



PRESENTED BY: AIRPAY PAYMENT SERVICES

AIRPAY PAYMENT SERVICES PVT LTD

MUMBAI

Cordova Integration Kit

Abstract

This document specifies the technical aspect of integrating Cordova Plugin.

Disclaimer

This documentation shall only be used for evaluating the planned services designated herein and may contain information that is privileged, confidential, Proprietary, or otherwise protected from disclosure. As a result, this document or content thereof shall not be disclosed, used, or duplicated, in whole or in part, for any purpose other than the Scope of Work assigned by airpay Payment Services to your company ("Recipient"). Upon completion of service or termination of service, the Recipient shall return all materials, including, without limiting the generality of the foregoing, all originals, copies, reproductions, and summaries of confidential information. Any unauthorized use or disclosure by the directors, officers, or employees of the Recipient shall be deemed to be unauthorized use or disclosure by the Recipient and the Recipient shall indemnify and hold harmless the airpay Payment Services from and against all damages, losses, costs, and expenses incurred because of such breach. airpay payment services may seek injunctive relief restraining the unauthorized disclosure or use of confidential information in addition to any other legal or equitable remedy otherwise available.

Version History

VERSION #	IMPLEMENTED BY	REVISION DATE	APPROVED BY	APPROVAL DATE	REASONS
1.0	Tushar Khandekar	[03/07/2025]	Mathew Thomas	[03/07/2025]	Initial Document

Contents

Abstract	2
Disclaimer	3
Version History.....	4
Airpay Cordova Plugin.....	6
Supported Platforms.....	6
Requirements	6
Installation.....	6
Navigate to your project directory:	6
Add the necessary platforms:	6
Install the Airpay plugin:	6
Executing program.....	7
AndroidManifest.xml	7
MainActivity.java	7
Code	7
build.gradle (app).....	11
repository.gradle.....	12
MyPlugin.java	12
Configuration and Logic.....	12
Private Key Logic –	13
Key for Checksum -	14
Checksum Logic –	14
Support.....	14
Version History	14
License.....	14
Response Message –	15
Code Reference for Secure hash logic -	15
Response From the Payment gateway.....	16
Request And Response Parameter Table	17
Request Parameter: -	17
Response Parameters	19

Airpay Cordova Plugin

This is the official Cordova plugin for integrating Airpay payment gateway into Cordova applications.

Supported Platforms

- Android (version compatibility details below) Upto Gradle version 8.7

Requirements

- Cordova Version: Supports Cordova version up to v12.0.0
- NodeJS Version: Supports NodeJS version up to v22.11.0
- Android Version: Compatible with Android SDK min version 24 to max version 34
- Android Build Tool: Supports Android Build Tool up to v22.11.0

Installation

Note: For Windows users, please run the following commands on Git Bash instead of Command Prompt. You can download Git for Windows [here](#).

Plugin Link - <https://github.com/Airpay2014/airpay-cordova-V4-India>

Navigate to your project directory:

```
cd your-project-folder
```

Add the necessary platforms:

```
cordova platform add android
```

Install the Airpay plugin:

```
cordova plugin add https://github.com/Airpay2014/airpay-cordova-V4-India#master
```

Executing program

The following Android files are crucial for the Airpay Cordova integration and ensure seamless functionality:

AndroidManifest.xml

Manages app-level configurations and permissions. Add below code inside Application tag.

```
<uses-library  
  android:name="org.apache.http.legacy"  
  android:required="false" />
```

Make sure to give Internet Permission.

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

MainActivity.java

Serves as the main entry point for your Cordova application.

Code

```
import android.content.Intent;  
import android.os.Bundle;  
import android.text.TextUtils;  
import android.util.Log;  
import android.widget.Toast;  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.zip.CRC32;  
import androidx.activity.result.ActivityResult;  
import androidx.activity.result.ActivityResultCallback;  
import androidx.activity.result.ActivityResultLauncher;  
import androidx.activity.result.contract.ActivityResultContracts;  
import com.airpay.airpaysdk_simplifiedotp.utils.Transaction;  
import com.airpay.airpaysdk_simplifiedotp.view.ActionResultListener;  
import com.example.myPlugin.MyPlugin;  
import org.apache.cordova.*;  
import org.json.JSONObject;
```

```
import java.util.zip.CRC32;
```

```
public class MainActivity extends CordovaActivity implements ActionListener
```

```

{
    public static ActivityResultLauncher<Intent> activityResultLauncher;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        activityResultLauncher = registerForActivityResult(
            new ActivityResultContracts.StartActivityForResult(),
            new ActivityResultCallback<ActivityResult>() {
                @Override
                public void onActivityResult(ActivityResult result) {
                    if (result.getResultCode() == RESULT_OK) {
                        Intent data = result.getData();
                        if (data != null) {
                            // Assuming Transaction is returned in the intent
                            Transaction transaction = (Transaction) data.getSerializableExtra("response");
                            if (transaction != null) {
                                Log.d("MainActivity", "Transaction status: " + transaction.getSTATUS());
                                // Handle successful transaction
                                Toast.makeText(MainActivity.this, ""+transaction.getSTATUS(),
                                    Toast.LENGTH_SHORT).show();

                                String transactionDetails =
                                    "\nSTATUS: " + transaction.getSTATUS() +
                                    "\nSTATUSMSG: " + transaction.getSTATUSMSG() +
                                    "\nTXN_MODE: " + transaction.getTXN_MODE() +
                                    "\nTRANSACTIONID: " + transaction.getTRANSACTIONID() +
                                    "\nTRANSACTIONAMT: " + transaction.getTRANSACTIONAMT() +
                                    "\nTRANSACTIONSTATUS: " + transaction.getTRANSACTIONSTATUS() +
                                    "\nMERCHANTTRANSACTIONID: " + transaction.getMERCHANTTRANSACTIONID()

                                +
                                    "\nMERCHANTPOSTTYPE: " + transaction.getMERCHANTPOSTTYPE() +
                                    "\nMERCHANTKEY: " + transaction.getMERCHANTKEY() +
                                    "\nSECUREHASH: " + transaction.getSECUREHASH() +
                                    "\nCUSTOMVAR: " + transaction.getCUSTOMVAR() +
                                    "\nTXN_DATE_TIME: " + transaction.getTXN_DATE_TIME() +
                                    "\nTXN_CURRENCY_CODE: " + transaction.getTXN_CURRENCY_CODE() +
                                    "\nTRANSACTIONVARIANT: " + transaction.getTRANSACTIONVARIANT() +
                                    "\nCHMOD: " + transaction.getCHMOD() +
                                    "\nBANKNAME: " + transaction.getBANKNAME() +
                                    "\nCARDISSUER: " + transaction.getCARDISSUER() +
                                    "\nFULLNAME: " + transaction.getFULLNAME() +
                                    "\nEMAIL: " + transaction.getEMAIL() +
                                    "\nCONTACTNO: " + transaction.getCONTACTNO() +
                                    "\nISRISK: " + transaction.getISRISK() +
                                    "\nMERCHANT_NAME: " + transaction.getMERCHANT_NAME() +
                                    "\nSETTLEMENT_DATE: " + transaction.getSETTLEMENT_DATE() +
                                    "\nSURCHARGE: " + transaction.getSURCHARGE() +
                                    "\nBILLEDAMOUNT: " + transaction.getBILLEDAMOUNT() +

```



```

        "\nCUSTOMERVPA: " + transaction.getCUSTOMERVPA() +
        "\nAdditional Data (myMap): " + transaction.getMyMap().toString();

        MyPlugin.greet(transactionDetails, MyPlugin.callbackContext);
    } else {
        Toast.makeText(MainActivity.this, "Transaction object is null",
Toast.LENGTH_SHORT).show();
    }
    } else {
        Log.d("MainActivity", "Intent data is null");
    }
    } else {
        // Toast.makeText(MainActivity.this, "Payment result not OK, Result Code: ",
Toast.LENGTH_SHORT).show();
        Log.d("MainActivity", "Payment result not OK, Result Code: " + result.getResultCode());
    }
    }
    }
    );

    // enable Cordova apps to be started in the background
    Bundle extras = getIntent().getExtras();
    if (extras != null && extras.getBoolean("cdvStartInBackground", false)) {
        moveTaskToBack(true);
    }

    // Set by <content src="index.html" /> in config.xml
    loadUrl(launchUrl);
}

public static ActivityResultLauncher<Intent> getActivityResultLauncher() {
    return activityResultLauncher;
}

@Override
public void onActivityResult(Object o) {
    if (o instanceof Transaction) {
        Transaction transaction = (Transaction) o;

        Toast.makeText(MainActivity.this, transaction.getTransactionStatus() + "\n" +
transaction.getStatusMsg(), Toast.LENGTH_LONG).show();
        if (transaction.getStatus() != null) {
            Log.e("STATUS -> ", "=" + transaction.getStatus());
        }
        if (transaction.getMerchantKey() != null) {
            Log.e("MERCHANT KEY -> ", "=" + transaction.getMerchantKey());
        }
        if (transaction.getMerchantPostType() != null) {
            Log.e("MERCHANT POST TYPE ", "=" +
transaction.getMerchantPostType());
        }
    }
}

```

```
if (transaction.getStatusMSG() != null) {
    Log.e("STATUS MSG -> ", "=" + transaction.getStatusMSG()); // success or fail
}
if (transaction.getTransactionAMT() != null) {
    Log.e("TRANSACTION AMT -> ", "=" + transaction.getTransactionAMT());
}
if (transaction.getTXN_MODE() != null) {
    Log.e("TXN MODE -> ", "=" + transaction.getTXN_MODE());
}
if (transaction.getMerchantTransactionID() != null) {
    Log.e("MERCHANT_TXN_ID -> ", "=" + transaction.getMerchantTransactionID()); // order id
}
if (transaction.getSecureHash() != null) {
    Log.e("SECURE HASH -> ", "=" + transaction.getSecureHash());
}
if (transaction.getCustomVar() != null) {
    Log.e("CUSTOMVAR -> ", "=" + transaction.getCustomVar());
}
if (transaction.getTransactionID() != null) {
    Log.e("TXN ID -> ", "=" + transaction.getTransactionID());
}
if (transaction.getTransactionStatus() != null) {
    Log.e("TXN STATUS -> ", "=" + transaction.getTransactionStatus());
}
if (transaction.getTXN_DATE_TIME() != null) {
    Log.e("TXN_DATETIME -> ", "=" + transaction.getTXN_DATE_TIME());
}
if (transaction.getTXN_CURRENCY_CODE() != null) {
    Log.e("TXN_CURRENCY_CODE -> ", "=" + transaction.getTXN_CURRENCY_CODE());
}
if (transaction.getTransactionVariant() != null) {
    Log.e("TRANSACTIONVARIANT -> ", "=" + transaction.getTransactionVariant());
}
if (transaction.getCHMOD() != null) {
    Log.e("CHMOD -> ", "=" + transaction.getCHMOD());
}
if (transaction.getBankName() != null) {
    Log.e("BANKNAME -> ", "=" + transaction.getBankName());
}
if (transaction.getCardIssuer() != null) {
    Log.e("CARDISSUER -> ", "=" + transaction.getCardIssuer());
}
if (transaction.getFullName() != null) {
    Log.e("FULLNAME -> ", "=" + transaction.getFullName());
}
if (transaction.getEmail() != null) {
    Log.e("EMAIL -> ", "=" + transaction.getEmail());
}
if (transaction.getContactNO() != null) {
    Log.e("CONTACTNO -> ", "=" + transaction.getContactNO());
}
```

```

    if (transaction.getMERCHANT_NAME() != null) {
        Log.e("MERCHANT_NAME -> ", "=" + transaction.getMERCHANT_NAME());
    }
    if (transaction.getSETTLEMENT_DATE() != null) {
        Log.e("SETTLEMENT_DATE -> ", "=" + transaction.getSETTLEMENT_DATE());
    }
    if (transaction.getSURCHARGE() != null) {
        Log.e("SURCHARGE -> ", "=" + transaction.getSURCHARGE());
    }
    if (transaction.getBILLEDAMOUNT() != null) {
        Log.e("BILLEDAMOUNT -> ", "=" + transaction.getBILLEDAMOUNT());
    }
    if (transaction.getISRISK() != null) {
        Log.e("ISRISK -> ", "=" + transaction.getISRISK());
    }
    String transid = transaction.getMERCHANTTRANSACTIONID();
    String apTransactionID = transaction.getTransactionID();
    String amount = transaction.getTransactionAMT();
    String transtatus = transaction.getTransactionSTATUS();
    String message = transaction.getStatusMSG();

    String customer_vpa = "";
    if (!TextUtils.isEmpty(transaction.getCHMOD()) && transaction.getCHMOD().equalsIgnoreCase("upi")) {
        customer_vpa = ":" + transaction.getCUSTOMERVPA();
        Log.e("Verified Hash ==", "INSIDE CHMODE UPI CONSIDTION");
    }

    String merchantid = ""; //Please enter Merchant Id
    String username = ""; //Please enter Username
    String sParam = transid + ":" + apTransactionID + ":" + amount + ":" + transtatus + ":" + message + ":" +
merchantid + ":" + username + customer_vpa;
    CRC32 crc = new CRC32();
    crc.update(sParam.getBytes());
    String sCRC = "" + crc.getValue();
    Log.e("Verified Hash ==", "sParam= " + sParam);
    Log.e("Verified Hash ==", "Calculate Hash= " + sCRC);
    Log.e("Verified Hash ==", "RESP Secure Hash= " + transaction.getSECUREHASH());

    if (sCRC.equalsIgnoreCase(transaction.getSECUREHASH())) {
        Log.e("Verified Hash ==", "SECURE HASH MATCHED");
    } else {
        Log.e("Verified Hash ==", "SECURE HASH MIS-MATCHED");
    }
}
}
}
}

```

build.gradle (app)

Defines build configurations and dependencies for the app. For latest dependency, kindly refer the kit on sanctum portal link.

```
implementation("com.airpay:Airpay-India-Kit-V4:1.0.0") {
    exclude group: 'androidx.core', module: 'core'
    // or
    exclude group: 'androidx.legacy', module: 'legacy-support-v4'
}

implementation 'com.android.volley:volley:1.2.1'
```

repository.gradle

Specifies repositories for dependency management. For Username and Password, kindly refer the kit on sanctum portal link.

```
maven {
    url "https://gitlab.com/api/v4/projects/69276629/packages/maven"
    credentials(HttpHeaderCredentials) {
        name = "Private-Token"
        value = "glpat-T1rgMKUrc686pzp_s9yw"
    }
    authentication {
        header(HttpHeaderAuthentication)
    }
}
```

MyPlugin.java

Implements the Cordova plugin's core logic. Please enter the merchant configuration details in required below fields

// Merchant details

```
String sMid = ""; // Please enter Merchant Id
String sSecret = ""; // Please enter Secret Key
String sUserName = ""; // Please enter Username
String sPassword = ""; // Please enter Password
String client_id = ""; //Please enter Client Id
String client_secret = ""; //Please enter Client Secret
```

```
.setSuccessUrl("") // Please enter Success URL
.setFailedUrl("") // Please enter Failed URL
```

Note :- For above merchant configuration details, Kindly contact with Airpay Support Team.

Configuration and Logic

Note:

Set environment variable value as per setup.

For Production - Please use the value of **PRODUCTION**

For Staging - Please use the value of **STAGING**

Private Key Logic –

```
String sSecret = ""; // Please enter secret key value
String sUserName = ""; // Please enter username value String sPassword = ""; // Please enter password value
String sTemp = sSecret+"@"+sUserName+":|"+sPassword;
String sPrivateKey = sha256(sTemp);
Checksum Logic –
String sAllData1;
DateFormat df1 = new SimpleDateFormat("yyyy-MM-dd");
String sCurDate1 = df1.format(new Date());
```

```
if (!TextUtils.isEmpty(txnSubtype) && txnSubtype.equals("12")) {
```

```
    if ("A".equalsIgnoreCase(period)) {
        sAllSubscriptionData = period +
            appendDecimal(subscriptionAmount) +
            "1" +
            retryAttempts;
```

```
    } else {
        sAllSubscriptionData = nextRunDate +
            frequency +
            period +
            appendDecimal(subscriptionAmount) +
            "1" +
            recurringCount +
            retryAttempts;
    }
```

```
sAllData1 = email + firstName
    + lastName + address
    + city + state
    + country + appendDecimal(amount)
    + orderId + sAllSubscriptionData + sCurDate1;
```

```
} else {  
    sAllData1 = email + firstName  
        + lastName + address  
        + city + state  
        + country + appendDecimal(amount)  
        + orderId + sCurDate1;  
}
```

Key for Checksum -

```
String sTemp3 = sUserName + "~::~" + sPassword;  
String sKey1 = Utils.sha256(sTemp3);
```

Checksum Logic –

```
// checksum  
sAllData1 = sKey1 + "@::" + sAllData1;  
String sChecksum1 = Utils.sha256(sAllData1);
```

Support

- Tech/integration Support Team
- Customer Support Team

Version History

- 0.1
 - Initial Release

License

This project is licensed under the [Airpay Payment Service] License

Response Message –

The response can be retrieved in either the ``onResult`` or ``onActivityResult`` method, depending on your preference. Please choose whichever method is more convenient for handling the response.

We can validate the server response by comparing the secure hash values. If the secure hash value from the server response matches the calculated secure hash, the response is considered valid from the server's end.

Code Reference for Secure hash logic -

Below code was handled inside the `onResult` method of `MainActivity.java` class.

```
String transid = transaction.getMERCHANTTRANSACTIONID();
String apTransactionID = transaction.getTransactionID();
String amount = transaction.getTransactionAMT();
String transtatus = transaction.getTransactionSTATUS();
String message = transaction.getStatusMSG();
String customer_vpa = "";
if (!TextUtils.isEmpty(transaction.getCHMOD()) && transaction.getCHMOD().equalsIgnoreCase("upi")) {
    customer_vpa = ":" + transaction.getCUSTOMERVPA();
    Log.e("Verified Hash ==", "INSIDE CHMODE UPI CONSIDTION");
}
String merchantid = ""; //Please enter Merchant Id
String username = ""; //Please enter Username
String sParam = transid + ":" + apTransactionID + ":" + amount + ":" + transtatus + ":" + message + ":" +
merchantid + ":" + username + customer_vpa;
CRC32 crc = new CRC32();
crc.update(sParam.getBytes());
String sCRC = "" + crc.getValue();
Log.e("Verified Hash ==", "sParam= " + sParam);
Log.e("Verified Hash ==", "Calculate Hash= " + sCRC);
Log.e("Verified Hash ==", "RESP Secure Hash= " + transaction.getSECUREHASH());
if (sCRC.equalsIgnoreCase(transaction.getSECUREHASH())) {
    Log.e("Verified Hash ==", "SECURE HASH MATCHED");
} else {
    Log.e("Verified Hash ==", "SECURE HASH MIS-MATCHED");
}
}
```

Response From the Payment gateway

getMERCHANTTRANSACTIONID() = orderid you have send to airpay system is returned back getTRANSACTIONID()

= airpay transaction reference number

getTRANSACTIONAMT() = transaction amount getSTATUS() =

successful = 200

getSTATUSMSG() = Response message received from payment gateway.

getSECUREHASH() = Secure hash generated by airpay

All fields are mandatory except MESSAGE.

(** This method is mandatory. You can get it from the settings page of your airpay merchant account. * This method is mandatory)

chmod variable contains Payment Modes available for user. for e.g. If you want to show only Credit Card/Debit Card, then value of the chmod variable will be "pg". If you want Netbanking and Prepaid card then value of the chmod variable will be "nb_ppc". If you want to show all payment options activated for you at airpay, then leave this variable blank.

Various modes passed this way separated by underscore element.

(If you want to show all payment options activated for you at airpay, then leave this variable blank.)

Request And Response Parameter Table

Request Parameter: -

<u>Method Name</u>	<u>Input Data Format</u>	<u>Data Length</u>	<u>Example</u>
setEmailId	String	6-50	abc@gmail.com
setMobileNo	String	8-15	9898989898
setBuyerFirstName	String	1-50	Name
setBuyerLastName	String	1-50	Name
setBuyerAddress	String	4-255	Mumbai
setBuyerCity	String	2-50	Mumbai
setBuyerState	String	2-50	Maharashtra
setBuyerCountry	String	2-50	India
setBuyerPinCode	String	4-8	400001
setAmount	String	1-7	50.00
setChmod	String	0-15	Payment Mode (not required) <ul style="list-style-type: none"> • ppc - prepaid card • pg - payment gateway • nb - Netbanking • pgcc - Credit card • pgdc - Debit card • cash - Cash • emi - EMI • rtgs - RTGS • upi - UPI • btqr - Bharat QR • payltr - Pay later • va - Virtual account • enach - eNACH • remit - Remittance

			chmod variable contains Payment Modes available for user. for e.g. If you want to show only Credit Card/Debit Card, then value of the chmod variable will be "pg". If you want Netbanking and Prepaid card then value of the chmod variable will be "nb_ppc".If you want to show all payment options activated for you at airpay, then leave this variable blank.
setChecksum	String	1-100	Generate Checksum **
setPrivateKey	String	1-100	Generate Private Key **
setMerchantId	String	1-15	Provide by airpay **
setAppPackage	String	10-255	Generated the Package Name **
setSuccessUrl	String	10-255	Success Url
setFailedUrl	String	10-255	
setCustomVar	String	1-100	Alphanumeric, Space, =
setTxnSubType	String	1	3
setWallet("0")	String	1	Numeric
setCurrency("356")	Integer	3	Eg. 356
setIsoCurrency("INR")	String	3	Eg. INR
setOrderId	String	20	Test123
setLanguage("EN")	String	2	en, hi, mr, ar, ta, gu, bn, or, as, ma
sb_nextrundate	String		Next subscription date (length 103 for enabling subscription) (required) mm/dd/yyyy date must be current date+1 (t+1)
sb_period	String	1	Subscription period (length 1 for enabling subscription) (required) - D W M Y A Day/Week/Month/Year/Adhoc
sb_frequency	String	1-999	Subscription frequency (length 1-999 for enabling subscription) (required)
sb_amount	String	1-7	Subscription amount (length 1-6 .2 for enabling subscription) (required)
sb_isrecurring	String	1	Is subscription recurring (length 1 for enabling subscription) (required)
sb_recurringcount	String	1-999	Subscription Recurring Count (length 1-999 for enabling subscription and Is Subscription Recurring is Yes , if recurring count is 999 then subscription is set as never ending end date its apply only for enach transaction) (required)
sb_retryattempts	String	1	Subscription retry attempts (length 1 for enabling subscription) (required)
sb_maxamount	String	1-7	Maximum amount can charge, and greater than or equal to the amount

Response Parameters

<u>Field</u>	<u>Type</u>	<u>Description</u>
STATUS	Numeric	Transaction Payment Status (required) Success - 200 Transaction in Process - 211 Failed - 400 Dropped - 401 Cancel - 402 Incomplete - 403 Bounced - 405 No Records – 503
TXN_MODE	Alphanumeric	Transaction mode LIVE or Sandbox
TXN_DATE_TIME	Date	Transaction Date and Time
TXN_CURRENCY_CODE	Numeric	Payment Currency (eg:- 356)
CURRENCY_CODE	Alphanumeric	Payment Currency (eg:- inr)
TRANSACTIONID	Numeric	orderid you have send to airpay system (required)
TRANSACTIONAMT	Numeric	Transaction amount (required)
TRANSACTIONSTATUS	Numeric	Transaction status code (eg:- 200(success),402(cancelled))
STATUSMSG	Alphanumeric	Transaction payment status SUCCESS TRANSACTION IN PROCESS FAILED DROPPED CANCEL INCOMPLETE BOUNCED NO RECORDS
MERCHANTTRANSACTIONID	Numeric	Merchant transaction id.
MERCHANTKEY	Numeric	Merchant key.
CUSTOMVAR	Alphanumeric	customvar value received from payment gateway.
AP_SECUREHASH	Alphanumeric	Secure hash generated by Airpay Server.

CHMOD	Alphanumeric	Chanel of Payment done
FULLNAME	Alphanumeric	Customer name.
EMAIL	Alphanumeric	Customer Email
CONTACTNO	Numeric	Customer phone
APTRANSACTIONID	Numeric	airpay transaction reference number (required)
ISRISK	Numeric	If the transaction is risk
BILLEDAMOUNT	Numeric	Billed Amount
CUSTOMERVPA	Alphanumeric	VPA will return if channel is upi
CARDISSUER	Alphanumeric	Card issuer name, this field is available in pg,emi,express payment
CARD_DETAILS	Numeric	Card number masked, this field is available in pg,emi,express payment
BANKNAME	Alphanumeric	Name of the bank, this field is available in pg,emi,pos
CARDCOUNTRY	Alphanumeric	Card issued country, this field is available in pg,emi,pos
CARDTYPE	Alphanumeric	Type of Card Credit/Debit/Unknown
TRANSACTIONREASON	Alphanumeric	Failed Reason