Number to records need to see per page, this value should be between 1 and 50	integer	optional

If above parameters are passed, response will have additional information about pagination as following


```
"page": {
      "total": 175,
      "per_page": 10,
```

```
    "current_page": 1,
    "last_page": 18,
    "from": 1,
    "to": 10
  },
```


This pagination information should be used in page_number filed for subsequent api request.

6. Refunds API

ZenPay provides a refund API which merchants can use to programmatically issue refunds instead of clicking the "refund" button in the ZenPay UI. This API can be invoked on any prior successful transaction. The transaction which is being refunded should be in either "paid" or "settled" state, or in "refunded" state (in case of partial amount refunds). Refunds can be either for the full amount paid by the customer, or any part of it.

The API needs a valid transaction ID as input.

Note: processing of refunds is subject to availability of funds in subsequent settlement cycles. This API will return a failure response in case sufficient funds are not available to process the refund.

5.1. Refund request API

URL: https://pay.zenpay.biz/v2/refundrequest

Request Parameters:

Parameter Name	Description	Data type	Optional / Mandatory
api_key	ZenPay would assign a unique 36- digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	Mandatory
transaction_id	The unique alphanumeric transaction ID generated by ZenPay for a prior transaction.	varchar(30)	Mandatory
merchant_refund_id	This is your (merchant) refund reference number. It must be unique for every refund request. If a refund request is sent with same merchant_refund_id we return the response of the previously successful refund request. Warning: If you are N OT using this filed then be careful, as each request will be treated as a new refund request. Thus it is recommended to use this field.	varchar(30)	Optional
merchant_order_id	This is your (merchant) reference number which you submitted while	varchar(30)	Optional

	making the original transaction. Note that if this value does not match with related transaction_id field then you will get error. In typical cases do not send this field.		
amount	The amount which needs to be refunded. This needs to be less than or equal to the transaction amount.	decimal(10,2)	Mandatory
description	Description of the refund. Usually the reason for issuing refund, as specified by merchant.	varchar(500)	Mandatory
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package	varchar(255)	Mandatory

Response Parameters:

The output is a JSON which contains the error(s), if any, in validation, or a simple success message which confirms that the refund request has been accepted and will be processed during subsequent settlement cycle.

If the request is successfully processed response you will get a "*data*" block, and in case of failure you will see "*error*" block, you will not get "*data*" key in case of error.

In case of success, **NOTE:** that *refund_reference_no* is returned by the bank and it can be null in case refunds are not initiate by bank immediately, but is done at end of the day.

```
{
  "data": {
    "transaction_id": "HDVISC7472820193",
    "refund_id": 4351,
    "refund_reference_no": null
    "merchant_refund_id": 76783_R_1,
    "merchant_order_id": 76783,
```

```
  }
}
```

In case of error,

```
{
  "error": {
    "code": 1039,
    "message": "The refund amount is greater than transaction amount"
  }
}
```

5.2. Refund Status API

If a refund is initiated either from merchant or payment gateway end and merchant wants to check its status (details such as if it is refunded or not how much amount was paid and how much is refunded will be posted in response). To check the status of any refund which was initiated merchant should post the API request.

URL: https://pay.zenpay.biz/v2/refundstatus

Request Parameters:

Parameter Name	Description	Data type	Optional / Mandatory
api_key	ZenPay would assign a unique 36- digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	Mandatory
transaction_id	The unique alphanumeric transaction ID generated by ZenPay for a prior transaction.	varchar(30)	Mandatory

merchant_order_id	This is your (merchant) reference number which you submitted while making the original transaction. Note that if this value does not match with related transaction_id field then you will get error. In typical cases do not send this field.	varchar(30)	Optional
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package	varchar(255)	Mandatory

Response Parameters:

The output is a JSON it will give all details about a refund if any initiated for this transaction, if not it will give json with error. Partial multiple refunds are also shown in refund_details

If the request is successfully processed response you will get a "*data*" block, and in case of failure you will see "*error*" block, you will not get "*data*" key in case of error.

NOTE: that *refund_reference_no* is returned by the bank and it can be null in case refunds are not initiate by bank immediately but is done at end of the day.


```
{
  "data": {
    "transaction_id": "FDAXIP9740656834",
    "merchant_order_id": "351177",
    "refund_amount": 2.04,
    "transaction_amount": "2.04",
    "refund_details": [
      {
        "refund_id": 3523,
        "refund_reference_no": "602201803257434370",
        "merchant_refund_id": null,
        "refund_amount": "2.04",
        "refund_status": "Customer Refunded",
```

```
      "date": "2018-02-01 11:19:49"
    }
   ]
  },
  "hash":
"20D8CB42D14C35AAEF06BB200C82E560DCC1D0C19EEFFBFD07CBEEB3BD39AE746AFB30A5803D6375
27CE1A45AE367565E8AF5933809E3F597D7CDDDCDB3C28FE"
}
```


In case of error,


```
{
  "error": {
    "code": 1050,
    "message": "No data record found for the given input"
  }
}
```

7. SPLIT API

6.1. Split Settlement API

6.1.1. Split transaction before settlement API

URL: ***https://pay.zenpay.biz/v2/splitsettlementrequest***

Request Parameters:

Parameter Name	Description	Data type	Optional / Mandatory
api_key	The unique key provided to the merchant	varchar(36)	Mandatory
order_id	The order id of the transaction		Mandatory
split_info	The json format data can contain vendor_code and vendor_percent or vendor_code and vendor_amount, see the json structure below.	json	Mandatory
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package hash = strtoupper(hash('sha512', salt|api_key|order_id|split_info))	varchar(255)	Mandatory

The split_info parameter will be in json format as shown below:

```
{
      "vendors":[
            {
                  "vendor_code":"2VEN449",
                  "split_amount_percentage":"80"
            },
            {
                  "vendor_code":"XYZ123",
                  "split_amount_fixed":"11"
            }
```

```
      ]
}
```

Response Parameters:

The response will be in json format as show below:

In case of success,

```
{
    "data": {
        "message": "The split settlement request is successful."
    }
}
```

In case of total split percentage or amount exceeds 100% or total settlement amount

```
{
    "error": {
        "code": 1024,
        "message": "Sum of split amount should be less than or equal
to settlement amount."
    }
}
```

In case of vendor code invalid or not approved

```
{
    "error": {
        "code": 1007,
        "message": "One or more Codes is either not added or not
approved."
    }
}
```

8. VENDOR API

7.1. Add Vendor API

URL: https://pay.zenpay.biz/v2/addvendor

This API allows the merchant to register new vendors with the ZenPay system. These vendors can also be added manually from the ZenPay dashboard.

When a vendor is added, it is "non-approved" by default. ZenPay will approve the vendors separately. This is for security purposes.

Parameter Name	Description	Data type	Optional / Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	mandatory
vendor_code	This is the vendor code that you wish to add in the ZenPay system. This has to be unique. Alphanumeric values are permitted here.	varchar(30)	mandatory
vendor_name	A descriptive name to identify the vendor.	varchar(100)	mandatory
vendor_contact_email	Email address where the vendor can be contacted. Has to be a valid email address.	varchar(200)	mandatory
vendor_contact_num	Phone number where the vendor can be contacted.	varchar(10)	mandatory
vendor_contact_address	Address where the vendor can be reached.	varchar(300)	optional
account_name	Account holder name (of the vendor bank account). Optional if UPI details are given.	varchar(300)	optional
account_number	Account number of the vendor. Optional if UPI details are given.	varchar(50)	optional
ifsc_code	IFSC code of the vendor's bank. Optional if UPI details are given.	varchar(50)	optional
bank_name	Bank name of the vendor's bank. Optional if UPI details are given.	varchar(200)	optional
bank_branch	Bank branch of the vendor's bank. Optional if UPI details are given.	varchar(300)	optional
upi_id	UPI VPA of the vendor. Optional if bank account details are given.	varchar(50)	optional
vendor_pan	PAN number of the vendor	varchar(10)	optional
description_1	Vendor description 1	varchar(200)	optional
description_2	Vendor description 2	varchar(200)	optional

activate_bharat_qr	Pass 'y' in case bharat qr needs to be generated for this vendor	varchar(1)	optional
aadhar_number	Aadhar number is required if bharat qr needs to be generated	varchar(12)	optional
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package Please ensure that the concatenated string (upon which the hash will be computed) is computed based on columns in alphabetical order. Please include only those columns in your hash calculation which you are actually passing to us. For example, if you are not passing vendor_contact_address do not include that in hash calculation.	varchar(200)	mandatory

Note: This API will return error if the vendor already exists in the system AND is active. If an inactive/disapproved vendor exists, this API will update the details for that vendor code.

The response will be in json format as show below:

In case of success,


```
{
    "data": {
        "code":"SUCCESS",
        "account_id":0026,
        "message": "Vendor is added Successfully"
        "static_qr_code":  "00020101021102164077580000037748061661004600000385150821hdfc00000016271312641001 0A0000005240116yap94026@equitas02031.027230010A0000005240105940262830001 0A00000052401121234123412345204739953033565802IN5907abc pay6032bangalore6106560001621207089402600123 14D41"
    }
}
```


In case vendor already exists


```
{
    "error": {
        "code": 1024,
```

```
        "message": "Vendor code already exists"
    }
}
```

7.2. Modify Vendor API

URL: https://pay.zenpay.biz/v2/modifyvendor

Pre-existing vendors in the system can be modified using this API. This API works on approved as well as non-approved vendors. However, any modification to a pre-existing active vendor will immediately disapprove that vendor, automatically. If the vendor that is being modified does not exist, the API will return an error and will NOT automatically add the vendor. This will change the default account for the vendor

Parameter Name	Description	Data type	Optional/ Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	mandatory
vendor_code	This is the vendor code that you wish to modify in the ZenPay system. This value must already exist in the system, failing which ZenPay will return an error.	varchar(30)	mandatory
vendor_name	A descriptive name to identify the vendor.	varchar(100)	optional
vendor_contact_email	Email address where the vendor can be contacted. Has to be a valid email address.	varchar(200)	optional
vendor_contact_num	Phone number where the vendor can be contacted.	varchar(10)	optional
vendor_contact_address	Address where the vendor can be reached. Optional.	varchar(300)	optional
account_id	Account id which needs to be modified. The account_id is returned in the addvendor API. If it is not passed, the last approved account of the vendor is updated	varchar(10)	optional
account_name	Account holder name (of the vendor bank account). Optional if UPI details are given.	varchar(300)	optional
account_number	Account number of the vendor. Optional if UPI details are given.	varchar(50)	optional
ifsc_code	IFSC code of the vendor's bank. Optional if UPI details are given.	varchar(50)	optional
bank_name	Bank name of the vendor's bank. Optional if UPI details are given.	varchar(200)	optional

bank_branch	Bank branch of the vendor's bank. Optional if UPI details are given.	varchar(300)	optional
upi_id	UPI VPA of the vendor. Optional if bank account details are given	varchar(50)	optional
vendor_pan	PAN number of the vendor	varchar(10)	optional
description_1	Vendor description 1	varchar(200)	optional
description_2	Vendor description 2	varchar(200)	optional
activate_bharat_qr	Pass 'y' in case bharat qr needs to be generated for this vendor	varchar(1)	optional
aadhar_number	Aadhar number is required if bharat qr needs to be generated	varchar(12)	optional
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package Please ensure that the concatenated string (upon which the hash will be computed) is computed based on columns in alphabetical order. Please include only those columns in your hash calculation which you are actually passing to us. For example, if you are not passing vendor_contact_address do not include that in hash calculation.	varchar(200)	mandatory

The response will be in json format as show below:
In case of success,

```
{
    "data": {
        "code":"SUCCESS",
        "message": " Vendor Details Updated Successfully"
    }
}
```

7.3. Add Vendor Accounts API

URL: https://pay.zenpay.biz/v2/addvendoraccount

Multiple accounts can be added to pre-existing vendors in the system using this API. This API works on approved as well as non-approved vendors. If the vendor that is being given does not exist, the API will return an error and will NOT automatically add details to the vendor.

Parameter Name	Description	Data type	Optional/ Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	mandatory
vendor_code	This is the vendor code that you wish to modify in the ZenPay system. This value must already exist in the system, failing which ZenPay will return an error.	varchar(30)	mandatory
account_name	Account holder name (of the vendor bank account). Optional if UPI details are given.	varchar(300)	optional
account_number	Account number of the vendor. Optional if UPI details are given.	varchar(50)	optional
ifsc_code	IFSC code of the vendor's bank. Optional if UPI details are given.	varchar(50)	optional
bank_name	Bank name of the vendor's bank. Optional if UPI details are given.	varchar(200)	optional
bank_branch	Bank branch of the vendor's bank. Optional if UPI details are given.	varchar(300)	optional
upi_id	UPI VPA of the vendor. Optional if bank account details are given	varchar(50)	optional
default_account	Whether this will be the default account of the vendor or not. Possible values are "y" or "n". IMPORTANT: System can have only one default account, if the value is passed as 'y' and default account exist, error will be displayed	varchar(1)	mandatory
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package Please ensure that the concatenated string (upon which the hash will be computed) is computed based on columns in alphabetical order. Please include only those columns in your hash calculation which you are actually passing to us. For example, if you are not	varchar(200)	mandatory

	passing vendor_contact_address do not include that in hash calculation.		

The response will be in json format as show below:
In case of success,

```
{
    "data": {
        "code":"SUCCESS",
        "account_id":"12313",
        "message": " Vendor account is added Successfully"
    }
}
```

7.4. Delete Vendor API

URL: https://pay.zenpay.biz/v2/deletevendor

This API can be used to delete a pre-existing vendor from the ZenPay system. Subsequent to deletion, there can be no further split payments to this vendor. Importantly, deletion of a vendor will NOT impact pending payouts to the vendor. Any pending settlements will still occur

Parameter Name	Description	Data type	Optional/ Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	mandatory
vendor_code	This is the vendor code that you wish to delete from the ZenPay system. This value must already exist in the system, failing which ZenPay will return an error.	varchar(30)	mandatory
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package hash = toUpper (sha512 (SALT| api_key |vendor_code))	varchar(255)	mandatory

This API can be used to delete a pre-existing vendor from the ZenPay system. Subsequent to deletion, there can be no further split payments to this vendor. Importantly, deletion of a vendor will NOT impact pending payouts to the vendor. Any pending settlements will still occur
The response will be in json format as show below:

In case of success,

```
{
    "data": {
        "code":"SUCCESS",
        "message": " Vendor is deleted Successfully"
    }
}
```

7.5. Get Vendor API

URL: https://pay.zenpay.biz/v2/vendorstatus

This API can be used to delete a pre-existing vendor from the ZenPay system.

Parameter Name	Description	Data type	Optional/ Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	mandatory
vendor_code	This is the vendor code that you wish to retrieve from the ZenPay system. This value must already exist in the system, failing which ZenPay will return an error.	varchar(30)	mandatory
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package hash = toUpper (sha512 (SALT| api_key |vendor_code))	varchar(255)	mandatory

This API can be used to get details of a pre-existing vendor from the ZenPay system.

The response will be in json format as show below:

In case of success,

```
{
    "data": {
        "vendor_code": "5d1ee6743a3asd2d476a669b",
        "vendor_name": "Sindhu",
        "vendor_contact_email": "sharathkumar@example.com",
        "vendor_contact_num":  "9900990099",
        "vendor_contact_address": "Domlur, Indira Nagar, Bangalore",
        "vendor_pan": "ARSPH1234Q",
        "vendor_approved": "y",
        "vendor_split_percentage": null,
        "vendor_split_amount": null,
        "earliest_settlement_time_frame": null,
        "latest_settlement_time_frame": null,
        "vendor_logo": null,
        "qr_code":
"
ABlBMVEX///8AAABVwtN+AAAAAXRSTlMAQObYZgAAAAlwSFlzAAAOxAAADsQBlSsOGwAA…
……vRou3ed6h+OmH3Ch8fMD+wfnU1Eyj4zFVQAAAABJRU5ErkJggg==",
        "bank_accounts": [
            {
                "account_id": 3288,
                "account_name": "Dummy",
                "account_number": "1000000001",
                "ifsc_code": "UTIB0000003",
                "bank_name": "Dummy Bank",
                "bank_branch": "Demo Road",
                "upi_id": null,
                "bank_approved": "y",
                "default_account": "y"
            }
        ]
    }
}
```

9. SETTLEMENT APIs

8.1. Get Settlements API

URL: https://pay.zenpay.biz/v2/getsettlements

This API allows a merchant to programmatically access the status of any of his past settlements and other pertinent information pertaining to a prior settlement. If this API returns a blank bank_reference_number, it means the amount is not yet settled. If the API returns no data, it means that the system has not calculated settlements yet, you would need to re-check after 12:30 AM.

Please note that this API will not provide any information for failed transactions since, there can be no settlement for a failed transaction. To obtain information about failed transactions, use the payment status API described in an earlier section.

8.1.1. Parameters to be POSTed in Request

Parameter Name	Description	Data type	Optional / Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	Mandatory
bank_reference	The bank reference number of the actual NEFT/IMPS/RTGS transaction performed by ZenPay to the merchant's current account	varchar(100)	Optional
date_from	The start date from which you need to retrieve settlement information. This needs to be passed in DD-MM-YYYY format.	varchar(10)	Optional
date_to	The end date at which you need to retrieve settlement information. This needs to be passed in DD-MM-YYYY format.	varchar(10)	Optional
completed	Whether settlement is completed or not. Pass in 'y' or 'n' here.	varchar(1)	Optional
settlement_id	The unique numeric settlement ID assigned to each settlement	number(20)	Optional
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package.	varchar(255)	Mandatory

This API returns a JSON in the following format:


```
{
   "data": [
      {
         "settlement_id": 10075,
         "bank_reference": "710061536126",
         "payout_amount": "2.06",
         "completed":  "y",
         "account_name": "Tester Sharma",
         "account_number": "50100012341231",
         "ifsc_code": "HDFC0000002",
         "bank_name": "HDFC BANK",
         "bank_branch": "CMH RD, INDIRA NAGAR BRANCH",
         “settlement_datetime": "2017-02-20 16:31:28",
         "sale_amount": "3.00",
         "chargeback_amount": “0.00”,
         "refund_amount": "0.00"
      }
   ],
   "hash":
"684CDA22F7A429D68281444A8F6809A5FEFEA7A055258984E129554AC359C956E58E36B67A4EB9F948
1E616888E722DDB95A81EFBED4416B24F19E3126077F5E"
}
```


In case there is no record found in the system for the combination of input parameter, following error is returned


```
{
  "error": {
    "code": 404,
    "message": "No record found"
  }
}
```


8.2. Get Settlement Details API

URL: https://pay.zenpay.biz/v2/getsettlementdetails

This API allows a merchant to programmatically access the status of any of his past **settlement details** (transaction level settlements).

Please note that this API will not provide any information for failed transactions since by definition, there can be no settlement for a failed transaction. To obtain information about failed transactions, use the payment status API described in an earlier section.

8.2.1. Parameters to be POSTed in Request

Parameter Name	Description	Data type	Optional / Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	mandatory
order_id	Order ID passed by the merchant during the original payment transaction request	varchar(30)	optional
transaction_id	Transaction ID assigned by ZenPay for this successful transaction	varchar(30)	optional
bank_code	Bank code signifying payment mode and channel	varchar(4)	optional
customer_phone	Phone number of customer as provided during the original paymentrequest API call	varchar(30)	optional
customer_email	Email ID of customer as provided during the original paymentrequest API call	varchar(255)	optional
customer_name	Name of customer as provided during the original paymentrequest API call	varchar(255)	optional
date_from	The start date from which you need to retrieve settlement detail information. This needs to be passed in DD-MM-YYYY format.	varchar(10)	optional
date_to	The end date at which you need to retrieve settlement detail information. This needs to be passed in DD-MM-YYYY format.	varchar(10)	optional
completed	Whether settlement is completed or not. Pass in 'y' or 'n' here.	varchar(1)	optional
settlement_id	The unique numeric settlement ID assigned to each settlement	number(20)	optional
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package.	varchar(255)	mandatory

This API returns a JSON in the following format:

```
{
  "data": [
    {
```

```
        "transaction_id": "HDMASC2746901262",
        "order_id": "225495",
        "settlement_id": 27837,
        "bank_reference": "710061536126",
        "settlement_datetime": null,
        "customer_name": "Tester",
        "customer_email": "tester@example.com",
        "customer_phone": "8050603774",
        "completed": "y",
        "description": "Settlement for Rs. 2.06 paid through transaction ID HDMASC2746901262 on 2017-09-28 13:36:19 for merchant hotel booking",
        "gross_transaction_amount": "2.06",
        "payment_mode": "Credit Card",
        "payment_channel": "Master",
        "applicable_tdr_percent": "3.00",
        "applicable_tdr_fixed_fee": "0.00",
        "percent_tdr_paid_by_merchant": "0",
        "tdr_amount": "0.06",
        "tax_on_tdr_amount":  "0.00",
        "amount_reimbursed":  "2.00"
    }
  ],
  "hash":
"D2EFF4776D973DA46563DA0F80139B84AFED77C58496A34DD0D653272A0EE1E5D09F4C94AD4394512B16341A5A44906B4B10FF5B6AA1F03DE98A164B39881C4E"
}
```


In case there is no record found in the system for the combination of input parameter, following error is returned


```
{
  "error": {
    "code": 404,
    "message": "No record found"
  }
}
```

10. CHALLAN PAYMENT API

9.1. Request challan payment API

URL: https://pay.zenpay.biz/v1/requestchallan

This API allows the merchant to create a link which can be sent to customers by email and/or SMS. This link allows the customer to make easy payments without data entry hassles.

On clicking this link, the customer is taken directly to a confirmation page where he can verify his details (email ID, name and amount), and on confirmation, he is taken to the payment page.

Parameter Name	Description	Data type	Optional/ Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	mandatory
name	Name of the person whom the invoice is addressed to.	varchar(100)	Mandatory
mobile	Phone number of the person whom the invoice is addressed to.	varchar(10)	mandatory
email	Email ID of the person whom the invoice is addressed to.	varchar(100)	mandatory
amount	Amount which the user needs to pay.	decimal(15,2)	mandatory
purpose	Purpose of payment - this should be a descriptive string which clearly tells the user what he is paying for.	varchar(100)	mandatory
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package.	varchar(255)	mandatory

9.2. Request challan payment API url

URL: https://pay.zenpay.biz/v1/generatechallanurl

This API allows the merchant to create a url which can be sent to customers by email and/or SMS. This url allows the customer to make easy payments without data entry hassles.

On clicking above url, the customer is taken directly to a confirmation page where he can verify his details (email ID, name and amount), and on confirmation, he is taken to the payment page.

Request parameters are as following:

Parameter Name	Description	Data type	Optional/ Mandatory
api_key	ZenPay would assign a unique 36-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	mandatory
name	Name of the person whom the invoice is addressed to.	varchar(100)	Mandatory
mobile	Phone number of the person whom the invoice is addressed to.	varchar(10)	mandatory
email	Email ID of the person whom the invoice is addressed to.	varchar(100)	mandatory
amount	Amount which the user needs to pay.	decimal(15,2)	mandatory
purpose	Purpose of payment - this should be a descriptive string which clearly tells the user what he is paying for.	varchar(100)	mandatory
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, using the following mechanism: toUpper (sha512 (SALT|amount| api_key | email|mobile|name|purpose)) Note: the SALT will be provided by ZenPay separately. Please ensure that the concatenated string (upon which the hash will be computed) is in alphabetical order. Note: NEVER PASS SALT IN A FORM	varchar(255)	Mandatory

Response from this API will be in JSON format:

On successful call to this API you will receive JSON response in following format.

```
{
  "data": {
    "url": "http://biz.localhost.com/challan/b39b0596-73c4-4b7e-b63d-
bbc13361e044",
    "uuid": "b39b0596-73c4-4b7e-b63d-bbc13361e044",
    "tnp_id": 81600
  }
}
```

data - successful response will have "**data**" tag.
url - this is the url what can be distributes as suitable.
uuid - this is the unique identifier for this request.
tnp_id - this is another unique identifier that can be used for getting the transaction details using paymentStatusById API.

On failure json response is as following:

```
{
  "error": {
    "code": 221,
    "message": "GEN-UNAUTHORIZED - The api key field is incorrect"
```

```
    }
}
```

error - erred response will have "**error**" tag.
code - this is error category code
message - this is more descriptive error tag and error message.

List of error codes and corresponding messages:

Code	Message
221	GEN-UNAUTHORIZED The api key field is incorrect The hash key field is invalid
998	GEN-INVALID-PARAMS The name field is required. The email field is required. The mobile field is required. The amount field is required. The purpose field is required. The hash field is required.

11. Server to Server Call Back (Web hooks)

10.1. Server to server response on Payment

To get server to server response, add callback URL in parameter named "Payment Callback URL" in your ZenPay dashboard. If this is not found contact ZenPay to set this up for you.

Whenever there is a successful payment done by your customer apart from receiving success or failure message on customers' browser, following response parameters are also posted to the mentioned callback URL.

These are very same response that we send as response to **paymentrequest** API.

Parameter name	Description
transaction_id	A unique ID that can be used to trace the transaction uniquely within ZenPay. Transaction IDs are alphanumeric.
payment_method	This tells the payment method used by customer - example: "credit card", "debit card", "netbanking", etc.
payment_datetime	Date and Time of this payment in "DD-MM-YYYY HH:MM:SS" format
response_code	Status of the transaction (return code). 0 signifies successful transaction. Non-zero values signify error.
response_message	Can have a value of "success" or "failure". Order
order_id	The same order_id that was originally posted by the merchant in the request.
amount	The same original amount that was sent by the merchant in the transaction request.
currency	This is the 3digit currency code (INR), it will be same value that was originally sent by merchant.
description	The same description that was originally sent by the merchant in the transaction request.
name	The same value that was originally sent by merchant
email	The same value that was originally sent by merchant
phone	The same value that was originally sent by merchant
address_line_1	The same value that was originally sent by merchant
address_line_2	The same value that was originally sent by merchant
city	The same value that was originally sent by merchant
state	The same value that was originally sent by merchant
country	The same value that was originally sent by merchant
zip_code	The same value that was originally sent by merchant
udf1	The same value that was originally sent by merchant
udf2	The same value that was originally sent by merchant
udf3	The same value that was originally sent by merchant
udf4	The same value that was originally sent by merchant
udf5	The same value that was originally sent by merchant
hash	ZenPay calculates the hash using the same algorithm which was outlined earlier. Hence, the merchant needs to check whether this returned hash matches the calculated hash.

10.2. Server to server response on Settlement

To get server to server response, add callback URL in parameter named "Settlement Callback URL" in your ZenPay dashboard. If this is not found contact ZenPay to set this up for you.

Whenever there is a successful settlement done by ZenPay to your bank account apart from receiving success or failure email message, following response parameters are also posted to the mentioned callback URL.

These are very same response that we send as response to **getsettlements** API.

Parameter name	Description
settlement_id	Settlement Id for this aggregated settlement
bank_reference	Bank reference Number
payout_amount	Aggregated Amount paid to merchant
completed	Settlement is completed or not, 'y' or 'n'
account_name	Account Holders Name to which the Amount is settled
account_number	Account Number to which the Amount is settled
ifsc_code	IFSC Code of the branch to which Account Number belongs
bank_name	Bank name to which Account Number belongs
settlement_datetime	Date of settlement
sale_amount	Total sale amount for the transactions included in this aggregated settlement
chargeback_amount	Amount deducted from the sale amount for chargeback adjustment
refund_amount	Amount deducted from the sale amount for refunds adjustment.

12. SEAMLESS PAYMENT REQUEST API

In case you perform the normal payment integration process as outlined earlier in the document, the customer will necessarily be redirected to the ZenPay payment page wherein he will be required to enter his card details or select the appropriate bank/payment instrument (such as wallets etc.) with which he would like to make the payment.

If you would like your customer to perform the bank selection and/or enter his card information on your site, without using the ZenPay payment page, this can be achieved using the seamless payment request API. ZenPay would provide you a list of appropriate bankcodes which you would be required to pass along with the payment request. These bankcodes are available in Appendix 3 of this document.

Please note that in order to use our seamless payment request API you would need to be PCI-DSS compliant. For more information on PCI compliance, or if you would like us to assist you in your PCI compliance efforts, please contact your ZenPay relationship manager.

11.1. Steps for Integration

- You need to submit a POST REQUEST to our server, at the below mentioned URL URL https://pay.zenpay.biz/v2/paymentseamlessrequest `
- Note: hash is a mandatory parameter. If your hash is not properly calculated or does not match for whatever reason, we will not be able to process the payment. The usage of hash is explained in subsequent sections.
- When you submit your transaction request to ZenPay, we assign a transaction ID to you.
- The customer is automatically redirected to the appropriate bank or 3D-secure page, as the case may be.
- After the customer pays the entire amount using one or more payment instruments, he is redirected back to the merchant site.
- We recommend that you check the hash at your end again, after we send back the response to you. This is essential to prevent user data tampering fraud.

11.2. Parameters to be POSTed in Seamless Payment Request

URL: https://pay.zenpay.biz/v2/paymentseamlessrequest

Parameter Name	Description	Data type	Optional / Mandatory
api_key	ZenPay would assign a unique 40-digit merchant key to you. This key is exclusive to your business/login account. If you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.	varchar(36)	Mandatory
order_id	This is your (merchant) reference number. It must be unique for every transaction.	varchar(30)	Mandatory

mode	This is the payment mode (TEST or LIVE are valid values). Defaults to LIVE.	varchar(4)	Optional
amount	This is the payment amount.	decimal(15,2)	Mandatory
currency	This is the 3digit currency code (INR)	varchar(3)	Mandatory
description	Brief description of product or service that the customer is being charged for.	varchar(500)	Mandatory
name	Name of customer.	varchar(100)	Mandatory
email	Customer email address.	varchar(100)	Mandatory
phone	Customer phone number	varchar(50)	Mandatory
address_line_1	Customer address	varchar(100)	Optional
address_line_2	Customer address 2	varchar(100)	Optional
city	Customer city	varchar(50)	Mandatory
state	Customer State	varchar(50)	Optional
country	Customer country must be IND	varchar(50)	Mandatory
zip_code	Customer zip code	varchar(20)	Mandatory
udf1	User defined field 1	varchar(300)	Optional
udf2	User defined field 2	varchar(300)	Optional
udf3	User defined field 3	varchar(300)	Optional
udf4	User defined field 4	varchar(300)	Optional
udf5	User defined field 5	varchar(300)	Optional
bank_code	Bank code identifies the payment mode and channel.	varchar(20)	Mandatory
card_number	Card number (11 to 19 digits)	varchar(19)	Conditional
expiry_date	Expiry date in mm/yyyy format	varchar(7)	Conditional
card_holder_name	Card holder name	varchar(30)	Conditional
cvv	CVV/CVC	varchar(4)	Conditional
payer_virtual_address	Virtual Payee Address – VPA, Mandatory for UPIU bankcode	varchar(100)	Conditional
return_url	Return URL success - ZenPay will make a POST request to this URL after successful transaction, with a set of parameters, which you can process as you want to.	varchar(300)	Mandatory
return_url_failure	Return URL failure - ZenPay will make a POST request to this URL after failed transaction, with a set of parameters, which you can process as you want to. Defaults to return_url.	varchar(300)	Optional
return_url_cancel	Return URL cancel - ZenPay will make a POST request to this URL after user-cancelled transaction, with a set of parameters, which you can process as you want to.	varchar(300)	Optional
hash	You need to compute a hash of all your parameters and pass that hash to ZenPay, for details about calculating hash refer Appendix 2. Note: the SALT will be provided by ZenPay separately. NEVER PASS SALT IN A FORM, DO NOT STORE SALT IN ANDROID APP APK or IPHONE APP package.	varchar(255)	Mandatory

11.3. Response Parameters

Response sent after complete amount is paid. This is a server to browser response

Parameter name	Description
transaction_id	A unique ID that can be used to trace the transaction uniquely within ZenPay. Transaction IDs are alphanumeric.
payment_mode	This tells the payment mode used by customer - example: "credit card", "debit card", "netbanking", etc.
Payment_channel	The actual payment channel - for example Visa, Master, Diners, HDFC Bank, MobiKwik, etc.
payment_datetime	Date and Time of this payment in "DD-MM-YYYY HH:MM:SS" format
response_code	Status of the transaction (return code). 0 signifies successful transaction. Non-zero values signify error.
response_message	Can have a value of "Transaction Successful" or "Transaction Failed" or “Transaction Cancelled”.
error_desc	Failure reason (if transaction is failed)
order_id	The same order_id that was originally posted by the merchant in the request.
amount	The same original amount that was sent by the merchant in the transaction request.
currency	This is the 3digit currency code (INR), it will be same value that was originally sent by merchant.
description	The same description that was originally sent by the merchant in the transaction request.
name	The same value that was originally sent by merchant
email	The same value that was originally sent by merchant
phone	The same value that was originally sent by merchant
address_line_1	The same value that was originally sent by merchant
address_line_2	The same value that was originally sent by merchant
city	The same value that was originally sent by merchant
state	The same value that was originally sent by merchant
country	The same value that was originally sent by merchant
zip_code	The same value that was originally sent by merchant
udf1	The same value that was originally sent by merchant
udf2	The same value that was originally sent by merchant
udf3	The same value that was originally sent by merchant
udf4	The same value that was originally sent by merchant
udf5	The same value that was originally sent by merchant
Cardmasked	The card number used by customer for making payment. This will NOT BE SENT by default, and is sent only in case the merchant has been explicitly approved to receive this information. Else this is always sent as null.
Hash	ZenPay calculates the hash using the same algorithm which was outlined earlier. Hence, the merchant needs to check whether this returned hash matches the calculated hash.

13. Appendix 2 - Hash calculation guide

12.1. How to Calculate Hash on API request

To calculate hash, you will need the salt provided by ZenPay.

Hashing generation algorithm

Following are the steps to calculate hash.

1. Create a | (pipe) delimited string called hash_data with first value as the salt.
2. Now sort the post fields based on their keys and create a | delimited string, for the fields with values.
3. Hash the hash_data string using SHA512 hashing algorithm and save the hash in secure_hash string
4. Convert the secure_hash string to upper case

Example PHP code to generate hash

```php
/**
 * @param array $parameters
 * @param string $salt
 * @param string $hashing_method
 * @return null|string
 */
function generateHashKey($parameters, $salt, $hashing_method = 'sha512')
{
    $secure_hash = null;
    ksort($parameters);
    $hash_data = $salt;
    foreach ($parameters as $key => $value) {
        if (strlen($value) > 0) {
            $hash_data .= '|' . trim($value);
        }
    }

    if (strlen($hash_data) > 0) {
        $secure_hash = strtoupper(hash($hashing_method, $hash_data));
    }

    return $secure_hash;
}
```

12.2. How to check the response Hash

It is important to make sure the response received from ZenPay is genuine, and to do so you will need to do a hash check on your server on receiving the response.

Every response received has a field called hash. Sometimes it is null, which means it in not important to check hash for the response, but if there is a hash present please perform hash check as described below and make sure integrity of the response received from ZenPay APIs.

To check hash, you will need the salt provided by ZenPay.

Hash checking algorithm

Example PHP code to check hash

```php
/**
 * @param string $salt
 * @param array $response_array
 * @return bool
 */
function responseHashCheck($salt, $response_array)
{

    /* If hash field is null no need to check hash for such response */
    if (is_null($response_array['hash'])) {
        return true;
    }

    $response_hash = $response_array['hash'];
    unset($response_array['hash']);

    /* Now we have response json without the hash */
    $calculated_hash = hashCalculate($salt, $response_array);

    return ($response_hash == $calculated_hash) ? true : false;
}

/**
 * @param string $salt
 * @param array $input
 * @return string
 */

function hashCalculate($salt, $input)
{
    /* Columns used for hash calculation, Donot add or remove values from $hash_columns
array */
    $hash_columns = array_keys($input);
    /*Sort the array before hashing*/
    sort($hash_columns);

    /*Create a | (pipe) separated string of all the $input values which are available
in $hash_columns*/
    $hash_data = $salt;
    foreach ($hash_columns as $column) {
        if (isset($input[$column])) {
            if (strlen($input[$column]) > 0) {
                $hash_data .= '|' . trim($input[$column]);
            }
        }
    }
    $hash = strtoupper(hash("sha512", $hash_data));
```

```
    return $hash;
}
```

Example PHP code to check hash if response is JSON

```
/**
 * @param $salt
 * @param $response_json
 * @return bool
 */
function responseHashCheck($salt, $response_array)
{

    /* If hash field is null no need to check hash for such response */
    if (is_null($response_array['hash'])) {
        return true;
    }

    $response_hash = $response_array['hash'];
    unset($response_array['hash']);
    $response_json = json_encode($response_array, JSON_UNESCAPED_SLASHES);

    /* Now we have response json without the hash */
    $calculated_hash = hashCalculate($salt, $response_json);

    return ($response_hash == $calculated_hash) ? true : false;
}

/**
 * @param $salt
 * @param $input_json
 * @return string
 */
function hashCalculate($salt, $input_json)
{

    /* Prepend salt with input json and calculate the hash using SHA512 */
    $hash_data = $salt . $input_json;
    $hash = strtoupper(hash('sha512', $hash_data));

    return $hash;
}
```

14. Appendix 3 - List of bank codes

Bankcode	Payment Mode	Payment Channel	Description
VISC	Credit Card	Visa	Visa Credit Card
MASC	Credit Card	Master	Master Credit Card
DINC	Credit Card	Diners	Diners Credit Card
AMXC	Credit Card	Amex	American Express Credit Card
RUPC	Credit Card	Rupay	Rupay Credit Card
VICC	Commercial Credit Card	Visa	Visa Commercial Credit Card
MACC	Commercial Credit Card	Master	Master Commercial Credit Card
VICI	International Credit Card	Visa	Visa International Credit Card
MACI	International Credit Card	Master	Master International Credit Card
AMXI	International Credit Card	Amex	American Express International Credit Card
DINI	International Credit Card	Diners	Diners International Credit Card
VISD	Debit Card	Visa	Visa Debit Card
MASD	Debit Card	Master	Master Debit Card
MAED	Debit Card	Maestro (non-SBI)	Maestro Debit Card (non-SBI)
MSED	Debit Card	Maestro (SBI)	Maestro Debit Card (SBI)
RUPD	Debit Card	Rupay	Rupay Debit Card
VIDI	International Debit Card	Visa	Visa International Debit Card
MADI	International Debit Card	Master	Master International Debit Card
IOBP	Debit Pin	Indian Overseas Bank	Indian Overseas Bank
SYDP	Debit Pin	Syndicate Bank	Syndicate Bank
AXIP	Debit Pin	AXIS Bank	AXIS Bank
ADBP	Debit Pin	Andhra Bank	Andhra Bank
UCOP	Debit Pin	UCO Bank	UCO Bank
SUBP	Debit Pin	Suryodaya Bank	Suryodaya Bank
ICIP	Debit Pin	ICICI Bank	ICICI Bank
IDFP	Debit Pin	IDFC Bank	IDFC Bank
SBIP	Debit Pin	State Bank Of India	State Bank Of India
CBIP	Debit Pin	Central Bank Of India	Central Bank Of India
ALLN	Netbanking	Allahabad Bank	Allahabad Bank NetBanking
ADBN	Netbanking	Andhra Bank	Andhra Bank
ADBM	Netbanking	Andhra Bank - Corporate	Andhra Bank - Corporate
AXIN	Netbanking	AXIS Bank	AXIS Bank NetBanking
BDNN	Netbanking	Bandhan bank	Bandhan Bank
BBKN	Netbanking	Bank of Bahrain and Kuwait	Bank of Bahrain and Kuwait

BBRM	Netbanking	Bank of Baroda - Corporate	Bank of Baroda Corporate Banking
BBRN	Netbanking	Bank of Baroda - Retail	Bank of Baroda Retail Banking
BOIN	Netbanking	Bank of India	Bank of India
BOMN	Netbanking	Bank of Maharashtra	Bank of Maharashtra
BCBN	Netbanking	Bassein Catholic Bank	Bassein Catholic Bank
BMBN	Netbanking	Bharatiya Mahila Bank	Bharatiya Mahila Bank
CANN	Netbanking	Canara Bank	Canara Bank
CSBN	Netbanking	Catholic Syrian Bank	Catholic Syrian Bank
CBIN	Netbanking	Central Bank Of India	Central Bank Of India
CITN	Netbanking	Citi Bank NetBanking	Citi Bank NetBanking
CUBN	Netbanking	City Union Bank	City Union Bank
CRPN	Netbanking	Corporation Bank	Corporation Bank
COSN	Netbanking	Cosmos Bank	Cosmos Bank
DBSN	Netbanking	DBS Bank	DBS Bank
DCBM	Netbanking	DCB Bank - Corporate	DCB Bank - Corporate Netbanking
DENN	Netbanking	Dena Bank	Dena Bank
DSHN	Netbanking	Deutsche Bank	Deutsche Bank
DCBN	Netbanking	Development Credit Bank	Development Credit Bank
DHNM	Netbanking	Dhanlakshmi Bank - Corporate Net Banking	Dhanalakshmi Bank - corporate
DHNN	Netbanking	Dhanlakshmi Bank	Dhanalakshmi Bank
FEDN	Netbanking	Federal Bank	Federal Bank
HDFN	Netbanking	HDFC Bank	HDFC Bank
ICIN	Netbanking	ICICI Bank	ICICI Netbanking
IDFN	Netbanking	IDFC Bank	IDFC Bank
ININ	Netbanking	Indian Bank	Indian Bank
IOBN	Netbanking	Indian Overseas Bank	Indian Overseas Bank
INDN	Netbanking	IndusInd Bank	IndusInd Bank
IDBN	Netbanking	IDBI	Industrial Development Bank of India
INGN	Netbanking	ING Vysya Bank	ING Vysya Bank
JAKN	Netbanking	Jammu and Kashmir Bank	Jammu and Kashmir Bank
JSBN	Netbanking	Janata Sahakari Bank	Janata Sahakari Bank
KJBN	Netbanking	Kalyan Janata Sahakari Bank	Kalyan Janata Sahakari Bank
KRKN	Netbanking	Karnataka Bank	Karnataka Bank
KRVN	Netbanking	Karur Vysya - Retail	Karur Vysya
KRVM	Netbanking	Karur Vysya - Corporate	Karur Vysya - Corporate Netbanking
KKBN	Netbanking	Kotak Mahindra Bank	Kotak Mahindra Bank

LVBN	Netbanking	Laxmi Vilas Bank	Laxmi Vilas Bank - Retail
LVBM	Netbanking	Laxmi Vilas Bank - Corporate	Laxmi Vilas Bank - Corporate
MSBN	Netbanking	The Mehsana Urban Co Op Bank Ltd	The Mehsana Urban Co Op Bank Ltd.
NKBN	Netbanking	NKGSB Bank	NKGSB Bank
OBCN	Netbanking	Oriental Bank of Commerce	Oriental Bank of Commerce
PMCN	Netbanking	Punjab & Maharashtra Coop Bank	Punjab & Maharashtra Coop Bank
PSBN	Netbanking	Punjab & Sind Bank	Punjab & Sind Bank
PNBN	Netbanking	Punjab National Bank - Retail	Punjab National Bank - Retail Banking
PNBM	Netbanking	Punjab National Bank - Corporate	Punjab National Bank-Corporate
RTNN	Netbanking	RBL Bank Limited	RBL Bank Limited
RTNM	Netbanking	RBL Bank Limited-Corporate Net Banking	RBL Bank Limited- Corporate Net Banking
SRSN	Netbanking	Saraswat Bank	Saraswat Bank
SVCN	Netbanking	Shamrao Vitthal Co-operative Bank	Shamrao Vitthal Co-operative Bank
SVCM	Netbanking	Shamrao Vitthal Co-operative Bank - Corporate	Shamrao Vitthal Co-operative Bank - Corporate
SOIN	Netbanking	South Indian Bank	South Indian Bank
SCBN	Netbanking	Standard Chartered Bank	Standard Chartered Bank
SBJN	Netbanking	State Bank of Bikaner and Jaipur	State Bank of Bikaner and Jaipur
SBHN	Netbanking	State Bank of Hyderabad	State Bank of Hyderabad
SBIN	Netbanking	State Bank of India	State Bank of India
SBMN	Netbanking	State Bank of Mysore	State Bank of Mysore
SBPN	Netbanking	State Bank of Patiala	State Bank of Patiala
SBTN	Netbanking	State Bank of Travancore	State Bank of Travancore
SYDN	Netbanking	Syndicate Bank	Syndicate Bank
TMBN	Netbanking	Tamilnad Mercantile Bank Ltd.	Tamilnad Mercantile Bank Ltd.
TSCN	Netbanking	Tamilnadu State Coop Bank	Tamilnadu State Coop Bank
TJSN	Netbanking	TJSB Bank	TJSB Bank
UCON	Netbanking	UCO Bank	UCO Bank
UBIM	Netbanking	Union Bank of India - Corporate	Union Bank - Corporate Netbanking

UBIN	Netbanking	Union Bank of India - Retail	Union Bank of India
UNIN	Netbanking	United Bank Of India	United Bank Of India
VIJN	Netbanking	Vijaya Bank	Vijaya Bank
YESN	Netbanking	Yes Bank	Yes Bank
ESFN	Netbanking	Equitas Bank	Equitas Small Finance Bank Limited
PNYN	Netbanking	PNB YUVA Bank	PNB YUVA Bank
KCCN	Netbanking	The Kalupur Commercial Cooperative Bank Limited	The Kalupur Commercial Cooperative Bank Limited
AIRN	Netbanking	Airtel Payment Bank	Airtel Payment Bank
BHAN	Netbanking	Bharat Bank	Bharat Bank
NBLN	Netbanking	Nainital Bank	Nainital Bank
DEMN	Netbanking	Demo Bank	Demo Netbanking
GPPN	Netbanking	GP Parsik Sahakari Bank	GP Parsik Sahakari Bank
AUSN	Netbanking	AU Small Finance Bank	AU Small Finance Bank
FESN	Netbanking	Federal Bank Scan And Pay	Federal Bank Scan And Pay
SUBN	Netbanking	Suryoday Small Finance Bank	Suryoday Small Finance Bank
VRBN	Netbanking	Varachha Co-Operative Bank	Varachha Co-Operative Bank
NEBN	Netbanking	North East Small Finance Bank	North East Small Finance Bank
IDBM	Netbanking	IDBI Bank - Corporate	Industrial Development Bank of India - Corporate
YESM	Netbanking	Yes Bank - Corporate	Yes Bank - Corporate
CRPM	Netbanking	Corporation Bank - Corporate	Corporation Bank - Corporate
ADCN	Netbanking	Ahmedabad District Co-operative Bank	Ahmedabad District Co-operative Bank
ESAN	Netbanking	ESAF Small Finance Bank	ESAF Small Finance Bank
BRLM	Netbanking	Barclays Corporate Banking	Barclays Corporate Banking
ZOBN	Netbanking	Zoroastrian Co-operative Bank	Zoroastrian Co-operative Bank
ABPN	Netbanking	Aditya Birla Payments Bank	Aditya Birla Payments Bank
ALLM	Netbanking	Allahabad Bank - Corporate	Allahabad Bank - Corporate
AXIM	Netbanking	AXIS Bank - Corporate	AXIS Bank - Corporate NetBanking

HSBE	EMI	HSBC BANK	HONG KONG AND SHANGHAI BANKING (HSBC BANK)
YESE	EMI	YES BANK	YES BANK
SCBE	EMI	STANDARD CHARTERED BANK	STANDARD CHARTERED BANK
AXIE	EMI	AXIS BANK	AXIS Bank Credit EMI
RBLE	EMI	RBL BANK	RBL BANK
HDFE	EMI	HDFC BANK	HDFC BANK
CITE	EMI	CITIBANK	CITIBANK
KKBE	EMI	KOTAK MAHINDRA BANK	KOTAK MAHINDRA BANK
INDE	EMI	INDUSIND BANK	INDUSIND BANK
AMXE	EMI	Amex	American Express EMI
ICIE	EMI	ICICI Bank	ICICI Bank EMI
SBIE	EMI	State Bank Of India	State Bank of India Credit Card
BOBE	EMI	Bank Of Baroda	Bank Of Baroda EMI
AXDE	EMI	AXIS BANK DEBIT EMI	AXIS Bank Debit EMI
KKBL	Direct EMI	Kotak Mahindra Bank	Kotak Mahindra Bank
BOBL	Direct EMI	Bank of Baroda	Bank of Baroda
ICIL	Direct EMI	ICICI Bank	ICICI Bank
BOIA	ATM Card	Bank of India	Bank of India ATM Card
PNBA	ATM Card	Punjab National Bank	Punjab National Bank ATM Card
BOBA	ATM Card	Bank of Baroda	Bank of Baroda ATM Card
BMBA	ATM Card	Bharatiya Mahila Bank	Bharatiya Mahila Bank ATM Card
BOMA	ATM Card	Bank of Maharashtra	Bank of Maharashtra ATM Card
CANA	ATM Card	Canara Bank	Canara Bank ATM Card
INDA	ATM Card	IndusInd Bank	IndusInd Bank ATM Card
IDFA	ATM Card	IDFC	IDFC ATM Card
IOBA	ATM Card	Indian Overseas Bank	Indian Overseas Bank ATM Card
KGBA	ATM Card	Kerala Gramin Bank	Kerala Gramin Bank ATM Card
LVBA	ATM Card	Laxmi Vilas Bank	Laxmi Vilas Bank ATM Card
PKGA	ATM Card	Pragathi Krishna Gramin Bank	Pragathi Krishna Gramin BankATM Card
RSCA	ATM Card	Rajasthan State Coop Bank	Rajasthan State Coop Bank ATM Card
SMCA	ATM Card	Shivalik Mercantile Co-Op Bank Limited	Shivalik Mercantile Co-Op Bank Limited ATM Card
UBIA	ATM Card	Union Bank of India	Union Bank of India ATM Card
ITZH	Cash Card	ITZ Cash Card	ITZ Cash card
ICSH	Cash Card	Icash Card	Icash card
DONH	Cash Card	DONE Cash Card	DONE Cash Card
PWMH	Cash Card	PayWorld Cash Card	PayWorld Cash Card
PTMW	Wallet	Paytm	Paytm wallet

MBKW	Wallet	Mobikwik	Mobikwik Wallet
OXIW	Wallet	Oxigen	Oxigen Wallet
MRPW	Wallet	mRupee	mRupee Wallet
JIOW	Wallet	Reliance JioMoney	Reliance Jio Money wallet
TMWW	Wallet	The Mobile Wallet	The Mobile Wallet
PYCW	Wallet	Paycash	Paycash wallet
OLAW	Wallet	Ola Money	Ola Money Wallet
JNCW	Wallet	Jana Cash	Jana Cash
ATLW	Wallet	Airtel Money	Airtel Money
PNBW	Wallet	PNB Wallet	PNB Wallet
FRCW	Wallet	FreeCharge	FreeCharge Wallet
EZCW	Wallet	ezeClick	ezeClick Wallet
VDFW	Wallet	mPesa Wallet	mPesa - Vodafone Wallet
SBIW	Wallet	SBI Buddy	SBI Buddy
AMPW	Wallet	Amazon Pay	Amazon Pay
PZAW	Wallet	Payzapp	Payzapp wallet
IDMW	Wallet	Idea Money	Idea Money
CITW	Wallet	Citibank Reward Points	Citibank Reward Points
YPAW	Wallet	YpayCash	YpayCash
ZIPW	Wallet	ZipCash	ZipCash
YESW	Wallet	Yes Pay	Yes Pay
BKSW	Wallet	bKash Wallet	bKash Wallet
DCBW	Wallet	DCB Cippy	DCB Cippy
PHPW	Wallet	PhonePe	PhonePe wallet
VISP	Prepaid Card	Visa	Visa Prepaid Card
MASP	Prepaid Card	Master	Master Prepaid Card
RUPP	Prepaid Card	Rupay	Rupay Prepaid Card
ZEMF	Cardless EMI	ZestMoney	ZestMoney
INCF	Cardless EMI	Instacred	Instacred
EASF	Cardless EMI	EarlySalary	EarlySalary
BHQR	Bharat QR	Bharat QR	Bharat QR Payments(Dynamic)
BSQR	Bharat QR(Static)	Bharat QR(Static)	Bharat QR Payments(Static)
ECLV	E-Collect	E-Collect	E-Collect(Virtual Accounts)
UPIU	UPI	Unified Payments Interface	Unified Payments Interface

15. Appendix 4 - List of error codes

Error Numeric Code	Error Code	Error Description
0	SUCCESS	Transaction successful
1000	FAILED	Transaction failed
1001	INVALID-API-KEY	The api key field is incorrect
1002	INVALID-LIVE-MODE-ACCESS	The live mode access is not allowed
1003	INVALID-ORDER-ID-FIELD	The order id field should to be unique
1004	ORDER-ID-FIELD-NOT-FOUND	The order id field is not found
1005	INVALID-AUTHENTICATION	Invalid authentication at bank
1006	WAITING-BANK-RESPONSE	Waiting for the response from bank
1007	INVALID-INPUT-REQUEST	Invalid input in the request message
1008	TRANSACTION-TAMPERED	Transaction tampered
1009	DECLINED-BY-BANK	Bank Declined Transaction
1010	INVALID-AMOUNT	Amount cannot be less than 1
1011	AUTHORIZATION-REFUSED	Authorization refused
1012	INVALID-CARD	Invalid Card/Member Name data
1013	INVALID-EXPIRY-DATE	Invalid expiry date
1014	DENIED-BY-RISK	Transaction denied by risk
1015	INSUFFICIENT-FUND	Insufficient Fund
1016	INVALID-AMOUNT-LIMIT	Total Amount limit set for the terminal for transactions has been crossed
1017	INVALID-TRANSACTION-LIMIT	Total transaction limit set for the terminal has been crossed
1018	INVALID-DEBIT-AMOUNT-LIMIT	Maximum debit amount limit set for the terminal for a day has been crossed
1019	INVALID-CREDIT-AMOUNT-LIMIT	Maximum credit amount limit set for the terminal for a day has been crossed
1020	MAXIMUM-DEBIT-AMOUNT-CROSS	Maximum debit amount set for per card for rolling 24 hrs has been crossed
1021	MAXIMUM-CREDIT-AMOUNT-CROSS	Maximum credit amount set for per card for rolling 24 hrs has been crossed
1022	MAXIMUM-TRANSACTION-CROSS	Maximum transaction set for per card for rolling 24 hrs has been crossed
1023	HASH-MISMATCH	Hash Mismatch
1024	INVALID-PARAMS	Invalid parameters
9999	UNKNOWN-ERROR	Unknown error occurred
1025	INVALID-BANK-CODE	Invalid bank code
1026	INVALID-MERCHANT	Merchant is not active

1027	INVALID-TRANSACTION	Invalid transaction
1028	TRANSACTION-NOT-FOUND	Transaction not found
1029	TRANSACTION-TERMINATED	Transaction terminated
1030	TRANSACTION-INCOMPLETE	Transaction incomplete
1031	AUTO-REFUNDED	Transaction auto refunded
1032	REFUNDED	Transaction refunded
1033	SINGLE-TRANSACTION-LOWER-LIMIT-CROSS	The amount provided is less than transaction lower limit
1034	SINGLE-TRANSACTION-UPPER-LIMIT-CROSS	The amount provided is more than transaction upper limit
1035	TRANSACTION-DAILY-LIMIT-CROSS	The daily transaction limit is exceeded for the merchant
1036	TRANSACTION-MONTHLY-LIMIT-CROSS	The monthly transaction limit is exceeded for the merchant
1037	DAILY-TRANSACTION-NUMBER-CROSS	The daily transaction number is exceeded for the merchant
1038	MONTHLY-TRANSACTION-NUMBER-CROSS	The monthly transaction number is exceeded for the merchant
1039	INVALID-REFUND-AMOUNT	The refund amount is greater than transaction amount
1040	INVALID-CVV	Invalid Card Verification Code
1041	AUTO-REFUNDED-TNP	Transaction is auto refunded by TnP
1042	FAILED-NO-RESPONSE	Transaction failed as there was no response from bank
1043	TRANSACTION-CANCELLED	Transaction cancelled
1044	UNAUTHORIZED	Unauthorized
1045	FORBIDDEN	Forbidden Access
1046	TRANSACTION-ALREADY-CAPTURED	Transaction already captured
1047	AUTHORIZED	Transaction authorized
1048	CAPTURED	Transaction captured
1049	VOIDED	Transaction voided
1050	NO-RECORD-FOUND	No data record found for the given input
1051	ACQUIRER-ERROR	Error occurred at the bank end
1052	INVALID-EMAIL	Invalid Email ID
1053	INVALID-PHONE	Invalid phone number
1055	SEAMLESS-NOT-ALLOWED	Seamless payment not allowed
1054	SESSION-TIMEOUT	Session expired. Please go back and try again.
1056	INVALID-VPA	Virtual Payee Address is invalid
1057	3DS-FAILED	3D Secure authentication failed
1058	ACCOUNT-BLOCKED	Bank Account is blocked
1059	ACQUIRER-MAX-TRANSACTION-LIMIT	Transaction amount limit has exceeded
1060	ACQUIRER-MIN-TRANSACTION-LIMIT	The amount provide is less than minimum transaction amount allowed

1061	ACQUIRER-VELOCITY-LIMIT	Transaction frequency limit has exceeded
1062	CARD-EXPIRED	Card has expired
1063	HOTLISTED	Stolen or Lost Card
1064	INACTIVE-CARD	The card is inactive
1065	INCORRECT-PIN	Incorrect PIN
1066	INSUFFICIENT-BALANCE	Insufficient Balance
1068	INVALID-BIN	Bin not found
1069	INVALID-CARDHOLDER-NAME	Invalid card holder name
1070	INVALID-CARD-INFO	Invalid brand or bin range not enabled
1071	INVALID-INPUT-DATA	Invalid input data
1072	ISSUER-DECLINE	Payment decline by bank
1073	ISSUER-DISALLOWED	Transaction not allowed by the issuer
1074	ISSUER-ERROR	Payment processing failed due to error at bank
1075	ISSUER-LIMIT-DECLINE	Exceeds withdrawal frequency or count limit
1076	MAX-RETRIES	Maximum number of PIN retries exceeded
1077	OTP-LIMIT-EXCEEDED	OTP validation attempts limit exceeded
1078	PIN-ATTEMPTS-EXCEEDED	PIN attempts limit exceeded
1079	RESTRICTED-CARD	Restricted card
1080	SUSPECTED-FRAUD	Suspected fraud
1081	TIMEOUT	Payment was not completed on time
1082	UPI-PIN-NOT-SET	UPI PIN is not set
1083	UPI-REQUEST-EXPIRED	UPI request expired
1084	USER-BACK-REFRESH	User pressed refresh button
1085	ACQUIRER-DOWN	Payment acquirer is down
1086	TOO-MANY-RECORDS	Response contains too many records, use pagination
1087	NOT-ENOUGH-PARAMETERS	Not enough parameters to get response, add one or more optional parameters
1088	TRANSACTION-IN-PROCESS	We are processing your transaction

16. Appendix 5 – Currency Codes

Currency	Currency Code
INR	Indian Rupee
USD	United States Dollar
EUR	Euro
SGD	Singapore Dollar
HKD	Hong Kong Dollar
GBP	British Pound
CAD	Canadian Dollar
AUD	Australian Dollar
AED	Arab Emirates Dinar