PAYMENT GATEWAY

Invoicing & Payments API

Contact

Tel: +91 9526 790 111, 222, 333
Email: mail@sparkitts.com
Website:
www.sparkitpay.com

Document version 1.5

1. OVERVIEW

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

Through SparkitPay, 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 debit

SparkitPay also offers you a business UI (https://biz.SparkitPay.in) 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.

Through this interface, you can also cancel past invoices (and in some cases, past transactions), manage your payables, vendor payments, set split ratios for vendor payments, process refunds, etc. This online interface can be accessed through https://biz.SparkitPay.in.

2. PAYMENT REQUEST API

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

2.1 Steps for Integration

- Initially your transaction limit would be set to a small amount and the said limit will be increased after a few successful test transactions.

- You need to submit a **POST REQUEST** to our server, at the below mentioned URL **https://biz.SparkitPay.in/v1/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 submit your transaction request to SparkitPay, we assign a transaction ID to you.

- The customer is necessarily re-directed to SparkitPay's payment page. After the customer makes the payment through SparkitPay (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 SparkitPay 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.

2.2 Parameters to be POSTed in Transaction Request

URL:

https://biz.SparkitPay.in/v1/paymentrequest

Parameter Name	Description	Data type	Optional / Mandatory
api_key	SparkitPay 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(40)	Mandatory
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.	Unsigned integer	Mandatory
Mode	This is the payment mode (TEST or LIVE are valid values)	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 has to be IND	Varchar(50)	Mandatory
Country	Customer country has to be IND	Varchar(50)	Mandatory
zip_code	Customer zip code	Varchar(20)	Mandatory
udf1	User defined field	Varchar(300)	Optional
udf2	User defined field 2		
udf3	User defined field 3		
udf4	User defined field 4		
udf5	User defined field 5		
return_url	Return URL success – SparkitPay will make a POST request to this URL after successful transaction, with a set of parameters, which you can process as you want to.		
Hash	You need to compute a hash of all your parameters and pass that hash to SparkitPay, using the following mechanism. If a value is not being passed, exclude it from the calculation : toUpper(sha512(SALT|address_line_1|address_line_2|amount|api_key|city|country|currency|description|email|mode|name|order_id|phone|return_url|state|udf1|udf2|udf3|udf4|udf5|zip_code)) Note: the SALT will be provided by SparkitPay separately. NEVER PASS SALT IN A FORM		

Here is a sample HTML page that you can use to test the API before actually doing the integration:

```
<html>
<head> </head>
 <body>
<form action="https://biz.sparkitpay.com/v1/paymentrequest"
name="payment" method="POST">
<input type="hidden" value="f141a2b3c-12ab-41s0-bd4e-de123456d4ff" name="api_key"/>
<input type="hidden" value="http://www.yoursite.com/payment_return_url"
name="return_url"/>
 <input type="hidden" value="TEST" name="mode"/>
<input type="hidden" value="00568" name="order_id"/>
<input type="hidden" value="100" name="amount"/>
<input type="hidden" value="INR" name="currency"/>
<input type="hidden" value="Some details about the transaction" name="description"/>
<input type="hidden" value="karmendra" name="name"/>
<input type="hidden" value="your_email_id@example.com" name="email"/>
```

```
<input type=”hidden” value=”9900261104” name=”phone”/>
<input type=”hidden” value=”245. 2nd Main” name=”address_line_1”/>
<input type=”hidden” value=”RT Nagar” name=”address_line_2”/>
<input type=”hidden” value=”Bengaluru” name=”city”/>
<input type=”hidden” value=”Karnataka” name=”state”/>
<input type=”hidden” value=”560037” name=”zip_code”/>
<input type=”hidden” value=”India” name=”country”/>
<input type=”hidden” value=”” name=”udf1”/>
<input type=”hidden” value=”” name=”udf2”/>
<input type=”hidden” value=”” name=”udf3”/>
<input type=”hidden” value=”” name=”udf4”/>
<input type=”hidden” value=”” name=”udf5”/>
<input type=”hidden” value=”” name=”hash”/>
<button style=”color: #fff;background-color: #5cb85c;border-color: #4cae4c;display: inline-block;
padding: 6px 12px; border: 1px solid transparent;”> SUBMIT </button>
</form>
</body>
</html>
```

2.3 Response Parameters returned by SparkitPay

Parameter name	Description
transaction_id	A unique ID that can be used to trace the transaction uniquely within SparkitPay. 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 “YYYY-MM-DD 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_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	SparkitPay 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. toUpper(sha512(SALT|address_line_1|address_line_2|amount|api_key|city|country|currency|description|email|mode|name|order_id|payment_method|phone|response_code|response_message|return_url|state|transaction_id|udf1|udf2|udf3|udf4|udf5|zip_code))
error_desc	Failure reason (if transaction is failed)

Note: Consider, a failed response sent from SparkitPay server to your server via a user's browser and user chares the response code to Success, even though transaction is failed it will now show Success on your website. To make sure the transaction response is same as what SparkitPay server sent please check the hash before considering the transaction response success or failure.

3. PAYMENT STATUS API

Payment Status by merchant's order id

SparkitPay 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:

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

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

URL:

https://biz.sparkitpay.com/v1/paymentstatus

3.1.1 Parameters to be POSTed

Parameter Name	Description
api_key	Unique merchant identifier key provided by SparkitPay.
Hash	This is a mandatory parameter, and is required for cross-verification. You need to calculate the hash as follows: **sha512(SALT|api_key|order_id)** **Note: NEVER PASS SALT IN A FORM**
order_id	This is the original order_id that you passed while making the transaction request.

3.1.2 Parameters in the Response

transaction_id	A unique ID that can be used to trace the transaction uniquely within SparkitPay. Transaction IDs are alphanumeric.
payment_method	This tells the payment method used by customer – example: “credit card”, “debit card”, “netbanking”, etc.
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_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.
Description	The same description that was originally posted by the merchant in the request.

Hash	SparkitPay calculates the hash using the same algorithm which was outlined earlier. Hence, the merchant needs to check whether this returned hash matches the hash that was originally passed.
error_desc	Failure reason (if transaction is failed)

3.2 Payment Status by Id

URL: https://biz.sparkitpay.com/v1/paymentstatusbyid

SparkitPay 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:

1. The response might not reach you due to network issues or other problems such as user clicking refresh button on their browser, etc.
2. This also protects against any tampering, since you have a second fall back check here.

Parameters to be POSTed:

Parameter name	Description
api_key	Unique merchant identifier key provided by SparkitPay.
hash	This is a mandatory parameter, and is required for cross-verification. You need to calculate the hash as follows: sha512(SALT|api_key|order_id) Note: NEVER PASS SALT IN A FORM
Id	This is the original id that you passed while making the transaction request.
type	Values in this field could be: ORDERID/TNPID/UUID

Parameters in the Response:

Parameter name	Description
transaction_id	A unique ID that can be used to trace the transaction uniquely within SparkitPay. 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 “YYYY-MM-DD 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 3 digit 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	SparkitPay 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.

Response from this API will be in JSON format:

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


```
"data": {
  "id": "8597",
```

```
    "invd_id": 8597,
    "order_id": "",
    "address_line_1": "",
    "address_line_2": "",
    "amount": "4.00",
    "city": "",
    "country": "",
    "currency": "",
    "description": "Web Payment for 8597",
    "email": "Sharathkumar.harishi@gmail.com",
    "name": "Sharathkumar",
    "payment_datetime": "2016-10-20 12:29:11",
    "payment_method": "wallet",
    "phone": "9535640653",
    "response_code": "0399",
    "response_message": "Invalid authentication at bank",
    "return_url": "",
    "state": "",
    "transaction_id": "ETRQ7530",
    "udf1": "",
    "udf2": "",
    "udf3": "",
    "udf4": "",
    "udf5": "",
    "zip_code": "",
    "hash": "8E1AA522B653B408602648E3F36ACBFC1E6D6C7A0BA692BE8D
D22390CF1280F18F13F9419588C99BBE0C0C4D2DB3856A452BDB
1E2AC771A39AF02163C6C51582"
  }

}
```

data – successful response will have "**data**" tag.

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 GEN-INVALID-TRANSACTION Transaction aborted by user or bank The transaction request is not found GEN-INVALID-PARAMS The id field is not found
998	GEN-INVALID-PARAMS The id field is required. The type field is required. The hash field is required. The selected type is invalid.

data – successful response will have "**data**" tag.

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.

4. SPLIT API

4.1 SPLIT PAYMENT REQUEST API

(4.1 deprecated, refer section 4.2)

4.1.1 Split Payment Request Overview

SparkitPay provides an API for split payment requests. These are payment requests where the money paid by the customer is distributed among the merchant and one or more vendors. The distribution is defined by the merchant, through this API. The prerequisites to call this API are as follows:

- The merchant should have been granted access to this API. Please contact your SparkitPay relationship manager to obtain access.
- The vendor list (to whom the payment is being split) should be predefined in the SparkitPay system, either via vendor APIs (see subsequent sections), or manually via the SparkitPay UI.
- The split information that is passed via this API should be a valid JSON string per RFC 4627.
- The merchant's IP address should be provided to us beforehand, and that IP address would be whitelisted by SparkitPay. Split payment requests (and more crucially, vendor addition/modification requests) would be permitted only from the designated IP address.

4.1.2 Parameters to Split Payment Request API

URL:
https://biz.sparkitpay.com/v1/splitpaymentrequest

Parameter Name			
api_key	SparkitPay would assign a unique 40-digit merchant key to you. Thiskeyisexclusivetoyour business/login account. Ifyouhavemultipleloginaccounts,therewill necessarilybe onedifferentapi_keyperloginaccount that is assigned toyou.	varchar(40)	Mandatory
Mode (TEST/LIVE)	For testing purpose TEST mode option is provided (you willbe redirected totest payment page), Incase of LIVEmode optionlivetransactionswillhappen.	varchar(4)	Optional
order_id	This is your (merchant) reference number. It must be unique foreverytransaction.Wedoperformavalida- tion atourendanddonotallowduplicateorder_ids for the same merchant.	Unsigned integer	Mandatory
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 chargedfor.	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)	Optional
state	Customer State	Varchar(50)	Optional
country	Customer country	Varchar(50)	Optional
zip_code	Customer zip code	Varchar(20)	Optional
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
return_url	Return URL– SparkitPay will make a POST requesttothis URLaftersuccessfultransaction,witha setofparameters, whichyoucanprocessasyouwant to.	Varchar(300)	
hash	You need to compute a hash of all your parameters and pass that hash to SparkitPay, using the following mechanism: toUpper (sha512 (SALT |address_line_1| address_line_2|amount|api_key|city| country|currency|description|email| mode|name|order_id|phone|return_url| state|udf1|udf2|udf3|udf4|udf5|zip_code))	Varchar(200)	

	Note: the SALT will be provided by SparkitPay separately. Please ensure that the concatenated string (upon which the hash will be computed) is 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 address_line_2 do not include that in hash calculation. Note: please do NOT include the columns "split_info" and "split_enforce_strict" in the hash calculation. Note: NEVER PASS SALT IN A FORM		
Split_info	This should be a valid JSON string which contains the split information (vendor-product split). This is explained in greater detail in subsequent portions of the document	Longtext (JSON)	Optional
split_enforce_strict	This can take only two values: "y" or "n". If "y" is passed here, SparkitPay will do thorough validation of the input split information. Any failure in validation (for example missing or inactive vendor) will cause the entire payment request to be aborted. If "n" is passed here, SparkitPay will process the payment regardless of errors in the split information and will credit the entire proceeds to merchant directly, ignoring the split instructions.	Varchar(1)	Mandatory

4.1.3 Providing Split Information

The parameter split_info should contain information about the split. The following format should be strictly followed while passing this parameter, wherein each line item detail is provided to us.
For example, assume that your customer has selected three products to purchase from your website and has checked out: a refrigerator, a mobile phone and an LCD TV. The following information would be provided to us:

```
{
  "line1":{
    "item":" Refrigerator",
    "price":"25000",
    "vendor_code":"XXYA99",
    "vendor_amount":"5000"
  },

  "line2":{
    "item":"Smartphone",
    "price":"14000",
    "vendor_code":"XXYA99",
    "vendor_amount":"13500"
  },
```

```
 "line3":{
    "item":"LCD TV 42\"",
    "price":"30000",
    "vendor_code":"YYHA98",
    "vendor_percent":"20",
  }
}
```

Each element containing the line item should contain the following format:

```
"line1":{
    "item":"Refrigerator",
    "price":"25000",
    "vendor_code":"XXYA99",
    "vendor_amount":"5000"
  }
```

In the above example, line item 1 contains an order for “Refrigerator” (“item”), having a price of Rs. 25000 (“price”) out of which Rs. 5000 needs to be paid to the vendor (“vendor_amount”).

Note: double quotes passed in this JSON will need to be escaped with \ character. The line item elements can also be passed with a vendor percent as follows:

```
"line3":{
    "item":"LCD TV 42\"",
    "price":"30000",
    "vendor_code":"YYHA98",
    "vendor_percent":"20",
  }
```

Instead of vendor_amount (which is an absolute number), you can also pass vendor_percent (which is a percentage of the price of that line item) as shown above.

Note: in case both, vendor_percent and vendor_amount are passed, vendor_amount always takes precedence.

In the above case, the payment would be distributed as follows:

Item	Price	Amount to vendor	Amount to merchant
Refrigerator	₹ 25,000	₹ 5,000	₹ 20,000
Smartphone	₹ 14,000	₹ 13,500	₹ 500
LCD TV 42"	₹ 30,000	₹ 6,000	₹ 24,000

Amount charged to customer	₹ 69,000
Aggregated amount to merchant	₹ 44,500
Aggregated amount to vendor YYHA98	₹ 6,000
Aggregated amount to vendor XXYA99	₹ 18,500

Note: The above example is illustrative in nature and does not cover SparkitPay's TDR deduction. This is covered in the subsequent section.
If some line items have no vendor split and the entire amount needs to be passed to merchant, it is permissible to exclude the vendor information from some of the elements. For example:

```
{
  "line1":{
    "item":" Refrigerator",
    "price":"25000",
  },

  "line2":{
    "item":"Smartphone",
    "price":"14000",
    "vendor_code":"XXYA99",
    "vendor_amount":"13500"
  },

  "line3":{
    "item":"LCD TV 42\"",
    "price":"30000",
    "vendor_code":"YYHA98",
    "vendor_percent":"20",
  }
}
```

In the above example, there is no vendor defined for line1. As such, the entire line1 total (₹ 25,000) will be credited to merchant whereas the conventional split logic will be followed for line2 and line3.

If any of the specified vendors are not present in SparkitPay system, or are inactive/disapproved, the following two scenarios may occur:

- Split_enforce_strict = 'y': in this case, we will return an error and decline the transaction.
- Split_enforce_strict = 'n': in this case, we will treat bad vendor data as equivalent to missing vendor data/vendor not supplied, and will credit that line item completely to merchant.

The same methodology is followed in case of other data errors such as "vendor split amount > purchase amount".

4.1.4 TDR Deduction in Split Payment

"percentage > 100", and other similar errors.
In a non-split conventional payment, the TDR + applicable service tax is deducted from the amount paid by customer and the remainder is credited to the merchant's registered account. In a split payment, the default SparkitPay setting is to deduct the TDR from the total amount paid by customer, and deduct that

payout amount from the amount due to merchant. In other words, the TDR is borne entirely by merchant in the default mode of operation.

For example, let us assume the following split scenario:

```
{
  "line1":{
    "item":" Item1",
    "price":"300",
    "vendor_code":"XXYA99",
    "vendor_amount":"50"
  },

  "line2":{
    "item":"Item2",
    "price":"300",
    "vendor_code":"XXYA99",
    "vendor_amount":"100"
  },

  "line3":{
    "item":"Item3",
    "price":"400",
    "vendor_code":"YYHA98",
    "vendor_percent":"20",
  }
}
```

In the above scenario, the following are the relevant numbers:

Grand total (amount paid by customer)	₹ 1,000
Transaction charges (assuming TDR of 2% + 14.5% ST on the TDR amount)	TDR = (2/100)*1,000 = ₹ 20 ST = (14.5/100)*20 = ₹ 2.9 Total transaction charge = 20+2.9 = ₹ 22.9
Amount to vendor XXYA99	₹ 150
Amount to vendor YYHA98	₹ 80
Amount to merchant	(1000 – (150+80)) – 22.9 = ₹ 747.1

Hence, the default settings in SparkitPay charge the TDR exclusively to merchant. If you would like to prorate the TDR among merchant and vendor(s), or add a convenience fee to the end-user, etc. please contact your relationship manager who will assist you in configuring your system appropriately

4.1.5 Response returned by SparkitPay

Parameter Name	Description
transaction_id	AuniqueIDthatcanbeusedtotracethetransactionuniquely within SparkitPay. Transaction IDs are alphanumeric.

payment_method	Thistellsthepaymentmethodusedbycustomer–example: “credit card”, “debit card”, “netbanking”, etc.
response_code	Statusofthetransaction(returncode).0signifiessuccessful transaction. Non-zero values signify error.
response_message	Can have a value of “success” or “failure”.
order_id	Thesameorder_idthatwasoriginallypostedbythemerchant in the request.
Amount	Thesameoriginalamountthatwassentbythemerchantin the transaction request.
Description	Thesamedescriptionthatwasoriginallysentbythemerchant in the transaction request.
Name	Thesamevaluethat was originallysent by merchant
Email	Thesamevaluethat was originallysent by merchant
phone	Thesamevaluethat was originallysent by merchant
address_line_1	Thesamevaluethat was originallysent by merchant
address_line_2	Thesamevaluethat was originallysent by merchant
city	Thesamevaluethat was originallysent by merchant
state	Thesamevaluethat was originallysent by merchant
Country	Thesamevaluethat was originallysent by merchant
zip_code	Thesamevaluethat was originallysent by merchant
udf1	Thesamevaluethat was originallysent by merchant
udf2	Thesamevaluethat was originallysent by merchant
udf3	Thesamevaluethat was originallysent by merchant
udf4	Thesamevaluethat was originallysent by merchant
udf5	Thesamevaluethat was originallysent by merchant
hash	SparkitPay calculates the hash using the same algorithm which was outlined earlier. Hence, the merchant needs to check whether this returned hash matches the hash that was originally passed.
error_desc	Failure reason (if transaction is failed)

4.2 Split transaction before settlement API

URL: http://biz.SparkitPay.in/v1/splitsettlement

Request Parameters:

Parameter Name	Description
api_key	The unique key provided to the merchant
order_id	The order id of the transaction
split_data	The json format data contains vendor_code and vendor_percent
Hash	strtoupper(hash(‘sha512’, salt|api_key|order_id|split_data))

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

```
[
  {
    "vendor_code": "XYZ",
    "vendor_percent": 10
  },
  {
    "vendor_code": "ABC",
    "vendor_percent": 60
  }
]
```

Response Parameters:

The response will be in json format as show below: In

case of success,

```
'data' => [
   'code'    => 'SUCCESS',
   'message' => 'The request was successful',
],
```

In case of error,

```
'error' => [
   'code'    => '221',
   'message' => 'GEN-INVALID-VENDOR - The vendor is not found'
],
```

List of error codes and corresponding messages

Code	Message
221	GEN-UNAUTHORIZED The api key field is incorrect The hash key field is invalid GEN-INVALID-TRANSACTION The transaction request is not found. GEN-ORDER-ID-NOT-FOUND The order id is not found.
998	GEN-INVALID-PARAMS The api_key field is required. The order_id field is required. The hash field is required. The split_data is required.

5. HOST-to-HOST SPLIT API

The method outlined in *section 4.3* ("providing split information") can be used to provide split information as part of the payment request, in one single step. However, for merchants who are sensitive about exposing their split information to a potentially malicious end-customer, a separate methodology has been provided to provide SparkitPay with the split information. This method relies on a server-to-server (or also called host-to-host) communication wherein merchant's servers directly communicate with SparkitPay's servers in a separate transaction.

Note: Irrespective of whether split_info was provided along with `the original` payment transaction, any split information provided via a host-to-host call will override and overwrite the old split information. In other words, the host-to-host call always takes precedence over information provided in the original payment transaction.

There are two ways in which the split information can be provided to merchant via host-to-host calls:

- Merchant providing split info to SparkitPay: merchant calls a designated SparkitPay API for this purpose.
- SparkitPay obtaining split information from merchant: SparkitPay calls a designated API developed by merchant specifically for this purpose.

5.1 Merchant Providing Split Info to SparkitPay

URL:

https://biz.sparkitpay.com/v1/splitinfoh2h

This API allows the merchant to provide split information for prior successful transactions. The merchant can call this API at any time, provided the following conditions are met:

- The order_id passed as parameter should be a valid order ID for the given merchant, which SparkitPay has a record of.
- The transaction referenced by the said order_id should be a successful transaction
- The transaction referenced by the said order_id should have been created by the "splitpaymentrequest" API and NOT the conventional API.
- The settlement initiation for the transaction referenced by the said order_id should not have happened yet. In other words, if the settlement for the transaction was already initiated at the time this API is called, SparkitPay will not be able to honour this split information request. Typically, SparkitPay performs a settlement compute in a batch at 12:00 AM every morning.

➢ A global business parameter called "split_info_breathing_time_h2h" has been provided, which can be set to a number. The settlement compute job will then, not consider any transaction requests in the settlement compute, up to those many seconds in the past. For example, if a

transaction (with split_flag = 'y') occurs at 11:59:59 AM and consider that the parameter is set to 600 seconds. The settlement batch job (which kicks off at 12 AM) will consider split transactions only till 11:50 AM, hence giving up to 600 seconds (10 minutes) of breathing time for the merchant to call the splitinfoh2h API for the said API.

5.1.1 Parameters POSTed by Merchant

URL:

https://biz.sparkitpay.com/v1/splitinfoh2h

Parameter Name	Description
api_key	Unique merchant identifier key provided by SparkitPay.
Hash	This is a mandatory parameter, and is required for cross-verification. You need to calculate the hash as follows: SHA512 (SALT | api_key | order_id) Note: NEVER PASS SALT IN A FORM
order_id	This is the original order_id that the merchant passed while making the transaction request.
split_info	A JSON field which contains the details of split for the given order_id.
split_enforce_strict	This takes on two possible values – 'y' or 'n'. If 'y' is passed here, SparkitPay will do thorough validation of the input split information. Any failure in validation (for example missing or inactive vendor) will cause the settlement to be held in abeyance (for this particular order_id). If 'n' is passed here, SparkitPay will process the settlement regardless of errors in the split information and will credit the entire proceeds to merchant directly, ignoring the split instructions.

5.2 SparkitPay obtaining Split Information from Merchant

URL:

<Merchant defined URL>

A global business parameter called "split_info_callback_url_h2h" has been provided, which can be set to any merchant-specified URL. The following conditions need to be met:

- The URL must be an https URL. Plain http URLs will not be accepted.
- The URL MUST accept the following three parameters: api_key, order_id and hash.
- The URL should strictly accept a POST request only.
- The merchant should compute the hash at his end and verify that the hash matches with the value sent by SparkitPay. Any sudden mismatch during course of operations should immediately be reported to SparkitPay.

- The merchant should compute hash while returning the information to SparkitPay, and the same would be verified by SparkitPay.
- The merchant should always return split information in the specified JSON format, which is further described in subsequent section.

5.2.1 Parameters POSTed by SparkitPay to merchant-defined URL

URL:

https://<merchantdefinedcallbackurl>/

Parameter Name	Description
api_key	Unique merchant identifier key provided by SparkitPay.
Hash	This is a mandatory parameter, and is required for cross-verification. You need to calculate the hash as follows: SHA512 (SALT|api_key|order_id) Note: NEVER PASS SALT IN A FORM
order_id	This is the original order_id that you passed while making the transaction request.

5.2.3 Value returned by merchant-defined URL to SparkitPay

Expected return information from merchant to SparkitPay is a JSON string.
The format and rules that need to be followed for this JSON are exactly the same as outlined in section 4.3 (providing split information). An example of such an output JSON is provided below. For a more comprehensive explanation, please refer to section 4.3.

```
{
“line1”:{
“item”:” Refrigerator”,
“price”:”25000”,
“vendor_code”:”XXYA99”,
“vendor_amount”:”5000”
},
“line2”:{
“item”:”Smartphone”,
“price”:”14000”,
“vendor_code”:”XXYA99”,
“vendor_amount”:”13500”
},
“line3”:{
“item”:”LCD TV 42\””,
“price”:”30000”,
“vendor_code”:”YYHA98”,
“vendor_percent”:”20”,
}
}
```

6. VENDOR API

6.1 Add Vendor API

URL:

https://biz.sparkitpay.com/v1/addvendor

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

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

Parameter Name	Description	Data type	Optional / Mandatory
api_key	SparkitPay would assign a unique 40-digit merchant key toyou.Thiskeyisexclusiveto your business/login account.Ifyouhave multiplelogin accounts,therewill necessarilybeonedifferentapi_keyperlogin account that is assigned toyou.	varchar(40)	Mandatory
vendor_code	This is the vendor code that you wish to add in the SparkitPay system. This has to be unique. Aplhanumeric 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. Optional.	Varchar(300)	Optional
account_name	Account holder name (of the vendor bank account)	Varchar(300)	Mandatory
account_number	Account number of the vendor.	Varchar(50)	Mandatory
ifsc_code	IFSC code of the vendor’s bank.	Varchar(50)	Mandatory
bank_name	Bank name of the vendor’s bank.	Varchar(200)	Optional
vendor_pan	PAN number of the vendor	Varchar(10)	Optional
bank_branch	Bank branch of the vendor’s bank.	Varchar(300)	Optional

hash	You need to compute a hash of all your parameters and pass that hash to SparkitPay, using the following mechanism: toUpper (sha512 (SALT|account_name |account_number| api_key|bank_branch| bank_name| ifsc_code|vendor_code | vendor_contact_address| vendor_contact_email |vendor_contact_num| vendor_name)) Note: the SALT will be provided by SparkitPay separately. 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. Note: NEVER PASS SALT IN A FORM	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.

6.2 Modify Vendor API

URL:

https://biz.sparkitpay.com/v1/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.

Parameter Name	Description	Data type	Optional/ Mandatory
api_key	SparkitPay would assign a unique 40-digit merchant key toyou.Thiskeyisexclusive toyourbusiness/login account.Ifyouhave multipleloginaccounts,therewill necessarilybeonedifferentapi_keyper loginaccount that is assigned toyou.	varchar(40)	Mandatory
vendor_code	This is the vendor code that you wish to modify in the SparkitPay system. This value must already exist in the system, failing which SparkitPay 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_name	Account holder name (of the vendor bank account)	Varchar(300)	Optional
account_number	Account number of the vendor.	Varchar(50)	Optional
ifsc_code	IFSC code of the vendor's bank.	Varchar(50)	Optional
bank_name	Bank name of the vendor's bank.	Varchar(200)	Optional
bank_branch	Bank branch of the vendor's bank.	Varchar(300)	Optional
hash	You need to compute a hash of all your parameters and pass that hash to SparkitPay, using the following mechanism: `toUpper ( sha512 (SALT|account_name |account_number| api_key|bank_branch| bank_name| ifsc_code|vendor_code | vendor_contact_address| vendor_contact_email| vendor_contact_num| vendor_name))` Note: the SALT will be provided by SparkitPay separately. Please ensure that the concatenated string (upon which the hash will be computed) is in alphabetical order. Please include only those columns in your hash calculation which you are	Varchar(200)	Mandatory

	actually passing to us. For example, if you are not passing vendor_contact_address do not include that in hash calculation.		

6.3 Delete Vendor API

URL:

https://biz.sparkitpay.com/v1/deletevendor

This API can be used to delete a pre-existing vendor from the SparkitPay 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	SparkitPay would assign a unique 40-digit merchant key toyou.Thiskeyisexclusivetoyour business/login account.Ifyouhavemultiplelogin accounts,therewill necessarilybeonedifferent api_keyperloginaccount that is assigned to you.	varchar(40)	Mandatory
vendor_code	Thisisthevendorcodethatyouwishtodeletefrom the SparkitPay system. This value must already exist in the system, failing which SparkitPay will return an error.	varchar(30)	Mandatory
hash	You need to compute a hash of all your parameters and pass that hash to SparkitPay, using the following mechanism: toUpper (sha512 (SALT| api_key |vendor_code)) Note: the SALT will be provided by SparkitPay separately. Pleaseensurethattheconcatenatedstring(upon which the hash will be computed) is in alphabetical order. Note:NEVER PASS SALT IN A FORM	varchar(40)	Mandatory

This API can be used to delete a pre-existing vendor from the SparkitPay 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

7. SETTLEMENT API

7.1 getsettlementsbyorderid API

This API allows a merchant to programmatically access the status of any of his past settlements and other pertinent information pertaining to a prior transaction. 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.

Parameter Name	Description	Data type	Optional / Mandatory
api_key	SparkitPay would assign a unique 40-digit merchant key toyou.Thiskeyisexclusiveto your business/login account. Ifyouhavemultiplelogin accounts,therewill necessarilybeonedifferent api_keyperloginaccount that is assigned toyou.	varchar(40)	Mandatory
order_id	This is the order_id that the merchant passed to SparkitPay while making the original transaction request.	Unsigned integer	Mandatory
hash	You need to compute a hash of all your parameters and pass that hash to SparkitPay, using the following mechanism: toUpper (sha512 (SALT| api_key |order_id)) Note: the SALT will be provided by SparkitPay 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(200)	

This API returns a JSON in the following format:

(below output JSON for illustrative purposes only)

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

```
  {
   "order_id": 10017,
   "customer_name": "Karmendra Suthar",
   "customer_email": "kkk@example.com",
   "customer_phone": "9900261104",
   "return_url":
   "http://yoursitename.com/SparkitPay_return.php",
   "settlement_id": 653,
   "payment_date": "2016-04-12 08:45:57",
   "transaction_id": "HTRQ1071",
   "amount_paid_by_payer": "2749.00",
   "SparkitPay_charges": "52.23",
   "SparkitPay_service_tax": "7.57",
   "receiver_name": "OMN Public School",
   "receiver_split_share": "100.00",
   "amount_reimbursed_to_receiver": "2689.20",
   "settlement_status": "Fulfilled",
   "bank_settlement_date": "2016-04-13 14:18:49",
   "bank_settlement_amount": "2689.20",
   "bank_reference_number": "17462819",
   "name_of_bank_settled_to": "ICICI Bank"
  }
 ]
}
```


If you require customizations to the above output or need additional fields, please contact your relationship manager.

8. CHALLAN PAYMENT API

8.1 Request challan payment API

URL:

https://biz.sparkitpay.com/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	SparkitPay would assign a unique 40-digit merchant key to you. This key is exclusive to your business/login account. If	varchar(40)	Mandatory

	you have multiple login accounts, there will necessarily be one different api_key per login account that is assigned to you.		
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 SparkitPay, using the following mechanism: toUpper (sha512 (SALT|amount| api_key | email|mobile|name|purpose)) Note: the SALT will be provided by SparkitPay 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(40)	Mandatory

8.2 Request challan payment API url

URL:

https://biz.sparkitpay.com/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	SparkitPay 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(40)	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 SparkitPay, using the following mechanism: toUpper (sha512 (SALT|amount|api_key | email|mobile|name|purpose)) Note: the SALT will be provided by SparkitPay 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(40)	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.

9. Server to Server Call Back

9.1 Server to server response on Payment

To get server to server response, add callback URL in parameter named "Payment Callback URL" in settings tab of the biz.SparkitPay.in dashboard. If this is not found contact SparkitPay 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 SparkitPay. 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 "YYYY-MM-DD 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	SparkitPay 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.