Razorpay POS

Integration Solution Doc

→ POS Bridge

→ DQR

→ Android SDK Solution

TABLE OF CONTENTS

RAZORPAY POS' S INTEGRATED PAYMENT SOLUTION - POS BRIDGE

- Prerequisites to use POS Bridge Solution
- What is the Android POS Bridge solution?
- How to get started with Razorpay's POS Bridge
- Integration?
- Advantages
- API Details
  - To send notification for POS Bridge from merchant side
  - To get the status of POS Bridge notification from Merchant's side
  - To cancel a notification from Merchant's side
- Best Practices
- Testing the Integration
- Solution Capabilities
- Steps to Go-Live
- What do we need for Go-Live?

RAZORPAY POS'S INTEGRATED DQR SOLUTION

- What is the POS's DQR solution?
- API Details
  - To Send notification for POS Bridge from merchant side:
  - To get the status of POS Bridge notification from Merchant's side
  - To cancel a notification from Merchant's side

RAZORPAY_POS_ ANDROID SDK

- Razorpay POS SDK Introduction
  - Why Razorpay POS SDK?
  - What does Razorpay POS Android SDK do?
- Getting started with Android SDK Integration
  - How to include Payments SDK in your Android Application
  - How does the POS Android SDK work?
- Getting started with Android SDK Integration
- Getting started with Android SDK Integration
- Getting started with Android SDK Integration
  - Testing the Integration
  - Go-Live

RAZORPAY POS' S INTEGRATED PAYMENT SOLUTION - POS BRIDGE

PREREQUISITES TO USE POS BRIDGE SOLUTION

Merchant system must have Internet Connectivity (Wi-Fi or Mobile Hotspot)
Merchant systems are able to call/ access any external API like Ezetap POS Bridge API from their own system using minimum TLS 1.2 protocol (Transport Layer Security).

WHAT IS THE ANDROID POS BRIDGE SOLUTION?

POS Bridge Solution Architecture:

Ezetap's POS Bridge solution allows seamless integration with Customer's billing application across any Operation system (Web/ Android/ Web/ Linux and more etc.) for payment processing. POS Bridge is a server-to-server communication which is enabled by the billing application to the POS terminal which gets payment notification for completion.

Merchant server
Agent initiates payment push
Merchant server calls Razorpay PUSH2PAY API
Billing application
Merchant calls Razorpay P2P API for displaying the transaction details in the application
Razorpay server will send the transaction response to merchant server
Razorpay server sends payment notification to the device
Device starts payment txn & receives txn response
VISA
MasterCard
RuPay
Payment authorization
Authentication & validation
Issuer Bank

POS Bridge Solution is an integration tool devised to record all transactions performed on the POS via the merchant billing system irrespective of the payment mode Integrated (inside Ezetap Terminal) and Non- integrated (outside of Ezetap Terminal). In addition to recording all transactions, the solution assists merchants POS Bridge to integrate with the merchant's billing system for better reconciliation and reporting needs (through Ezetap VAS services).

Action Flow for Android POS:

The Android POS is an Android based Smart device. It is a standalone SIM + Wi-Fi enabled device, which can enable various modes of payments such as

- Cash
- Cards
- Wallets
- UPI
- Remote Pay – Payment Links

The transaction flow in case of Android POS is as follows:

- Agent logins into the device
- After billing is done on the Billing system, ‘Pay’ button is pressed on the Billing System (or Billing Application), (payment mode e.g. CARD/UPI etc. can be selected on Billing System).
- At this point an API call is made to the Ezetap Server which is routed to the payment device.
- Device receives notification along with the payment context (e.g. unique reference number, amount, etc.) .
- Considering card transactions over here, the Agent asks the customer to insert the card and further the customer enters the pin.
- Further authentication happens based on the pin entry and once the payment is authenticated the same response is updated back to the device.
- Billing system is updated (if Integrated) with the transaction details as well via API call.

How to get started with Ezetap’s POS Bridge Integration?

We recommend that you get started with a Demo environment test, validate the flow and use cases and then move on to the Production environment

Here is what you will need to start the Demo Integration

- Demo merchant account (Please provide a Mobile Number and address for us to create your Merchant Account)
- Demo App Key: You will use this as a part of the code to Initiate the Application (more details in the API details Segment)
- Demo Device (Please provide an address where we can deliver a Demo Device to you to start off the Integration process)

You can reach-out to the Ezetap point of contact **during the entire process of Integration in** case of support or troubleshooting.

Advantages

- One stop solution for collecting payment across multiple payment modes
- Avoid any manual intervention to accept payments faster and error free
- Reduces cash pilferage
- Act as a single source of truth to get all your transaction for any payment mode present on your billing system
- Save effort and time for reconciliation due to data availability across pay modes
- Flexibility to record payment modes without any changes at merchant end

API Details

Note: All request/responses are of type application/json

To Send notification for POS Bridge from merchant side

Description: API to send POS Bridge notification on Ezetap device.

- Demo URL:**https://demo.ezetap.com/api/3.0/p2padapter/pay**
- Production URL**: https://www.ezetap.com/api/3.0/p2padapter/pay**
- HTTP Method: POST
- Request HTTP Headers: Content-Type: application/json

Request Body

Parameter	Datatype	Mandatory	Description
appKey	String(50)	Yes	API key shared by Ezetap
pushTo	Json with “deviceId” as String (250)	Yes	PushTo is a json with destination info like Device id of the pos on which notification will be sent
username	String (20)	Yes	Username must be passed
description	String (50)	No	It describes the notification purpose
amount	BigDecimal	Yes	Transaction amount

amountCashBack	BigDecimal	No	CashBack amount if required
amountAdditional	BigDecimal	No	Additional Amount if required
externalRefNumber	String	Yes	Reference Number of the Bill
customerMobile Number	String	No	Customer mobile number
customerEmail	String	No	Customer’s email id
customerName	String	No	Customer’s Name
accountLabel	String	No	Label is a tag. It is used for passing MID/TID label in multi-TID case
externalRef Number2	String	No	

externalRef Number3	String	No	
externalRef Number4	String	No	
externalRef Numbers	Json String	No	
additionalData	String	No	AdditionalData is json
orgCode	String	No	Merchant’s orgCode
paymentBy	String	No	For service fees configuration, this is DEBIT|CREDIT

mode	String	Yes	Mandatory field values accepted(ALL,CARD, CASH, UPI, BHARATQR, etc.). Note : “ALL” needs to be passed in case all the available payment modes are to be shown on the device. Specific non-integrated payment modes to be configured on a case to case basis and mode value to be provisioned.
Emi Type	String	No	EMI” to be used for CARD/NBFC related EMI transactions

Response Body

Parameter	Datatype	Description
success	Boolean	Indicates notification has been started successful or not
errorCode	String(48)	Error code if notification has not been initiated due to either token issue or wrong deviceId . It is empty for a successful creation
errorMessage	String(256)	Descriptive error message for the above error code on notification initiation. This field will be empty for a successful initiation.
p2pRequestId		Unique identifier generated for each payment request sent to the device successfully.

Sample Request & Response

Parameter	Sample Response

{
"username": "",
"appKey": “",
"amount": "",
"accountLabel":"",
"customerMobileNumber": "",
"externalRefNumber": "",
"externalRefNumber2": "",
"externalRefNumber4": "",
"externalRefNumbers": [
],
"pushTo":{"deviceId":"<<enter
device serial number>>|
ezetap_android
"},
"mode
	{
	"success": true,
	"messageCode": null,
	"message": null,
	"errorCode": null,
	"errorMessage": null,
	"realCode": null,
	"apiMessageTitle": null,
	"apiMessage": null,
	"apiMessageText": null,
	"apiWarning": null,
	"p2pRequestId": ""
	}

Sample Request & Response for Multi MID /TID Scenario using Account Label

Account Label is used in a scenario when a single device has to be used for payment settlement to multiple settlement accounts/ TIDs. In such a scenario each TID is marked to one account label. While initiating a payment request the specific account label is passed in the Pay API request.

For instance , there are 2 account labels 'AC1' corresponding to TID1 and 'AC2' corresponding to TID2. For the above 2 the request will look as below:

Sample Request for AC1	Sample Response

{
"appKey":"demo app key",
"username":"4444001234",
"amount":"2100",
"customerMobileNumber":"",
"externalRefNumber":"Order ID",
"externalRefNumber2":"",
"externalRefNumber3":"",
"accountLabel": "AC1",
"customerEmail":"Testmail@gmail.com",
"pushTo":{"deviceId":"1850319432|ezetap_android""}
mode:"CARD"
}
	{
	"success": true,
	"messageCode": null,
	"message": null,
	"errorCode": null,
	"errorMessage": null,
	"realCode": null,
	"apiMessageTitle": null,
	"apiMessage": null,
	"apiMessageText": null,
	"apiWarning": null,
	"p2pRequestId":
	"190124182233567E010078004"
	}

Sample Request for AC2	Sample Response

{
"appKey":"demo app key",
"username":"4444001234",
"amount":"2100",
"customerMobileNumber":"",
"externalRefNumber":"Order ID",
"externalRefNumber2":"",
"externalRefNumber3":"",
"accountLabel": "AC2",
"customerEmail":"Testmail@gmail.c
om",
"pushTo":{"deviceId":"1850319432|
ezetap_android""}
mode:"CARD"
}
	{
	"success": true,
	"messageCode": null,
	"message": null,
	"errorCode": null,
	"errorMessage": null,
	"realCode": null,
	"apiMessageTitle": null,
	"apiMessage": null,
	"apiMessageText": null,
	"apiWarning": null,
	"p2pRequestId":
	"190124182233567E010078005"
	}

Error Codes

Error Code	Description	Resolution Steps
EZETAP_0000382	Device not found - The correct device ID is not passed	Pass correct device serial number
EZETAP_0000385	Device is not in the Network.	Check WiFi connectivity or Mobile Data or MPOS App is login
EZETAP_0000039	Payment amount unsupported.	Pass correct value for amount
EZETAP_0000050	Transaction amount greater than Limit.	Pass correct value for amount
EZETAP_0000162	Transaction amount less than Limit.	Pass correct value for amount
EZETAP_0000048	Payment tip not enabled.	Check with integration team (This is required only when tip needs to be collected at the time of payment)

EZETAP_0000148	Invalid Org, device does not belong to the org.	Check with integration team
EZETAP_0000047	Payment tip amount error.	Check with integration team (This is required only when tip needs to be collected at the time of payment)
EZETAP_0000387	ExternalRefNumber field is empty.	ExternalRefNumber is a mandatory parameter. Please pass a unique reference id
EZETAP_6000001	No such payment mode exists.	Pass correct payment mode
EZETAP_0000623	Device is busy with pending notification	Before initiating a new request complete/cancel the payment request on the device

To get the status of POS Bridge notification from Merchant's side

Description: API to get status of POS Bridge notification sent on ezetap device

- Demo URL: **https://demo.ezetap.com/api/3.0/p2padapter/status**
- Production: **https://www.ezetap.com/api/3.0/p2padapter/status**
- HTTP Method:: POST
- Request HTTP Headers: Content-Type: application/json

The status = AUTHORIZED field is highlighted in yellow in response.

Parameter	Datatype	Description
username	String	Any username (Mandatory)
appKey	String	App key (Mandatory)
origP2pRequestId		POS Bridge request ID returned by the Pay API call (Mandatory)

Response Body

Parameter	Datatype	Description
success	Boolean	Indicates notification found
errorCode	String(48)	Error code if notification is not found.
errorMessage	String(256)	Descriptive error message for the above error code on notification initiation. This field will be empty for a successful initiation.
message	String	It is the descriptive information of POS Bridge status, it also tells what the current status is.
messageCode	String	Message Code if notification status found

Sample Request & Response

Sample Request for AC2	Sample Response

{
"username":"<Your username here>",
"appKey":"<your appKey here>",
"origP2pRequestId":"<Use the one returned by the pay API>"
}
	Success Response:
	{
	"success": true,
	"messageCode": "P2P_DEVICE_TXN_DONE",
	"message": "Transaction done on device, Please look at Txn status.",
	"realCode": "P2P_DEVICE_TXN_DONE",
	"sessionKey": "",
	"username": "",
	"setting": {},
	"apps": [],
	"amount":,
	"amountAdditional": 0.00,
	"amountOriginal":,
	"amountCashBack": 0.00,
	"authCode": "",
	"batchNumber": "",
	"cardLastFourDigit": "",
	"currencyCode": "INR",
	"customerName": "",
	"customerMobile": "",
	"customerReceiptUrl": "http/",
	"deviceSerial": "",
	"externalRefNumber": "",
	"externalRefNumber2": "",
	"externalRefNumber3": "",
	"externalRefNumber4": "",
	"externalRefNumber5": "",
	"externalRefNumbers": [
	""
	],
	"formattedPan": "4639-17XX-XXXX-2274",
	"txnId": "221012081024502E010066434",
	"latitude": "22.1046222",
	"longitude": "82.1480449",
	"merchantName": "",
	"mid": "",
	"nonceStatus": "OPEN",
	"orgCode": "",
	"merchantCode": "",
	"payerName": "",
	"paymentCardBin": "",
	"paymentCardBrand": "VISA",
	"paymentCardType": "CREDIT",
	"pgInvoiceNumber": "210",
	"postingDate": 1665582029000,
	"processCode": "_DEF_PROC",
	"rrNumber": "RR25071",
	"settlementStatus": "PENDING",
	"status": "AUTHORIZED",
	"states": [
	"AUTHORIZED"
	],
	"tid": "22223340",
	"userMobile": "",
	"txnType": "CHARGE",
	"dccOpted": false,
	"cardHolderCurrencyExponent": 0,
	"userAgreement": "I agree to pay as per the card issuer agreement and receive chargeslip by electronic means.",
	"signable": true,
	"voidable": true,
	"refundable": false,
	"chargeSlipDate": "2022-10-12T13:40:29+0530",
	"readableChargeSlipDate": "12/10/2022 13:40:29",
	"cardTxnTypeDesc": "EMV with Online PIN ByPass",
	"issuerCode": "HDFC",
	"maximumPayAttemptsAllowed": 1,
	"maximumSuccessfulPaymentAllowed": 1,
	"noExpiryFlag": false,
	"dxMode": "SERIAL",
	"receiptUrl": "http://d.eze.cc/r/o/R74xNQuJ/",
	"signReqd": true,
	"txnTypeDesc": "Charge",
	"acquirerCode": "DUMMY",
	"additionalParamJson": "{\"\"}",
	"createdTime": 1665582025000,
	"customerNameAvailable": true,
	"callbackEnabled": false,
	"accountLabel": "",
	"onlineRefundable": false,
	"additionalAmount": 0.00,
	"orderNumber": "",
	"reverseReferenceNumber": "",
	"totalAmount": 0,
	"displayPAN": "",
	"nameOnCard": "",
	"invoiceNumber": "210",
	"cardType": "VISA",
	"tipEnabled": false,
	"callTC": false,
	"acquisitionId": "",
	"acquisitionKey": "",
	"processCronOutput": false,
	"externalDevice": false,
	"tipAdjusted": false,
	"txnMetadata": [],
	"middlewareStanNumber": 0,
	"otpRequired": false,
	"p2pRequestId": "221012081009089E010092433",
	"mode": "CARD",
	"reload": false,
	"redirect": false,
	"twoStepConfirmPreAuth": false
	}

Failed Response:


```
{
    "success": true,
    "messageCode": "P2P_DEVICE_TXN_DONE",
    "message": "Transaction done on device, Please look at Txn status.",
    "errorCode": "EZETAP_1000003",
    "errorMessage": "Card Declined. Please try again. If the problem persists, please try another card or contact card issuer.",
    "realCode": "P2P_DEVICE_TXN_DONE",
    "sessionKey": "da3c872e-f922-4b71-a564-b7f0fb05659c",
    "username": "4488221103",
    "setting": {},
    "apps": [],
    "amount": 531.00,
    "amountAdditional": 0.00,
    "amountOriginal": 531.00,
    "amountCashBack": 0.00,
    "authCode": "N/A",
    "batchNumber": "1",
    "cardLastFourDigit": "2274",
    "currencyCode": "AED",
    "customerName": "SHASHANK A",
    "customerReceiptUrl": "",
    "deviceSerial": "0821260712",
    "externalRefNumber": "INVOICE_POS Bridge_070055",
    "formattedPan": "4639-17XX-XXXX-2274",
    "txnId": "220221130321031E010836798",
    "merchantName": "Test",
    "mid": "200599999029",
    "nonceStatus": "CLOSED",
    "orgCode": "Test_4952948",
    "merchantCode": "Test_4952948",
    "payerName": "SHASHANK A",
    "paymentCardBin": "463917",
    "paymentCardBrand": "VISA",
    "paymentCardType": "CREDIT",
    "paymentMode": "CARD",
    "pgInvoiceNumber": "2",
    "postingDate": 1645463002000,
    "processCode": "_DEF_PROC",
    "rrNumber": "N/A",
    "settlementStatus": "FAILED",
    "status": "FAILED",
    "states": [
        "FAILED"
    ],
    "tid": "45200006",
    "userMobile": "4488221103",
    "txnType": "CHARGE",
    "dccOpted": false,
    "cardHolderCurrencyExponent": 0,
    "userAgreement": "",
    "signable": false,
    "voidable": false,
    "refundable": false,
    "chargeSlipDate": "2022-02-21T17:03:22+0400",
    "readableChargeSlipDate": "21/02/2022 17:03:22",
    "cardTxnTypeDesc": "EMV with Online PIN ByPass",
    "issuerCode": "HDFC",
    "maximumPayAttemptsAllowed": 1,
    "maximumSuccessfulPaymentAllowed": 1,
    "noExpiryFlag": false,
    "dxMode": "WIFI",
    "receiptUrl": "",
    "signReqd": false,
    "txnTypeDesc": "Charge",
    "acquirerCode": "MASHREQ",
    "additionalParamJson": "{\"FPIKSN\":\"FFFF005A610623000002\",\"MCC\":\"7399\"}",
    "createdTime": 1645463001000,
    "customerNameAvailable": true,
    "callbackEnabled": false,
    "onlineRefundable": false,
   "additionalAmount": 0.00,
    "orderNumber": "INVOICE_POS Bridge_070055",
    "reverseReferenceNumber": "N/A",
    "totalAmount": 531.00,
    "displayPAN": "2274",
    "nameOnCard": "SHASHANK A",
    "invoiceNumber": "2",
    "cardType": "VISA",
    "tipEnabled": false,
    "callTC": false,
    "acquisitionId": "",
    "acquisitionKey": "",
    "processCronOutput": false,
    "externalDevice": false,
    "tipAdjusted": false,
    "txnMetadata": [],
    "middlewareStanNumber": 0,
    "otpRequired": false,
    "p2pRequestId": "220221130303280E010067797",
    "redirect": false,
    "reload": false,
    "twoStepConfirmPreAuth": false
}
```

Error Codes

EZETAP_0000383 – Notification not found for this refNumber

Message Codes

Message Code	Description
P2P_DEVICE_RECEIVED	Notification received on POS device
P2P_DEVICE_SENT	If notification is sent to the device
P2P_STATUS_QUEUED	If notification is queued on the server
P2P_STATUS_IN_EXPIRED	On Notification expiration
P2P_DEVICE_TXN_DONE	Transaction completed on POS device, need to look at txn fields.
P2P_STATUS_UNKNOWN	POS Bridge Notification Status is in Unknown state.
P2P_DEVICE_CANCELED	POS Bridge Notification has been Canceled on Receiving device.
P2P_STATUS_IN_CANCELED _FROM_EXTERNAL_SYSTEM	Canceled successfully by POS Bridge Cancel API

NOTE – Success response will have Message_Code which tells what the current status of POS Bridge Notification is. For any successful transaction please refer to the field called Status in the response of status API. If the Status = AUTHORIZED only then the payment is considered as successful.

To cancel a notification from Merchant’s side

Description:

- API to cancel a notification that is queued up on the server or has been received on the device.
- Cancellation will happen even when the device has received notification. In such a scenario it will prompt the user “notification cancel” message and remove the notification.
- If the device user went on with the flow of taking PIN (in case of card transaction), then cancellation only happens on submission of payment with an error prompt “Notification canceled from billing system”
- In the above scenario, the cancellation request will be accepted until the device has submitted payment. Otherwise, it will respond with P2P_PAYMENT_INITIATED error code.

- Demo URL: **https://demo.ezetap.com/api/3.0/p2padapter/cancel**
- Production: https://www.ezetap.com/api/3.0/p2padapter/cancel
- HTTP Method:: POST
- Request HTTP Headers: Content-Type: application/json

Request Body

Parameter	Datatype	Description
username	String	Any username (Mandatory)
appKey	String	App key (Mandatory)
origP2pRequestId	String	Request ID returned by the pay API call (Mandatory)
pushTo	Json with “deviceId” as String (250)	PushTo is a json with destination info like Device id of the pos on which notification will be sent (Mandatory)

Sample Request & Response

Sample Request	Sample Response

{
"username":"<Your username here>",
"appKey":"<your appKey here>",
"origP2pRequestId":"190113035348970E0200",
}
	{
	"success": true,
	"messageCode": null,
	"message": null,
	"errorCode": null,
	"errorMessage": null,
	"realCode": null,
	"apiMessageTitle": null,
	"apiMessage": null,
	"apiMessageText": null,
	"apiWarning": null,
	"origP2pRequestId": "1901131948341E010055004"
	}

BEST PRATCISES

We suggest the following sequence for calling these APIs

- Call Pay API to initiate the transaction

- Call the Status Check API every 10 seconds till the 150th second from 30th second. Look for the status API response fields “status” and “messageCode”:

- If the “status” field is not present and
  - Case 1: If “message code” = P2P_DEVICE_RECEIVED then keep calling status API further till 150th second every 10 seconds.
  - Case 2: If “message code” = P2P_DEVICE_CANCELED then transaction flow is complete. Calls to Status API can be stopped and new transaction can be initiated.

- If the “status” = AUTHORIZED and “message code” = P2P_DEVICE_TXN_DONE, then transaction flow is complete. Calls to Status API can be stopped and new transaction can be initiated.

- If the “status” = FAILED and “message code” = P2P_DEVICE_TXN_DONE then then transaction flow is complete. Calls to Status API can be stopped and new transaction can be initiated.

- In case the “status” field is not present and “message code” = P2P_DEVICE_RECEIVED even on the 150th second instance, then call the cancel API to cancel the notification present on the device at that instance.

- Note: Time instances highlighted in blue above are suggested best practices from Ezetap and are configurable.

Below is the table on what action is to be taken based on specific status API responses :

API filed = "messageCode"	API filed = "message"	API filed = "status"	Action to be taken	Txn Status on Ezetap
P2P_DEVICE_RECEIVED	Notification has been received on POS Device.	API Field not present	Keep checking the status calling Pay API	Pending
P2P_STATUS_QUEUED	P2P_STATUS_QUEUED	API Field not present	Keep checking the status calling Pay API	Pending
P2P_DEVICE_TXN_DONE	Transaction done on device, Please look at Txn status.	EXPIRED	Stop calling status API	Expired
P2P_DEVICE_TXN_DONE	Transaction done on device, Please look at Txn status.	FAILED	Stop calling status API	Txn Failed
P2P_DEVICE_TXN_DONE	Transaction done on device, Please look at Txn status.	AUTHORIZED	Stop calling status API	Txn performed successfully
P2P_DEVICE_CANCELED	PushToPay Notification has been Canceled on Receiving device.	API Field not present	Stop calling status API	Txn notification has been cancelled
P2P_STATUS_IN_CANCELED_FROM_EXTERNAL_SYSTEM	PushToPay Notification has been Canceled from Billing/ External System.	API Field not present	Stop calling status API	Txn notification has been cancelled

Testing the Integration

A detailed testing scenario is a crucial and critical step before the roll-out to production. You can test all payment related integration features End-to-End including failure cases, PG errors etc by using the accompanying Amounts on the right in the below table. Please pass the following amount value such as 501, 505 etc. as mentioned below and you will get the corresponding error messages.

408	Payment Gateway Takes 3 minutes to respond
410	Timeout
501	Call Issuer
502	Call Referral
504	Pick up
505	Do not Honor
508	Honor with Id
512	Invalid Transaction
513	Invalid Amount
514	Invalid Card Number
515	No such Issuer
519	Try after 1 min
522	Susp Malfunction
523	Trans Fee Error
526	Duplicate Record
531	Declined
533	Expired Card
534	Suspected Card
535	Contact Acquirer
536	Restricted Card
539	No Credit Account
542	Timeout
556	No Card Record
561	Above Amount Limit
585	Batch Not Found
590	Cutoff in Process
591	Host Unavailable
592	Routing Problem
596	Err – Invalid Message
598	Kx Required
599	Session Expired
666	Payment Gateway takes 1.5 minutes to respond

Solution Capabilities

Ezetap POS Bridge Solution will work with all the below mentioned platforms:

- Windows
- IOS
- Android
- Linux

Steps to Go-Live

- Having confirmed, tested the integration and various scenarios, here are the steps you need to follow for Going-live
- Procure a Production device via the bank/ Ezetap as per your agreement
- Inform Ezetap once the device has been procured,
- Ezetap team will provide you the Production App Key, just add the Production App key in your demo code to the current one
- Replace the demo URLs with the production URLs
- Replace your current username, appkey with the one provided to you for production environment

Note: Ezetap's POS Bridge supports Java JDK of 1.7 v and above. This is due to mandates from banks and certifying authorities to support a minimum standard of security certifications and ensure secure transactions and support.

What do we need for Go-Live?

- Ezetap would need a list of users who would be accessing the Portal as well as the Device
- Who is procuring the SIM cards? Please confirm with your procurement team/decision makers who will be procuring the devices
- Shipment address of all the stores where installation has to be done.

RAZORPAY POS'S INTEGRATED DQR SOLUTION

What is the POS's DQR solution?

Razorpay POS has availability of customer facing Dynamic QR devices, wherein Dynamic QRs can be generated from the billing system in an integrated mode itself. This ensures transparency and smooth reconciliation of transaction at billing system's end.

API Details

Note: All request/responses are of type application/json

To Send notification for POS Bridge from merchant side:

Description: API to send POS Bridge notification on Ezetap device.

- Demo URL:**https://demo.ezetap.com/api/3.0/p2padapter/pay**
- Production URL: **https://www.ezetap.com/api/3.0/p2padapter/pay**
- HTTP Method: POST
- Request HTTP Headers: Content-Type: application/json

Request Body

Parameter	Datatype	Mandatory	Description
appKey	String(50)	Yes	API key shared by Ezetap
pushTo	Json with "deviceId" as String (250)	Yes	PushTo is a json with destination info like Device id of the pos on which notification will be sent
username	String (20)	Yes	Username must be passed
description	String (50)	No	It describes the notification purpose
amount	BigDecimal	Yes	Transaction amount
amountCashBack	BigDecimal	No	CashBack amount if required
amountAdditional	BigDecimal	No	Additional Amount if required
externalRefNumber	String	Yes	Reference Number of the Bill
customerMobile Number	String	No	Customer mobile number
customerEmail	String	No	Customer's email id
customerName	String	No	Customer's Name
accountLabel	String	No	Label is a tag. It is used for passing MID/TID label in multi-TID case
externalRef Number2	String	No	
externalRef Number3	String	No	
externalRef Number4	String	No	
externalRef Numbers	Json String	No	
additionalData	String	No	AdditionalData is json

Response Body

Parameter	Datatype	Description
success	Boolean	Indicates notification has been started successful or not
errorCode	String(48)	Error code if notification has not been initiated due to either token issue or wrong deviceId . It is empty for a successful creation
errorMessage	String(256)	Descriptive error message for the above error code on notification initiation. This field will be empty for a successful initiation.

Sample Request & Response

Sample Request	Sample Response
{	{
"username": "",	"success": true,
"appKey": “",	"messageCode": null,
"amount": "",	"message": null,
"accountLabel":"",	"errorCode": null,
"customerMobileNumber": "",	"errorMessage": null,
"externalRefNumber": "",	"realCode": null,
"externalRefNumber2": "",	"apiMessageTitle": null,
"externalRefNumber4": "",	"apiMessage": null,
"externalRefNumbers": [	"apiMessageText": null,
],	"apiWarning": null,
"pushTo":{"deviceId":"<<enter device serial number>>|razorpay_pos_soundbox	"p2pRequestId": ""
"},	}
"mode": "UPI"	
}	

Sample Request for AC2	Sample Response

{
"appKey":"demo app key",
"username":"4444001234",
"amount":"2100",
"customerMobileNumber":"",
"externalRefNumber":"Order ID",
"externalRefNumber2":"",
"externalRefNumber3":"",
"customerEmail":"Testmail@gmail.c
om",
"pushTo":
{"deviceId":"38230908450035|
razorpay_pos_soundbox"}
mode:"UPI"
}
	{
	"success": true,
	"messageCode": null,
	"message": null,
	"errorCode": null,
	"errorMessage": null,
	"realCode": null,
	"apiMessageTitle": null,
	"apiMessage": null,
	"apiMessageText": null,
	"apiWarning": null,
	"p2pRequestId": "190124182233567E010078004"
	}

Error Codes

Error Code	Description	Resolution Steps
EZETAP_0000382	Device not found - The correct device ID is not passed	Pass correct device serial number
EZETAP_0000381	Android FCM token not found (Android Device)	Check WiFi connectivity or Mobile Data or MPOS App is login
EZETAP_0000384	Firebase FCM error (Android Device)	Pass correct value for amount
EZETAP_0000385	Device is not in the Network.	Pass correct value for amount
EZETAP_0000039	Payment amount unsupported.	Pass correct value for amount
EZETAP_0000050	Transaction amount greater than Limit.	Check with integration team (This is required only when tip needs to be collected at the time of payment)
EZETAP_0000162	Transaction amount less than Limit.	Check with integration team
EZETAP_0000048	Payment tip not enabled.	Check with integration team (This is required only when tip needs to be collected at the time of payment)
EZETAP_0000148	Invalid Org, device does not belong to the org.	ExternalRefNumber is a mandatory parameter. Please pass a unique reference id
EZETAP_0000047	Payment tip amount error.	Pass correct payment mode
EZETAP_0000387	ExternalRefNumber field is empty.	Before initiating a new request complete/cancel the payment request on the device
EZETAP_6000001	No such payment mode exists.	
EZETAP_0000623	Device is busy with pending notification	

To get the status of POS Bridge notification from Merchant's side

Description: API to get status of POS Bridge notification sent on ezetap device

- Demo URL: https://demo.ezetap.com/api/3.0/p2padapter/status
- Production: **https://www.ezetap.com/api/3.0/p2padapter/status**
- HTTP Method:: POST
- Request HTTP Headers: Content-Type: application/json

The status = AUTHORIZED field is highlighted in yellow in response.

The status = AUTHORIZED field is highlighted in yellow in response.

Request Body

Parameter	Datatype	Description
username	String	Any username (Mandatory)
appKey	String	App key (Mandatory)
origP2pRequestId	String	POS Bridge request ID returned by the pay

Response Body

Parameter	Datatype	Description
success	Boolean	Indicates notification found
errorCode	String(48)	Error code if notification is not found.
errorMessage	String(256)	Descriptive error message for the above error code on notification initiation. This field will be empty for a successful initiation.
message	String	It is the descriptive information of POS Bridge status, it also tells what the current status is.
messageCode	String	Message Code if notification status found

Sample Request & Response

Sample Request	Sample Response

{
"username":"<Your username here>",
"appKey":"<your appkey here>",
"origP2pRequestid":"<Use the one returned by the pay API>"
}
	Success Response:
	{
	"success": true,
	"messageCode": "P2P_DEVICE_TXN_DONE",
	"message": "Transaction done on device, Please look at Txn status.",
	"realCode": "P2P_DEVICE_TXN_DONE",
	"sessionKey": "",
	"username": "",
	"setting": [],
	"apps": [],
	"amount",
	"amountAdditional": 0.00,
	"amountOriginal",
	"amountCashBack": 0.00,
	"authCode": "",
	"batchNumber": "",
	"cardLastFourDigit": "",
	"currencyCode": "INR",
	"customerName": "",
	"customerMobile": "",
	"customerReceiptUrl": "http]",
	"deviceSerial": "",
	"externalRefNumber1": "",
	"externalRefNumber2": "",
	"externalRefNumber3": "",
	"externalRefNumber4": "",
	"externalRefNumber5": "",
	"externalRefNumber6": [
	"",
	],
	"formattedPan": "4639-17XX-XXXX-2274",
	"txnid": "22101208102450201010066434",
	"latitude": "22.1046222",
	"longitude": "82.1480449",
	"merchantName": "",
	"mid": "",
	"nonceStatus": "OPEN",
	"orgCode": "",
	"merchantCode": "",
	"payerName": "",
	"paymentCardBin": "",
	"paymentCardBrand": "VISA",
	"paymentCardType": "CREDIT",
	"pgInvoiceNumber": "210",
	"postingDate": 1665558229000,
	"processCode": "_DEF_PROC",
	"rrNumber": "RR25071",
	"settlementStatus": "PENDING",
	"status": "AUTHORIZED",
	"states": [
	"AUTHORIZED"
	],
	"tid": "22223340",
	"userMobile": "",
	"txnType": "CHARGE",
	"dcoOpted": false,
	"cardHolderCurrencyExponent": 0,
	"userAgreement": "I agree to pay as per the card issuer agreement and receive chargeslip by electronic means.",
	"signable": true,
	"voidable": true,
	"refundable": false,
	"chargeSlipDate": "2022-10-13T13:40:29+0530",
	"readableChargeSlipDate": "12/10/2022 13:40:29",
	"cardTxnTypeDesc": "EMV with Online PIN ByPass",
	"IssuerCode": "HDFC",
	"maximumPayAttemptsAllowed": 1,
	"maximumSuccessfulPaymentAllowed": 1,
	"noExpiryFlag": false,
	"dsMode": "SERIAL",
	"receiptUrl": "http://d.eze.cc/l/a/R74sNQuJ/",
	"signReqd": true,
	"txnTypeDesc": "Charge",
	"acquirerCode": "DUMMY",
	"additionalParamJson": "{\"\":\"\"}",
	"createdTime": 1665558205000,
	"customerNameAvailable": true,
	"callbackEnabled": false,
	"accountLabel": "",
	"onlineRefundable": false,
	"additionalAmount": 0.00,
	"orderNumber": "",
	"reverseReferenceNumber": "",
	"totalAmount": 0,
	"displayPAN": "",
	"nameOnCard": "",
	"invoiceNumber": "210",
	"cardType": "VISA",
	"tipEnabled": false,
	"callTC": false,
	"acquisitionId": "",
	"acquisitionKey": "",
	"processCronOutput": false,
	"externalDevice": false,
	"tipAdjusted": false,
	"txnMetadata": [],
	"middlewareStanNumber": 0,
	"otpRequired": false,
	"p2pRequestId": "22101208100908961010092433",
	"mode": "CARD",
	"reload": false,
	"redirect": false,
	"twoStepConfirmPreAuth": false
	}
	
	Failed Response:
	{
	"success": true,
	"messageCode": "P2P_DEVICE_TXN_DONE",
	"message": "Transaction done on device, Please look at Txn status.",
	"errorCode": "EZETAP_1000003",
	"errorMessage": "Card Declined. Please try again. If the problem persists, please try another card or contact card issuer.",
	"realCode": "P2P_DEVICE_TXN_DONE",
	"sessionKey": "ec3c8f2e-1922-4b71-a564-b7f0fb05659c",
	"username": "4488221103",
	"setting": [],
	"apps": [],
	"amount": 531.00,
	"amountAdditional": 0.00,
	"amountOriginal": 531.00,
	"amountCashBack": 0.00,
	"authCode": "N/A",
	"batchNumber": "1",
	"cardLastFourDigit": "2274",
	"currencyCode": "AED",
	"customerName": "SHASHANK A",
	"customerReceiptUrl": "",
	"deviceSerial": "08212160712",
	"externalRefNumber": "INVOICE_POS Bridge_070055",
	"formattedPan": "4639-17XX-XXXX-2274",
	"txnid": "22022113030210316010836798",
	"merchantName": "Test",
	"mid": "20059999029",
	"nonceStatus": "CLOSED",
	"orgCode": "Test_4952948",
	"merchantCode": "Test_4952948",
	"payerName": "SHASHANK A",
	"paymentCardBin": "463917",
	"paymentCardBrand": "VISA",
	"paymentCardType": "CREDIT",
	"paymentMode": "CARD",
	"pgInvoiceNumber": "2",
	"postingDate": 1645463002000,
	"processCode": "_DEF_PROC",
	"rrNumber": "N/A",
	"settlementStatus": "FAILED",
	"status": "FAILED",
	"states": [
	"FAILED"
	],
	"tid": "45200006",
	"userMobile": "4488221103",
	"txnType": "CHARGE",
	"dcoOpted": false,
	"cardHolderCurrencyExponent": 0,
	"userAgreement": "",
	"signable": false,
	"voidable": false,
	"refundable": false,
	"chargeSlipDate": "2022-02-21T17:03:22+0400",
	"readableChargeSlipDate": "21/02/2022 17:03:22",
	"cardTxnTypeDesc": "EMV with Online PIN ByPass",
	"IssuerCode": "HDFC",
	"maximumPayAttemptsAllowed": 1,
	"maximumSuccessfulPaymentAllowed": 1,
	"noExpiryFlag": false,
	"dsMode": "WIFI",
	"receiptUrl": "",
	"signReqd": false,
	"txnTypeDesc": "Charge",
	"acquirerCode": "MASHREQ",
	"additionalParamJson": "{\"TPIKEN\":\"[\\\"FFFFD0A5B0B2D000D01\\\",\\\"C1\\\",\\\"7399\\\"]\"}",
	"createdTime": 1645463001000,
	"customerNameAvailable": true,
	"callbackEnabled": false,
	"onlineRefundable": false,
	"additionalAmount": 0.00,
	"orderNumber": "INVOICE_POS Bridge_070055",
	"reverseReferenceNumber": "N/A",
	"totalAmount": 531.00,
	"displayPAN": "2274",
	"nameOnCard": "SHASHANK A",
	"invoiceNumber": "2",
	"cardType": "VISA",
	"tipEnabled": false,
	"callTC": false,
	"acquisitionId": "",
	"acquisitionKey": "",
	"processCronOutput": false,
	"externalDevice": false,
	"tipAdjusted": false,
	"txnMetadata": [],
	"middlewareStanNumber": 0,
	"otpRequired": false,
	"p2pRequestId": "22022113030328010101067797",
	"redirect": false,
	"reload": false,
	"twoStepConfirmPreAuth": false
	}

Error Codes

EZETAP_0000383 – Notification not found for this refNumber

Message Codes

Message Code	Description
P2P_DEVICE_RECEIVED	Notification received on POS device
P2P_DEVICE_SENT	If notification is sent to the device
P2P_STATUS_QUEUED	If notification is queued on the server
P2P_STATUS_IN_EXPIRED	On Notification expiration
P2P_DEVICE_TXN_DONE	Transaction completed on POS device, need to look at txn fields.
P2P_STATUS_UNKNOWN	POS Bridge Notification Status is in Unknown state.
P2P_DEVICE_CANCELED	POS Bridge Notification has been Canceled on Receiving device.
P2P_STATUS_IN_CANCELED _FROM_EXTERNAL_SYSTEM	Canceled successfully by POS Bridge Cancel API

NOTE – Success response will have Message_Code which tells what the current status of POS Bridge Notification is. For any successful transaction please refer to the field called Status in the response of status API. **If the Status = AUTHORIZED only then the payment is considered as successful.**

To cancel a notification from Merchant's side

Description:

- API to cancel a notification that is queued up on the server or has been received on the device.
- Cancellation will happen even when the device has received notification. In such a scenario it will prompt the user "notification cancel" message and remove the notification.
- If the device user went on with the flow of taking PIN (in case of card transaction), then cancellation only happens on submission of payment with an error prompt "Notification canceled from billing system"
- In the above scenario, the cancellation request will be accepted until the device has submitted payment. Otherwise, it will respond with P2P_PAYMENT_INITIATED error code.

- Demo URL: **https://demo.ezetap.com/api/3.0/p2padapter/cancel**
- Production: https://www.ezetap.com/api/3.0/p2padapter/cancel
- HTTP Method:: POST
- Request HTTP Headers: Content-Type: application/json

Request Body

Parameter	Datatype	Description
username	String	Any username (Mandatory)
appKey	String	App key (Mandatory)
origP2pRequestId	String	Request ID returned by the pay API call (Mandatory)
pushTo	Json with “deviceId” as String (250)	PushTo is a json with destination info like Device id of the pos on which notification will be sent (Mandatory)

Sample Request & Response

Sample Request	Sample Response

{
"username":"<Your username here>",
"appKey":"<your appKey here>",
"origP2pRequestId":"190113035348970E0200
",
}
	{
	"success": true,
	"messageCode": null,
	"message": null,
	"errorCode": null,
	"errorMessage": null,
	"realCode": null,
	"apiMessageTitle": null,
	"apiMessage": null,
	"apiMessageText": null,
	"apiWarning": null,
	"origP2pRequestId": "190113194834 1E010055004"
	}

Please Note : For DQR Solution - Best Practices, Integration Testing, Solution Capabilities and Steps to Go-Live remain the same as for POS Bridge Solution.

RAZORPAY POS'S ANDROID SDK

RAZORPAY POS SDK INTRODUCTION

Razorpay POS is a digital payments platform that makes your payment acceptance simple and configurable. This Universal Payment Acceptance platform is a smart, simple end-to-end payments platform which is unique in its vision and architecture.
It can be deeply integrated with any enterprise system/sales channel to offer a unified and seamless payments experience to the end customer.

WHY RAZORPAY POS SDK?

- Enterprises who have sales and collection application already built and would like to accept digital Payments via the application. Razorpay POS's android SDKs are available to be plugged in to the existing application for payments acceptance.
- This integration is fast and easy, and typically takes around 2-3 days of time to complete.
- Enterprises can maintain one single application for business process as well as payments and can future proof their payments capabilities with easy enablement of newer modes of payments.
- **NOTE:** Razorpay POS android SDK will only support Android version 5.0 and above. This is due to PCI DSS compliance.

WHAT DOES RAZORPAY POS ANDROID SDK DO?

Razorpay POS SDK enables enterprises field application with for universal payments acceptance. The SDK helps you to integrate with the Service Application of Razorpay by calling an API (Which can be initiated via a single “Pay” button on your application), Razorpay POS Service Application manages all interactions between the Card, Device and the , Razorpay POS server, and encapsulates a smooth user experience during the entire payment cycle. The below figure showcases the flow of payment using Razorpay POS SDK.

GETTING STARTED WITH ANDROID SDK INTEGRATION

We recommend that you get started with integration in the demo environment, validate the flow and use cases on your application and then move to the Production environment.
You will receive the below details from Razorpay POS Solution Consultant for integration with the demo environment.

- Appkey for Authentication
- Credentials to access the demo portal

If you haven't received these details yet, please share the below details to respective Sales or Solution Consultant for an account to be created in the Demo Environment.

- Your Organisation's Name
- Your Organisation's Address
- Contact Number & Email ID

Please note: Devices which are in debug mode have a watermark - "DEBUG only, Not for Commercial", at the bottom right corner of the screen.

01
Merchant agent logs into the android app integrated with Razorpay SDK
Merchant Busines app
Razorpay SDK
Merchant Busines app
Razorpay SDK
06
Updated merchant CRM with payment details
02
Merchant app fetches the detail from merchant server
05
Payment response sent back from Razorpay server to app
03
Payment request is sent from SDK to Razorpay server
4A
Payment horization
4B
Authentication & validation
VISA
MasterCard
RuPay
Customer server
Payment authorization
Issuer Bank

HOW TO INCLUDE PAYMENTS SDK IN YOUR ANDROID APPLICATION

Razorpay POS payments SDK can be integrated with different types of android SDK such as Native, react native, Cordova, etc. Integrating the SDK will allow you to access Razorpay POS services in your application.
For step-by-step guide with code and other details, please refer to our GitHub portal here . You will find the following details there :

- What you need for Integration
- Sample App
- Documentation to create Cordova plugin for Android
- Sample code for **Native** / **Cordova** / **React Native**

HOW DOES THE POS ANDROID SDK WORK?

- Include the SDK in your mobile application to collect digital payments.
- SDK interfaces with Razorpay Service Application, which has a pet name Service App.
- The SDK inspects the availability of Service App on Android device and if not present it will be installed at run-time.
- **Service App** interfaces with the Card Device (in card of card payments) and Razorpay POS Servers to finish payment processing and notifies the final status to SDK/Client App.

For more details, please refer to our **portal**. You will find the following details

- Invoking Razorpay POS API
- Handling Responses
- Sample code to fetch the response
- Response Structure
- Sample Response

01 An API call launches the service app

02 Service app communicates with Razorpay

Service app

Server

04 Service app finishes the txn processing & notifies the SDK

03 Server sends the request back to service app

STEP 1 : INITIALISING THE SDK

The first API that needs to be integrated with is the “Initialize” API, this API performs the following 3 key activities and 1 optional activity:

- Initializes the SDK with global configuration settings
- Connects to the appropriate Razorpay POS server based on Application mode (AppMode) = DEMO/ PROD
- Connects to the appropriate merchant account, this depends on the app key entered
- Optionally invokes prepare Device to initialize the device (card reader) with the updated encryption keys from the corresponding bank

Initialize is the first method to be called. It is recommended to call this method after the user logs in to your application or (if login is not available) when he reaches the home screen of your application.
For step-by-step guide to initialize the SDK, please refer to our portal here -
You will find the following details:

- How to call initialize API
- How to prepare input for initialize API
- How to invoke the initialize API
- How to handle the response of Initialize API
- Sample Request and Response

STEP 2: TRANSACTION PAYMENT RESPONSE VIA AND SERVER TO SERVER CALL-BACK

The transaction response can be provisioned back to the client application using the SDK itself using onResultActivity method. However, if the client is looking for transaction response provisioning in some other way, Razorpay POS can provide a sever to server call back for each and every successful transaction. Also there will be a retrial mechanism to retry posting up to 3 times if http status 200 is not received as an acknowledgement in the previous attempts. Client will have to provide an endpoint on which the details will be sent for successful transactions.

STEP 3: UNIVERSAL PAY API FOR PAYMENT TRANSACTION

Razorpay POS has a universal pay API through which all the payment modes (that is enabled for the merchant) can be invoked through a single API call. With this API there will be no need of calling the individual methods for different payment modes (like Card, Remote Pay, QR etc).
Please refer to our portal here -
You will find the following details:

- How to prepare Input for the Pay API
- How to invoke Pay API
- How to handle the response of Pay API
- Basic structure of Pay API response
- Sample Pay API response for payment by Card
- How to handle payment failures

STEP 4: CARD PAYMENT API FOR PAYMENT VIA CARD

In addition to the universal Pay API described above, Card Payment API is also available for card-based payment transaction.
Please refer to our portal here -
You will find the following details:

- How to invoke Card Payment API
- JSON request
- Sample Request
- Handling the payment response
- Sample Response

STEP 5: HOW TO IDENTIFY A SUCCESSFUL CARD TRANSACTION

For Card transaction, please rely on the Status Field in the Pay API or Card Payment API response to identify a successful transaction

Status	What it means
Authorized	Transaction has been successfully executed
Failed	Transaction has not been executed and has failed; the money won't be deducted in this scenario
Voided	The transaction was authorized, and which is now has been voided.
Refunded	The transaction was completed and after which it was refunded

APIS FOR OTHER PAYMENT MODES SUCH AS UPI, QR, ETC.

APIs for other payment modes such as UPI, Cash, Cheque, etc. are also available. Please refer below for details of all such APIs and respective links to our portal.

API for payment menthods	Link to portal
Cash	**Link**
Cheque	**Link**
UPI	**Link**
Remote Pay	**Link**
QR Code	**Link**
Wallet	**Link**

Upon clicking the link provided in the table above, you will find below information

- How to prepare input for API
- How to invoke the API
- How to handle response of the API
- Sample Request & Response

ADDITIONAL APIS FOR OTHER OPERATIONS

Some other useful APIs that may be required for payment and related operations are also available. Please refer below for details of all such APIs and respective links to our portal.

API	Usage	Link to portal
Service Fee	To add Service Fee	**Link**
Accepting Meal Cards	To accept payment via meal cards	**Link**
Void Payment API	To process Refund on same day	**Link**
Check for Incomplete Payment	To check for the incomplete transaction	**Link**
Fetch Payment Details	To retrieve the details of a payment transaction	**Link**
Sending Receipts	To send e-receipts via SMS/e-mail	**Link**
Print Receipts	To print receipt/charge-slip/custom slip	**Link**
Close SDK	To exit from Razorpay POS SDK	**Link**

Upon clicking the link provided in the table above, you will find below information

- How to prepare input for API
- How to invoke the API
- How to handle response of the API
- Sample Request & Response

PRINT CUSTOM RECEIPTS, BILLS, INVOICE, IN ANY FORMAT

Razorpay POS provides custom print SDK which allows printing of custom bills, invoices receipts in addition to the chargeslip. Bills can be printed via the device in any format, layout of our choice.
Text Size & Font:

- Text Size: setTextSize(TypedValue.COMPLEX_UNIT_PX,15sp)
- font family: lato_bold

For sample code, SDK, please refer to the link.
The way to use print bitmap function is to arrange the details in xml format and then generate a bitmap of the same and print it further using the printer SDK.
Sample code for Bitmap Printing :

```
JSONObject jsonRequest = new JSONObject();
JSONObject jsonImageObj = new JSONObject();
try {
  img.buildDrawingCache();
  Bitmap bmap = img.getDrawingCache();
  String encodedImageData = getEncoded64ImageStringFromBitmap(bmap);
  // Building Image Object
  jsonImageObj.put("imageData", encodedImageData);
  jsonImageObj.put("imageType", "JPEG");
  jsonImageObj.put("height", "");// optional
  jsonImageObj.put("weight", "");// optional
  jsonRequest.put("image", jsonImageObj); // Pass this attribute when you have a valid captured
signature image
  EzeAPI.printBitmap(this, REQUEST_CODE_PRINT_BITMAP, jsonRequest);
} catch (JSONException e) {
  e.printStackTrace();
}
```

ERROR CODES

Typically, you will notice 4 types of errors during transactions:

- Common Errors
- Server Errors
- PG Errors
- SDK Errors

Full list of these error codes is available on our **portal** here .
Please refer to this for prominent error codes and their definitions.

STEPS FOR GO LIVE

Testing the Integration

A detailed testing scenario is crucial and critical step before the roll-out to production, you can test all payment related integration features End-to-End including failure cases, payment gateways errors etc. To facilitate this on our demo environment we do simulation of error mapped to certain amounts. Below you can find the amounts and mapped error cases.

Best Practice: Razorpay POS recommends testing at least 50% of these scenarios to conclude your UAT

Please refer to our portal **here.**
You will find the details of amounts to be passed to simulate a specific error.

GO-LIVE

Once you closed your integration and testing on our DEMO environment, you should consult the Razorpay POS team and get your integration verified. Post we have confirmed and tested the integration and various scenarios; you will be eligible to go live. Here are the steps you need to follow for Going-live:

- Procure a Production device via the bank (only for card payments)
- Inform the Razorpay POS once the device has been procured
- Razorpay POS team will provide you the Production App Key, add the Production App key in your initialise API request.
- Change the App mode to PROD, from DEMO

Best Practice: Razorpay POS recommends that you perform some test Re.1 Transactions and verify the transaction flow, portal information etc.

Razorpay POS

This document has been crafted to offer an in-depth exploration of the integration steps and artefacts required for Razorpay POS Integration. For any additional inquiries, please feel free to contact us at pos-integrations@razorpay.com.