



## **BasisPay Payment Gateway Android SDK Integration Kit**

044 4018 4444 | [info@basispay.in](mailto:info@basispay.in) | [basispay.in](https://basispay.in)

New # 9, Old # 11, 1st Floor, Palayakaran Street, Kalaimagal Nagar, Ekkatuthangal, Chennai, Tamil Nadu 600032

Thanks for choosing BasisPay as your payment facilitator for your business. This document helps business owners or developers to understand how to integrate BasisPay payment gateway in android Platform. If you have any queries or clarifications, please email to [support@basispay.in](mailto:support@basispay.in) or visit [www.tychepayment.com](http://www.tychepayment.com)

## Prerequisites

Below are the prerequisites to integrate with Payment Gateway:

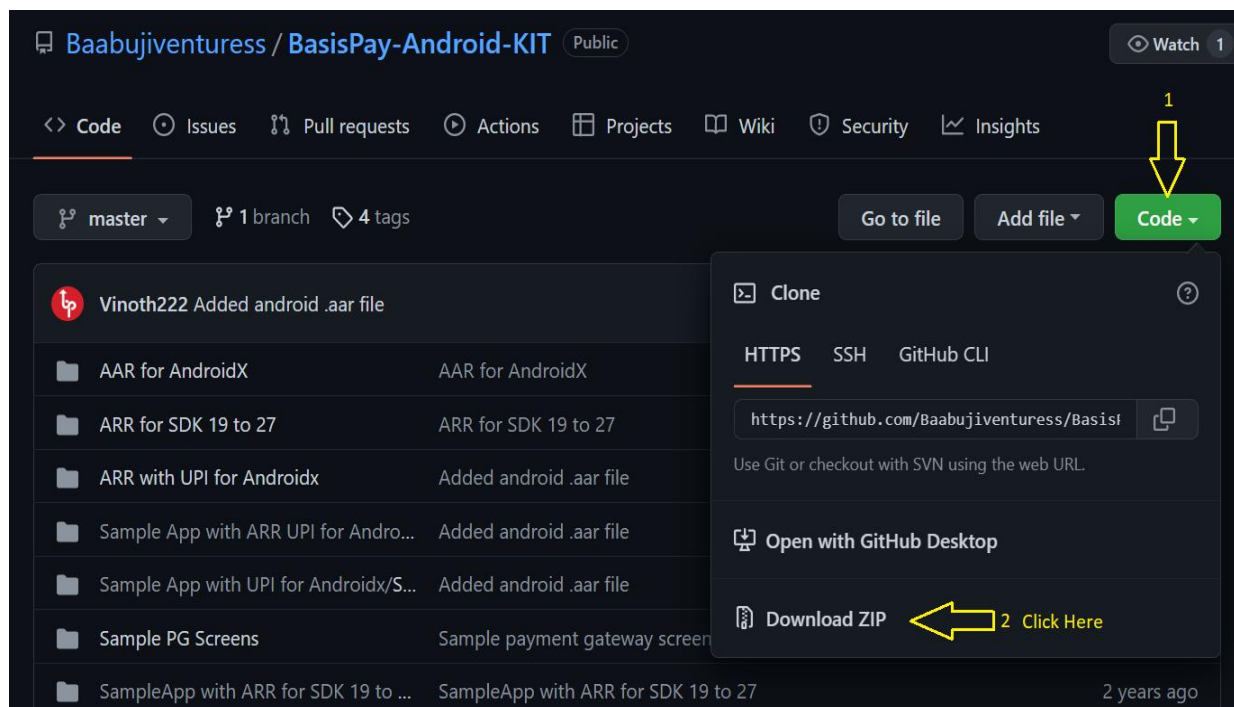
1. Be an Approved Merchant: To use our payment gateway, you should be a registered and approved merchant by us. If not registered, please register to get started!
2. Obtain API Key: You should have received an API key from us on Approval.
3. API to receive Payment Response: You should have a self-hosted web server to receive the response and verify the payment details post payment.
4. Min SDK Requirements: You should ensure that your Android App's SDK version should be greater than or equal to 21.

## Downloads

Basispay Payment gateway android integration link here:

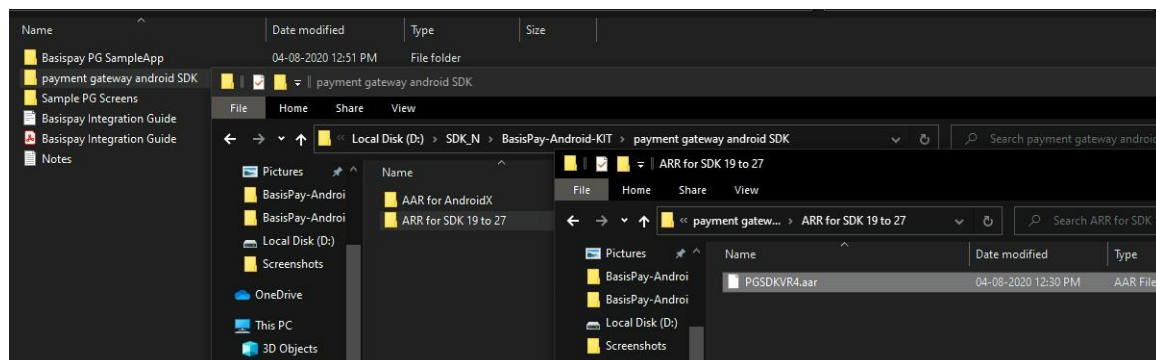
### **DOWNLOAD BASISPAY ANDROID PG SDK KIT**

- Our PG android SDK .aar file can be downloaded here:  
[Download AAR for SDK 19 to 27](#)  
[Download AAR for AndroidX](#)  
[Download UPI AAR for AndroidX](#) **New**
- A sample Android App that illustrates the integration of any app with our SDK can be downloaded here:  
[Download SampleAPP with AAR for SDK 19 to 27](#)  
[Download SampleAPP with AAR for AndroidX](#)  
[Download Sample App with UPI AAR for AndroidX](#) **New**



## Integration Steps

1. Download the BasisPay-Android-KIT zip folder and unzip it.



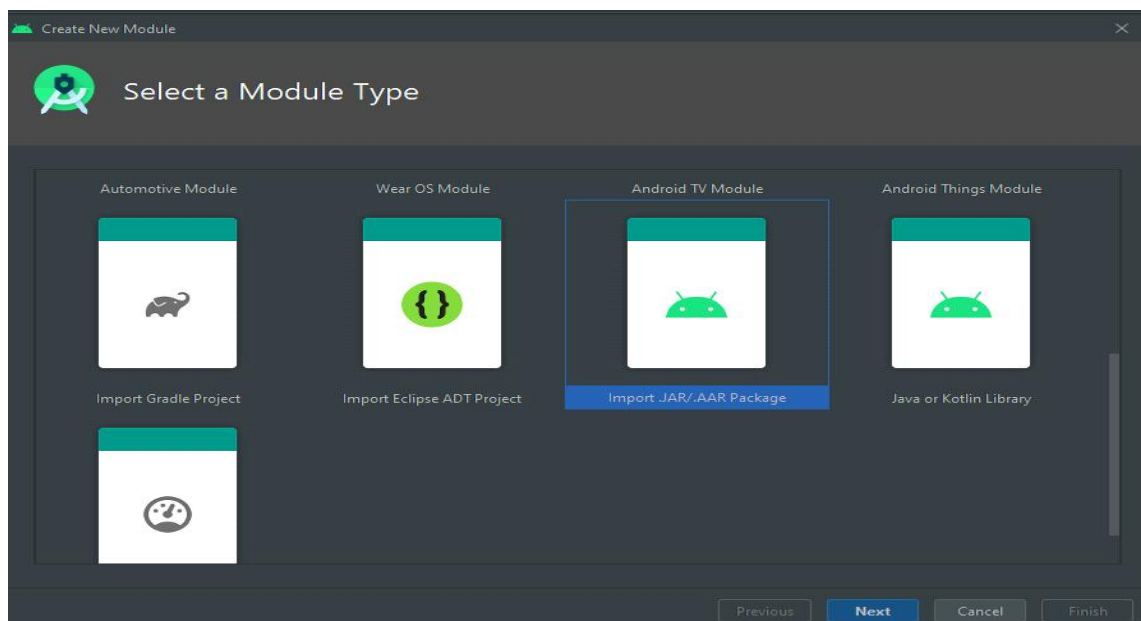
Copy the PGSDKVR4 AAR file and save this in your Desktop.

## Prerequisites

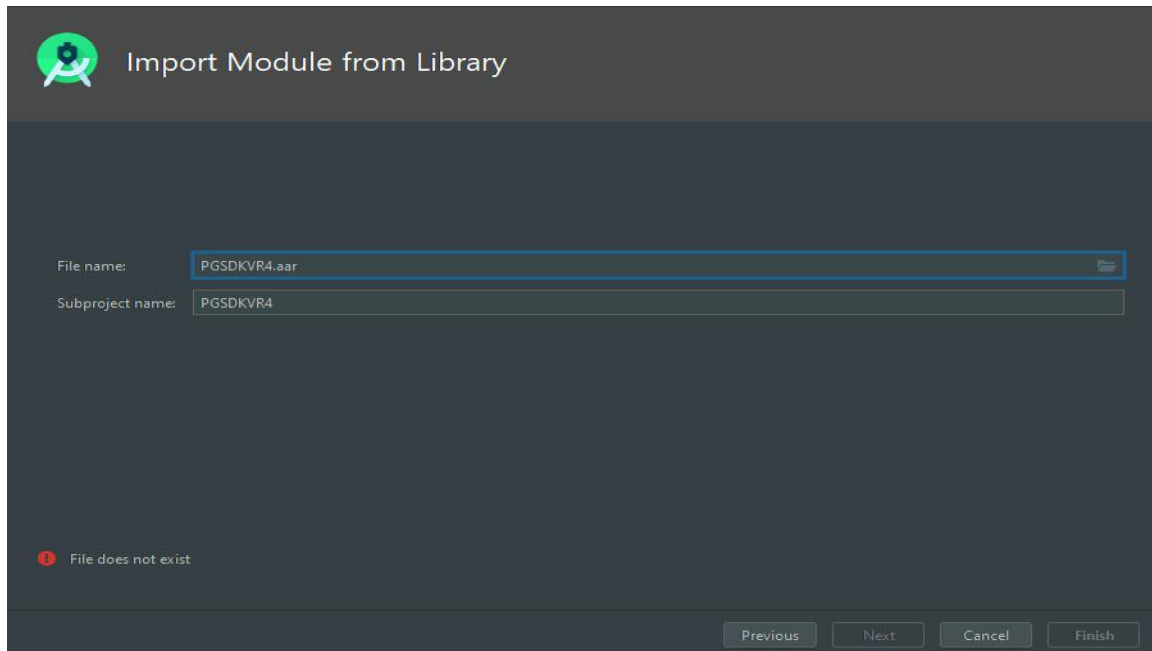
Your Android App's SDK Version must be greater than or equal to 19.

Add the PGSDKVR4 AAR library to your app:

- Click File > New > New Module.

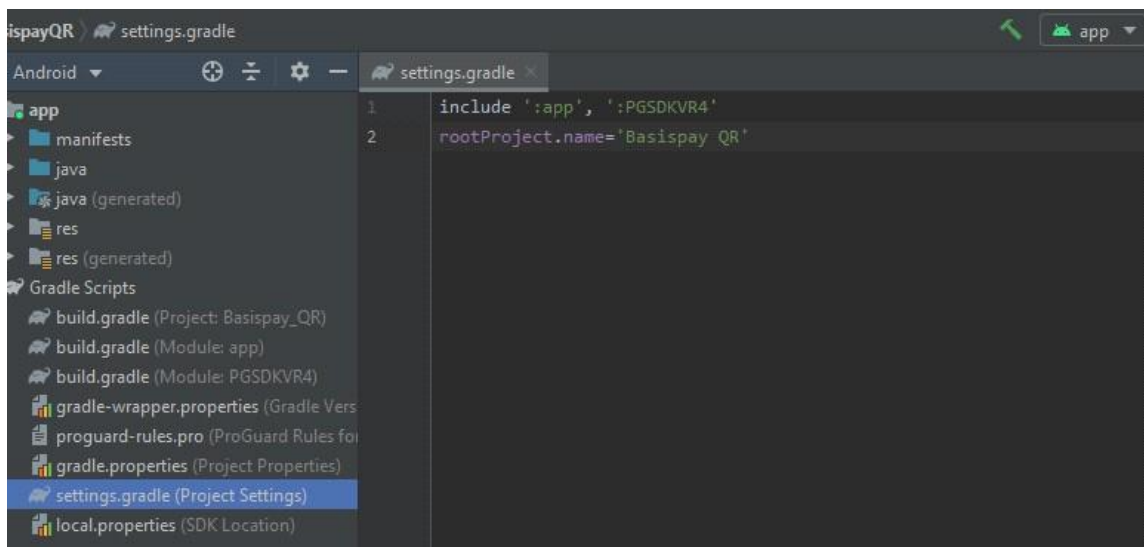


- Click Import .JAR/.AAR Package then click Next.



- Enter the location of the PGSDKVR4 AAR file then click Finish.

Make sure PGSDKVR4 library is listed at the top of your settings.gradle file, as shown below:



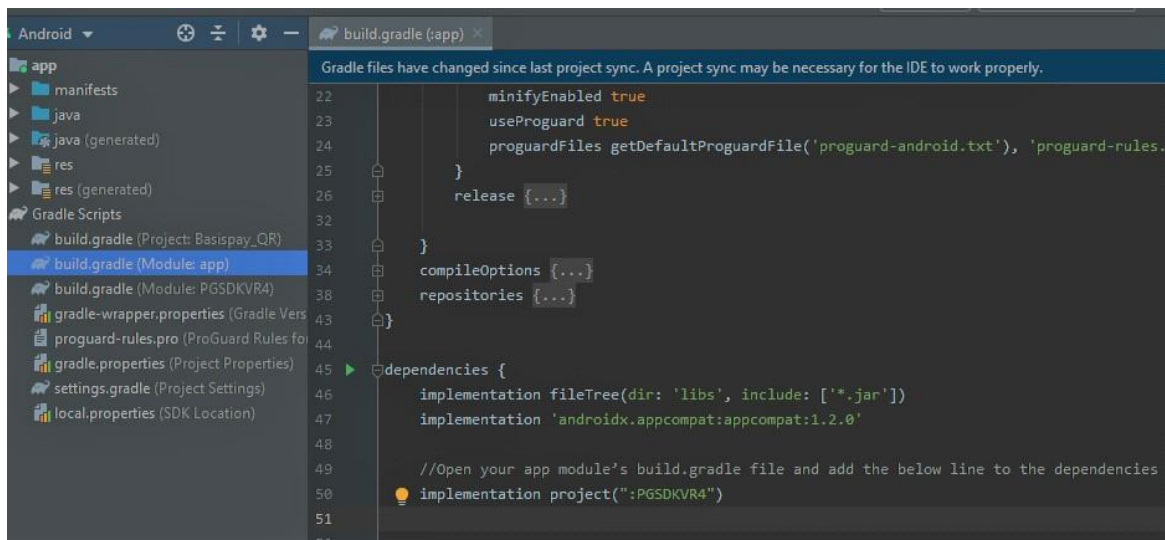
For SDK 19-27

```
include ':app', ':PGSDKVR4'
```

## For AndroidX

`include ':app', ':pgsdkv5ax'`

Open your app module's build.gradle file and add the below line to the dependencies block as shown in the following snippet:



## For SDK 19-27

`dependencies`

`{`

`implementation project(":PGSDKVR4")`

`}`

For AndroidX

dependencies

```
{  
  
    implementation project(":pgsdkv5ax")  
  
}
```

Latest Android Studio Version 4.2 and Above **Latest**

Your Android App's Target SDK Version must be greater than or equal to 21.

Add the Latest .AAR Dependency to your app: **New**

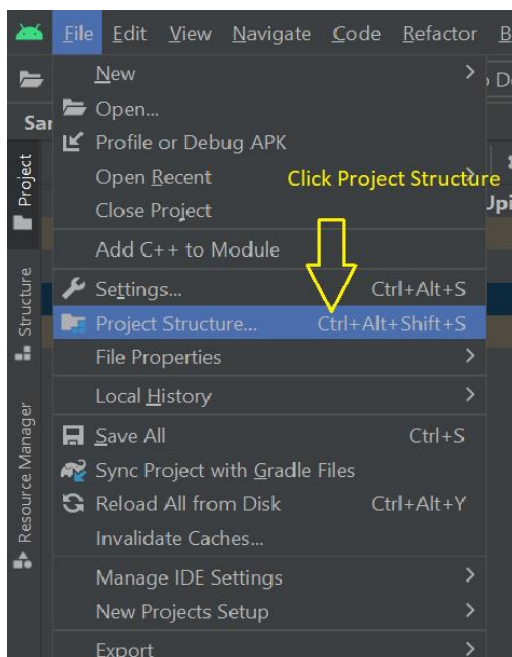
Once copy the .AAR file and save this in your android app.

[Download UPI AAR for AndroidX](#) **New**

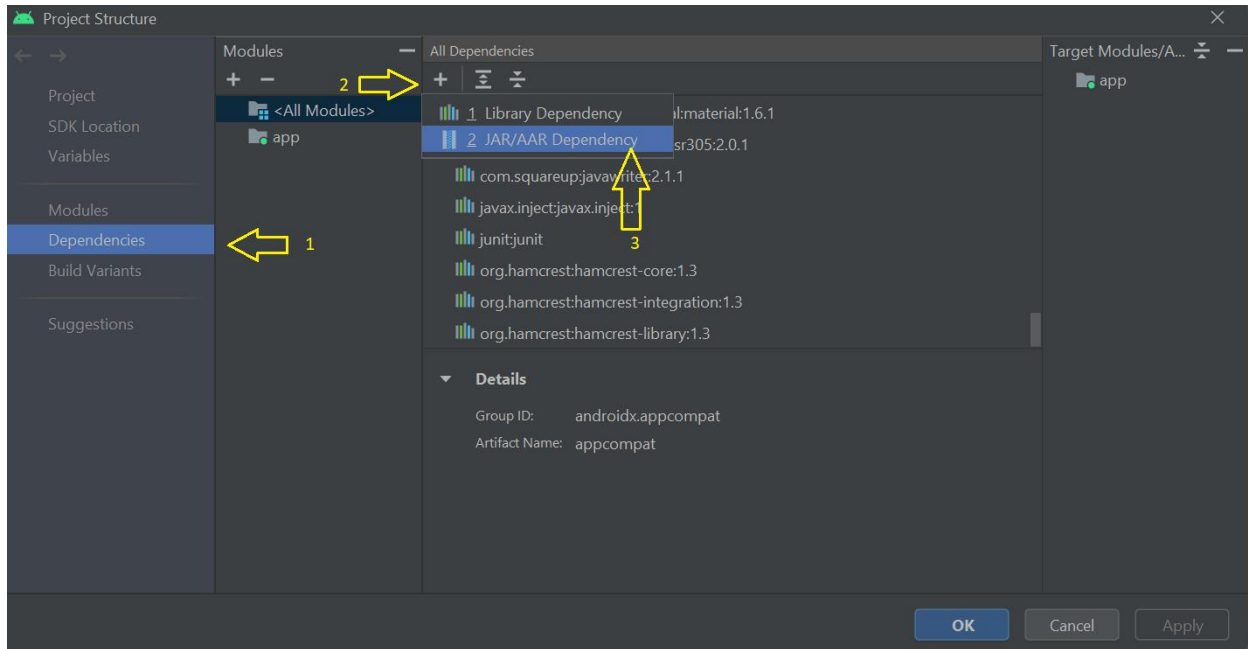
App Location: [Project Directory](#) > [app](#) > [libs](#).

Steps:

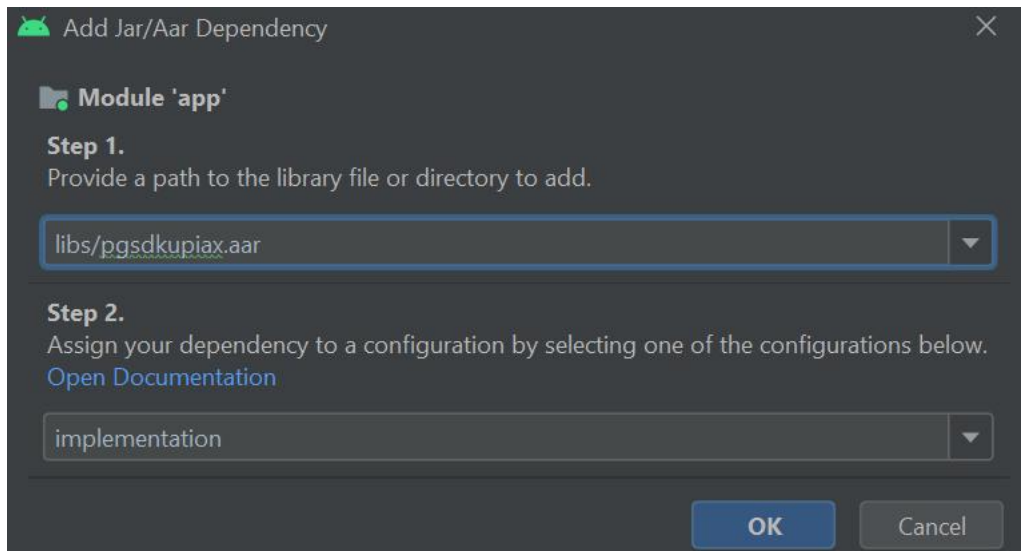
1. Navigate to File > Project Structure.



2. In the Declared Dependencies tab, click + and select JAR/AAR Dependency in the dropdown.

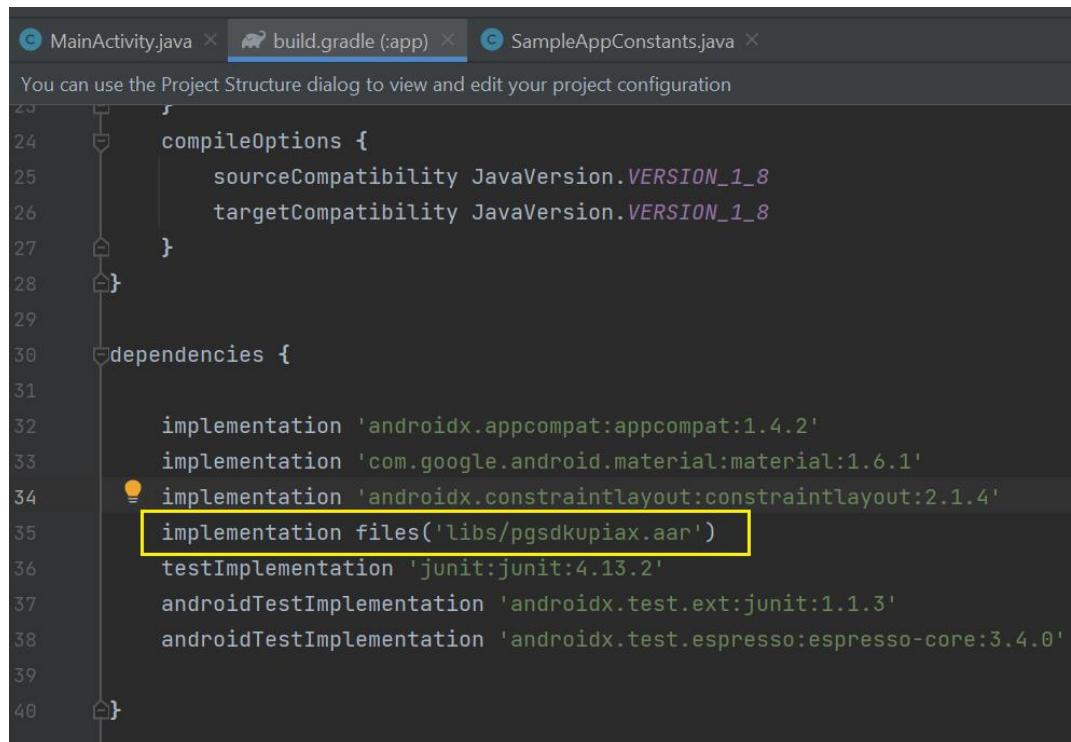


3. In the Add JAR/AAR Dependency dialog, first enter the path to your .aar or .jar file, then select the configuration to which the dependency applies.





4. Check your app's build.gradle file to confirm a declaration similar to the following



Alternatively, if you're running Gradle builds outside of Android Studio, it's possible to import a dependency by adding a path to the dependency in your app's build.gradle file. For example:

```
dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar", "*.aar"])
}
```

5. If you added dependency gradle file and then sync now in your project.

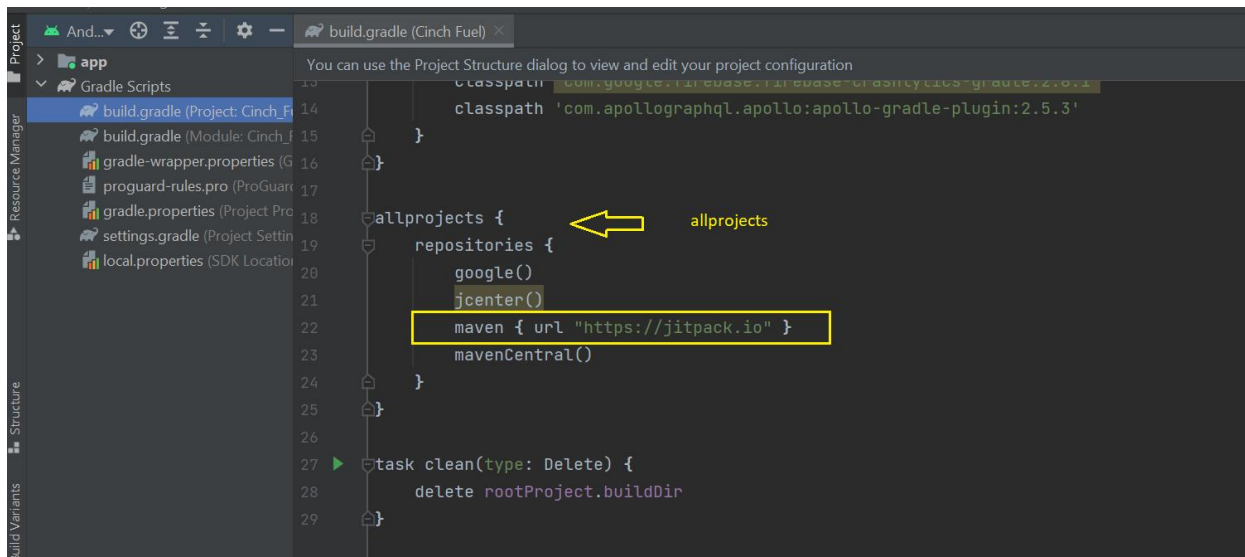
[Add the Latest Library Dependency to your app:](#) **New**

To get a Git project into your build.

Step:

1. Add it in your project root build.gradle at the end of repositories.

```
allprojects {
    repositories {
        ...
        maven { url 'https://jitpack.io' }
    }
}
```

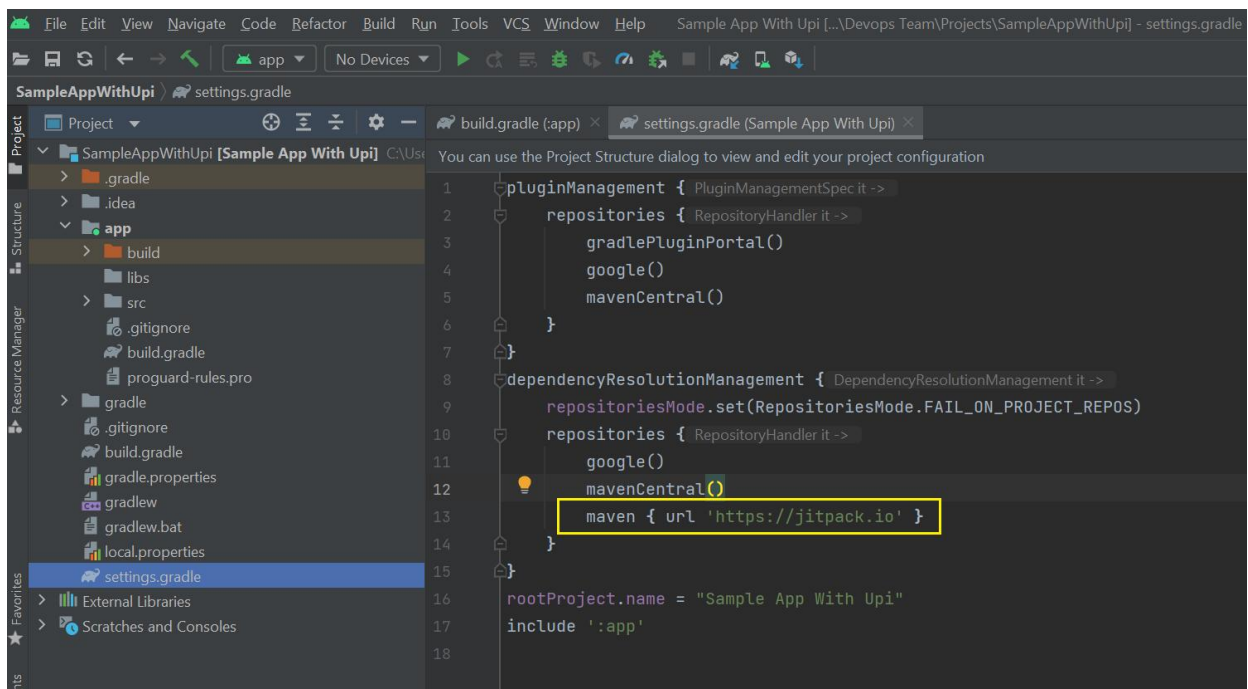


OR

In Android Studio IDE Version above 4.

Add it in your project root settings.gradle at the end of repositories.

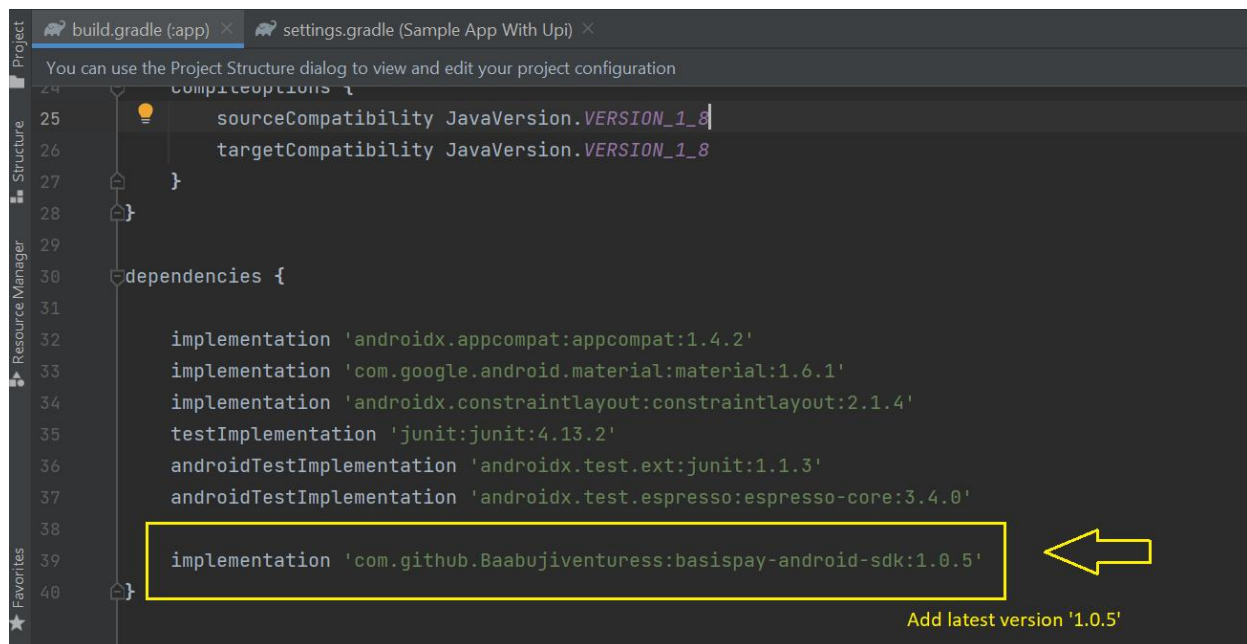
Example:



2. Add the dependency in app module build.gradle file.

```
dependencies {  
  
    implementation 'com.github.Baabujiventuress:basispay-android-sdk:1.0.5'  
  
}
```

Example:



3. Check your app's build.gradle file to confirm and added the dependency then sync now in your project. That's it.

Click Sync Project with Gradle Files

Make sure you have the below permissions in your manifest file:

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

## Code Explanation

Below the code if you added latest library or .aar file:

Set the payment parameters in latest library and .aar dependency

```
(MainActivity.java)
Random rnd = new Random();
int n = 100000 + rnd.nextInt(900000);
SampleAppConstants.PG_ORDER_ID=Integer.toString(n);

PaymentParams pgPaymentParams = new PaymentParams();
pgPaymentParams.setApiKey(SampleAppConstants.PG_API_KEY);/*required
pgPaymentParams.setSaltKey(SampleAppConstants.PG_SALT_KEY);/*required
pgPaymentParams.setAmount(SampleAppConstants.PG_AMOUNT);/*required
pgPaymentParams.setEmail(SampleAppConstants.PG_EMAIL);/*required
pgPaymentParams.setName(SampleAppConstants.PG_NAME);/*required
pgPaymentParams.setPhone(SampleAppConstants.PG_PHONE);/*required
pgPaymentParams.setOrderId(SampleAppConstants.PG_ORDER_ID);/*required
pgPaymentParams.setCurrency(SampleAppConstants.PG_CURRENCY);/*required
pgPaymentParams.setDescription(SampleAppConstants.PG_DESCRIPTION);/*required
pgPaymentParams.setCity(SampleAppConstants.PG_CITY);/*required
pgPaymentParams.setState(SampleAppConstants.PG_STATE);/*required
pgPaymentParams.setAddressLine1(SampleAppConstants.PG_ADD_1);
pgPaymentParams.setAddressLine2(SampleAppConstants.PG_ADD_2);
pgPaymentParams.setZipCode(SampleAppConstants.PG_ZIPCODE);/*required
pgPaymentParams.setCountry(SampleAppConstants.PG_COUNTRY);/*required
pgPaymentParams.setReturnUrl(SampleAppConstants.PG_RETURN_URL);/*required
pgPaymentParams.setMode(SampleAppConstants.PG_MODE);/*required
pgPaymentParams.setUdf1(SampleAppConstants.PG_UDF1);
pgPaymentParams.setUdf2(SampleAppConstants.PG_UDF2);
pgPaymentParams.setUdf3(SampleAppConstants.PG_UDF3);
pgPaymentParams.setUdf4(SampleAppConstants.PG_UDF4);
pgPaymentParams.setUdf5(SampleAppConstants.PG_UDF5);
pgPaymentParams.setEnableAutoRefund("n");
pgPaymentParams.setOfferCode("testcoupon");
pgPaymentParams.setSplitEnforceStrict("y");
pgPaymentParams.setInterface_type("android_sdk");/*required
```

## Initiate the payment action

```
PaymentGatewayPaymentInitializer pgPaymentInitializer =
    new PaymentGatewayPaymentInitializer(pgPaymentParams,MainActivity.this);
pgPaymentInitializer.initiatePaymentProcess();
```

## Payment Response Code

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == PGConstants.REQUEST_CODE) {
        if (resultCode == Activity.RESULT_OK) {
            try {
                String sdkResponse = data.getStringExtra(PGConstants.PAYMENT_RESPONSE);
                System.out.println("paymentResponse: " + sdkResponse);
                if (sdkResponse.equals("null")) {
                    System.out.println("Transaction Error!");
                    Toast.makeText(this, "Transaction Error!", Toast.LENGTH_SHORT).show();
                } else {

                    JSONObject response = new JSONObject(sdkResponse);
                    if (response.has("payment_response") && response.getString("status").equals("success")) {
                        JSONObject paymentResponse = new JSONObject(response.getString("payment_response"));
                        Log.d("payment_response", paymentResponse.toString());
                        String transaction_id = paymentResponse.getString("transaction_id");
                        String response_code = paymentResponse.getString("response_code");
                        String response_message = paymentResponse.getString("response_message");
                        String amount = paymentResponse.getString("amount");
                        String payment_mode = paymentResponse.getString("payment_mode");
                        String payment_channel = paymentResponse.getString("payment_channel");
                        String payment_datetime = paymentResponse.getString("payment_datetime");
                        if (response_code.equals("0")) {
                            //Transaction Success
                            Toast.makeText(this, response_message, Toast.LENGTH_SHORT).show();
                        } else if (response_code.equals("1000")) {
                            //Transaction Failed
                            Toast.makeText(this, response_message, Toast.LENGTH_SHORT).show();
                        } else {
                            //Transaction Cancel
                            Toast.makeText(this, response_message, Toast.LENGTH_SHORT).show();
                        }
                    } else {
                        System.out.println("Transaction Status: " + response.getString("status"));
                        System.out.println("Transaction Message: " + response.getString("error_message"));
                        Toast.makeText(this,response.getString("error_message"), Toast.LENGTH_SHORT).show();
                    }
                }
            }

        } catch (JSONException e) {
            e.printStackTrace();
        }

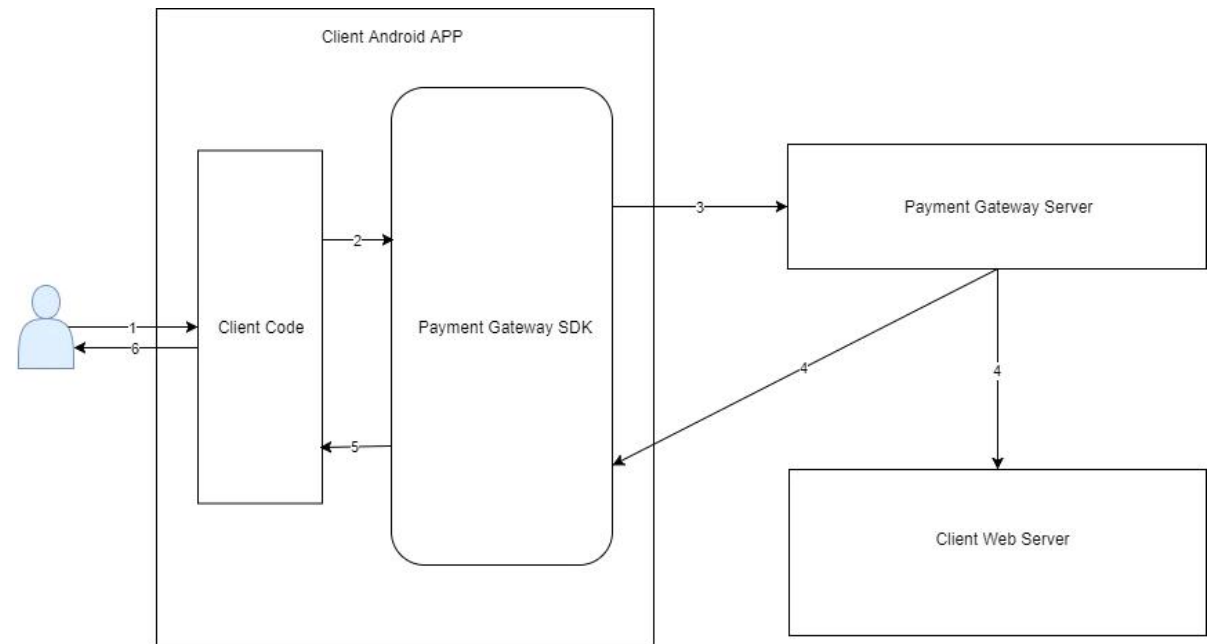
    }
    if (resultCode == Activity.RESULT_CANCELED) {

    }
}
```

# Code Explanation

## Note

Your App must use the latest security standards to prevent your code being compromised



- User starts the payment in the client's app.
- Client's code then sets the payment parameters and initiates the payment process via the SDK.
- The SDK in-turn interacts with the Payment Gateway server during the payment process.
- After the payment, the Payment Gateway sends the payment response to the Client's web server(Via the Return URL)and to the SDK.At this point, the client's code in the Client's web server code should re verify the hash in the payment response and store the response in the Database.
- The SDK parses the payment response and converts into json and provides it to the Client's code. At this point, the client should compare the amount and order id with their corresponding value DB in the web server.
- If the values match,Client's Code then displays the payment response to the User.

Initiate the "com.test.pg.secure.pgsdkv4.PaymentParams" CLASS to

set the payment parameters:

## Set the payment parameters

```
PaymentParams pgPaymentParams = new PaymentParams();
pgPaymentParams.setAPIKey(SampleAppConstants.PG_API_KEY);
pgPaymentParams.setAmount(SampleAppConstants.PG_AMOUNT);
pgPaymentParams.setEmail(SampleAppConstants.PG_EMAIL);
pgPaymentParams.setName(SampleAppConstants.PG_NAME);
pgPaymentParams.setPhone(SampleAppConstants.PG_PHONE);
pgPaymentParams.setOrderId(SampleAppConstants.PG_ORDER_ID);
pgPaymentParams.setCurrency(SampleAppConstants.PG_CURRENCY);
pgPaymentParams.setDescription(SampleAppConstants.PG_DESCRIPTION);
pgPaymentParams.setCity(SampleAppConstants.PG_CITY);
pgPaymentParams.setState(SampleAppConstants.PG_STATE);
pgPaymentParams.setAddressLine1(SampleAppConstants.PG_ADD_1);
pgPaymentParams.setAddressLine2(SampleAppConstants.PG_ADD_2);
pgPaymentParams.setZipCode(SampleAppConstants.PG_ZIPCODE);
pgPaymentParams.setCountry(SampleAppConstants.PG_COUNTRY);
pgPaymentParams.setReturnUrl(SampleAppConstants.PG_RETURN_URL);
pgPaymentParams.setMode(SampleAppConstants.PG_MODE);
pgPaymentParams.setUdf1(SampleAppConstants.PG_UDF1);
pgPaymentParams.setUdf2(SampleAppConstants.PG_UDF2);
pgPaymentParams.setUdf3(SampleAppConstants.PG_UDF3);
pgPaymentParams.setUdf4(SampleAppConstants.PG_UDF4);
pgPaymentParams.setUdf5(SampleAppConstants.PG_UDF5);
```

Initialize the `com.test.pg.secure.pgskv4.PaymentGatewayPaymentInitializer` class with payment parameters and initiate the payment:

## Initiate the payment

```
PaymentGatewayPaymentInitializer pgPaymentInitializer = new
PaymentGatewayPaymentInitializer(pgPaymentParams,MainActivity.this);
```



```
pgPaymentInitializer.initiatePaymentProcess();
```

To receive the json response, override the `onActivityResult()` using the `REQUEST_CODE` and `PAYMENT_RESPONSE` variables from `com.test.pg.secure.pgsdkv4.PaymentParams` class

## Payment Response Code

```
1  @Override
2  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
3
4      if (requestCode == PGConstants.REQUEST_CODE) {
5          if (resultCode == Activity.RESULT_OK) {
6              try {
7                  String paymentResponse = data.getStringExtra(PGConstants.PAYMENT_RESPONSE);
8                  System.out.println("paymentResponse: " + paymentResponse);
9                  if (paymentResponse.equals("null")) {
10                     System.out.println("Transaction Error!");
11                     transactionIdView.setText("Transaction ID: NIL");
12                     transactionStatusView.setText("Transaction Status: Transaction Error!");
13                 } else {
14                     JSONObject response = new JSONObject(paymentResponse);
15                     transactionIdView.setText("Transaction ID: " + response.getString("transaction_id"));
16                     transactionStatusView.setText("Transaction Status: " + response.getString("response_message"));
17                 }
18             } catch (JSONException e) {
19                 e.printStackTrace();
20             }
21         }
22     }
23
24     if (resultCode == Activity.RESULT_CANCELED) {
25         //Write your code if there's no result
26     }
27
28 }
29 }
```

## Note

Request parameters are the parameters that will be sent to our server API for payment initiation. Client should store the order id and the amount before payment initiation and compare it with the order id and amount in the response Json from our server post payment process to ensure no end user tampering on the requested parameters.

## List of Request Parameters

PARAMETER NAME	DESCRIPTION	REQUIRED	DATATYPE
api_key	We 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.	Mandatory	String - Max:40.
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.	Mandatory	String - Max:30.
mode	This is the payment mode ("TEST" or "LIVE" are valid values). "LIVE" is the default value when not specified.	Optional	String - Max:4.
amount	This is the payment amount.	Mandatory	Decimal - Max Digits Before Decimal:15, Max Digits after Decimal:2.
currency	This is the 3-digit currency code (INR).	Mandatory	String - Max:3.
description	Brief description of product or service that the customer is being charged for.	Mandatory	String - Max:200.
name	Name of customer.	Mandatory	String - Max:200.
email	Customer email address.	Mandatory	String - Max:200.
phone	Customer phone number.	Mandatory	String - Max:30.
address_line_1	Customer's address line 1.	Optional	String - Max:255.
address_line_2	Customer's address line 2.	Optional	String - Max:255.
city	Customer City.	Mandatory	String - Max:50.
state	Customer State.	Optional	String - Max:50.
country	Customer Country.	Mandatory	String - Max:50.
zip_code	Customer Zipcode.	Mandatory	String - Max:10.
timeout_duration	Timeout duration (in seconds).	Optional	Integer - Min:0,Max:1000.
udf1	User defined field 1.	Optional	String - Max:300.
udf2	User defined field 2.	Optional	String - Max:300.
udf3	User defined field 3.	Optional	String - Max:300.
udf4	User defined field 4.	Optional	String - Max:300.
udf5	User defined field 5.	Optional	String - Max:300.
return_url	This API needs to be created by the Client using their web server to which, Traknpay will automatically send all the response after a payment as a POST request after a payment is completed. Client is required to verify the hash and store the data to their database.This must be HTTPS, not HTTP to ensure a secured line.	Mandatory	String - Max:200.

return_url_failure	We will send all failed transaction response parameters to this URL if specified, else, it will send the failed response to the "return_url" parameter	Optional	String - Max:200.
return_url_cancel	We will send all cancelled transaction response parameters to this URL if specified, else, it will send the cancelled response to the "return_url" parameter.	Optional	String - Max:200.
percent_tdr_by_user	Percent of tdr amount paid by user.	Optional	Integer - Min:0,Max:100.
flatfee_tdr_by_user	Fixed fee paid by user.	Optional	Integer - Min:0,Max:99999999.
show_convenience_fee	Controls whether the convenience fee amount (for surcharge merchants) is displayed to the customer (on the payment page) or not.	Optional	String - Max:1.
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.	Optional	String - Max:1.
payment_options	payment options to be displayed such credit card(cc), netbanking(nb), wallet(w) and atm card(atm). Tabs will be displayed by order in which values are sent. Values accepted are: cc,nb,w,atm (comma separated string).	Optional	String - Max:20.
payment_page_display_text	This text will be displayed below the logo on payment page.	Optional	String - Max:200.

## List of Response Codes

### Note

Below are the response codes that come in the payment response post payment from our server, that must be handled by the client.

Code	Response Message	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

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
9999	UNKNOWN-ERROR	Unknown error occurred
997	-	These are unhandled errors coming from banks directly.

**THANK YOU**

If you are facing any issues while integration you can email to [support@basispay.in](mailto:support@basispay.in)