

REACTJS GENERATE INVOICE



1. Introduction:

- Overview of PG Kit- Easily integrated, simplest, and secure technology for merchants that bridges your business and your customer.

2. Prerequisites:

System Requirement:

- Node JS must be installed in the System.
- After extracting the PG Kit.
- Run the **npm i** command for node Modules.
- Installed the Crypto JS **npm i crypto-js**
- Run the Project **npm start**

3.) API Specification:

Base URL:-

Below is the API base URL's

UAT

<https://pay1.getepay.in:8443/getepayPortal/pg/v2/generateInvoice>

Production

Will be shared with the merchant post-UAT sign off.

4.) Argument:

```
const data = {  
  mid: 108,  
  amount: "100.00",  
  merchantTransactionId: "sd12121",  
  transactionDate: "Mon Oct 03 13:54:33 IST 2023",  
  terminalId: "Getepay.merchant61062@icici",  
  udf1: "1234567890",  
  udf2: "test@gmail.com",  
  udf3: "Test",  
  udf4: "",  
  udf5: "",  
  udf6: "",  
  udf7: "",  
  udf8: "",  
  udf9: "",  
  udf10: "",  
  ru: "https://pay1.getepay.in:8443/getepayPortal/pg/pgPaymentResponse",  
  callbackUrl: "",  
  currency: "INR",  
  paymentMode: "ALL",  
  bankId: "",  
  txnType: "single",  
  productType: "IPG",  
  txnNote: "Test Txn",  
  vpa: "Getepay.merchant61062@icici",  
};
```



5.) Getepay UAT Keys:

```
"Getepay MID": "108"  
"Getepay Terminal Id": "Getepay.merchant61062@icici",  
"Getepay Key": "JoYPd+qso9s7T+Ebj8pi4Wl8i+AHLv+5UNJxA3JkDgY=",  
"Getepay IV": "hlnuyA9b4YxDq6oJSZF18g==",  
"Getepay Url": "https://pay1.getepay.in:8443/getepayPortal/pg/v2/generateInvoice",
```

6.) Request:

Parameter	Description	Data Type	Mandatory
mid	Merchant ID	String	M
amount	Amount	String	M
MerchantTransactionId	Transaction ID of the Merchant	String	M
transactionDate	Date of Transaction	String	M
terminalId	Terminal ID	String	M
udf1	Merchant MobileNo.	String	M
udf2	Merchant Email-Id	String	M
udf3	Merchant Name	String	M
udf4	User define field	String	O
udf5	User define field	String	O
udf6	Reserved Field (Split)	String	R
udf7	User define field	String	O
udf8	User define field	String	O
udf9	User define field	String	O
udf10	User define field	String	O
ru	Return URL	String	M
callbackUrl	Call Back URL	String	O
currency	INR	String	M
paymentMode	Mode of Payment	String	M
bankId	Bank ID	String	O
txnType	Type of Transaction	String	M
productType	Type of Product	String	M
txnNote	Note/Remarks	String	O
vpa	Virtual PaymentAddress (Terminal ID)	String	M

According to the argument passed, the user will be redirected to PG-----

7.) Request & Response Format:

Below is the request format:

```
{
  "mid": <Provided by Getepay>,
  "terminalId": <Provided by
Getepay>,
  "req": <Hex encoded AES/GCM encrypted json request>,
}
```

Below is the response format:

```
{
  "mid": <Provided by Getepay>,
  "terminalId": <Provided by
  Getepay>,"status": <status of
  request>, "message":
  <success/error message>,
  "res": <Hex encoded AES/GCM encrypted json request>
}
```

8.) Response:

Parameter	Description	Data Type
paymentId	Payment Id	String
paymentUrl	Payment Page	String
qrIntent	QR intent URL	String
qr	DynamicQR Image	String
token	Generated Token	String
qrpath	OR Path URL	String

9.) Decrypted Response Format:

[paymentId=19172652,
paymentUrl=https://pay1.getepay.in:8443/getepayPortal/pg/v2/payment?token=fc5ce2a3-775d-4562-a28d-d30342411d46, qrIntent=upi://pay?pa=Getepay.merchant129118@icici&mam=61.00&am=61.00&pn=Siddharth Purohit&tr=GETgpdr314295, qr=/media/shared/dynamicqrpath/GETgpdr314295.png, token=fc5ce2a3-775d-4562-a28d-d30342411d46]

10.) Encryption/Decryption:

Getepay uses AES/GCM encryption with hex-encoded for data communication. Below is the way to generate the actual request and response.

- Generate the respective API request.
- Convert the request in JSON string.
- Encode the request with **AES/GCM** using the Getepay provided key and iv.
- Convert the output into a hex string
- Set the hex-encoded string in the "req" or "res" parameter.

Encryption Logic:

```
function base64ToBytes(base64) {
  return Uint8Array.from(Buffer.from(base64, 'base64'));
}

function bytesToBase64(bytes) {
  return Buffer.from(bytes).toString('base64');
}

class GcmPgEncryption {
  constructor(iv, ivKey) {
    this.iv = iv;
    this.ivKey = ivKey;
    this.mKey = null;
  }

  async init() {
    const combined = this.ivKey + this.iv;
    const combinedBytes = new TextEncoder().encode(combined);
    const hash = crypto.createHash('sha256').update(combinedBytes).digest();
    this.mKey = bytesToBase64(hash);
  }

  async encryptWithMKeys(plainMessage) {
    if (!this.mKey) await this.init();

    const salt = crypto.randomBytes(16);
    const iv = crypto.randomBytes(12);
    const passwordBytes = Buffer.from(this.mKey, 'utf-8');

    const derivedKey = crypto.pbkdf2Sync(passwordBytes, salt, 65535, 32, 'sha512');

    const cipher = crypto.createCipheriv('aes-256-gcm', derivedKey, iv);
    const plaintext = Buffer.from(plainMessage, 'utf-8');

    const encrypted = Buffer.concat([cipher.update(plaintext), cipher.final()]);
    const tag = cipher.getAuthTag();

    const combined = Buffer.concat([salt, iv, encrypted, tag]);
    return bytesToBase64(combined);
  }
}
```

Decryption Logic:

```
async decryptWithMKeys(cipherContent) {
  if (!this.mKey) await this.init();

  const combined = base64ToBytes(cipherContent);
  const salt = combined.slice(0, 16);
  const iv = combined.slice(16, 28);
  const ciphertext = combined.slice(28, -16);
  const tag = combined.slice(-16);

  const passwordBytes = Buffer.from(this.mKey, 'utf-8');
  const derivedKey = crypto.pbkdf2Sync(passwordBytes, salt, 65535, 32, 'sha512');

  const decipher = crypto.createDecipheriv('aes-256-gcm', derivedKey, iv);
  decipher.setAuthTag(tag);

  const decrypted = Buffer.concat([decipher.update(ciphertext), decipher.final()]);
  return decrypted.toString('utf-8');
}
}
// data (the encrypted information), key (the decryption key), and iv (the initialization
vector).
// The result is converted to a UTF-8 string using .toString(enc.Utf8) and is then returned
from the function.
```

11.) Response Handling

After making the payment you can handle the response in the following two ways:-

- 1.) You can use our Return URL it will redirect to the receipt.

For UAT:

<https://pay1.getepay.in:8443/getepayPortal/pg/pgPaymentResponse>

For Portal:

<https://portal.getepay.in:8443/getepayPortal/pg/pgPaymentResponse>

- 2.) If you are using your RU (Return URL) you can use the below sample for your reference:

- Note: Call this function on the Deployed URL to handle the response.
- Set the Deployed URL on the RU

```
function handleSuccessPayment(responseData) {
var decryptedData = decryptEas(
    responseData,
    "JoYPd+qso9s7T+Ebj8pi4Wl8i+AhLv+5UNJxA3JkDgY=",
    "hlnuyA9b4YxDq6oJSZF18g=="
);
var parsedData = JSON.parse(decryptedData);
}
```

12.) Sample Decrypted Response (After Payment):

```
{ "getepayTxnId": "19246958", "mid": "108", "txnAmount": "100.0", "txnStatus": "SUCCESS", "merchantOrderNo": "sd12121", "udf1": "1234567890", "udf2": "mailto:test@gmail.com", "udf3": "Test", "udf4": "", "udf5": "", "udf6": "", "udf7": "", "udf8": "", "udf9": "", "udf10": "", "udf41": "http://localhost:8080/success_payment", "custRefNo": "", "paymentMode": "DC", "discriminator": "L21IZGIhL3NoYXJlZC9keW5hbWljcXJwYXRoL2dwZHIuMDAwMjUzODlucG5n", "message": "", "paymentStatus": "SUCCESS", "txnDate": "2024-08-02 11:23:59", "surcharge": "", "totalAmount": "102.36" }
```

13.) How To Use Split Payment Option:

Step 1- Encode the below data using base64 Platform

Step 2- After encoding set encoded data in udf 6 which is mentioned in the code.

Step 3- set txnType: "multi",

Note: The amount mentioned in the code below should be the total sum in the PG Code.

```
<products>
  <product>
    <code>Code001</code>           //Set product Id
    <name>ACC1</name>             // Set Name
    <amount>100</amount>         //Set amount
  </product>
</products>
```